UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

UNIVERSITY OF MODENA AND REGGIO EMILIA

"Enzo Ferrari" Department of Engineering

Master's Degree in Artificial Intelligence Engineering (LM-32)

Fine-Grained Cross-Modal Alignment for Improving Visual Grounding in Multimodal Large Language Models

Supervisors:
Prof. Lorenzo Baraldi
Prof. Marcella Cornia
Co-supervisors:
Dott. Sara Sarto
Dott. Davide Caffagni

Candidate:
Filippo Garagnani
Student ID 199670

ACADEMIC YEAR 2025/2026

Contents

List of Figures

List of Algorithms

Riassunto in lingua italiana

Multimodal Large Language Models (MLLMs) forniscono a un language model la possibilità di gestire input visuali, permettendo così a un singolo modello di lavorare e ragionare sul testo e sulle immagini contemporaneamente. Sono diventati uno strumento centrale nell'AI moderna, poiché svolgono compiti quali l'*image captioning*, il *visual question answering* e una generica comprensione multimediale, dove il contesto visuale e il linguaggio naturale devono essere interpretati assieme. Nonostante i recenti progressi, gli MLLM presentano ancora problemi nel fornire una corrispondenza dettagliata tra le rappresentazioni visuali e testuali. Questo debole allineamento tra le modalità spesso comporta il fenomeno delle allucinazioni: il modello genera un contenuto che non è supportato o è incoerente con l'immagine di input. Questo comportamento diminuisce la loro affidabilità, in particolare in contesti che richiedono risposte accurate e verificabili. Questa tesi studia una strategia di associazione tra testo e immagine, con l'obiettivo di migliorare l'allineamento nella generazione multimodale. Il metodo proposto collega esplicitamente intervalli del testo generato alle regioni dell'immagine che descrive, forzando interazioni più localizzate tra le *features* visuali e i *tokens* testuali. Invece che basarsi solo su rappresentazioni globali fra le modalità, questo collegamento è forzato attraverso un obiettivo aggiuntivo di allineamento durante l'allenamento del modello, con lo scopo di produrre risposte che sono maggiormente supportate dall'input visuale preservando, però, le capacità visuali del modello multimodale inalterate. Il metodo è valutato su architetture basate su LLaVA, costruite con componenti diverse ed allenate con varie configurazioni. Un confronto sistematico delle strategie di allineamento valuta il contributo di ciascuna nel ridurre le allucinazioni.

Abstract

Multimodal Large Language Models (MLLMs) couple a language model with visual inputs, letting a single model process and reason over both text and images. They have become a central tool in modern AI, supporting tasks such as image captioning, visual question answering and broader multimedia understanding, where visual context and natural language must be interpreted together. Despite this progress, MLLMs still struggle to establish fine-grained correspondences between visual and textual representations. This weak cross-modal grounding often surfaces as hallucination: the model produces content that is not supported by, or is inconsistent with, the input image. Such behavior undermines their reliability, especially in applications that require accurate and verifiable answers. This thesis studies a visual-to-text alignment strategy aimed at improving grounding in multimodal generation. The proposed method explicitly links spans of generated text to the image regions they describe, encouraging more localized interactions between visual features and language tokens. Rather than relying only on global cross-modal representations, this link is enforced through an auxiliary alignment objective during training, with the goal of producing responses that are more faithfully grounded in the visual input while preserving the model's general multimodal ability. The method is evaluated on LLaVA-based architectures, instantiated with different components and trained under varying configurations. A systematic comparison of the alignment strategies assesses their respective contributions to reducing hallucination.

Chapter 1

Introduction

In recent years, the field of Artificial Intelligence has seen a significant surge, both for its superhuman capabilities and for its increasing number of possible applications. This has also been thanks to the advent of Large Language Models, systems capable of generating coherent text and following human instructions [1]. In order to apply these models in several fields, the more advanced Multimodal Large Language Models (MLLMs) have been developed [52]. These are able to understand, reason and solve problems via different modalities, in particular text and vision. This new category of models has been able to tackle tasks with excellent results, such as visual question answering and image captioning [32].

Despite the capabilities these models possess, a phenomenon in which the response given by the MLLMs is not coherent with the visual input provided has emerged, and it has been called *hallucination* [7]. More specifically, hallucinations may manifest as the model omitting certain elements of the visual input, describing incorrect objects' attributes, mistaking the relationships between the objects themselves, or even mentioning non-existent objects altogether [48, 74]. Especially in critical applications, this phenomenon is highly problematic, as it can cause dangerous misinterpretations.

The aim of this thesis is to directly tackle one possible cause of hallucinations, which is a weak alignment of visual and textual information, often referred to as *modality gap*. Rather than a coarse-grained additional alignment between the full image representation and the caption describing it, the proposed approach enforces a fine-grained alignment between specific portions of the caption and regions of the image described by them at the level of their latent representations, extracted from a

frozen visual teacher model. This alignment is introduced as an auxiliary training objective, which forces the MLLM to caption a given image while keeping visually-consistent textual representations during training. It will be shown that the method is able to reduce object-level hallucination in MLLMs, while preserving general visual capabilities of the model. The work of this thesis is organized as follows.

Chapter 2 provides the necessary background information on which the proposed method builds upon. The chapter firstly overviews the broad field of Deep Learning, with a particular look at the paradigms of supervised and self-supervised learning, before introducing the subfield of Natural Language Processing (NLP), and in particular, the language modeling task and the main approaches proposed for it. It then introduces the Transformer, the current state-of-the-art architecture for NLP applications and for language modeling, analyzing it both in its single components and globally, and positioning it in the current literature for other applications, such as computer vision. Building on these, a comprehensive study of Large Language Models is given, stemming from their architectural differences with the original Transformer and from the current paradigms in their training procedures, and then a selected set of Large Language Model families that will be used as foundations for the proposed method is analyzed. The chapter continues with representation learning, overviewing how strong representations have been built for natural language and visual data, and how they have been jointly aligned into a shared latent space, before examining MLLMs: this is done from the analysis of the state-of-the-art LLaVA architecture to a description of how these models are built, trained and evaluated. Finally, the chapter closes with an analysis of the phenomenon of hallucinations, delving into its manifestations, its causes and how it can be analytically computed and evaluated, and ends with a brief literature overview of how bridging the aforementioned modality gap helps in reducing hallucinations.

Chapter 3, instead, introduces the method proposed in this thesis. It first describes the MLLM configuration employed and the GranD dataset exploited by the methodology, and provides the reader with a few useful notations. It then describes the Region-Level Alignment additional objective, specifying its internal mechanisms, and enumerates the design choices available for each component of the method and discusses them in detail. It finally provides a brief overview of an additional studied methodology, which, at its core, aligns the internal patch representations of the image throughout the layers of the model with the output of a frozen teacher.

Chapter 4 develops on the methods and presents the results obtained via its application. It first provides a step-by-step decomposition of the whole experimental setup, starting from the model and the training specifics, to a brief analysis of the evaluation benchmarks employed to understand the impact of the method. Then, the results of the proposed methodology are detailed, with a particular focus on the reduction of the hallucination phenomenon. The chapter then presents an ablation study over a set of different implementations of the components of the method, and how they impact the final result. A brief qualitative set of results is then followed by a discussion of what the method is achieving and what are its limitations. Finally, the chapter ends with the experimental analysis of the alternative method proposed alongside the Region-Level alignment.

Finally, Appendix A reports the complete benchmark results underlying the ablation studies of Chapter 4, of which only a subset, for simplicity purposes, is presented in the main chapter.

Chapter 2

Background

2.1 Deep Learning

Deep Learning is a methodological branch of Artificial Intelligence based on parametrized models composed of multiple layers of interconnected nonlinear transformations, capable of learning representations from data. Its capabilities have made it a powerful tool, enabling advances across both industry and research [62].

A *model* is a mathematical representation that approximates an underlying relationship between inputs and outputs, enabling predictions or decisions. In Deep Learning, a model consists of the previously mentioned layers; each of these is defined via real-valued parameters. These parameters are *learned* through an optimization procedure aimed at minimizing errors on given tasks [44]. There are various paradigms in Deep Learning, mainly taken from the broader field of Machine Learning: *supervised learning*, *unsupervised learning*, *self-supervised learning*, and *reinforcement learning*.

In supervised learning, the task requires a mapping between inputs and their associated labels. The tasks can be mainly divided into two categories: *classification*, where the outputs belong to a discrete set, and *regression*, where the output is a continuous value. In unsupervised learning, there is no predetermined output; instead, the model learns underlying patterns in the input data distribution. Among the unsupervised tasks, the most common are *clustering*, aiming at grouping similar samples, and *dimensionality reduction*, which tries to reduce the overall dimension of the data while preserving its structure. Self-supervised learning is a paradigm that lies between

supervised and unsupervised learning. The model learns to predict a hidden portion or manipulated part of the input data based on its visible parts. The labels to predict are automatically created from the unlabelled data itself. The self-supervised approach is mainly meant to create meaningful internal representations of the data. Finally, reinforcement learning involves a model learning to perform a task by choosing actions to maximise a given reward signal.

Supervised Learning. Supervised Learning is a paradigm that applies to tasks requiring a map between a set of *data points* $X = \{x_1, x_2, \ldots, x_m\}$ and a set of *labels* $Y = \{y_1, y_2, \ldots, y_m\}$.

A *hypothesis* is a candidate function h_ϕ that is used instead of the real, unknown, mapping between data points and labels; ϕ represents instead the set of parameters that the hypothesis function requires, which represents a model of the relationship between inputs and outputs. For instance, the proposed label by an hypothesis $h_\phi(\cdot)$ for an input point x_i is $\hat{y}_i = h_\phi(x_i)$; the predicted value may be different from the *real* label y_i, mainly depending on the goodness of the hypothesis function itself.

To evaluate the quality of the given hypothesis, a *loss* – also called *error* – function $\mathcal{L}$ is generally defined, giving a numerical estimate of the distance between a prediction and the real label of a data point x_i:

$$\mathcal{L} : (h_\phi(x_i), y_i) \to \mathbb{R}$$

The loss computation can be performed over the entirety of X through the *cost function* J:

$$J(\phi) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(h_\phi(x_i),\ y_i\right)$$

As expressed above, the cost function J solely depends on the set of parameters ϕ. This means that, ideally, knowing the closed form of J could give a way to determine the minimum value of J by computing its derivative w.r.t. ϕ:

$$\nabla_\phi\, J(\hat{\phi}) = 0$$

Conceptually, what this means is that it could be possible to determine – given h – the set of parameters that yields the overall best results on the datasets X, Y.

Unfortunately, generally speaking, determining the global optimum $\hat{\phi}$ analytically is intractable; in order to reduce the overall error of the hypothesis, approximate methods are needed. The most commonly used is the *gradient descent* procedure. Intuitively, it updates the parameters ϕ by

employing the opposite direction of the gradient of the cost function. The direction of the negative gradient is employed because it returns the direction of maximum decrease of the function; the general update rule is the following:

$$\phi \leftarrow \phi - \alpha \nabla_{\phi} J(\phi)$$

where $\alpha \in \mathbb{R}$ is called *learning rate* and determines the length of the step towards the antigradient direction.

Self-Supervised Learning. The Self-Supervised Learning paradigm is quite similar to the Supervised Learning one, with the main difference that the set of data points X and the set of labels Y are dynamically determined by applying an appropriate function g. Just like in Unsupervised Learning, a single input set $Z = \{z_1, z_2, \ldots, z_m\}$ of unlabelled inputs is required. Then, by applying the possibly non-deterministic function g to any element $z_i \in Z$, one can create an input-output pair:

$$g(z_i) \rightarrow (x_i, y_i)$$

Then, on these newly generated couples, it is possible to apply the notions of Supervised Learning discussed above.

Among the most notable examples of Self-Supervised Learning, it's enough to mention *textual reconstruction* – in which, from a block of raw text words (y_i) are reconstructed from a masked portion of the text (x_i) – and *inpainting* – in which, from an image, pixels are deleted and required to be reconstructed from the masked version of the input image.

Generally speaking, Self-Supervised Learning has been shown to enhance the capabilities of a model to generate semantically meaningful representations of data. For instance, BERT [21], a model trained on a large corpus of text to predict randomly missing words, has been shown to learn novel concepts that do not specifically adhere to any pre-defined linguistic groups [19].

Another example of the capabilities of self-supervised learning is DINO [12], a vision model trained to classify distorted and cropped instances of the same image; its learned features have been shown to explicitly capture the scene layout and the main object boundaries, as discussed in detail in Section 2.3.2.

2.1.1 Language Modeling

Natural Language Processing (NLP) is a field of Computer Science aimed at making computers understand and manage human language. Even if its origins trace back to the 1950s, the field has gained massive attention recently thanks to surprising results with Deep Learning methods.

One of the main problems in NLP is *language modeling*; that is, approximating the probability of a given word continuing a natural language sequence. Formally, given a set sequence $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, (such that every element belongs to a finite dictionary: $x_i \in V$, $\mathbf{x} \in V^n$), the goal of language modeling is to predict the probability:

$$p(y \mid [x_1, x_2, \ldots, x_n])$$

Where $y \in V$ is a possible word that could continue the sequence $\mathbf{x}$. Before the advent of Deep Learning approaches, language modeling was tackled via statistical estimations computed on large text corpora.

n-grams. n-gram is a method that approximates the probability density as:

$$p(y \mid \mathbf{x}) \simeq \frac{\text{count}\,(\mathbf{x} \circ y)}{\text{count}\,(\mathbf{x})}$$

where count$(\cdot)$ returns the number of occurrences of the sequence provided, and $\circ$ denotes a concatenation operator. Depending on the application, the vocabulary may consist either of individual characters or entire words.

In principle, increasing the value of n improves the expressive power of the model, since longer contexts can be taken into account. However, this also requires significantly larger datasets; otherwise, many sequences may never appear in the corpus, leading to sparse statistics and cases where count$(\mathbf{x}) = 0$, thus making the estimation unreliable [76].

The n-gram model is grounded in the *Markov assumption*: the probability of a word depends only on the preceding $n-1$ words, rather than on the entire preceding sequence. Formally:

$$p(x_i \mid x_1, \ldots, x_{i-1}) \approx p(x_i \mid x_{i-n+1}, \ldots, x_{i-1})$$

This simplification makes estimation tractable but introduces a fundamental limitation: any dependency beyond a window of $n-1$ words is ignored entirely.

To address the sparsity problem, various *smoothing* techniques have been proposed. Simple approaches such as Laplace (add-one) smoothing assign a small nonzero probability to unseen sequences. More sophisticated methods, such as Kneser–Ney smoothing [61] and its variants, redistribute probability mass from observed to unobserved sequences in a data-driven manner, and remained the standard for statistical language modeling for decades.

Recurrent Neural Networks. A natural way to overcome the fixed-context limitation of n-gram models is to process sequences with a *recurrent* architecture. A Recurrent Neural Network (RNN) maintains a hidden state $h_t \in \mathbb{R}^d$ that is updated at each step as a function of both the previous hidden state and the current input:

$$h_t = f_\phi(h_{t-1}, x_t)$$

In principle, the hidden state accumulates information from every preceding element, allowing the model to capture dependencies of arbitrary length. In practice, however, training RNNs via back-propagation through time causes gradients to either vanish or explode as they are propagated over many timesteps [8], making it difficult to learn long-range dependencies. The introduction of *gating mechanisms* - most notably in Long Short-Term Memory (LSTM) networks [30] - substantially mitigated the vanishing gradient problem by providing learned pathways for gradient flow. Despite this, a fundamental bottleneck remained: the sequential nature of recurrent computation prevents parallelization over the time dimension, limiting scalability on modern hardware and making it costly to train on the large corpora required for high-quality language modeling. These limitations directly motivated the development of attention-based architectures.

Neural Language Models and Pre-training. The shift from statistical to neural language models brought two fundamental changes. First, instead of counting discrete co-occurrences, a neural model learns a continuous *embedding* for each token $x_i \in V$, mapping the vocabulary into a dense vector space $\mathbb{R}^d$ where geometric proximity should reflect semantic relatedness [60] (i.e., a higher cosine similarity implies a very strong semantic correlation). Second, the conditional probability $p(y \mid \mathbf{x})$ is parameterized by a neural network rather than estimated from frequency tables; this enables greater generalization capabilities to unseen sequences. Early neural language models were based on Recurrent Neural Networks and their gated variants. As discussed above, however, their

sequential computation and difficulty with long-range dependencies limited their scalability. The introduction of the Transformer architecture [86] resolved both issues, and became the dominant backbone for language modeling.

A further decisive development was the *pre-training and fine-tuning* paradigm. Rather than training a model from scratch for a single task, a large Transformer architecture is first trained on a massive unlabelled corpus via a self-supervised objective – typically either *masked language modeling*, where randomly selected tokens are hidden and the model must predict them [21] (detailed in Section 2.3.1), or *causal language modeling*, where the model is trained to predict each token autoregressively from its left context [68], also called *next-token prediction*:

$$p_\phi\left(y_t \mid y_0, y_1, \ldots, y_{t-1}\right) \approx p(y_t \mid y_0, y_1, \ldots, y_{t-1})$$

where $p_\phi(\cdot)$ parametrizes the conditional probability on the causal context over the whole vocabulary.

Training neural models on the next-token prediction task simply requires the definition of a loss function quantifying how distant the produced probability distribution and an estimate of the actual probability distribution are. More specifically, assuming we have a vocabulary $V = \{v_1, \ldots, v_N\}$ and a sequence $[y_0, y_1, \ldots, y_n]$ with $y_j \in V$; the expected probability distribution of the input $[y_0, \ldots, y_{i-1}]$ should be the vector $\mathbf{w}$:

$$\mathbf{w}_k = \begin{cases} 1, & v_k \equiv y_i \\ 0, & \text{else} \end{cases}$$

which would be the best output the model could give us – perfectly understanding how does the sequence actually continue. Hence, the loss function can be simply computed via the so-called *cross-entropy* function:

$$\begin{aligned} \mathcal{L}_{\text{CE}}\left(\mathbf{w}, p_\phi\right) &= -\sum_{k=1}^{|V|} \mathbf{w}_k \log p_\phi(v_k \mid y_{<i}) \\ &= -\log p_\phi(y_i \mid y_{<i}), \end{aligned}$$

pushing the model towards increasing the probability assigned to the vocabulary element y_i – the continuation of the sequence – in order to decrease the value of the error function $\mathcal{L}_{\text{CE}}$.

This pre-training stage allows the model to build a strong internal representation for token embeddings. It can then be adapted to specific downstream tasks with less additional data by full fine-tuning. One could train different fine-tuned models by always starting, however, from the same base pre-trained model.

2.1.2 The Transformer

The Transformer was first introduced in 2017 [86] as a language modeling architecture, and since then it has been vastly employed in many different fields with successful results. Among its advantages are *parallelizable operations*, making the architecture scalable across different nodes, and the capability of the model to efficiently capture long-range dependencies. Moreover, the full architecture is still employed as the fundamental block of Large Language Models.

Attention Function. The *attention* mechanism is the key feature of the Transformer, enabling its aforementioned advantages. It's a function $f_a : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d}$, that processes a sequence of vectors. Each element is projected into three different spaces, each one representing a different semantic: the space of the *keys* – representing what each element embodies – the space of the *queries* – representing what each element is looking for inside the sequence – and the space of the *values* – representing the actual content of each element.

The idea is to compare one element's query with all the other elements' keys, since the latter encode the information that could be retrieved in other elements in the sequence. The goal of this comparison is to later extract the values from elements for which a query-key matching has been found.

Formally, given an element $x \in \mathbb{R}^d$, it can be projected into the three different spaces:

$$q = x\, W^Q, \quad k = x\, W^K, \quad v = x\, W^V$$

where $W^Q,\ W^K \in \mathbb{R}^{d \times d_k}$, and $W^V \in \mathbb{R}^{d \times d_v}$.

In order to calculate the attention function, it's enough to have a single query $q \in \mathbb{R}^{d_k}$ and a collection of n couples of keys and values $(k_i,\ v_i)_{i=1}^n$. Then, the core attention function can simply be computed via:

$$y = \sum_{i=1}^{n} \mathrm{sim}\,(q,\ k_i)\ v_i$$

which is nothing more than a weighted sum of the value vectors. The weights are actually a function of the similarity between the query q and the key k_i relative to the value v_i. Conceptually, the more similar q and k_i, the higher the contribution that v_i will bring to the end result. Specifically, the weights are a softmax-normalized vector, where the similarity is computed as a simple scaled dot-product:

$$\text{sim}\,(q,\ k_i) = \text{softmax}\left(\left[\frac{q \cdot k_1^T}{\sqrt{d_k}},\ \frac{q \cdot k_2^T}{\sqrt{d_k}},\ \ldots,\ \frac{q \cdot k_n^T}{\sqrt{d_k}}\right]\right)_i = \frac{e^{q \cdot k_i^T/\sqrt{d_k}}}{\sum_{j=1}^{n} e^{q \cdot k_j^T/\sqrt{d_k}}}$$

As said before, the queries are compared with the keys through the dot product and the *softmax* operation converts raw similarity values into a probability distribution. At the end, the operation returns a single vector $y \in \mathbb{R}^{d_v}$; usually, this is reprojected back via another matrix:

$$x' = yW^O$$

where $W^O \in \mathbb{R}^{d_v \times d}$, returns the output element to its original, pre-attention, dimensionality.

What actually happens while employing the values is that each output element incorporates the representations of value elements with a high matching similarity.

Attention in Matrix Form. If the input is not a single vector but a sequence of vectors, the attention function can be computed efficiently in matrix form. More specifically, given an input sequence $\mathbf{x} = [x_1,\ x_2,\ \ldots,\ x_n]$, where $x_i \in \mathbb{R}^d$, it's possible to stack each element one on top of the other in a single matrix $X \in \mathbb{R}^{n \times d}$. In this way, the i-th row of X is exactly x_i.

The projections in the three spaces are still untouched:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

The core attention function can also be extended to manage matrices instead of a single query vector. More specifically, given a query matrix $Q \in \mathbb{R}^{n \times d_k}$, and a pair of key and value matrices $K \in \mathbb{R}^{n \times d_k}, V \in \mathbb{R}^{n \times d_v}$, the output becomes:

$$Y = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right)V$$

Where implicitly $\text{softmax}(A)$ with A matrix is the $\text{softmax}(\cdot)$ function applied by row. Focusing, for instance, on a single row i of the matrix Q, we see that the core function is still untouched:

$$y_i = \text{softmax}\left(\frac{q_i \cdot K^T}{\sqrt{d_k}}\right)V$$

(see Figure 2.1).

The scaling factor $1/\sqrt{d_k}$ is introduced to prevent the dot products from growing large in magnitude as d_k increases. Since $Y \in \mathbb{R}^{n \times d_v}$, a reprojection back to the original d-dimensional space is still needed:

$$X' = YW^O$$

This formulation is quite better since it can be easily parallelized – being the function mainly a composition of matrix products.

Another interesting advantage of the attention is subtler and relies on its capability as a sequence-to-sequence function. The *path length complexity* is the number of computational steps required for information to flow between two positions in a model. In the attention function, we determine a matrix $A = QK^T$ of similarities between each query vector and every key in a single step of the operation: this gives us a path length complexity = $O(1)$. Other sequence-to-sequence approaches – such as the use of Convolutional Neural Networks or Recurrent Neural Networks – required a number of steps dependent on k before comparing x_i and x_{i+k}. Specifically, RNNs have a path length complexity $\propto O(k)$, while CNNs have it $\propto O(\log k)$.

Keeping low the path length complexity is crucial, because there is a much lower risk of losing information while it traverses the model; exactly one of the downsides of RNNs architectures [86].

Multi-Head Attention. Up until now, only a single set of projection matrices W^Q, W^K, W^V has been considered. However, it could be possible to employ a collection of H different projection matrices: $\{W_h^Q, W_h^K, W_h^V\}_{h=1}^H$. Each triplet is employed to obtain an output attention matrix $\{Y_h\}_{h=1}^H$. In order to aggregate the information from each output, every matrix is concatenated and then reprojected back to the initial d-dimensional space:

$$X' = [Y_1 \circ Y_2 \circ \cdots \circ Y_H] \cdot W^O$$

where $\circ$ is the concatenation operation.

In this case, the output reprojection matrix is $W^O \in \mathbb{R}^{Hd_v \times d}$. Each triplet is generally called *head*, and the whole operation is named *Multi-Head Attention* (see Figure 2.1).

Self-Attention and Cross-Attention. The only hard requirement of the attention operation is that the dimensionality of the queries is the same as the dimensionality of the keys – otherwise, $Q \cdot K^T$

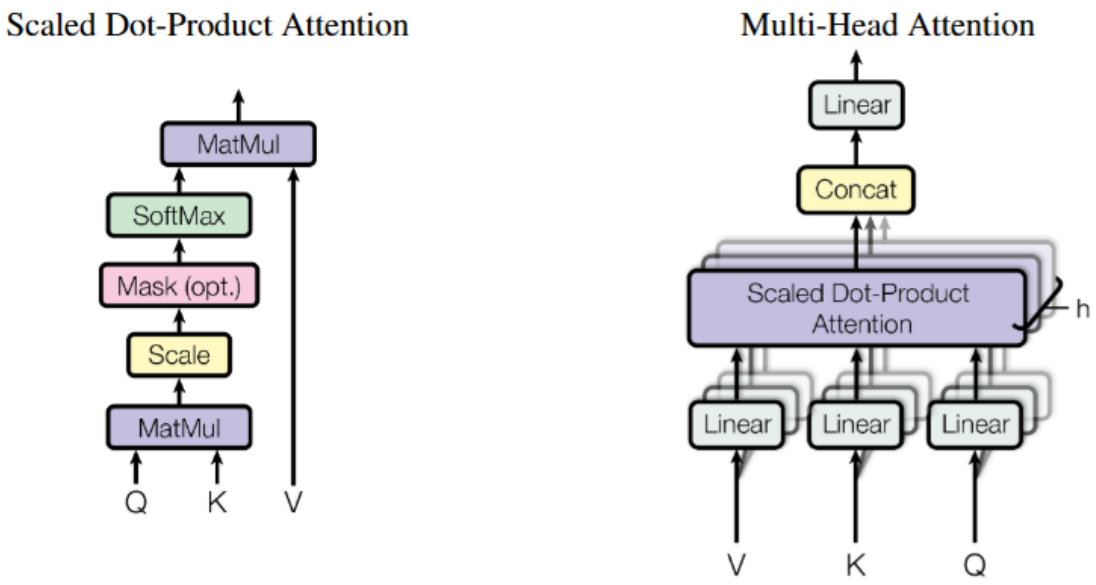

Figure 2.1: Scaled Dot-Product Attention (left) and Multi-Head Attention (right). From [86].

could not be computed – and that the overall number of keys matches the overall number of values – otherwise, $K^T \cdot V$ could not be computed.

So, attention could be computed *not necessarily* from a single sequence of elements. If all the projections come from the same sequence – i.e.:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

then the operation is usually named *Self-Attention*.

But, as said above, the projections may come from two different sequences $X_{(1)} \in \mathbb{R}^{n \times d}$, $X_{(2)} \in \mathbb{R}^{m \times d}$:

$$Q = X_{(1)}W^Q, \quad K = X_{(2)}W^K, \quad V = X_{(2)}W^V$$

in this case, the operation is called *Cross-Attention*.

Positional Encoding. The attention operation is position-agnostic. This means that by randomly permuting the elements in a sequence (i.e., performing a permutation on the rows of the input matrix), the resulting matrix is still the same, except for the order of the rows. Formally, given a permutation matrix $P_r \in \mathbb{R}^{n \times n}$, and given a permutation $\tilde{X} = P_r X$ of the input matrix, it can be

shown that:

$$\tilde{Y} = P_r Y$$

with $Y = \text{attn(X)}$ and $\tilde{Y} = \text{attn}(\tilde{X})$.

Generally speaking, however, the position of the elements in a sequence is extremely relevant. To address this issue, the usage of position-dependent vectors has been proposed. Specifically, given a matrix $P \in \mathbb{R}^{n \times d}$, instead of computing the attention function directly on $X \in \mathbb{R}^{n \times d}$, it is done on $P + X$. Of course, P has to be specifically designed to inject position-wise information in the input X, and this can be done in several ways.

In the original Transformer architecture [86] it was implemented via periodic functions:

$$P[i, 2j] = \sin\left(\frac{i}{10000^{2j/d}}\right)$$
$$P[i, 2j+1] = \cos\left(\frac{i}{10000^{2j/d}}\right)$$

This specific pair of functions had been employed because it returns values whose difference only depends on the distance between the elements. This would allow the model to possibly generalize over unseen positions during training.

Another proposed approach, which has been largely used [22], is to employ fully learnable positional encoding vectors; these simply become additional parameters to learn during the training procedure.

However, in more recent years, approaches based on fixed embeddings have spread. This is due to a number of reasons: for instance, fixed embeddings do not require additional parameters that need to be learned. Moreover, it's almost impossible for a model with learned embeddings to extend it to longer input sequences without an additional training stage, whereas certain fixed embeddings actually enable a model to do so. Recently, more complex and more efficient positional embedding techniques – namely, ALiBi [66] and RoPE [81] – have been developed and have become state-of-the-art for Transformer-based architectures; RoPE, in particular, is discussed in Section 2.2.1.

Transformer Architecture. The original Transformer architecture was proposed as an Encoder-Decoder model (see Figure 2.2). The Encoder was delegated with enhancing the original input

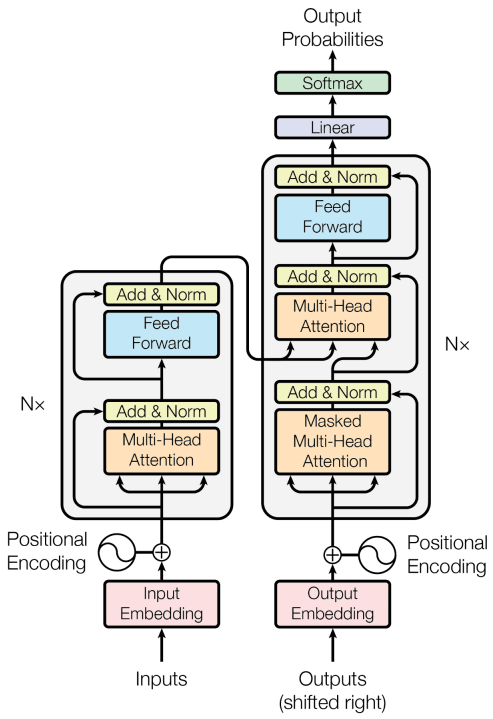

Figure 2.2: The original Transformer architecture. From [86].

sequence, while the Decoder had to autoregressively generate new tokens. Formally:

$$\mathbf{z} = \text{encoder}(\mathbf{x}),$$

$$y_{t+1} = \text{decoder}(\mathbf{x}, [y_0, y_1, \ldots, y_t])$$

where $\mathbf{x}$ is the provided input sequence and y_i is the i-th generated token by the Transformer.

The Encoder is made of a stack of $N_E = 6$ equal blocks. Each block is a sequence of four layers: a self-attention layer, an “Add & Norm" block, a feed-forward network, and another “Add & Norm" block.

The feed-forward network is a concatenation of two linear layers with a ReLU activation function in between:

$$\text{FFN}(X) = \max(0, XW_1 + b_1) \cdot W_2 + b_2$$

with $W_1 \in \mathbb{R}^{d \times 4d}$, $b_1 \in \mathbb{R}^{4d}$, $W_2 \in \mathbb{R}^{4d \times d}$, $b_2 \in \mathbb{R}^d$.

Instead, the “Add & Norm" block is a residual skip connection followed by a layer normalization. Denoting by $\mathcal{F}(\cdot)$ the output of a sub-layer (e.g., the self-attention function):

$$\text{Add \& Norm}(X) = \text{LayerNorm}(X + \mathcal{F}(X))$$

Where LayerNorm$(\cdot)$ applies a normalisation to the features of the input matrix, managing each token (i.e., row of the matrix) independently:

$$\text{LayerNorm}(Y) = \gamma \cdot \frac{Y - \mu_Y}{\sqrt{\sigma_Y^2}} + \beta$$

where μ_Y, σ_Y^2 are respectively the arithmetic mean and variance of the features, computed per token, and γ, $\beta \in \mathbb{R}^d$. (For pseudocode, see Algorithm 1).

Algorithm 1 Transformer Encoder Block

Require: Input sequence representations $\mathbf{X}$

1: $\mathbf{A} \leftarrow \text{SelfAttention}(\mathbf{X})$

2: $\mathbf{X}' \leftarrow \text{AddNorm}(\mathbf{X}, \mathbf{A})$

3: $\mathbf{F} \leftarrow \text{FeedForward}(\mathbf{X}')$

4: $\mathbf{Z} \leftarrow \text{AddNorm}(\mathbf{X}', \mathbf{F})$

5: **return Z**

The Decoder, similarly, is made of a stack of $N_D = 6$ identical blocks, each composed of three sub-layers, each followed by an *Add & Norm* block: a *masked* self-attention layer, a cross-attention layer, and a feed-forward network.

The masked self-attention layer is a standard self-attention operation in which a *causal mask M* is applied to the attention weight matrix before the softmax:

$$A = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}} + M\right)$$

with $M \in \mathbb{R}^{n \times n}$.

The mask sets to $-\infty$ all positions $j > i$ for each query i, ensuring that the i-th output element can not attend to future tokens (in fact, then, the softmax function would yield a 0 for positions having $-\infty$). This is necessary for autoregressive generation: at inference time, token y_{t+1} must be produced using only preceding tokens, and the masking enforces this constraint during training. The cross-attention layer then allows each decoder position to attend over the full encoder output $\mathbf{z}$: queries are projected from the decoder's current state, while keys and values are projected from $\mathbf{z}$. Finally, the feed-forward sub-layer is identical in structure to that of the encoder. (For pseudocode, see Algorithm 2).

Algorithm 2 Transformer Decoder Block

Require: Decoder input $\mathbf{X}$, encoder output $\mathbf{Z}$

$\mathbf{A}_1 \leftarrow \text{MaskedSelfAttention}(\mathbf{X})$

$\mathbf{X}^{(1)} \leftarrow \text{AddNorm}(\mathbf{X}, \mathbf{A}_1)$

$\mathbf{A}_2 \leftarrow \text{CrossAttention}(\mathbf{X}^{(1)}, \mathbf{Z})$

$\mathbf{X}^{(2)} \leftarrow \text{AddNorm}(\mathbf{X}^{(1)}, \mathbf{A}_2)$

$\mathbf{F} \leftarrow \text{FeedForward}(\mathbf{X}^{(2)})$

$\mathbf{Z}_{\text{out}} \leftarrow \text{AddNorm}(\mathbf{X}^{(2)}, \mathbf{F})$

return $\mathbf{Z}_{\text{out}}$

After the data flows through the Encoder module and the Decoder module, the last hidden state is used to predict the next token of the sequence. More specifically, this embedding is passed through a linear layer that maps the hidden representation to a vector of logits of the size of the vocabulary V, and then a softmax function is applied over the features to generate a probability distribution over the entire vocabulary. Formally:

$$y_t \sim \text{softmax}(W_V \mathbf{Z}^t_{\text{out}} + b_v) \equiv P(y_t \mid \mathbf{Z}^t_{\text{out}})$$

with $W_V \in \mathbb{R}^{d \times |V|}$, $b_v \in \mathbb{R}^{|V|}$. The token may be either sampled from the estimated probability distribution or, more simply, the most probable token may be chosen.

In order to skew the predicted probability distribution over the vocabulary, a certain parameter $\mathcal{T}$ – called *temperature* – may be leveraged. The higher the parameter is set, the more uniform the resulting distribution will be; conversely, the lower the temperature, the more 'sharp' the distribution becomes. Formally, by defining $\mathbf{z}_i = W_V \mathbf{Z}^t_{\text{out}} + b_v$ the logit (i.e., the raw score) given to the i-th token, its probability gets modified as:

$$P(y_t = i \mid \mathbf{Z}^t_{\text{out}}) = \frac{\exp\left(\mathbf{z}_i / \mathcal{T}\right)}{\sum_{j=1}^{|V|} \exp\left(\mathbf{z}_j / \mathcal{T}\right)}$$

In the limiting case $\mathcal{T} \to 0^+$, the resulting probability distribution will be a one-hot encoding of the most probable token, while $\mathcal{T} \to +\infty$ will instead make the distribution tend towards a uniform distribution.

Vision Transformers. As seen above, the original Transformer architecture was proposed for managing pure textual data and to estimate a causal conditional probability over all the possible tokens. It was shown, however, that with targeted architectural changes, the same architecture could achieve state-of-the-art performance also on visual data.

The first works trying to apply self-attention to visual data simply injected self-attention layers inside Convolutional Neural Networks. After this injection procedure, the model could have still employed convolutional layers – like in [88], where Convolutional blocks are followed by 'Non-Local Blocks', similar in nature to the self-attention function.

Other works tried to completely replace Convolutional Layers with Self-Attention layers, like in [72], where they substituted every Convolutional block in a ResNet [29] with a Self-Attention layer, followed by an Average Pooling in order to progressively reduce the dimension of the input tensor.

Nonetheless, none of this line of works could actually reach state-of-the-art like the original ResNet models. The Vision Transformer [22], instead, shifted the perspective entirely: it adapted the original Transformer architecture to be able to work with images, instead of trying to adapt CNNs to the attention function.

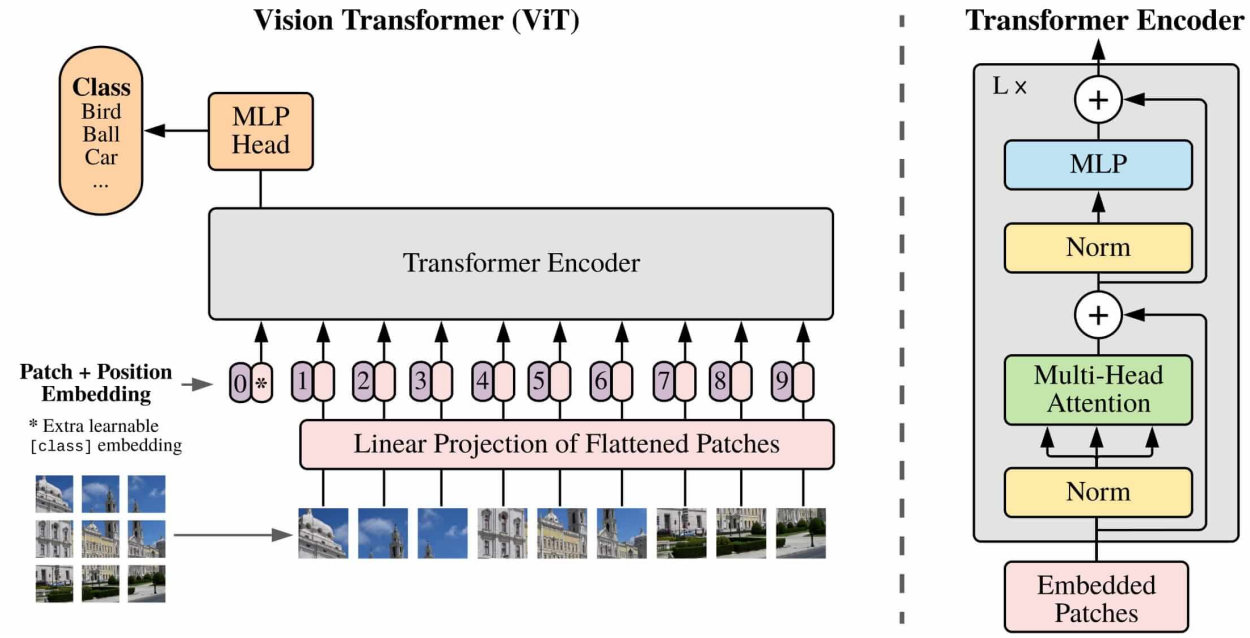

Figure 2.3: The Vision Transformer (ViT) architecture. From [22].

The original ViT architecture (see Figure 2.3) is, at its core, a single Transformer Encoder – since it's designed as a discriminative model, not as a generative one. The input image is 'tokenized'

in the following way: firstly, it gets broken down in a collection of non-overlapping patches; each one of these is flattened and passed through a linear projection in order to be mapped to the model's inner dimensionality d; then, learned 1-D positional encodings are added to each flattened vector before passing them as input to the Encoder. Formally:

$$\mathbf{X} = X_p E + P_e$$

where $X_p \in \mathbb{R}^{N \times (P^2 \times C)}$ with $N = HW/P^2$ which is the overall patch number, $(H,\ W)$ the size of the original image, (P, P) the size of each patch, C the number of channels of the image, $E \in \mathbb{R}^{(P^2 \times C) \times d}$ the linear projection and $P_e \in \mathbb{R}^{N \times d}$ the positional encoding matrix.

A small difference in the data preprocessing is the addition of the `[CLS]` token, inspired by the BERT model [21] (Section 2.3.1). It is a token with a learnable embedding prepended to the input sequence, whose hidden state is used at the end to predict on down-stream tasks. The usage of this added token is mainly due to the fact that it attends to every other embedding while carrying no input content of its own, making it a perfect candidate for content summary.

Following the pre-training and fine-tuning paradigm introduced in Section 2.1.1, the ViT is first pre-trained on massive datasets in supervised tasks, and later fine-tuned with a from-scratch prediction head on smaller datasets. The pre-training stage with very large corpora of data is almost a requirement: ViTs need a higher quantity of training samples than CNNs because they lack the *inductive bias*, which can be overcome by having more examples to learn upon. The inductive bias is an inherent assumption made by Deep Learning models over the provided data; for instance, by how the convolution operation works, CNNs are translational equivariant (i.e., a translation in the input won't affect the output if not for the same translation).

2.2 Large Language Models

A Large Language Model (LLM) is a Transformer-based architecture, trained on massive text corpora to perform the next-token prediction task. The 'large' adjective – differentiating them from other language models – refers both to the size of the massive datasets employed to train these architectures, but also to the exceptional number of parameters these models tend to reach.

2.2.1 Foundations

LLMs nowadays are generally, but not necessarily, *decoder-only* architectures; meaning that, instead of the original Transformer architecture, the encoder component is fully discarded and only the decoder module is kept. Among decoder-only LLMs, the most famous are certainly the GPT family of models [68, 10, 1] and the LLaMA models [85, 84, 26]. On the other hand, a known LLM employing an encoder-decoder structure is, for instance, T5 [70], which, by having an encoder-decoder architecture, allocates roughly twice the parameters of a decoder-only model at a comparable computational cost, since each component only processes part of the sequence.

Architectural Changes. During the years, small changes to the original Transformer architecture [86] have been proposed, and have become the de-facto standard for LLMs.

Instead of applying the LayerNorm$(\cdot)$ function after each block, the current approach is to apply it before layers; this *pre-normalization* approach mainly helps to stabilize the training procedure; in fact, the residual connection, not being normalized after a layer, allows the gradient to flow more easily through it:

$$\text{(Post-Normalization)} \qquad \mathbf{Y} \leftarrow \text{LayerNorm}(\mathbf{X} + \mathcal{F}(\mathbf{X}))$$

$$\text{(Pre-Normalization)} \qquad \mathbf{Y} \leftarrow \mathbf{X} + \mathcal{F}\left(\text{LayerNorm}(\mathbf{X})\right)$$

It is now employed by many LLMs, most notably by the GPT family (since GPT-2 [68] onwards) and by the LLaMA models [85].

Moreover, the LayerNorm$(\cdot)$ has been largely substituted by the RMSNorm$(\cdot)$ (Root Mean Square) function. While the former is robust against shift noises in weights (re-centering invariance) and against scale noises (re-scaling invariance), the latter entirely focuses on re-scaling invariance – which seems to be what matters most for Transformers [97]:

$$\text{RMSNorm}(\mathbf{X})_i = \frac{\mathbf{X}_i}{\sqrt{\frac{1}{d}\sum_{j=1}^{d}\mathbf{X}_j^2}}\gamma_i$$

This formulation is equivalent to the LayerNorm function when the sum of the features is 0. It is particularly employed by the LLaMA models.

The absolute positional encoding – either fixed or learned – has also been mainly rejected in favor of more advanced techniques. Among these, *RoPE* [81] has been mainly used by LLMs such

as LLaMA and GPT-Neo [9]. The core idea is to have positional encoding vectors that only depend upon the relative position between two tokens; i.e., such that the dot product between two elements in the sequence is a function of the two elements themselves and of the distance between them:

$$x_i \cdot x_j \equiv g(x_i, x_j, i - j)$$

The chosen function satisfies the *long-term decay* property: the higher the distance between the two elements, the smaller the dot product between them; implying that, during the attention function, a token will attend more easily to adjacent tokens.

Another change that has been made affects the activation function inside the FFN layer: instead of using the ReLU, modern approaches tend to prefer the SwiGLU [79] activation function:

$$\text{SwiGLU}(x) = \text{Swish}\,(xW + a) \otimes (xV + b)$$

with $\text{Swish}(z) = z/(1 + e^{-z})$, where W, V, a, b are all learned parameters and where $\otimes$ is element-wise multiplication. The shift towards the SwiGLU function has been done because it has been shown to consistently improve Transformers' capabilities with respect to the ReLU activation. The LLaMA models and the PaLM model [16] are two LLMs that have notably adopted it.

Another significant improvement in terms of inference time has been the usage of *Multi-Query Attention* (MQA). Instead of projecting each element of the sequence as query and as a key *for each head* in multi-head attention, they are projected once and shared across heads. Meaning, the matrix parameters W^K and W^V are the same for each head. Instead, each head still retains its own W^Q matrix. Surprisingly, this reduction in the number of distinct key and value projections does not negatively affect performance [78]; however, it significantly reduces the overall number of computation steps required at inference time. A middle-ground between MQA and MHA is the so-called *Grouped-Query Attention* (GQA) [2]. In this approach, each head is divided into a group; inside every group, keys and values are shared (i.e., W^K and W^V are the same), while each head still retains its own independent query projection. This approach has been adopted by models such as LLaMA2 [84] and Mistral-7B [36].

Datasets. Since the main training phase is entirely self-supervised – being next-token prediction – the data can be unlabelled. However, it's important to note that the more data is used for training,

the better the final results will be for the model. Not only does quantity matter: it has been shown that feeding an LLM poor quality data negatively affects the end result [68].

The main technique to retrieve the data is the usage of *crawlers*, software specifically designed to scrape the web. Some heuristics are employed to ensure that the quality of the downloaded data is high. Alongside scraped web pages, usually, LLMs are also trained on copyright-free books, ranging a variety of literary genres, such as novels, essays and philosophy; in particular for specialized models, the dataset could be expanded with data from professional domains, such as open-source code and scientific data [54].

Training Paradigms. Usually, LLMs are trained following a common pipeline: first, a pre-training stage on massive text corpora is carried out; secondly, a fine-tuning step is performed on specific target datasets to shift the model's distribution towards the desired one; finally, a Reinforcement Learning with Human Feedback (RLHF) stage is employed to further enhance the model's capabilities into providing human likeable answers.

Building on the pre-training and fine-tuning paradigm of Section 2.1.1, the pre-training stage is employed mainly to give the model broad knowledge on grammar, vocabulary and text structure. It has been shown that pre-training a language model before adapting it to the desired task yields performance gains [67].

The fine-tuning stage usually requires a more curated selection of input data. A known approach is *supervised fine-tuning* (SFT) [64], which requires a labelled dataset of input-output pairs and adapts the model to more specific tasks, such as closely following human instructions. Building on this, *alignment tuning* aims to steer the model away from harmful or false outputs; this is typically achieved by continuing SFT on carefully selected data that reflects the desired behavior, rather than through a separate training procedure.

The last stage in the training pipeline is usually RLHF [64]. Through an application of reinforcement learning, the model is queried certain questions and is penalized for 'bad' answers, while it is rewarded for answers reflecting the desired behavior. An external Reward Model (RM) is tasked with assigning a score to the answers provided by the LM, in order to 'instruct' it on how to answer; for instance, it could be penalized for using toxic words or citing harmful content. The RM itself is trained on human preference data, which consists of pairs of (answers, score) manually annotated

by human evaluators.

Emergent Capabilities. For LLMs, an ability is said to be *emergent* if it is present only in models with a number of parameters above a certain threshold; these are capabilities that show up the moment the models are scaled up [89]. Among these, for instance, are the ability to answer questions in the Persian language, understanding a word from its scrambled letters and performing addition of numbers with 8 digits. Interestingly, these abilities are not explicitly present in the training objective, but nonetheless unexpectedly arise as a consequence of scale. Models show almost random-like performance on emergent capabilities until they reach a certain critical scale threshold. Beyond it, performance on that ability improves sharply. This qualitative abrupt shift is called *phase transition*.

However, this assumption has been heavily challenged [75]. By using more fine-grained metrics, in fact, it is possible to see how emergent capabilities do not show up abruptly, but gradually develop in smaller models. The performance itself seems to be roughly scaling linearly with the number of the parameters of the model. Moreover, in the same work, it has been shown that the apparent manifestation of the abilities is partly due to the chosen metric: different evaluation criteria may change, or even reverse, the interpretation of a given capability.

Scaling Laws. Apart from a few aforementioned additions, the architecture of LLMs is at its core a Transformer, albeit decoder-only. Thanks to this substantially stable structure, the main changes one could exploit in order to increase the performance of a model would be in the size of the data, in the overall size of the model and in the total training time. So, the performance (i.e., the loss value) of Transformer-like architectures has been studied [39] as if it were a function of the three mentioned variables: data (D), number of parameters (N) and total compute time (C), where D is expressed as total number of tokens inside the dataset, N as number of non-embedding parameters and C as PF-days.

Among the many results that have emerged, the most important ones are that the performance of the model has a power-law dependency with each of the three factors; i.e.:

$$\mathcal{L}_{\text{CE}} \propto (X_0/X)^{-\alpha}$$

where each variable has its own set of fixed parameters. Interestingly, these results are practically

architecture-independent (i.e., similar Transformer architectures still follow the same power laws with small errors). The usefulness of having universal scaling laws is that the performance of a model can be *predicted* before training, which is particularly critical in the case of large-scale models: these require days, weeks or even months to be fully trained. The laws are also used to understand *how* to allocate the overall budget for a model training between model size and data [31].

Another result that follows from the scaling laws is that larger, non-converged models have actually better performance than smaller converged models; implying that a converged model is not always the optimal choice in terms of the value of the loss function.

2.2.2 LLM Families

A *family* of LLMs is a collection of models developed by the same organization, sharing a very similar architecture and a common name, but that differ on the number of parameters, on training methodologies or on specific optimizations.

GPT. The Generative Pre-trained Transformer (GPT) family, introduced by OpenAI, was the first to propose the decoder-only architecture for language modeling. Starting with GPT-1 [67], they showed the power of generative pre-training on next-token prediction followed by discriminative fine-tuning for specific tasks. GPT-2 [68], instead, entirely removed the down-stream fine-tuning: the authors believed that a strong enough pre-training stage would be enough for natural language tasks. GPT-3 [10] scaled up to 175 billion parameters, and marked a paradigm shift toward in-context learning: by passing a few examples of the task to perform inside the prompt, the model showed the capability to perform it as if it was fine-tuned over it. Subsequent iterations, such as GPT-4 [1], integrated multimodal inputs and reinforced human alignment through Reinforcement Learning from Human Feedback (RLHF), setting the foundational standard for modern commercial LLMs.

LLaMA. Developed by Meta AI, the Large Language Model Meta AI (LLaMA) family focused on publishing open-source LLMs, advancing the research by optimizing performance under strict compute budgets. Differently from the results of GPT-3, LLaMA-1 [85] showed that training smaller models on significantly more tokens than typically expected by scaling laws could yield increased

downstream performances during inference. The family introduced architectural refinements, including the SwiGLU activation functions, Rotary Position Embeddings (RoPE), and Root Mean Square (RMS) normalization, all described in Section 2.2.1. Following iterations, like LLaMA 2 [84] and LLaMA 3 [26], scaled up context windows, increased the vocabulary size, added multilingual support, and provided high-performing base and instruction-tuned variants that are currently vastly employed by researchers.

Vicuna. Developed by the Large Model Systems Organization (LMSYS), the Vicuna family [15] represents a fundamental milestone in efficient instruction-tuning and chatbot alignment. Built upon LLaMA, Vicuna-13B was trained by fine-tuning the base architecture on multi-turn conversations that had been shared by users. The significance of the Vicuna family lies in its instruction tuning stage achieved at a fraction of the compute cost required for proprietary models. It leveraged managed optimizations like gradient checkpointing to achieve over 90% of OpenAI's ChatGPT quality at a training cost under $300. The family also pioneered early LLM-as-a-judge evaluation frameworks by utilizing GPT-4 to systematically benchmark the chatbot performance.

Qwen. Introduced by Alibaba Cloud, the Qwen family [5] highlights the scaling of Mixture-of-Experts (MoE) architectures specifically for robust multilingual capability. The Qwen lineage is structurally characterized by its adoption of advanced RoPE configurations to support ultra-long context windows, SwiGLU activation variations, and a specialized visual-language offshoot (Qwen-VL) [6] that handles native multimodal perception.

2.3 Representation Learning

The success of pre-training on web-scale data in NLP prompted a broader question: could the same principle yield general-purpose representations in other domains? Having strong and robust representations enables down-stream applications to build on them more easily.

This section surveys the main approaches to learn representations from raw data that emerged.

2.3.1 Textual Representations.

Learning useful representations of language requires solving a fundamental challenge: text is discrete and sequential, and the meaning of a word is deeply entangled with its context. Early approaches relied on distributional semantics, producing static embeddings such as Word2Vec [60] and GloVe [65], which assigned a single fixed vector to each word regardless of its context. While effective as a first approximation, static embeddings collect different meanings of the same word into the same representation and cannot capture how meaning shifts with surrounding context.

Thanks to the rise of the Transformer architecture it became possible to extract *contextual* embeddings of words: each representation became a function of the whole sequence, not of its respective word alone. This shift made it possible to apply self-supervised tasks to raw, large-scale textual data. The most natural of these is, of course, next-token prediction. While this objective is very expressive, it requires the fundamental constraint that the model is generative, since it needs to predict a subsequent token in a sequence.

Thus, for discriminative models, a new set of tasks was developed that asked the model to reconstruct a corrupted version of the input. The most influential instantiation of this idea was BERT.

BERT. The Bidirectional Encoder Representations for Transformer (BERT) [21] is an encoder-only model based on the Transformer architecture. Its bidirectional nature (i.e., attention is performed without any kind of masking) is fundamental, since contextual embeddings require the entire sequence to be computed correctly. This implies that any token in the sequence attends to any other, without forcing causality relationships.

The model is first pre-trained on a pair of pre-text tasks, and then it can be fine-tuned to generalize on down-stream tasks of other kinds. The first pre-training task is Masked Language Modeling (MLM). Given a sequence of tokens $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, a random subset of indices $\mathcal{M} \subset \{1, \ldots, n\}$ is selected, and the corresponding tokens are replaced with a special mask token $[\texttt{MASK}]$, yielding the corrupted sequence $\mathbf{x}'$:

$$\mathbf{x}'_i = \begin{cases} [\texttt{MASK}], & \text{if } i \in \mathcal{M}, \\ \mathbf{x}_i, & \text{otherwise.} \end{cases}$$

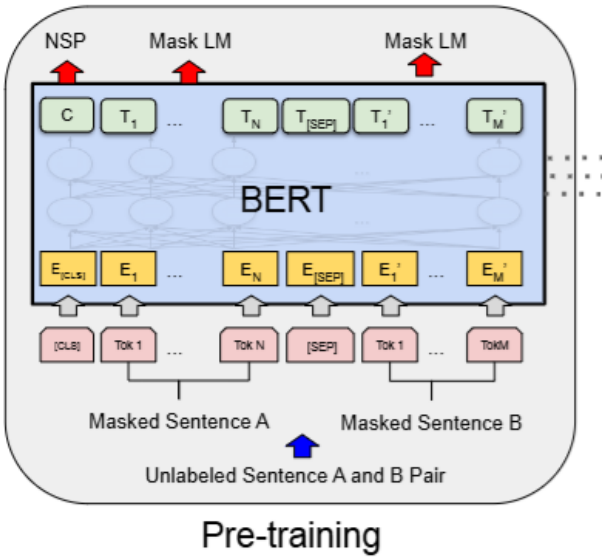

Figure 2.4: The BERT pre-training stage. From [21].

The encoder maps the corrupted sequence to a sequence of contextual representations, $\text{Enc}(\mathbf{x}') \in \mathbb{R}^{n \times d}$, and a linear projection followed by a softmax produces a distribution over the vocabulary V at each masked position. The MLM objective minimizes the negative log-likelihood of the original tokens at masked positions, pushing the model to learn a representation of the `[MASK]` tokens which were as close as possible to the original representations.

The second pre-training task is Next Sentence Prediction. Assuming that the large corpus of text is divided into sentences, each training sample is a pair (A, B), where B is with equal probability either the sentence that genuinely follows A in the corpus or a randomly sampled sentence. The two sentences are concatenated into a single input sequence with a special separator token `[SEP]`, and with a prepended classification token `[CLS]`. Moreover, each element of the two sentences is added to a special 'sentence vector' – either E_A or E_B – to further help the model distinguish between each sequence. At the output of the encoder, it is expected that the `[CLS]` token contains aggregate information about both sequences and is used as an overall sequence representation to classify whether B is a sentence that can follow A or not.

The two pre-training tasks are then applied together (see Figure 2.4). This self-supervised approach lets the model create robust contextual representations for tokens, which can be later employed to easily fine-tune the model to down-stream tasks. They are especially useful because

each representation is now context-dependent, and is enhanced by the surrounding representations.

2.3.2 Visual Representations.

Learning representations for visual data is inherently more challenging compared to language. Images are continuous and high-dimensional, and are not naturally tokenizable like text. Early deep learning approaches relied on training neural networks to map images to category indices, and to later employ the inner activation functions as features. Of course, this approach was heavily dataset-dependent, required a labelling cost, and the output representations were naturally bounded by the possible classification indices.

Many self-supervised approaches require, instead, to exploit the signal itself, without any additional human annotation. An example of this would be the so-called *instance classification*: the idea is to set each image of the raw dataset to be a distinct class [91], and to extract feature vectors that could discriminate between every image. This approach, however, does not scale well with the size of the dataset.

Two more advanced techniques, still employed in state-of-the-art models, emerged. The first one is the *instance discrimination* task: given two augmented views of the same image, a model is required to output similar representations for both, while pushing apart representations of different images. An example of this is BYOL [27], which tries to match the output representations of a student model – which will be the actual model later used for down-stream tasks – with the output of a teacher model, both receiving different views of the same image. Another instance of this approach is DINO [12], described below.

The second one is the *reconstruction* task: the model is given a partial or corrupted view of an image and trained to predict the missing content, either in pixel space or in a latent feature space. A clear example of this is I-JEPA [4], which will be discussed later.

DINO. The methodology proposed by DINO is a mixture between self-supervised learning and knowledge distillation. The latter approach is a learning paradigm in which a *student network* g_ϕ is trained to match the output of a *teacher network* g_ψ, where the latter is a fixed model. So, generally, the update rule is only performed on the student network's parameters ϕ.

DINO adapts this into a self-supervised approach by generating, from an image, a set V of

different views of it, two of which are *global* views. All distortions are passed through the student network, but the teacher network only receives the global views. In this way, the student should learn to output similar features as the teacher by only seeing local views of the image. In this setting, moreover, the teacher network is not given *a priori*, but is built on previous iterations of the student network. Both networks share the same ViT-based architecture.

Curiously, it has been shown that – by probing the activation functions – the attention maps computed by DINO tend to focus more on foreground elements of the image rather than the background, meaning that it managed to generalize well on the visual data (see Figure 2.5). Its capabilities of focusing on the foreground also outperform supervised models trained on the same datasets.

Figure 2.5: Attention maps produced by DINO. From [12].

I-JEPA. Joint Embedding Predictive Architectures (JEPA) [43] are non-generative models trying to predict the representation of an input y from another input x. This general paradigm has been applied to images via I-JEPA.

The idea behind I-JEPA is to predict, from a single crop of an image, other crops of the same image. The full image y is passed through a target encoder, from which the desired block representations are determined; then, a single crop is passed to a context encoder, which outputs a local representation of the image; then conditioning on positional encodings, a predictor tries to

predict the unseen block representations from the local crop representation. The model is non-generative, since every representation is computed in the latent space, and not converted back to the output space.

While not relying on data-augmentation, the I-JEPA features are able to outperform other approaches to down-stream tasks, such as image classification, object counting and depth prediction.

2.3.3 Multimodality

Multimodal Learning refers to Deep Learning or Machine Learning algorithms able to process and jointly reason over inputs expressed through different modalities (e.g., vision, natural language, sound...). Multimodality is a common problem in a wide range of tasks. Some problems are inherently multimodal – for instance, image captioning, visual question answering and speech transcription all require grounding across different modalities by definition. Other tasks benefit by enhancing the input with different multimodal sources – for instance, a medical diagnosis is richer when it contains both written notes and scan images.

The central challenge is finding a common representation across modalities. The dominant paradigm is that of a *joint representation space*. The idea is to create a d-dimensional latent space into which all the different modalities are jointly projected. A desired property of this shared space is semantic clustering: that is, the same concepts expressed via different modalities should map to nearby points. This property is so fundamental that it is typically enforced directly in multimodal models as part of the loss function [69]. Interestingly, however, it has been shown [49] that naively applying this contrastive approach does not lead to true alignment, but rather results in a *modality gap*: that is, a clear geometric distance between representations of different modalities; meaning that representations tend to cluster also by modality, rather than on semantics alone.

The proposed approaches to deal with different modalities in the same shared space have been various and distinct. One line of work – called *Early Fusion* – simply proposed the generation of an embedding vector for each modality that needs processing and a concatenation of these vectors would produce the multimodal input embedding. A clear example is the ViLT [41] architecture, concatenating the visual embedding $\bar{v}$ of an image with the textual embedding $\bar{t}$ of its description:

$$\mathbf{z} = [\bar{v} \circ \bar{t}]$$

One of the main benefits of this approach is that modalities are jointly analyzed as soon as possible, enabling the model to capture dependencies in the raw features. However, since embeddings are in their original representation space from the start, the dimensionality can grow large quite fast; moreover, if one modality is particularly informative, it may lead the model to completely discard the other modalities altogether.

Another line of work – called *Late Fusion* – first pre-processes each modality independently via intra-modality functions, and only then merges or concatenates the two together inside a shared embedding representation. An instance of this approach is VSE++ [23], in which visual and textual modalities are mapped into a joint embedding space through learned linear projections:

$$f(I; W_f) = W_f^T \phi(I), \quad g(t; W_g) = W_g^T \psi(c)$$

where $\phi(\cdot)$ and $\psi(\cdot)$ are feature extractors for images and text captions, respectively. Then, operations can be made in the joint embedding space; e.g., a similarity computation via the dot product:

$$s(I;\, c) = f(I; W_f) \cdot g(c;\, W_g)$$

A clear advantage of Late Fusion is its modularity; it's possible to later remove, edit or add modalities to the pipeline procedure – as long as there is an effective way to pre-process it accordingly. However, unlike Early Fusion, it's more difficult to capture inter-modality information, since modalities are pre-processed individually.

Another fusion strategy can be performed by employing Attention layers; usually, set of queries come from a modality, while keys and values come from a different modality. It's the case of Flamingo [3], which adds gated attention blocks between layers of a pre-trained LLM. The modalities are then processed, interleaved and jointly understood by the model throughout the entire inference pipeline: the gated attention block receives the queries from the auto-regressively text generated sequence, and keys and values from visual embeddings. While this approach allows for rich inter-modality interactions throughout the model, it comes at the cost of architectural complexity and tight coupling between modalities.

Among these approaches, contrastive dual-encoder models – and CLIP in particular – showed that late fusion with a contrastive objective, applied to web-scale data, yields representations with remarkable zero-shot generalization. This is discussed in the following.

CLIP. Early work in multimodality was encouraging. For instance, [37] trained a CNN over one hundred million Flickr images and comments, showing that the learned visual representations adapted well to down-stream tasks. Other concurrent works [20], instead, showed the capabilities of applying transformer-based architectures onto this contrastive objective. Nonetheless, all of these works didn't actually exploit the wider possibilities offered by natural language, which, of course, doesn't limit itself to a fixed amount of classes, but can express much more.

The CLIP architecture [69] proposed a new way to employ late-fusion paradigm on the image-text problem, in order to extract robust semantic embeddings through a contrastive objective at large scale. Being late fusion, each modality had its own independent encoder:

$$e_I^i = \text{Enc}_I(I_i); \quad e_T^i = \text{Enc}_T(T_i)$$

whereas the text encoder $\text{Enc}_T(\cdot) : \mathbb{R}^{d_T} \to \mathbb{R}^d$ is a Transformer [86], while the image encoder $\text{Enc}_I(\cdot) : \mathbb{R}^{d_I} \to \mathbb{R}^d$ is a ViT [22]. These are trained on a massive scale dataset, called WIT-400M, of four hundred million image-text pairs scraped from the web. Scale is an essential ingredient: without it, the contrastive objective alone would not suffice to ground the two modalities together.

Differently from other works [20], the pre-training strategy simply tries to let the model understand which text representation refers to a given image, and vice-versa, rather than generating the textual caption word by word. Formally, given a batch of N image-text pairs, CLIP is trained to maximize the N similarities between image and text representations for the same pair while minimizing the $N^2 - N$ similarity values for images and text representations not coming from the same pair. To do so, it first computes a matrix of scaled cosine similarities:

$$S_{ij} = \tau \cdot \frac{e_I^i}{\|e_I^i\|} \cdot \frac{e_T^j}{\|e_T^j\|}$$

where τ is a learned temperature parameter controlling the sharpness of the distribution. The contrastive loss is then a symmetric cross-entropy over this similarity matrix:

$$\mathcal{L} = -\frac{1}{2N} \sum_{i=1}^{N} \left(\log \frac{\exp(S_{ii})}{\sum_{k=1}^{N} \exp(S_{ik})} + \log \frac{\exp(S_{ii})}{\sum_{k=1}^{N} \exp(S_{ki})} \right)$$

The two terms enforce the objective symmetrically: the first treats each image as a query over all texts in the batch, while the second does the converse. The softmax function is required to convert raw similarity values to a probability distribution over the batch; of course, the loss tries to push as

high as possible the similarities S_{ii} both for each text fixing the i-th image and for each image while fixing the i-th caption (see Figure 2.6).

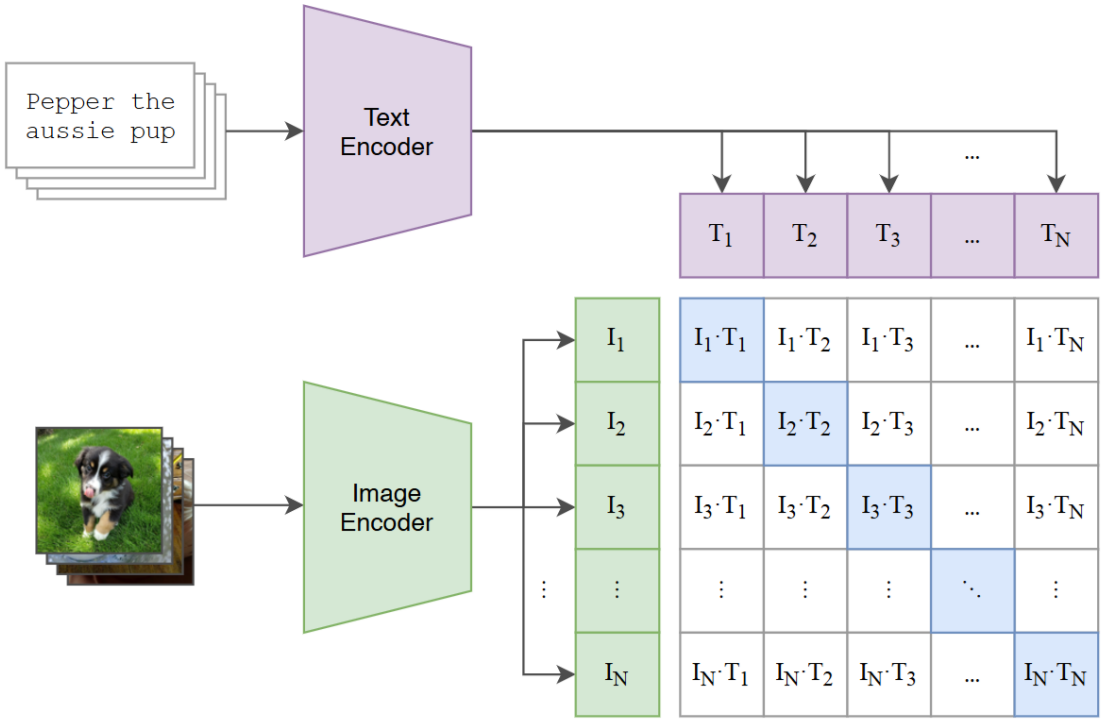

Figure 2.6: The CLIP training paradigm. From [69].

This *contrastive* pre-training allows the model to better align similar concepts together while distancing different elements. Moreover, by employing a representation in natural language of the visual data, the limitation of having a fixed number of classes a-priori completely vanishes.

Of course, CLIP can be later fine-tuned for different tasks but it has been shown to possess great zero-shot generalization capabilities as-is. One experiment that was conducted, for instance, was to use it out-of-the-box for classification: they computed the similarities between CLIP's text representations of the class names and the target image, choosing as the 'predicted' class the highest similarity value. This approach not only did show performances comparable with models specifically trained on classification, but also showed great improvement over distribution shifts of the image space – e.g., a doodle of a banana was still recognized as such by CLIP, and usually not by models trained on real-world photos. Curiously, due to the fact that the labels inside the datasets are rarely made of a single word, but rather of sentences, the best performance is reached when the textual input is not the single class word, but rather the sentence "`An image of {class}`".

However, some limitations were still present even in this approach [38]. For instance, CLIP has been shown to have very poor spatial reasoning [82] (i.e., the same object, in different locations, still has the same embedding); also, it struggles with attribute bindings [46] and with negations (for instance, the sentence "`An image with no cars`" has an embedding very similar to embeddings of car images). These shortcomings gave rise to other models aimed at addressing them from different angles: some refined the contrastive objective itself – improving training stability and scalability – while others abandoned the language supervision altogether in favor of purely visual self-supervised training, yielding richer spatial and structural representations.

SigLIP. SigLIP's [96] contribution mainly tried to overcome one bottleneck of CLIP, specifically the large batch size required to stabilize the contrastive loss. The batch size employed in the original work was of $32'768$ – which was almost necessary in order to let the model learn to push away embeddings sampled from different pairs. Training would usually require the collection of all the similarity scores in a single device, in order to normalize the values, which is a computationally heavy requirement.

SigLIP, instead of employing the samples for a multi-class classification problem, rewrites it as a simple binary classification setting. More specifically, the model is trained to predict from each possible image-caption pair (I_i, T_j) whether $i = j$ or if $i \neq j$ – i.e., if the image and the caption came from the same sample of the dataset or not. More specifically, the loss employed is a sigmoid:

$$\mathcal{L}_{ij} = \log \frac{1}{1 + \exp\left(z_{ij}\left(-t\,\mathbf{x}_i\mathbf{y}_j + b\right)\right)}$$

with $\mathbf{x}_i$ and $\mathbf{y}_j$, respectively, the encoded embedding of image I_i and the encoded embedding of caption T_j. Instead, z_{ij} is the label to predict, which is either 1 if $i = j$ or -1 if $i \neq j$.

In this scenario, each possible pair is considered independently, avoiding large batch sizes like CLIP. Of course, the dataset is vastly imbalanced: for N image-caption pairs, exactly $N^2 - N$ would require a label of -1. In order to alleviate this, the learnable temperature parameter t is initialized at 10, whereas the learnable bias term b is initially set at -10. This helps the model: whenever it finds a small positive alignment for a pair ($\mathbf{x}_i\mathbf{y}_j > 0$), the temperature term helps it overcome the bias; whereas a small negative alignment would be pushed to very negative values already, forcing the model to focus almost only on positive samples.

2.4 Multimodal Large Language Models

Multimodal Large Language Models (MLLMs) [11] are a set of models able to integrate visual and textual modalities, providing multi-turn dialogue support and instruction-following capabilities. These models employ different techniques to aggregate cross-modality features and training procedures.

The general framework consists of a visual encoder $V_E(\cdot)$, an Adapter $A_\psi(\cdot)$ and a language model $f_\phi(\cdot)$. The visual encoder embeds the image, while the adapter projects the visual embedding into another representation, more akin to the input expected by the language model. The latter generates the predicted answer by receiving as input a concatenation of the visual and the textual embeddings (see Figure 2.7).

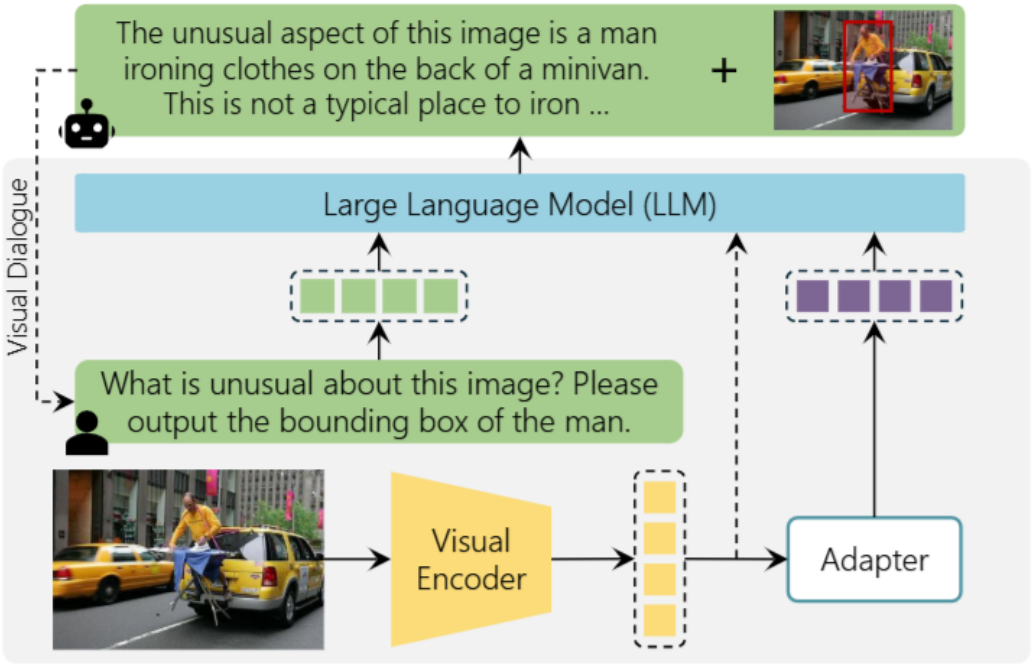

Figure 2.7: The general architecture of MLLMs. From [11].

The LLM choice usually falls into the open-source families, like the LLaMA or the Vicuna suite of models, for easier access. The vision encoder, instead, which is usually kept frozen, is generally based on ViT architectures. Finally, the adapter, whose task is to carefully align visual embeddings in order for them to be manageable by an LLM, ranges from simple MLP or linear layers to more

advanced architectures, e.g. a Transformer-based model.

Multimodal architectures generally undergo either a one-stage training or a two-stage training process. In both cases, however, the common training task is next-token prediction over autoregressive generation. MLLMs can undergo different visual understanding tasks, such as Visual Question Answering (VQA) or captioning.

2.4.1 LLaVA

One of the current most prominent, open-source MLLMs is LLaVA [52]. LLaVA, which stands for "Large Language and Vision Assistant", introduces a straightforward, trainable linear projection matrix as an adapter to map the visual tokens directly into the word embedding space of the language model (see Figure 2.8).

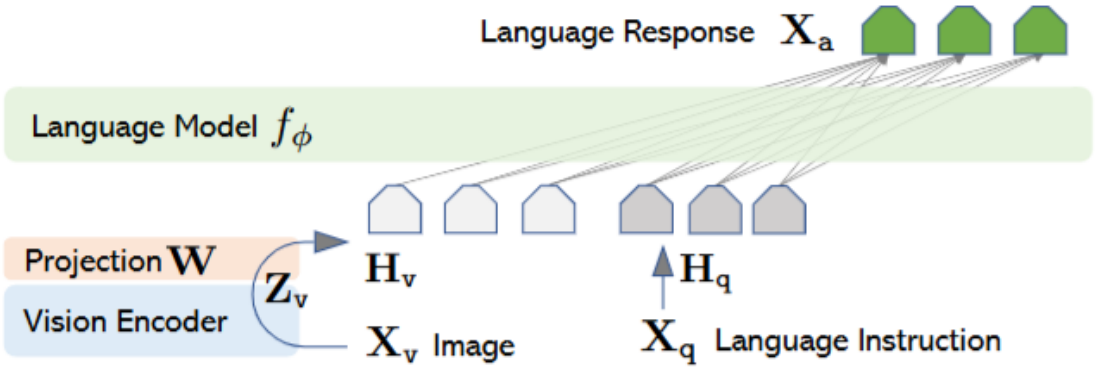

Figure 2.8: The LLaVA architecture network. From [52].

A set of upgrades to the methodology followed in the next iteration, LLaVA 1.5 [51]. More specifically, they replaced a few backbones (e.g., the adapter module) in the architecture and enhanced the training data with additional academic datasets.

Architecture. The architecture follows the general framework of a MLLM. In the first version of LLaVA, they employed Vicuna as the backbone LLM, pre-trained CLIP visual encoder ViT-L/14 (managing resolutions of up to 224×224) and a simple linear layer as an adapter. In the following iteration, they replaced the linear layer with a two-layer MLP, increased the resolution of the visual encoder (pushing the resolution to 336×336) and employed Vicuna v1.5 as LLM.

Dataset. LLaVA was intended as a model to support multi-turn dialogues, but data regarding multimodal instruction-following was quite limited. In order to create it, the LLaVA authors decided to leverage a GPT-4 model to generate a collection of data to employ.

More specifically, given an image X_v, and its associated caption X_c, they asked the model to create a set of questions $\mathbf{X}_q$ involving the visual data provided. To support the generation better, they also provide *bounding boxes* to localize objects in the scene. To create such data, they used the COCO dataset [50] as a base. The possible data samples generated are either a *conversation* between the assistant model and the user, a *reasoning* process question, requiring the assistant to perform step-by-step analysis to produce the correct answer, or a *detailed description* of the visual input.

In addition to this machine-generated data, in LLaVA 1.5 they also included additional VQA academic-task-oriented datasets.

Training. The training procedure of LLaVA is two-stage. The first stage focuses on training the adapter to output LLM word embeddings from visual data, whereas the second stage fine-tunes the architecture end-to-end on the instruction-following data, while keeping frozen the visual encoder.

The first stage – called **Feature Alignment** – keeps both the LLM and the visual encoder frozen, while only training the adapter. The employed data are 595K image-caption pairs from the CC3M dataset [77]. The visual encoder receives the input image, the adapter projects the output encoded visual embeddings and the LLM performs autoregressive generation on the projected embeddings concatenated with one natural language question asking for a description of the provided image. The ground-truth the model should generate is, of course, the caption of the image found in the dataset.

The second stage – the **Instruction Tuning** stage – trains both the LLM and the adapter. The data they employ is the aforementioned machine-assisted instruction-following data. While the *conversation* data type is a multi-turn conversation, the other two are single-turn instead. During training, data is uniformly sampled from the dataset.

Notably, each input to the model is given via role-prefix. For instance, for a user question X_q^1 and for a model's answer X_a^1, the prompt is built as: `Human:` X_q^1 `<STOP> Assistant:` X_a^1.

LLaVA-MORE. LLaVA-MORE [17] extends the standard LLaVA framework by analyzing the effects of combining different language models and visual backbones. The architecture keeps the typical structure of multimodal models and adopts a two-stage training procedure.

2.4.2 Training Procedures

LLaVA popularized the two-stage training paradigm, but other architectures also employed, with varying degrees of success, a single-stage training.

Two-Stage Training. In this training paradigm, the MLLM is firstly trained in order to map to a shared latent space both visual and textual embeddings; then, the second stage consists only of instruction-tuning to strengthen the multimodal assistance capabilities.

Other than the already discussed LLaVA, another notable MLLM employing a two-stage training pipeline is MiniGPT-4 [100]. Differently from LLaVA, it only trains the adapter – a simple linear layer – during both stages. Curiously, they also report that increasing the complexity of the adapter causes the model to underperform. Another interesting example, instead, is Instruct-BLIP [18]. They employ both a Q-Former architecture – which outputs instruction-conditioned visual embeddings – followed by a linear layer as an adapter, and only train both of them during the two stages.

Single-Stage Training. In this training paradigm, the model is concurrently aligned both on visual understanding and instruction-following data.

It is the case of Flamingo [3], in which the cross-attention layers inside the pre-trained LLM and the Perceiver Resampler module are jointly trained. Similarly, LLaMA-Adapter [98] introduces a small set of trainable adapter parameters that separately capture visual knowledge while preserving the pretrained language model, allowing image-text alignment and instruction learning to be performed jointly.

2.4.3 Performance Evaluation

Evaluating MLLMs differs greatly from evaluating LLMs, since the addition of a new input modality opens up many different tasks. The abilities of an MLLM can be divided into four major classes

[32]. The first one is **recognition**, in which the model is tested to see if it truly perceives the visual input data, by performing tasks such as *object recognition* or *object counting*; the second is **reasoning**, in which the model is evaluated to understand whether it is capable of reasoning over the perceived data, by employing frameworks like visual question-answering (VQA); the third is **trustworthiness**, where the MLLM is tested to understand how faithful the produced answers are; finally, the fourth is **domain-specific competence**, where certain knowledge-intensive benchmarks are employed to evaluate the model's capabilities of answering in certain scenarios.

The evaluation metrics, instead, are mainly performed via exact matches (i.e., an answer is deemed correct only if it coincides exactly like the ground-truth value) or through the LLM-as-a-judge technique, in which an external LLM gives its own opinion over the produced answer. While the first approach may miss partially correct answers, the second one is certainly less reproducible and more expensive.

2.5 Visual Hallucinations in MLLMs

A hallucination is a generative phenomenon of MLLMs occurring whenever part – if not all – of the text produced by the MLLM has no corroboration in the visual data. This usually happens when the answer fails to mention an object that is present in the image, or when it confidently references an object that is absent from it.

This type of object hallucination can be actually classified as one of three different types: *object category*, *object attribute* and *object relation* [7]. In the category scenario, the MLLM identifies nonexistent object categories; in the attribute case, the model wrongly describes an existing object's attributes (e.g., color, shape, counting...); finally, the object relation type happens whenever the relationship between correctly described objects does not align with the image content.

2.5.1 Causes of Hallucinations

Correctly understanding where and why hallucinations happen is difficult to do; however, certain origins have been shown to be root causes of hallucinations in MLLMs. By mitigating these, these models showcase a reduction in the hallucinative behavior.

Data. Bridging the gap between the two modalities is a complex task, requiring large datasets to build robust MLLMs. Insufficient or low-quality data, therefore, may lead to problematic cross-modal alignment results. Many datasets for the training stages are either scraped from the web or synthetically generated; in both cases, there could be inaccurate or noisy data samples inside, further polluting the final model into producing hallucinations.

Model. As seen in Section 2.4, MLLMs are built of many different components, some of which are usually pre-trained. Since they are not entirely trained end-to-end from scratch, some errors may propagate during the pipeline. Having either a weak vision model or a weak adapter may produce inaccurate or noisy visual embeddings. The LLM, moreover, is usually much stronger than the vision model – meaning that the parametric knowledge (i.e., the information the model possesses inside its own parameters) may as well override visual content altogether.

Training. The training objective itself – which is, usually, a standard cross-entropy loss on the predicted tokens – may not be suitable for learning visual content, mainly due to the visual data's complex spatial structure.

2.5.2 Evaluating Hallucinations

Evaluating hallucinations in MLLMs is specifically done on object hallucinations in the generated answer.

CHAIR. Being one of the first hallucination evaluation works, CHAIR [74] focuses almost entirely on the image captioning task. The computation of the CHAIR metric is straightforward, and can be done in two variants:

$$\text{CHAIR}_i = \frac{|\{\text{hallucinated objects}\}|}{|\{\text{all objects mentioned}\}|}$$
$$\text{CHAIR}_s = \frac{|\{\text{sentences with hallucinated objects}\}|}{|\{\text{all sentences}\}|}$$

The metric is a simple computation of how many hallucinated objects (resp., sentences with at least one hallucination) were present with respect to the total number of mentioned objects (resp.,

sentences). When it was first presented, they restricted the computation to the 80 COCO [50] object classes and also employed synonyms mapping to correctly evaluate an object hallucination.

A few other evaluation metrics were also proposed during time. The *Cover* measures the object coverage of responses:

$$\text{Cover}(R) = \frac{|\{\text{correctly mentioned objects}\}|}{|\{\text{total objects present}\}|}$$

The *Hal* metric, instead, computes the fraction of responses having $\text{CHAIR}_s \neq 0$, while the *Cog* metric computes how many of the generated objects match the set of known hallucinated targets: this is a set mimicking the human cognitive biases.

Except for the *Cover* metric, for all the others, the lower the value a model reaches for them, the better it is at avoiding hallucinations.

POPE. Since the CHAIR metric could be affected by the overall generated length, a new metric was proposed, called Polling-based Object Probing Evaluation (POPE) [48]. The main idea is to evaluate hallucinations via a binary classification task, with simple *yes-or-no* questions about objects: positive questions are formed from ground-truth objects present in the input image, while negative questions are built by sampling objects absent from it. In both cases, the model is asked whether a given object is present in the image.

AMBER. POPE, unfortunately, while being flexible and more robust than the CHAIR metric, suffers from a lack of fine-grained hallucination type evaluation, such as attributes and relations. AMBER [87] employs both generative and discriminative tasks, and unifies the results under a single metric:

$$\text{AMBER} = \frac{1}{2}\left(1 - \text{CHAIR} + \text{F1}\right)$$

where the CHAIR metric is computed over generative tasks, and the F1 score is computed on discriminative tasks.

2.5.3 Latent-Space Alignment in MLLMs

As discussed in the previous section, a possible cause of hallucination in MLLMs is a *weak alignment* between the visual and textual information inside the model: the two modalities are

encoded by separately pre-trained components and their representations may remain geometrically separated (the *modality gap* of Section 2.3.3), possibly causing the language model to override the weaker visual signal with its parametric knowledge. A growing body of work tackles hallucination directly at this level, acting on the model's internal representations so that the visual and textual features describing the same physical object or concept are forced to be more similar to one another, rather than only supervising the model's output. These approaches differ in two respects. The first concerns *when* the intervention is applied: *training-time* methods add an auxiliary objective that reshapes the representation space during the training stage of MLLMs, whereas *inference-time* methods leave the trained weights untouched and intervene only during inference. The second concerns *what* is acted upon: most methods steer the model's latent *representations* (the hidden states), while others operate directly on the output *distribution* over tokens. The two families are discussed below.

Training-Time Alignment. A first family introduces an auxiliary objective that forcefully changes the representation space during training, leaving the architecture unchanged. HACL [35] starts from two empirical observations: the modality gap noted above, and an entanglement between the representations of faithful and hallucinative text, making the two almost impossible to tell apart. To address both, it adds a cross-modal contrastive objective on the representations produced by the language model: matching visual and textual representations are pulled together, while *hallucinative* captions – automatically generated to look like plausible descriptions – are used as negatives to be pushed away. Equipping an MLLM with this objective is reported not only to reduce hallucination, but also to improve performance across several captioning and question-answering benchmarks. In a similar spirit, SANTA [13] (Self-Augmented Contrastive Alignment) targets object and action hallucinations by contrasting the model's representations against self-generated perturbed variants of the same input – by analyzing the portions of the input that the model is more likely to hallucinate –, encouraging it to rely on the genuine visual evidence rather than on the language priors.

A different idea within the same family aligns the *visual* representations of the MLLM to those of a strong, frozen vision model used as a teacher. The motivation is that, under text-only supervision, the internal visual features of an MLLM tend to drift away from those of its vision encoder, progressively losing visual information across layers. VIRAL [94] (Visual Representation Alignment)

counteracts this by adding a regularization term that aligns the intermediate visual representations of the MLLM with the frozen features of a vision foundation model – such as DINOv2 – through a cosine-similarity loss, reporting consistent gains on fine-grained visual benchmarks (see Figure 2.9). Along the same line, VaLR (Vision-aligned Latent Reasoning) [34] aligns the hidden intermediate states of the MLLM with the per-patch features of a pre-trained vision encoder, so that visual information is retained rather than progressively discarded across layers, and additionally teaches the model to perform latent reasoning with vision-aligned tokens. Common to all these methods is that the alignment signal targets either *global* (sequence-level) representations or the *whole* set of visual features, rather than individual image regions matched to the text that describes them; this is a direct consequence of their supervision, which is derived from synthetic negatives or a frozen whole-image teacher, none of which encodes *where* a given piece of text is grounded in the image. They thus differ in what is aligned – text against text (HACL, SANTA) or the MLLM's visual features against a vision teacher (VIRAL, VaLR) – but share this coarse, region-agnostic granularity.

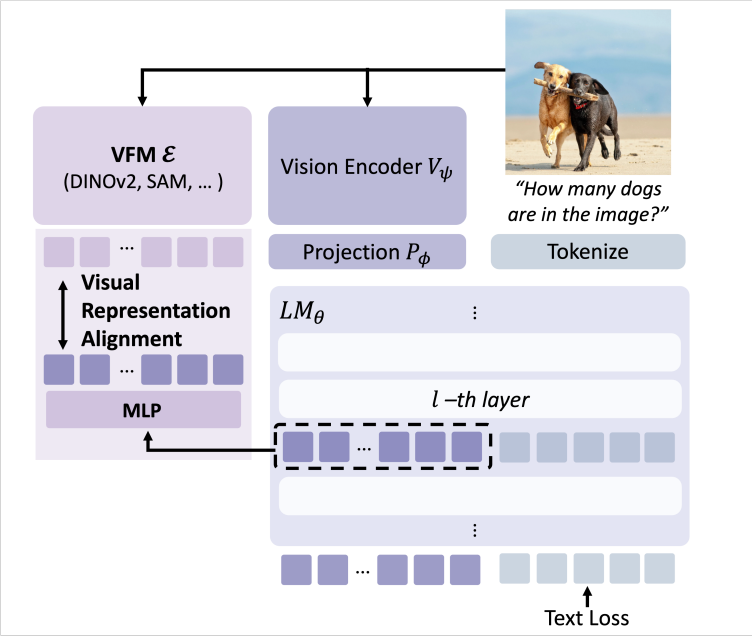

Figure 2.9: The VIRAL methodology. From [94].

Inference-Time Steering. A second family leaves the trained weights untouched and instead intervenes on the latent representations or on the output distribution at decoding time. VTI [53]

(Visual-Textual Intervention) pre-computes, from perturbed images, a set of shift directions that stabilize the visual features. These directions are only determined after training – thereby not requiring additional training cost – using a few samples from one single dataset. The directions are applied then to new queries during generation, actually reducing hallucinations thanks to this steering towards more stable features. Visual Contrastive Decoding (VCD) [45], instead, contrasts the token distributions obtained from the original image and from a distorted, hallucination-inducing, version of it. The language-prior component, that is responsible for ungrounded predictions, is in this way diminished. Because they act only at inference time, these methods are lightweight and broadly applicable, but they do not modify the underlying representations that gave rise to the misalignment in the first place.

Towards Region-Level Alignment. A common trait of the training-time methods above is that their alignment signal is *coarse*: it operates on global, sequence-level representations or on the whole set of visual features, without specifying *which* visual region each piece of text should be grounded in. Object hallucinations, however, are inherently *local* phenomena: each object category, attributes or relationships usually concern a limited region of the whole image. This suggests that supervising the alignment at the level of individual informative patches, rather than the whole image, could target the representations responsible for hallucination more directly. Such fine-grained supervision requires data that explicitly links textual phrases to image regions. Chapter 3 builds on this observation, proposing a training-time objective that aligns the internal representation of each phrase with the visual region it describes.

Chapter 3

Proposed Method

3.1 Motivation

As already discussed in Section 2.5.3, while much work has been done towards training-time alignment for MLLMs, this line of research is entirely focused on sequence-level alignment. More specifically, geometrically bringing closer visual and textual data is performed in literature only on the entire sequence, with no prior work on specific matchings between the language embeddings describing an object and visual embeddings of the patches actually containing the object itself.

We hypothesize that explicitly aligning the internal representation of each object-referring phrase with the visual features of the region the object is contained in strengthens the grounding of the language model's hidden states, counteracting the tendency of the LLM's parametric priors to override a weak visual signal (Section 2.5), thereby reducing object-level hallucination while leaving the model's general visual capabilities intact.

3.2 Preliminaries

In the following section, an overview of the framework for the proposed methodology is given. The first section contains a description of the base multimodal model that is augmented, then the second one details the grounding dataset that provides the region-level supervision, and the last one fixes the notation used throughout the rest of the chapter. The alignment objective itself is introduced in Section 3.3.

3.2.1 Model

The baseline MLLM employed was a LLaVA-1.5 [51] (see Section 2.4.1). The backbone LLM was the open-source Vicuna-1.5 [15], the vision encoder was CLIP ViT-L/14 @336 [69], and the adapter was a standard 2-layer MLP; the latter maps each visual token from the encoder dimension d_v to the language model's hidden dimension d through a linear layer, followed by a GELU non-linearity and a second linear layer of size $d \times d$. The fine-tuning of the models was done over the full set of weights, without employing any PEFT techniques. The teacher that was employed was a distinct vision encoder, a DINOv2 model [63], whose weights remained frozen throughout the whole training process.

3.2.2 Dataset

In order to perform the object-aware grounding, a specific dataset is required. Specifically, a dataset containing pairs of captioned images with additional annotations linking specific sub-strings of the caption to the image regions they describe. For this purpose, the Grounding-anything Dataset (GranD) [73] was employed. The dataset has automatically annotated SAM [42] images, encompassing 7.5M unique concepts, grounded in 810M regions (see Figure 3.1).

Figure 3.1: Images, captions and annotations samples of the GranD dataset. From [73].

Each image is associated with a `dense caption`, describing in length its contents. The caption, then, has a list of `details`. Each detail contains a sub-string of the caption alongside a bounding-box reference. The sub-string references an object alongside its possible attributes, whereas the bounding box references the position of that object inside the image.

3.2.3 Notation

In the dataset, an image I is paired with a dense caption. Once both elements are independently processed and concatenated, they form the input to the LLM. The caption references a set of objects O, in which each object $o \in O$ is characterized by two pieces of information. On the textual side, o corresponds to a *phrase span*, the contiguous sub-string of the caption that refers to it: let $[i+1 : i+n]$ be the generic positions of this span within the full input sequence (i.e., after the inserted visual tokens), so that $\mathbf{h}^{\ell}_{[i+1:i+n]} \in \mathbb{R}^{n \times d}$ collects the hidden states that layer ℓ of the LLM produces over those n tokens. On the visual side, o is localized by a *region* of I, a bounding box, whose appearance is summarized by the frozen teacher into a single embedding $\mathbf{v}_o \in \mathbb{R}^k$. The proposed objective operates on these object-level pairs $\left(\mathbf{h}^{\ell}_{[i+1:i+n]}, \mathbf{v}_o\right)$: a pooling operation P reduces the span to a single textual vector and a projection p_ϕ maps it into the teacher space $\mathbb{R}^k$, where the two modalities can actually be compared.

3.3 Region-Level Alignment Objective

The main change to the LLaVA methodology is the introduction of a penalty term in the loss function, forcing the model to increase the similarity between the textual embeddings in the LLM describing an object with the visual embeddings, outputted from a frozen teacher vision model. An overview of the resulting pipeline is given in Figure 3.2.

Fixing an object $o \in O$, recall from the notation above its span hidden states $\mathbf{h}^{\ell}_{[i+1:i+n]}$ and its teacher embedding $\mathbf{v}_o$. To aggregate the variable number of textual embeddings into a single vector, a *pooling* operation $P : \bigcup_{n=1}^{\infty} \mathbb{R}^{n \times d} \to \mathbb{R}^d$ is applied, yielding $\mathbf{p}^{\ell}_o = P(\mathbf{h}^{\ell}_{[i+1:i+n]})$. Finally, in order to match the dimensionality of the two modalities, a projection $p_\phi : \mathbb{R}^d \to \mathbb{R}^k$ is computed over the pooled textual vector.

The penalty term is hence given by:

$$\mathcal{L}_{align} = \frac{1}{|O|} \sum_{o \in O} \left(1 - \cos\left(p_\phi(\mathbf{p}^{\ell}_o), \mathbf{v}_o\right)\right)$$

Of course, the complete loss function on which the model is trained is:

$$\mathcal{L} = \mathcal{L}_{AR} + \lambda \cdot \mathcal{L}_{align}$$

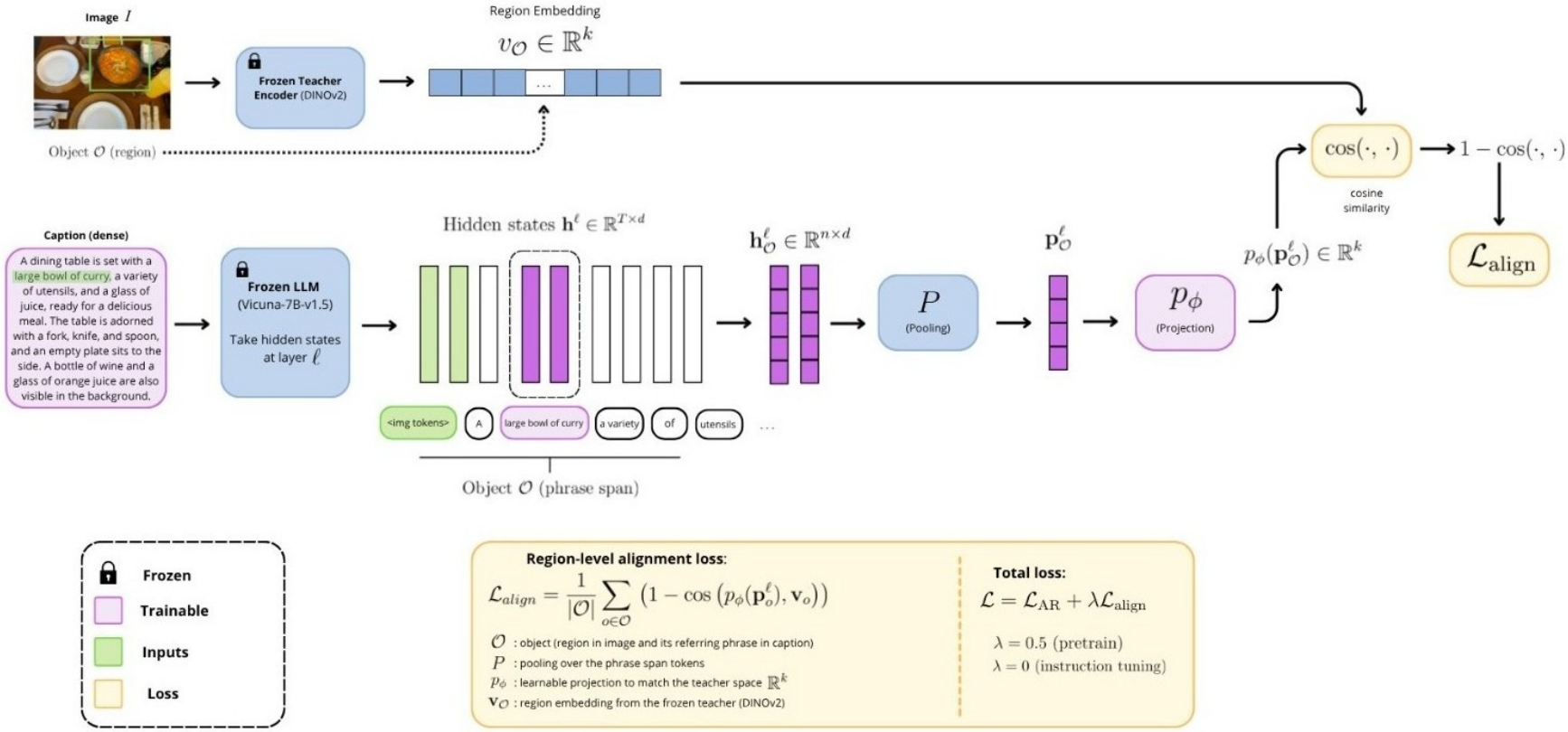

Figure 3.2: Overview of the region-level alignment objective.

where λ is a regularization term. In the case at hand, during the pre-training stage of LLaVA, λ is set to 0.5, whereas in the instruction tuning stage is set to 0, relying fully on the auto-regressive generation loss.

During the pre-training stage, the LLM, the backbone visual encoder and the teacher remain frozen – only the adapter and the projection p_ϕ are trained as a lightweight alignment head.

The proposed objective differs (as briefly stated in Section 3.1) from previous works mainly on the fact that the supervision is *region-level*: each constraint ties the hidden states of a specific phrase span – located inside the LLM's own token stream – to the visual features of the single region that the phrase describes. Prior latent-space methods (Section 2.5.3) instead operate at the sequence level, aligning global or whole-image representations and never matching an individual object's text to its corresponding image region.

3.4 Alignment Variants

In the additional objective specifications (Section 3.3), there were various components that can be implemented in several different ways. These split into choices acting on the *textual* side of the alignment – the layer at which it is applied (Section 3.4.1) and the way the phrase span is pooled (Section 3.4.2) – and choices acting on the *visual* side – the teacher backbone (Section 3.4.3) and the way the region embedding is produced (Section 3.4.4). All examined variants are hereafter explained.

3.4.1 Alignment Depth

It has been shown [99] that visual information flows differently at different layers of a LLaVA model: earlier layers transfer general visual information to the textual tokens whereas middle layers focus on fine-grained details of the vision tokens to pass to the textual embeddings. VaLR [34] (Section 2.5.3), which aligns the MLLM's intermediate states with per-patch visual features, similarly experimented with applying its alignment at different layers, finding that different depths behave differently in this information flow.

Motivated by this phenomenon, the alignment was applied at different depths – i.e., changing the hyperparameter ℓ. Two target layers were considered: the *middle* layer, sitting precisely at half

the number of layers $\left(\ell = \lfloor \frac{L}{2} \rfloor\right)$, and the *next-to-last* layer $(\ell = L - 1)$.

3.4.2 Pooling Strategy

As already stated, the number of textual embeddings $\mathbf{h} \in \mathbb{R}^{n \times d}$ referencing an object o and, possibly, its attributes is variable. Hence, a pooling technique must be employed to reduce the tokens into a single, aggregated, vector. Two different techniques were examined.

The first was *mean-pooling*, which performed a simple average over the set of vectors:

$$P(\mathbf{h}) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{h}_i$$

The second was *last-pooling*, which kept only the last embedding of the sequence:

$$P(\mathbf{h}) = \mathbf{h}_n$$

The idea behind last-pooling is that, due to the causal attention mask of the decoder LLM, the last token of the span attends to all the preceding ones. Therefore it is already capable of aggregating the information of the whole phrase: the same reason its final hidden state is commonly taken as a sequence-level representation in decoder-only models.

3.4.3 Teacher Backbone

While the main teacher backbone employed was as already stated DINOv2, whose purely self-supervised features are known to capture object boundaries and scene layout well (Section 2.3.2), other backbones were experimented with in order to understand the impact of the teacher's own provided representation space. In particular, two language-supervised encoders were tested: CLIP ViT-L/14 [69] and SigLIP [96] (Section 2.3.3). The motivation behind the comparison is that these backbones work in structurally different representation spaces: DINOv2 is optimized purely for visual self-supervision, whereas the contrastive encoders are already shaped by textual supervision.

3.4.4 Region Source

Another important aspect of the alignment objective regards *how* the visual embedding $\mathbf{v}_o$ is computed, given the object's bounding box, the image and the teacher model. Two different strategies were studied.

The first method was *cropping*. The input image was cropped in order to keep only the bounding box region, and then resized and center-cropped to the teacher's expected input resolution by its native image preprocessor. Hence, the frozen encoder would output a certain number of output embeddings. These were summarized into a single feature vector by taking the <CLS> token embedding, which acts as a global summary of the whole crop. This was the case for both DINOv2 and the contrastive encoders, except that, for the CLIP backbone, the <CLS> token was read from the penultimate layer rather than the last one, following common practice.

The second method was *full-image patch pooling*. Instead of passing to the teacher model only a crop, the entire image was provided – which, theoretically speaking, would further enhance the result because it could have more details to produce a more robust representation. Then, only the patch embeddings that could be referred to the original bounding box, were kept and mean-pooled into the single vector $\mathbf{v}_O$.

3.5 Training Procedure

The alignment objective is integrated into the standard two-stage training pipeline of LLaVA (Section 2.4.1) without altering its overall structure.

Two-stage integration. The alignment penalty is applied only during the first stage, which is *feature alignment*. In this stage, the backbone LLM, the visual encoder and the teacher are kept frozen, while the multimodal adapter and the projection p_ϕ are trained. Accordingly, the regularization weight is set to $\lambda = 0.5$. In the following *instruction tuning* stage, the procedure reverts to standard LLaVA fine-tuning: the alignment term is switched off ($\lambda = 0$) and the alignment head is discarded, so that the model is optimized purely on the auto-regressive objective. The reason for this choice is to use the region-level alignment as a *pre-training regularizer* that shapes the representation space before instruction following is learned. Not applying the regularization during the second-stage should avoid any perturbation of the instruction-tuning distribution, which actually leads the model's final conversational behavior.

Batching grounded and ungrounded data. Within the feature-alignment stage, GranD samples are interleaved with the regular image-caption data. Since not every sample carries region an-

notations, each batch is accompanied by a boolean mask $\mathbf{s} \in \{0,1\}^B$ (the `grand_source_mask`), where $s_b = 1$ marks a GranD sample. The auto-regressive loss $\mathcal{L}_{AR}$ is computed over the whole batch as usual, whereas the alignment loss $\mathcal{L}_{align}$ is accumulated only over the grounded samples; ungrounded samples contribute nothing to it. To bound the per-sample cost, and not overshadow the autoregressive loss, at most C objects (a hyperparameter, called `max_crops`) are aligned for each grounded sample.

Phrase-span localization. The span $[i+1 : i+n]$ of an object o must be *located* inside the model's token stream at runtime. For each object, its phrase is tokenized and the resulting sub-token sequence is searched, as a contiguous sub-sequence, among the *supervised* caption tokens – those whose label differs from the ignore index `IGNORE_INDEX`, i.e., the tokens the model is actually trained to generate. When a match is found, the matched positions are shifted by the number of visual tokens inserted before them, so that they index correctly into the hidden-state sequence $\mathbf{h}^\ell$ (whose length includes the expanded visual tokens, not only the textual embeddings). Objects whose phrase cannot be matched (e.g., because the caption contains the phrase with small differences) are silently skipped and do not contribute to $\mathcal{L}_{align}$. So only the subset of annotated objects that are actually found in the generated caption is aligned. The procedure is summarized in Algorithm 3.

Algorithm 3 Phrase-Span Localization

Require: Phrase w, label ids $\mathbf{y}$, input ids $\mathbf{x}$, tokenizer T

$\mathbf{t} \leftarrow T(w)$ ▷ token ids of the sentence
$G \leftarrow \{\, j : \mathbf{y}_j \neq \texttt{IGNORE\_INDEX} \,\}$ ▷ supervised caption positions
for each start position $j \in G$ **do**
  if $\mathbf{y}_{[j:j+|\mathbf{t}|]} = \mathbf{t}$ **then**
    return ShiftByVisualOffset($[\, j : j + |\mathbf{t}| \,]$, $\mathbf{x}$)
  end if
end for
return $\varnothing$ ▷ phrase not found; object will be skipped

Loss computation. At each training step, a single forward pass of the (frozen) LLM yields both the auto-regressive loss and the hidden states at the target layer ℓ, which are requested explicitly. The

alignment loss is then accumulated over the GranD samples of the batch, as detailed in Algorithm 4. For every grounded sample, each annotated object is localized in the token stream, its span is pooled and projected on the textual side, and its region is embedded by the teacher on the visual side; the two are compared through the cosine term of Section 3.3. The per-object terms are averaged within a sample, and the resulting values are averaged across grounded samples, before being added to the auto-regressive loss with weight λ.

Algorithm 4 Region-Level Alignment Loss (per batch)

Require: Batch of B samples, mask $\mathbf{s}$, layer ℓ, weight λ, pooler P, projection p_ϕ, teacher t, cap C

$\mathcal{L}_{AR}$, $\mathbf{H}^\ell \leftarrow$ LlmForward(batch) ▷ request hidden states
$\mathcal{B} \leftarrow [\,]$ ▷ per-sample alignment losses
for each sample b with $s_b = 1$ **do**
    load image I_b and its annotated objects O_b
    acc $\leftarrow [\,]$ ▷ accumulated captions
    for each object $o \in O_b$, up to C matched **do**
        span $\leftarrow$ LocatePhraseSpan(o) ▷ Algorithm 3
        if span $= \varnothing$ **then**
            continue
        end if
        $\mathbf{p} \leftarrow p_\phi\Big(P\Big(\mathbf{H}_b^\ell[\text{span}]\Big)\Big)$ ▷ textual side
        $\mathbf{v} \leftarrow t(\text{Region}(I_b, o))$ ▷ visual side
        append $(1 - \cos(\mathbf{p}, \mathbf{v}))$ to acc
    end for
    if acc $\neq [\,]$ **then**
        append mean(acc) to $\mathcal{B}$
    end if
end for
$\mathcal{L}_{align} \leftarrow \text{mean}(\mathcal{B})$ **if** $\mathcal{B} \neq [\,]$ **else** 0
return $\mathcal{L}_{AR} + \lambda \cdot \mathcal{L}_{align}$

3.6 Sequence-Level Layer Distillation

In parallel to the region-level alignment objective, a complementary methodology was explored. The idea was to still perform grounding the LLM's hidden states through a frozen visual teacher, but at the level of the whole visual-token sequence rather than at the region level. Differently from the objective of Section 3.3, this alternative does not require region annotations: therefore it can be applied to every training sample of the LLaVA pre-training mixture.

3.6.1 Motivation

The region-level loss ties a specific, localized phrase span to the specific region it describes. A different hypothesis is that a more general representational grounding could already be induced without this fine-grained matching, simply by encouraging the LLM's own internal representation of the visual tokens to remain as close as possible to that of a strong frozen visual teacher throughout the network, and not at a single layer.

3.6.2 Layer Aggregation

Rather than intervening at a single fixed depth ℓ (Section 3.4.1), this methodology aggregates the supervision over a contiguous block of L deep decoder layers of the LLM, $\{\ell_1, \ldots, \ell_L\} \subset \{1, \ldots, N\}$. This was entirely performed on the last five layers of the backbone, but it is not a hard requirement. For each selected layer ℓ_i, the hidden states $\mathbf{h}^{\ell_i} \in \mathbb{R}^{T \times d}$ over the full input sequence are collected. A per-layer linear projection $p_{\phi_i} : \mathbb{R}^d \to \mathbb{R}^k$ maps them into the teacher's embedding space. This differs from the pooling operator P of Section 3.4.2, since no span is pooled into a single vector and no phrase location algorithm is performed. The alignment is instead independently computed for every visual token, it is averaged over positions and then aggregated *across layers*.

3.6.3 Objective

On the visual side, the frozen teacher is a DINOv2 [63] model, chosen for the same reasons discussed in Section 3.4.3. Differently from the region-level formulation, the teacher processes the *entire* input image. One embedding per patch, $\mathbf{v}_j \in \mathbb{R}^k$ is kept, rather than storing a single region

summary. Then, the patch embeddings are matched to the visual-token positions j of the LLM's own input sequence. A single loss formulation was considered for the resulting pairs $\left(p_{\phi_i}(\mathbf{h}_j^{\ell_i}), \mathbf{v}_j\right)$: it is a direct *cosine-similarity* penalty, analogous to $\mathcal{L}_{align}$ of Section 3.3, averaged over layers and visual-token positions:

$$\mathcal{L}_{seq} = \frac{1}{L \cdot T_v} \sum_{i=1}^{L} \sum_{j=1}^{T_v} \left(1 - \cos\left(p_{\phi_i}\left(\mathbf{h}_j^{\ell_i}\right), \mathbf{v}_j\right)\right)$$

where T_v is the total output number of the visual patch embeddings.

As with the region-level objective, only the per-layer projections p_{ϕ_i} are trainable, while the LLM, the visual encoder and the teacher remain frozen and the penalty is applied during the feature-alignment stage only.

Chapter 4

Experiments

4.1 Experimental Setup

This section describes the protocol used to validate the proposed method. The implementation details such as datasets, models and training configuration are reported (Section 4.1.1), and then the benchmarks adopted to evaluate both the general visual capabilities and the hallucination tendencies of the resulting models are presented (Section 4.1.2).

4.1.1 Implementation Details

All the compared models share the same architecture and the same two-stage training pipeline, differing only in the alignment configuration under study. The common setup is explained in details below.

Datasets. The datasets employed for the experimental procedure are both the GranD [73] dataset (see Section 3.2.2) and the LLaVA v1.5 [51] original training datasets. In the first *feature alignment* stage, samples from GranD and from LLaVA are interleaved; whereas the whole dataset of LLaVA is employed, only a subset of GranD is used for the training procedure – both for auto-regressive caption generation and for the additional alignment loss (see Section 3.3). In order to make the two datasets equal in terms of data, a subset of 500K samples from GranD are selected and employed.

Models. All experiments build upon the LLaVA-1.5 architecture (Section 2.4.1). The backbone language model is Vicuna-7B-v1.5 [15], the visual encoder is the CLIP ViT-L/14 model at a 336×336 resolution [69] and the adapter is the two-layer `mlp2x_gelu` projector described in Section 3.3. The visual features are extracted from the next-to-last layer from the CLIP model. The frozen teacher providing the region targets is instead a DINOv2-base model [63], whose 768-dimensional `<CLS>` embedding is used as a summary of each region.

The aligned-variant models are compared with a baseline LLaVA v1.5 model. To make the comparison as fair as possible, the baseline is trained with the exact same codebase, with the exception that the additional loss is never applied, in order to keep only the standard auto-regressive loss function.

Training Configuration. The training follows the two-stage paradigm presented in Section 3.5. In the *feature alignment* stage, only the multimodal adapter and the alignment head are updated, while the language model, the visual encoder and the teacher are kept frozen. The alignment weight is set to $\lambda = 0.5$, with last-token pooling of the phrase span at the middle layer of the language model. This stage runs for one epoch over the interleaved 558K-caption LLaVA pre-training data and the 500K GranD subset, with a learning rate of 1×10^{-3} (cosine schedule, 0.03 warm-up ratio, no weight decay) and an effective batch size of 256:

$$8 \text{ GPUs } \times \text{ a per-device batch size of } 8 \times 4 \text{ gradient-accumulation steps } = 256$$

In the subsequent *instruction tuning* stage the alignment term is turned off ($\lambda = 0$) and both the language model and the adapter are fine-tuned end-to-end for one epoch on the 665K-sample LLaVA instruction mixture, with a learning rate of 2×10^{-5} and an effective batch size of 128. The schedule employed is the same as the one of the first stage.

All training runs use `bf16` precision, a maximum sequence length of 2048 tokens, gradient checkpointing, and the DeepSpeed ZeRO optimizer [71]: stage 2 is used during feature alignment and stage 3 during instruction tuning. Following LLaVA-1.5 specifics, input images are padded to a square aspect ratio before being fed to the visual encoder.

4.1.2 Evaluation Benchmarks

The resulting models are evaluated in two different scenarios. The first measures their general visual capabilities, in order to verify that the alignment objective does not degrade the multimodal skills of the model. The second specifically quantifies their tendency to hallucinate.

Visual Capabilities. A suite of public benchmarks is employed, each one identifying different skills. General visual question answering and perception are evaluated through GQA [33], VizWiz [28], ScienceQA [57], SEED-Bench [47], MMBench [55], MME [24] and MMStar [14]. Text and document-oriented understanding is evaluated with TextVQA [80], ChartQA [59], OCRBench [56] and AI2D [40]. Knowledge-intensive and mathematical reasoning are measured by MMMU [95] and MathVista [58]. Finally, benchmarks probing detailed visual perception – RealWorldQA [92], BLINK [25], MMVP [83] and V* [90] – were also employed.

Hallucination Benchmarks. Object hallucination is evaluated with the three benchmarks introduced in Section 2.5. POPE [48] casts hallucination as a binary yes/no answering task over present and absent objects, reporting the accuracy, precision, recall and F1 score. AMBER [87] is employed in both of its settings: the *generative* one reports the CHAIR, Cover, Hal and Cog scores over generated descriptions, while the *discriminative* one reports accuracy, precision, recall and F1 over binary questions. Lastly, the original CHAIR [74] metric is computed on MSCOCO [50] captions, in both its instance-level ($CHAIR_i$) and sentence-level ($CHAIR_s$) variants, together with the object recall and the average caption length. For all of these metrics, except the coverage and the accuracy ones, lower values indicate fewer hallucinations.

4.2 Main Results

This section compares the proposed region-aligned model against the LLaVA-1.5 baseline. The proposed method, which will be called *Ours*, employs region-level alignment by employing last-token pooling at the middle layer, with the cropping technique for the teacher output.

Hallucination. Table 4.1 reports the hallucination benchmarks. The proposed model improves over the baseline on the majority of them. On POPE, accuracy rises from 87.5 to 87.8, and on AMBER's generative setting the fraction of hallucinated responses (Hal) drops sharply, from 43.2 to 38.5, with the CHAIR and Cog scores decreasing as well. This implies that when the model freely generates a description, it mentions fewer non-existent objects, which is what the objective was meant to encourage. The benefit also carries over to AMBER's *discriminative* setting, where accuracy, recall and the overall F1 score all increase (F1 from 78.6 to 79.5). The only exceptions are a marginal drop in AMBER discriminative precision and a small increase of the MSCOCO CHAIR_s and CHAIR_i scores. As will be seen in Section 4.3, the MSCOCO CHAIR metric favors the baseline over *every* aligned variant.

Benchmark	Metric	Baseline	Ours
POPE	Accuracy ↑	87.5	**87.8**
AMBER (gen.)	CHAIR ↓	9.7	**9.0**
	Cover ↑	50.0	**50.8**
	Hal ↓	43.2	**38.5**
	Cog ↓	5.3	**4.6**
AMBER (disc.)	Accuracy ↑	75.7	**76.3**
	Precision ↑	**94.4**	93.4
	Recall ↑	67.3	**69.2**
	F1 ↑	78.6	**79.5**
MSCOCO	CHAIR_s ↓	**55.0**	56.6
	CHAIR_i ↓	**16.3**	16.8
	Recall ↑	78.3	**79.1**

Table 4.1: Hallucination benchmarks: the proposed region-aligned model (*Ours*) versus the baseline. Arrows denote the direction of improvement.

Visual Capabilities. Tables 4.2 and 4.3 report instead the general visual benchmarks. The alignment leaves the model's overall multimodal ability essentially unchanged: the vast majority of benchmarks remain within about one point of the baseline in either direction, with VizWiz (+1.5) and BLINK (+0.4) even improving slightly. These results tell us that the method avoids hallucinations without losing its visual capabilities. The main exceptions are the fine-grained perception benchmarks MMVP and V* (Table 4.3), which drop by 4.7 and 1.6 points.

Model	GQA	VizWiz	SciQA	TextVQA	MME	MMB_{en}	MMB_{cn}	SEED	MMMU	MathVista	AI2D	ChartQA
Baseline	**63.4**	52.0	**69.5**	**58.1**	**1480.5**	**66.0**	**57.9**	**67.2**	34.8	**8.0**	**55.7**	**17.5**
Ours	63.1	**53.5**	69.4	**58.1**	1480.1	65.7	57.0	67.1	**34.9**	**8.0**	**55.7**	16.7

Table 4.2: General visual capability benchmarks: *Ours* versus the baseline.

Model	OCRBench	MMStar	RealWorldQA	BLINK
Baseline	**31.8**	**34.9**	**55.2**	46.3
Ours	31.6	34.4	54.8	**46.7**
Model	MMVP		V*	
Baseline	**33.3**		**52.4**	
Ours	28.7		50.8	

Table 4.3: Fine-grained and OCR-oriented perception benchmarks.

Overall, the proposed method reduces object hallucination across both the generative and the discriminative protocols while preserving the general visual capabilities of the base model. The main exceptions are the MSCOCO CHAIR metric and the two smallest fine-grained perception sets.

4.3 Ablation Studies

This section examines the effect of the individual design choices introduced in Sections 3.4.1–3.4.4, varying one component at a time from the configuration of the main model. Each ablation is reported on a subset of the presented metrics: POPE, the generative CHAIR and Hal scores, and

the discriminative F1.The complete set of benchmark results for every configuration is reported in Appendix A.1.

4.3.1 Alignment Depth and Pooling Strategy

Table 4.4 analyzes the alignment layer and the pooling operator at the same time. Moving the alignment from the next-to-last to the middle layer is an improvement to the method: with last-token pooling it improves the discriminative F1 from 75.9 to 79.5 and the generative CHAIR from 9.5 to 9.0, at no cost on POPE. The effect is smaller, and less consistent, for mean pooling.

Comparing the two pooling operators at the middle layer, last-token pooling clearly dominates on the AMBER metrics – F1 79.5 against 76.9, CHAIR 9.0 against 9.6, Hal 38.5 against 40.7 – whereas mean pooling is marginally better only on POPE (88.2 against 87.8).

Based on these ablations, the last-token and middle-layer configuration is therefore adopted as the main model.

Pooling	Layer	POPE ↑	CHAIR ↓	Hal ↓	F1 ↑
Baseline		87.5	9.7	43.2	78.6
last	next-to-last	87.8	9.5	**38.5**	75.9
last	middle (*Ours*)	87.8	**9.0**	**38.5**	**79.5**
mean	next-to-last	87.6	9.5	41.2	77.8
mean	middle	**88.2**	9.6	40.7	76.9

Table 4.4: Joint ablation of the alignment depth and the pooling operator.

4.3.2 Teacher Backbone

Table 4.5 replaces the DINOv2 teacher with the language-supervised CLIP and SigLIP encoders at the middle layer. DINOv2 with last-token pooling remains the strongest on generative metrics and on F1, supporting the intuition that a purely visual teacher provides a more informative region target than a language-aligned one. CLIP is competitive – it attains the best POPE and a comparable F1 under mean pooling, probably because it is the same model that is used as vision encoder of the

MLLM – while SigLIP is consistently the weakest. These results motivate the choice of DINOv2 as the default teacher.

Teacher	Pooling	POPE ↑	CHAIR ↓	Hal ↓	F1 ↑
DINOv2	last (*Ours*)	87.8	**9.0**	**38.5**	**79.5**
DINOv2	mean	88.2	9.6	40.7	76.9
CLIP	last	**88.3**	9.3	40.7	77.3
CLIP	mean	87.5	9.6	40.6	78.9
SigLIP	mean	87.6	9.6	41.2	75.9

Table 4.5: Teacher backbone ablation, at the middle layer.

4.3.3 Region Source

Table 4.6 compares the two region sources which were explained in Section 3.4.4. Their effect on most metrics is small; the notable exception is the MSCOCO CHAIR score, in which full-image patch pooling consistently lowers with respect to cropping – from 59.4 to 56.4 at the middle layer, and from 57.6 to 54.2 at the next-to-last. Providing the teacher with the whole image, rather than an isolated crop, thus partially mitigates the one systematic regression observed in the main results (Section 4.2). This can be explained by the fact that having the teacher model access the whole image grants it much more information about the single region than the crop alone.

Layer	Region	POPE ↑	CHAIR ↓	Hal ↓	F1 ↑	CHAIR_s ↓
Baseline		87.5	9.7	43.2	**78.6**	55.0
next-to-last	crop	87.6	**9.5**	41.2	77.8	57.6
next-to-last	full	87.8	9.7	41.0	76.5	**54.2**
middle	crop	**88.2**	9.6	**40.7**	76.9	59.4
middle	full	88.0	**9.5**	41.0	77.9	56.4

Table 4.6: Region source ablation (mean-token pooling).

Image-to-Image Alignment. Finally, an alternative formulation was tested in which the alignment is applied not between the textual phrase span and the region, but between the language model's own visual tokens falling inside the region and the teacher embedding – an *image-to-image* objective. This variant did not yield consistent improvements over the text-to-region objective: relative to the equivalent full-image text-to-region model, it improved the discriminative F1 (79.2 against 77.9) but worsened the generative CHAIR (9.7 against 9.5) and Hal (42.9 against 41.0) scores, while leaving POPE unchanged. It was therefore not pursued further, and the textual phrase span is used as the anchor of the alignment throughout.

4.4 Qualitative Results

In the present section, a few qualitative examples of our method are presented (see Figures 4.1–4.3). Sentences marked with **red** represent object, attribute or relationship hallucinations, whereas sentences in **green** signal an improvement – either by deleting the hallucination or by being more fine-grained – on the baseline.

4.5 Discussion

The results give evidence for the hypothesis of Chapter 3: tying the representation of an object-referring phrase to the visual features of the region that contains the object itself reduces object hallucination, without damaging the general visual ability of the base model.

The best configuration. The employed configuration uses the middle-layer, last-token pooling methodology. Both of them have benefits over their counterparts. Last-token pooling exploits the causal structure of the language model: the final token of a phrase span has already attended to the whole span, so its hidden state is a natural summary of the object description. Mean pooling, instead, dilutes this signal by averaging over tokens of different informativeness.

The middle layer is where the visual information transferred from the image tokens is richest, before it is progressively abstracted away in the upper layers (as it was seen in Section 3.4.1). Hence, applying the objective there grounds the textual representation while the visual evidence is

LLaVA-1.5 (Baseline)

The image depicts a cloudy day with a dark sky, and there are multiple traffic lights visible in the scene. **Some of the traffic lights are located near the top of the image**, while others are positioned closer to the bottom. The traffic lights are situated at various heights and angles, creating a sense of depth in the scene. In addition to the traffic lights, there are two street signs in the image, one located near the top left corner and the other towards the center of the scene. The presence of these signs and traffic lights indicates that this is an urban area with a well-regulated traffic system.

Ours (Region-Aligned)

The image depicts a cloudy day with a street scene featuring multiple traffic lights. **There are at least nine traffic lights visible in the scene, with some located closer to the foreground and others further away**. The traffic lights are positioned at various heights and angles, creating a sense of depth in the image. In addition to the traffic lights, **there is a street sign and a stop sign present in the scene**. The stop sign is located near the center of the image, while the street sign is positioned slightly to the left. The overall atmosphere of the image is that of a typical city street on a cloudy day.

Figure 4.1: Qualitative comparison on the MSCOCO dataset (image 269682).

LLaVA-1.5 (Baseline)

The image features a street corner with a street sign and a Canadian flag. The street sign is mounted on a pole, and the Canadian flag is flying nearby. The scene is set in a city, with a tall building visible in the background. **There are several people in the scene, with one person standing closer to the left side of the image, another person in the middle, and two more people on the right side. A clock can be seen in the upper right corner of the image, possibly indicating the time for passersby**.

Ours (Region-Aligned)

The image features a street scene with a street sign and a Canadian flag on a pole. The street sign is located near the top of the pole, while the flag is positioned further down. The street sign is also accompanied by a street name sign, indicating the location. **In the background, there are two tall buildings, one on the left side and another on the right side of the scene. The buildings are situated behind the street sign and the flag**, creating a sense of depth in the image.

Figure 4.2: Qualitative comparison on the MSCOCO dataset (image 335800).

LLaVA-1.5 (Baseline)

The image features a person holding a sandwich in their hand, with a bite taken out of it. The sandwich appears to be a large, juicy steak sandwich. The person is also holding a cup of soda, which is placed next to the sandwich. In the scene, **there are two other people present, one on the left side and another on the right side. The person on the left is holding a sandwich as well, while the person on the right is holding a cup**. The image captures a casual dining moment with friends enjoying their meal together.

Ours (Region-Aligned)

The image features a close-up of a person holding a sandwich in their hands, with a bite taken out of it. The sandwich appears to be a large, juicy steak sandwich. The person is also holding **a cup of Pepsi**, which is placed near the sandwich. In the background, **another person can be seen, partially visible on the left side of the image**. The dining table occupies most of the background, extending from the left edge to the right edge of the image.

Figure 4.3: Qualitative comparison on the MSCOCO dataset (image 268734).

still accessible and useful to be used. Combined, the two give the lowest hallucination scores.

The MSCOCO CHAIR regression. The one benchmark that worsens in every configuration is the MSCOCO CHAIR score, on which the baseline is better than all aligned variants. A plausible explanation is that exposure to GranD's dense, object-rich captions encourages the model to enumerate more objects when freely captioning, which raises the raw count of hallucinated objects measured by CHAIR even as the response-level hallucination rate decreases. The region-source ablation (Section 3.4.4) partially backs this up: feeding the teacher the whole image instead of an isolated crop consistently lowers CHAIR_s, which suggests that a broader visual context discourages the model from inventing objects that a tight crop cannot disambiguate. The regression is thus not intrinsic to the region-level objective, but a side effect of the supervision distribution that a better region source could partially correct.

Fine-grained perception. The alignment improves object-level grounding, but does not carry over to fine-grained perception: the small drops on MMVP and V^* show this. Part of the phenomenon can be expected from how the objective is built: the teacher summarizes each region into a single `<CLS>` embedding, which captures *what* an object is rather than its fine intra-region detail. Such a target helps suppress hallucinated objects, but gives little signal for the subtle attribute and spatial distinctions that MMVP and V^* ask for.

4.6 Sequence-Level Layer Distillation Results

This section reports the results of the alternative methodology introduced in Section 3.6, in which a DINOv2 teacher is used to align at the patch level and without region annotations the last five decoder layers of the LLM. The method was tested across different vision-encoder combinations: Vicuna-7B, Qwen2.5-7B and Llama-3-8B, paired with CLIP or SigLIP. Each configuration is compared against the same pipeline without the additional layer-distillation objective. Table 4.7 reports the hallucination-oriented metrics (POPE accuracy, AMBER generative Hal and CHAIR, AMBER discriminative F1) alongside a general-capability indicator (MMBench-en), for every pair; the complete set of benchmark results is reported in Appendix A.2.

Backbone	Vision	Variant	POPE ↑	CHAIR ↓	Hal ↓	F1 (disc.) ↑	MMBench-en ↑
Llama-3-8B	CLIP	Baseline	85.9	9.1	40.1	65.7	68.9
		5-Layer	**86.8**	**8.2**	**36.5**	**73.2**	**70.2**
Qwen2.5-7B	SigLIP	Baseline	85.6	7.0	31.1	72.7	74.6
		5-Layer	**85.8**	**6.8**	**30.8**	**74.2**	**76.5**
Vicuna-7B	CLIP	Baseline	**87.2**	**8.3**	**38.0**	77.3	64.9
		5-Layer	87.0	**8.3**	40.3	**78.6**	**66.0**
Vicuna-7B	SigLIP	Baseline	**85.9**	8.4	**37.4**	74.9	65.9
		5-Layer	85.4	**8.2**	37.6	**75.3**	**66.2**

Table 4.7: Sequence-level layer distillation versus matched baselines, across backbones and vision encoders.

4.6.1 Discussion

The most consistent improvement is obtained with the Llama-3-8B backbone. The distilled variant improves every hallucination metric considered: POPE rises from 85.9 to 86.8, the AMBER generative CHAIR drops from 9.1 to 8.2 and Hal from 40.1 to 36.5, while the AMBER discriminative F1 jumps from 65.7 to 73.2, driven by a large recall gain. The most important observation is that this increase is not achieved at the expense of general capability: MMBench-en improves as well (from 68.9 to 70.2), together with GQA, SEED and VizWiz. This is the only configuration in which the sequence-level distillation improves hallucination metrics, discriminative accuracy *and* general visual capability simultaneously, making it the strongest evidence in favor of this alternative methodology.

The second-best result is obtained by pairing Qwen2.5-7B with the SigLIP vision encoder: the AMBER discriminative F1 improves from 72.7 to 74.2, both MSCOCO CHAIR variants decrease (CHAIR_s: 49.6 → 48.6, CHAIR_i: 15.1 → 13.4), and POPE improves slightly. General capability also benefits, with MMBench-en rising by nearly two points (from 74.6 to 76.5) and MME by 57 points.

The Vicuna-7B pairs, instead, show smaller but still positive effects. The model, paired with either vision encoder, shows a small but consistent gain in AMBER discriminative F1 and in MMBench-en, with MMVP and BLINK improving markedly under SigLIP (+3.3 and +2.9 points, respectively). The hallucinations benchmarks are not negatively affected, and still remain comparable with the baseline.

Overall, distilling a frozen DINOv2 teacher into the last five layers of the LLM at the patch level yields a consistent, if generally modest, improvement in hallucination metrics without harming general visual capability. The Llama-3-8B and Qwen2.5-7B/SigLIP pairings provide the clearest evidence that this region-agnostic formulation is a viable complement to the region-level objective of Chapter 3.

Chapter 5

Conclusion

5.1 Key Findings

The main contribution of this thesis was to implement, analyze and evaluate a new methodology for reducing the phenomenon of visual hallucinations in MLLMs, through a fine-grained alignment setup. Specifically, an additional training-time objective was added, aimed at further aligning the textual representations of objects with the representations of the specific image regions describing them, extracted from a frozen visual teacher model.

The experimental results show evidence in support of the effectiveness of the proposed method. A model trained via this additional objective shows a decrease in the overall manifestation of hallucinations during generation. The model improves over a baseline, in which the additional objective is not applied, in a suite of hallucinatory benchmarks: it improves on the challenging POPE benchmark and on the AMBER dataset, both in its generative and discriminative frameworks, while having a slight regression over the MSCOCO CHAIR metric. This general improvement signals the additional objective's capability in reducing the omission of existing objects and in the mention of non-existing objects, attributes or relationships.

Moreover, the broad set of general visual benchmarks employed also signals that the method does not, in fact, improve on hallucinations at the cost of losing its capabilities; instead, while it does not show increased capabilities if not for a few datasets, the difference is negligible and comparable with the baseline for the vast majority of benchmarks. Within this setup, the best configuration pairs the alignment applied at the middle layer with last-token pooling of the phrase span, as it

consistently yields the lowest hallucination scores among the studied variants.

The alternative, sequence-level layer distillation methodology, explored in parallel, also proved beneficial in reducing hallucinations without harming general capabilities, strengthening the claim that the inherent modality gap in MLLMs is actually one of the causes behind hallucinative behaviors.

5.2 Limitations and Future Works

Despite these encouraging results, the proposed method still presents a few limitations. First of all, it requires region-annotated data such as that provided by the GranD dataset. This specific data requirement is not always available, and when it is, it is usually machine-annotated, meaning it could be inaccurate or erroneous and consequently pollute the model's capabilities. Secondly, the teacher summarizes each region into a single, coarse embedding, which is enough to suppress object-level hallucinations but provides little information for the fine-grained attribute and spatial distinctions that benchmarks like MMVP and V^* require. Moreover, the additional alignment was only forced during the first stage of the training pipeline, without ever being added to the second stage of the procedure. Finally, despite promising results in many hallucinative benchmarks, on the CHAIR MSCOCO dataset, the methodology still presents a slight regression that could be attributed to the object-rich captions of the GranD dataset.

Future work could address these points from different angles: exploring more fine-grained region representations in place of the single teacher embedding, reducing the dependency on region-annotated datasets, and devising an additional dialogical dataset containing region annotations to wire in the alignment objective during the second stage of the training pipeline. Combining the two proposed objectives, so that the model is aligned both region-by-region and across the whole visual-token sequence, could also be another direction to pursue.

5.3 Final Remarks

This thesis has shown that explicitly grounding an MLLM's internal representations to a strong visual teacher, at a fine-grained region-level depth, is an effective way to counteract the modality gap. This approach consequently reduces object-level hallucinations at a lightweight cost at training

time, while having no additional cost at inference time. The results obtained, together with the promising evidence given by the sequence-level alternative, suggest that acting directly on the model's latent representations is a fruitful direction to investigate further, alongside the more established decoding-based approaches, to provide hallucination mitigation in Multimodal Large Language Models.

Bibliography

[1] Josh Achiam et al. “Gpt-4 technical report”. In: *arXiv preprint arXiv:2303.08774* (2023).

[2] Joshua Ainslie et al. “Gqa: Training generalized multi-query transformer models from multi-head checkpoints”. In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 2023, pp. 4895–4901.

[3] Jean-Baptiste Alayrac et al. “Flamingo: a visual language model for few-shot learning”. In: *Advances in neural information processing systems* 35 (2022), pp. 23716–23736.

[4] Mahmoud Assran et al. “Self-supervised learning from images with a joint-embedding predictive architecture”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 15619–15629.

[5] Jinze Bai et al. “Qwen technical report”. In: *arXiv preprint arXiv:2309.16609* (2023).

[6] Jinze Bai et al. *Qwen-VL: A Versatile Vision-Language Model for Understanding, Localization, Text Reading, and Beyond*. 2023. arXiv: 2308.12966 [cs.CV]. URL: https://arxiv.org/abs/2308.12966.

[7] Zechen Bai et al. “Hallucination of multimodal large language models: A survey”. In: *arXiv preprint arXiv:2404.18930* (2024).

[8] Y. Bengio, P. Simard, and P. Frasconi. “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166. DOI: 10.1109/72.279181.

[9] Sidney Black et al. “Gpt-neox-20b: An open-source autoregressive language model”. In: *Proceedings of BigScience Episode# 5–Workshop on Challenges & Perspectives in Creating Large Language Models*. 2022, pp. 95–136.

[10] Tom Brown et al. “Language models are few-shot learners”. In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[11] Davide Caffagni et al. “The revolution of multimodal large language models: A survey”. In: *Findings of the association for computational linguistics: ACL 2024* (2024), pp. 13590–13618.

[12] Mathilde Caron et al. “Emerging properties in self-supervised vision transformers”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 9650–9660.

[13] Kai-Po Chang et al. “Mitigating object and action hallucinations in multimodal llms via self-augmented contrastive alignment”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2026, pp. 3172–3181.

[14] Lin Chen et al. “Are We on the Right Way for Evaluating Large Vision-Language Models?” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2024. arXiv: 2403.20330.

[15] Wei-Lin Chiang et al. *Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality*. Mar. 2023. URL: https://lmsys.org/blog/2023-03-30-vicuna/.

[16] Aakanksha Chowdhery et al. “Palm: Scaling language modeling with pathways”. In: *Journal of machine learning research* 24.240 (2023), pp. 1–113.

[17] Federico Cocchi et al. “Llava-more: A comparative study of llms and visual backbones for enhanced visual instruction tuning”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2025, pp. 4278–4288.

[18] Wenliang Dai et al. “Instructblip: Towards general-purpose vision-language models with instruction tuning”. In: *Advances in neural information processing systems* 36 (2023), pp. 49250–49267.

[19] Fahim Dalvi et al. “Discovering latent concepts learned in BERT”. In: *arXiv preprint arXiv:2205.07237* (2022).

[20] Karan Desai and Justin Johnson. “VirTex: Learning Visual Representations From Textual Annotations”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 11162–11173.

[21] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*. 2019, pp. 4171–4186.

[22] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929* (2020).

[23] Fartash Faghri et al. “Vse++: Improving visual-semantic embeddings with hard negatives”. In: *arXiv preprint arXiv:1707.05612* (2017).

[24] Chaoyou Fu et al. “MME: A Comprehensive Evaluation Benchmark for Multimodal Large Language Models”. In: *arXiv preprint arXiv:2306.13394* (2023).

[25] Xingyu Fu et al. “BLINK: Multimodal Large Language Models Can See but Not Perceive”. In: *European Conference on Computer Vision (ECCV)*. 2024. arXiv: 2404.12390.

[26] Aaron Grattafiori et al. “The llama 3 herd of models”. In: *arXiv preprint arXiv:2407.21783* (2024).

[27] Jean-Bastien Grill et al. “Bootstrap your own latent-a new approach to self-supervised learning”. In: *Advances in neural information processing systems* 33 (2020), pp. 21271–21284.

[28] Danna Gurari et al. “VizWiz Grand Challenge: Answering Visual Questions from Blind People”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 3608–3617.

[29] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

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

[31] Jordan Hoffmann et al. “Training compute-optimal large language models”. In: *arXiv preprint arXiv:2203.15556* 10 (2022).

[32] Jiaxing Huang and Jingyi Zhang. “A survey on evaluation of multimodal large language models”. In: *arXiv preprint arXiv:2408.15769* (2024).

[33] Drew A Hudson and Christopher D Manning. “Gqa: A new dataset for real-world visual reasoning and compositional question answering”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 6700–6709.

[34] Byungwoo Jeon et al. “Vision-aligned Latent Reasoning for Multi-modal Large Language Model”. In: *arXiv preprint arXiv:2602.04476* (2026).

[35] Chaoya Jiang et al. “Hallucination augmented contrastive learning for multimodal large language model”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 27036–27046.

[36] Yihang Jiang et al. “6G non-terrestrial networks enabled low-altitude economy: Opportunities and challenges”. In: *arXiv preprint arXiv:2311.09047* (2023).

[37] Armand Joulin et al. “Learning Visual Features from Large Weakly Supervised Data”. In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, 2016, pp. 67–84. ISBN: 978-3-319-46478-7.

[38] Raphi Kang et al. “Is CLIP ideal? No. Can we fix it? Yes!” In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2025, pp. 22436–22446.

[39] Jared Kaplan et al. “Scaling laws for neural language models”. In: *arXiv preprint arXiv:2001.08361* (2020).

[40] Aniruddha Kembhavi et al. “A Diagram Is Worth A Dozen Images”. In: *European Conference on Computer Vision (ECCV)*. 2016, pp. 235–251.

[41] Wonjae Kim, Bokyung Son, and Ildoo Kim. “Vilt: Vision-and-language transformer without convolution or region supervision”. In: *International conference on machine learning*. PMLR. 2021, pp. 5583–5594.

[42] Alexander Kirillov et al. “Segment Anything”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2023, pp. 4015–4026.

[43] Yann LeCun et al. “A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27”. In: *Open Review* 62.1 (2022), pp. 1–62.

[44] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539.

[45] Sicong Leng et al. “Mitigating object hallucinations in large vision-language models through visual contrastive decoding”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 13872–13882.

[46] Martha Lewis et al. “Does CLIP Bind Concepts? Probing Compositionality in Large Image Models”. In: *Findings of the Association for Computational Linguistics: EACL 2024*. Ed. by Yvette Graham and Matthew Purver. St. Julian’s, Malta: Association for Computational Linguistics, Mar. 2024, pp. 1487–1500. DOI: 10.18653/v1/2024.findings-eacl.101. URL: https://aclanthology.org/2024.findings-eacl.101/.

[47] Bohao Li et al. “SEED-Bench: Benchmarking Multimodal LLMs with Generative Comprehension”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024. arXiv: 2307.16125.

[48] Yifan Li et al. “Evaluating object hallucination in large vision-language models”. In: *Proceedings of the 2023 conference on empirical methods in natural language processing*. 2023, pp. 292–305.

[49] Victor Weixin Liang et al. “Mind the gap: Understanding the modality gap in multi-modal contrastive representation learning”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 17612–17625.

[50] Tsung-Yi Lin et al. “Microsoft coco: Common objects in context”. In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[51] Haotian Liu et al. “Improved baselines with visual instruction tuning”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2024, pp. 26296–26306.

[52] Haotian Liu et al. “Visual instruction tuning”. In: *Advances in neural information processing systems* 36 (2023), pp. 34892–34916.

[53] Sheng Liu, Haotian Ye, and James Y Zou. “Reducing hallucinations in large vision-language models via latent space steering”. In: *International Conference on Learning Representations*. Vol. 2025. 2025, pp. 72402–72419.

[54] Yiheng Liu et al. “Understanding llms: A comprehensive overview from training to inference”. In: *Neurocomputing* 620 (2025), p. 129190.

[55] Yuan Liu et al. “MMBench: Is Your Multi-modal Model an All-around Player?” In: *European Conference on Computer Vision (ECCV)*. 2024. arXiv: 2307.06281.

[56] Yuliang Liu et al. “OCRBench: On the Hidden Mystery of OCR in Large Multimodal Models”. In: *Science China Information Sciences* 67.12 (2024).

[57] Pan Lu et al. “Learn to Explain: Multimodal Reasoning via Thought Chains for Science Question Answering”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022.

[58] Pan Lu et al. “MathVista: Evaluating Mathematical Reasoning of Foundation Models in Visual Contexts”. In: *International Conference on Learning Representations (ICLR)*. 2024. arXiv: 2310.02255.

[59] Ahmed Masry et al. “ChartQA: A Benchmark for Question Answering about Charts with Visual and Logical Reasoning”. In: *Findings of the Association for Computational Linguistics: ACL 2022*. 2022, pp. 2263–2279.

[60] Tomas Mikolov et al. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).

[61] Hermann Ney, Ute Essen, and Reinhard Kneser. “On structuring probabilistic dependences in stochastic language modelling”. In: *Computer Speech & Language* 8.1 (1994), pp. 1–38. ISSN: 0885-2308. DOI: https://doi.org/10.1006/csla.1994.1001. URL: https://www.sciencedirect.com/science/article/pii/S0885230884710011.

[62] Mohd Halim Mohd Noor and Ayokunle Olalekan Ige. “A survey on state-of-the-art deep learning applications and challenges”. In: *Engineering Applications of Artificial Intelligence* 159 (2025), p. 111225.

[63] Maxime Oquab et al. “Dinov2: Learning robust visual features without supervision”. In: *arXiv preprint arXiv:2304.07193* (2023).

[64] Long Ouyang et al. “Training language models to follow instructions with human feedback”. In: *Advances in neural information processing systems* 35 (2022), pp. 27730–27744.

[65] Jeffrey Pennington, Richard Socher, and Christopher D Manning. “Glove: Global vectors for word representation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[66] Ofir Press, Noah A Smith, and Mike Lewis. “Train short, test long: Attention with linear biases enables input length extrapolation”. In: *arXiv preprint arXiv:2108.12409* (2021).

[67] Alec Radford et al. “Improving language understanding by generative pre-training”. In: (2018).

[68] Alec Radford et al. “Language models are unsupervised multitask learners”. In: *OpenAI blog* 1.8 (2019), p. 9.

[69] Alec Radford et al. “Learning transferable visual models from natural language supervision”. In: *International conference on machine learning*. PmLR. 2021, pp. 8748–8763.

[70] Colin Raffel et al. “Exploring the limits of transfer learning with a unified text-to-text transformer”. In: *Journal of machine learning research* 21.140 (2020), pp. 1–67.

[71] Samyam Rajbhandari et al. “Zero: Memory optimizations toward training trillion parameter models”. In: *SC20: international conference for high performance computing, networking, storage and analysis*. IEEE. 2020, pp. 1–16.

[72] Prajit Ramachandran et al. “Stand-alone self-attention in vision models”. In: *Advances in neural information processing systems* 32 (2019).

[73] Hanoona Rasheed et al. “Glamm: Pixel grounding large multimodal model”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 13009–13018.

[74] Anna Rohrbach et al. “Object hallucination in image captioning”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 4035–4045.

[75] Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. “Are emergent abilities of large language models a mirage?” In: *Advances in neural information processing systems* 36 (2023), pp. 55565–55581.

[76] Claude Elwood Shannon. “A mathematical theory of communication”. In: *The Bell system technical journal* 27.3 (1948), pp. 379–423.

[77] Piyush Sharma et al. “Conceptual Captions: A Cleaned, Hypernymed, Image Alt-text Dataset For Automatic Image Captioning”. In: *Proceedings of ACL*. 2018.

[78] Noam Shazeer. “Fast transformer decoding: One write-head is all you need”. In: *arXiv preprint arXiv:1911.02150* (2019).

[79] Noam Shazeer. “Glu variants improve transformer”. In: *arXiv preprint arXiv:2002.05202* (2020).

[80] Amanpreet Singh et al. “Towards VQA Models That Can Read”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 8317–8326.

[81] Jianlin Su et al. “Roformer: Enhanced transformer with rotary position embedding”. In: *Neurocomputing* 568 (2024), p. 127063.

[82] Shengbang Tong et al. “Eyes Wide Shut? Exploring the Visual Shortcomings of Multimodal LLMs”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2024, pp. 9568–9578.

[83] Shengbang Tong et al. “Eyes Wide Shut? Exploring the Visual Shortcomings of Multimodal LLMs”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024, pp. 9568–9578.

[84] Hugo Touvron et al. “Llama 2: Open foundation and fine-tuned chat models”. In: *arXiv preprint arXiv:2307.09288* (2023).

[85] Hugo Touvron et al. “Llama: Open and efficient foundation language models”. In: *arXiv preprint arXiv:2302.13971* (2023).

[86] Ashish Vaswani et al. “Attention is all you need”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964.

[87] Junyang Wang et al. “Amber: An llm-free multi-dimensional benchmark for mllms hallucination evaluation”. In: *arXiv preprint arXiv:2311.07397* (2023).

[88] Xiaolong Wang et al. “Non-local neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7794–7803.

[89] Jason Wei et al. “Emergent abilities of large language models”. In: *arXiv preprint arXiv:2206.07682* (2022).

[90] Penghao Wu and Saining Xie. “V*: Guided Visual Search as a Core Mechanism in Multimodal LLMs”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024.

[91] Zhirong Wu et al. “Unsupervised feature learning via non-parametric instance discrimination”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 3733–3742.

[92] xAI. *Grok-1.5 Vision Preview*. `https://x.ai/news/grok-1.5v`. 2024.

[93] Ruibin Xiong et al. “On layer normalization in the transformer architecture”. In: *International conference on machine learning*. PMLR. 2020, pp. 10524–10533.

[94] Heeji Yoon et al. “Visual representation alignment for multimodal large language models”. In: *arXiv preprint arXiv:2509.07979* (2025).

[95] Xiang Yue et al. “MMMU: A Massive Multi-discipline Multimodal Understanding and Reasoning Benchmark for Expert AGI”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024.

[96] Xiaohua Zhai et al. “Sigmoid loss for language image pre-training”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2023, pp. 11975–11986.

[97] Biao Zhang and Rico Sennrich. “Root mean square layer normalization”. In: *Advances in neural information processing systems* 32 (2019).

[98] Renrui Zhang et al. “Llama-adapter: Efficient fine-tuning of language models with zero-init attention”. In: *arXiv preprint arXiv:2303.16199* (2023).

[99] Zhi Zhang et al. “Cross-modal information flow in multimodal large language models”. In: *Proceedings of the Computer Vision and Pattern Recognition Conference*. 2025, pp. 19781–19791.

[100] Deyao Zhu et al. “MiniGPT-4: Enhancing Vision-Language Understanding with Advanced Large Language Models”. In: *International Conference on Learning Representations*. Ed. by B. Kim et al. Vol. 2024. 2024, pp. 18378–18394. URL: https://proceedings.iclr.cc/paper_files/paper/2024/file/50623630a2372839c078474efa6c0cb8-Paper-Conference.pdf.

Appendix A

Full Ablation Results

This appendix reports the complete benchmark results underlying the ablation studies of Chapter 4, of which only a subset of metrics is presented in the main chapter. Results are grouped by benchmark family, mirroring the split of Section 4.2: hallucination benchmarks (POPE, AMBER generative and discriminative, MSCOCO CHAIR), general visual capability benchmarks, and OCR-oriented and fine-grained benchmarks.

A.1 Region-Level Alignment: Full Ablation Results

This section reports the complete set of benchmark results for the ablations of Section 4.3.

A.1.1 Alignment Depth and Pooling Strategy

Full results for the configurations of Table 4.4.

Pooling	Layer	Notes	POPE	CHAIR	Cover	Hal	Cog	Acc	Prec	Rec	F1	CHAIRs	CHAIRi	Rec	Len
Baseline	–	–	87.5	9.7	50.0	43.2	5.3	75.7	94.4	67.3	78.6	55.0	16.3	78.3	103.4
last	next-to-last		87.8	9.5	51.0	38.5	4.6	73.1	93.8	63.7	75.9	57.4	16.5	79.3	103.0
last	middle	(Ours)	87.8	9.0	50.8	38.5	4.6	76.3	93.4	69.2	79.5	56.6	16.8	79.1	104.3
mean	next-to-last		87.6	9.5	50.8	41.2	5.2	74.7	93.3	66.7	77.8	57.6	16.9	79.6	103.4
mean	middle		88.2	9.6	50.0	40.7	5.0	74.0	93.7	65.2	76.9	59.4	17.8	78.9	102.4

Table A.1: Alignment depth and pooling strategy: hallucination benchmarks.

Pooling	Layer	Notes	GQA	VizWiz	SciQA	TextVQA	MME	MMB-en	MMB-cn	SEED	MMMU	MathVista	AI2D	ChartQA
Baseline	–	–	63.4	52.0	69.5	58.1	1480.5	66.0	57.9	67.2	34.8	8.0	55.7	17.5
last	next-to-last		63.0	57.2	67.9	57.6	1461.9	67.1	58.5	67.2	36.1	6.3	55.2	16.3
last	middle	(Ours)	63.1	53.5	69.4	58.1	1480.1	65.7	57.0	67.1	34.9	8.0	55.7	16.7
mean	next-to-last		63.4	53.5	68.8	58.0	1513.7	65.8	58.3	67.4	35.2	7.3	55.4	16.7
mean	middle		63.2	52.6	68.8	57.7	1491.8	66.1	57.1	66.6	34.9	7.5	55.4	16.6

Table A.2: Alignment depth and pooling strategy: general visual capability benchmarks.

Pooling	Layer	Notes	OCRBench	MMStar	RealWQA	BLINK	MMVP	V*
Baseline	–	–	31.8	34.9	55.2	46.3	33.3	52.4
last	next-to-last		31.5	35.6	54.9	46.2	29.3	47.6
last	middle	(Ours)	31.6	34.4	54.8	46.7	28.7	50.8
mean	next-to-last		31.7	33.5	54.8	46.9	34.0	50.3
mean	middle		31.5	33.6	54.0	45.9	26.7	45.0

Table A.3: Alignment depth and pooling strategy: fine-grained and OCR-oriented benchmarks.

A.1.2 Teacher Backbone

Full results for the configurations of Table 4.5, at the middle layer.

Teacher	Pooling	POPE	CHAIR	Cover	Hal	Cog	Acc	Prec	Rec	F1	CHAIRs	CHAIRi	Rec	Len
Baseline	–	87.5	9.7	50.0	43.2	5.3	75.7	94.4	67.3	78.6	55.0	16.3	78.3	103.4
DINOv2	last	87.8	9.0	50.8	38.5	4.6	76.3	93.4	69.2	79.5	56.6	16.8	79.1	104.3
DINOv2	mean	88.2	9.6	50.0	40.7	5.0	74.0	93.7	65.2	76.9	59.4	17.8	78.9	102.4
CLIP	last	88.3	9.3	50.7	40.7	5.2	74.3	93.2	66.0	77.3	57.4	17.2	79.4	102.6
CLIP	mean	87.5	9.6	50.6	40.6	5.0	75.6	92.9	68.5	78.9	57.2	17.1	79.0	103.7
SigLIP	mean	87.6	9.6	50.9	41.2	5.0	73.1	93.5	63.9	75.9	57.4	17.4	79.5	104.2

Table A.4: Teacher backbone: hallucination benchmarks.

A.1.3 Region Source

Full results for the configurations of Table 4.6 (mean-token pooling).

Teacher	Pooling	GQA	VizWiz	SciQA	TextVQA	MME	MMB-en	MMB-cn	SEED	MMMU	MathVista	AI2D	ChartQA
Baseline	–	63.4	52.0	69.5	58.1	1480.5	66.0	57.9	67.2	34.8	8.0	55.7	17.5
DINOv2	last	63.1	53.5	69.4	58.1	1480.1	65.7	57.0	67.1	34.9	8.0	55.7	16.7
DINOv2	mean	63.2	52.6	68.8	57.7	1491.8	66.1	57.1	66.6	34.9	7.5	55.4	16.6
CLIP	last	63.2	54.9	68.7	57.5	1478.9	66.7	56.9	67.5	35.2	6.1	54.9	16.8
CLIP	mean	63.4	53.1	69.9	57.8	1492.1	66.4	57.7	67.2	35.2	10.3	55.2	16.8
SigLIP	mean	63.0	55.8	69.8	57.3	1530.8	65.8	56.7	67.0	36.8	7.3	54.7	17.2

Table A.5: Teacher backbone: general visual capability benchmarks.

Teacher	Pooling	OCRBench	MMStar	RealWQA	BLINK	MMVP	V*
Baseline	–	31.8	34.9	55.2	46.3	33.3	52.4
DINOv2	last	31.6	34.4	54.8	46.7	28.7	50.8
DINOv2	mean	31.5	33.6	54.0	45.9	26.7	45.0
CLIP	last	31.1	33.1	53.7	46.3	29.3	47.1
CLIP	mean	31.9	33.5	54.4	46.3	29.3	49.2
SigLIP	mean	31.6	32.7	54.6	43.6	31.3	46.1

Table A.6: Teacher backbone: fine-grained and OCR-oriented benchmarks.

Layer	Region	POPE	CHAIR	Cover	Hal	Cog	Acc	Prec	Rec	F1	CHAIRs	CHAIRi	Rec	Len
next-to-last	crop	87.6	9.5	50.8	41.2	5.2	74.7	93.3	66.7	77.8	57.6	16.9	79.6	103.4
next-to-last	full	87.8	9.7	50.7	41.0	5.0	73.7	93.7	64.6	76.5	54.2	16.7	80.0	102.2
middle	crop	88.2	9.6	50.0	40.7	5.0	74.0	93.7	65.2	76.9	59.4	17.8	78.9	102.4
middle	full	88.0	9.5	50.8	41.0	4.8	74.9	93.7	66.7	77.9	56.4	16.8	78.9	102.1

Table A.7: Region source: hallucination benchmarks.

Layer	Region	GQA	VizWiz	SciQA	TextVQA	MME	MMB-en	MMB-cn	SEED	MMMU	MathVista	AI2D	ChartQA
next-to-last	crop	63.4	53.5	68.8	58.0	1513.7	65.8	58.3	67.4	35.2	7.3	55.4	16.7
next-to-last	full	63.3	54.5	68.5	57.8	1499.3	66.7	57.9	67.0	35.6	7.0	54.8	17.0
middle	crop	63.2	52.6	68.8	57.7	1491.8	66.1	57.1	66.6	34.9	7.5	55.4	16.6
middle	full	63.3	53.4	69.4	57.5	1500.9	67.5	58.8	66.7	35.0	7.0	54.9	17.2

Table A.8: Region source: general visual capability benchmarks.

Layer	Region	OCRBench	MMStar	RealWQA	BLINK	MMVP	V*
next-to-last	crop	31.7	33.5	54.8	46.9	34.0	50.3
next-to-last	full	31.7	33.9	54.0	46.8	32.7	50.8
middle	crop	31.5	33.6	54.0	45.9	26.7	45.0
middle	full	30.8	34.4	54.2	45.8	29.3	49.7

Table A.9: Region source: fine-grained and OCR-oriented benchmarks.

A.1.4 Image-to-Image Alignment

Full results for the image-to-image variant discussed at the end of Section 4.3, compared against the equivalent full-image, middle-layer, text-to-region model.

Objective	POPE	CHAIR	Cover	Hal	Cog	Acc	Prec	Rec	F1	CHAIRs	CHAIRi	Rec	Len
text-to-region	88.0	9.5	50.8	41.0	4.8	74.9	93.7	66.7	77.9	56.4	16.8	78.9	102.1
image-to-image	88.0	9.7	50.8	42.9	5.1	76.0	93.4	68.7	79.2	59.4	17.1	79.1	103.9

Table A.10: Image-to-image alignment: hallucination benchmarks.

Objective	GQA	VizWiz	SciQA	TextVQA	MME	MMB-en	MMB-cn	SEED	MMMU	MathVista	AI2D	ChartQA
text-to-region	63.3	53.4	69.4	57.5	1500.9	67.5	58.8	66.7	35.0	7.0	54.9	17.2
image-to-image	63.5	55.0	69.8	57.1	1488.4	67.3	57.5	67.2	35.0	7.0	56.3	16.8

Table A.11: Image-to-image alignment: general visual capability benchmarks.

A.2 Sequence-Level Layer Distillation: Full Results

This section reports the complete set of benchmark results for the alternative methodology of Section 3.6, complementing the curated results of Table 4.7 (Section 4.6).

Objective	OCRBench	MMStar	RealWQA	BLINK	MMVP	V*
text-to-region	30.8	34.4	54.2	45.8	29.3	49.7
image-to-image	31.1	33.8	56.1	43.5	32.7	48.2

Table A.12: Image-to-image alignment: fine-grained and OCR-oriented benchmarks.

Backbone	Vision	Variant	POPE	CHAIR	Cover	Hal	Cog	Acc	Prec	Rec	F1	CHAIRs	CHAIRi	Rec	Len
Vicuna-7B	CLIP	Baseline	87.2	8.3	51.3	38.0	4.4	74.2	93.1	66.1	77.3	51.4	15.2	77.9	101.7
Vicuna-7B	CLIP	5-Layer	87.0	8.3	52.1	40.3	4.6	75.2	91.9	68.6	78.6	50.6	14.2	78.1	101.2
Vicuna-7B	SigLIP	Baseline	85.9	8.4	51.2	37.4	4.4	72.1	92.8	62.8	74.9	51.6	15.3	76.8	102.4
Vicuna-7B	SigLIP	5-Layer	85.4	8.2	51.5	37.6	4.2	72.4	92.7	63.4	75.3	52.6	15.5	76.1	103.2
Qwen2.5-3B	CLIP	Baseline	87.0	7.6	51.0	34.5	3.6	70.7	94.8	59.0	72.7	49.2	14.0	76.5	100.6
Qwen2.5-3B	CLIP	5-Layer	87.4	7.6	51.3	35.4	4.1	70.6	93.7	59.6	72.9	48.0	13.7	77.4	102.7
Qwen2.5-3B	SigLIP	Baseline	85.9	10.2	50.8	42.8	4.6	67.6	93.8	54.8	69.2	55.6	18.0	74.4	102.9
Qwen2.5-3B	SigLIP	5-Layer	86.5	10.1	51.6	44.1	4.7	68.0	94.4	55.1	69.6	53.2	16.8	75.2	105.3
Qwen2.5-7B	SigLIP	Baseline	85.6	7.0	51.4	31.1	3.5	70.4	93.8	59.3	72.7	49.6	15.1	75.9	100.2
Qwen2.5-7B	SigLIP	5-Layer	85.8	6.8	51.3	30.8	3.5	71.8	94.6	61.0	74.2	48.6	13.4	77.6	99.4
Llama-3-8B	CLIP	Baseline	85.9	9.1	51.1	40.1	4.3	65.5	96.0	50.0	65.7	51.8	15.1	78.2	101.0
Llama-3-8B	CLIP	5-Layer	86.8	8.2	50.6	36.5	3.9	71.2	95.3	59.4	73.2	51.4	14.4	78.2	100.4

Table A.13: Sequence-level layer distillation: hallucination benchmarks.

Backbone	Vision	Variant	GQA	VizWiz	SciQA	TextVQA	MME	MMB-en	MMB-cn	SEED	MMMU	MathVista	AI2D	ChartQA
Vicuna-7B	CLIP	Baseline	62.7	55.5	68.9	57.0	1501.5	64.9	58.4	66.8	36.2	8.5	56.5	18.2
Vicuna-7B	CLIP	5-Layer	63.2	54.5	69.7	57.8	1477.0	66.0	57.9	67.2	34.9	10.1	56.8	18.6
Vicuna-7B	SigLIP	Baseline	63.3	55.4	70.6	57.7	1461.9	65.9	59.6	67.7	35.6	7.5	57.2	15.0
Vicuna-7B	SigLIP	5-Layer	63.4	55.3	69.6	57.9	1461.0	66.2	58.8	67.1	36.3	8.1	57.4	14.8
Qwen2.5-3B	CLIP	Baseline	61.7	53.2	73.8	55.2	1443.2	69.7	67.7	68.2	40.6	6.0	62.1	16.1
Qwen2.5-3B	CLIP	5-Layer	61.8	48.5	74.8	53.8	1488.6	69.6	68.4	68.1	39.7	7.4	62.8	16.4
Qwen2.5-3B	SigLIP	Baseline	59.9	45.4	73.1	47.9	1308.6	60.8	60.3	64.5	40.4	6.3	60.4	11.0
Qwen2.5-3B	SigLIP	5-Layer	59.6	45.6	73.8	48.8	1357.0	63.9	63.8	65.4	39.2	6.6	62.9	11.6
Qwen2.5-7B	SigLIP	Baseline	63.6	59.5	76.9	57.2	1490.4	74.6	74.4	70.0	45.9	7.1	67.6	14.9
Qwen2.5-7B	SigLIP	5-Layer	63.8	55.3	77.7	57.3	1547.5	76.5	74.6	70.9	47.1	7.7	68.0	15.2
Llama-3-8B	CLIP	Baseline	63.3	50.7	73.5	56.1	1472.2	68.9	65.3	68.2	37.1	6.6	60.5	16.3
Llama-3-8B	CLIP	5-Layer	63.7	56.5	73.1	55.7	1490.6	70.2	65.7	69.6	38.2	6.4	59.0	16.4

Table A.14: Sequence-level layer distillation: general visual capability benchmarks.

Backbone	Vision	Variant	OCRBench	MMStar	RealWQA	BLINK	MMVP	V*
Vicuna-7B	CLIP	Baseline	32.1	34.5	54.5	46.9	30.7	48.2
Vicuna-7B	CLIP	5-Layer	31.0	33.2	56.2	47.0	30.0	49.7
Vicuna-7B	SigLIP	Baseline	34.5	34.9	52.0	45.6	31.3	41.9
Vicuna-7B	SigLIP	5-Layer	33.3	36.0	54.1	48.5	34.7	41.9
Qwen2.5-3B	CLIP	Baseline	29.1	40.7	52.7	48.2	25.3	45.5
Qwen2.5-3B	CLIP	5-Layer	31.0	39.5	53.6	46.8	31.3	49.2
Qwen2.5-3B	SigLIP	Baseline	23.5	36.7	48.5	45.8	24.7	39.3
Qwen2.5-3B	SigLIP	5-Layer	27.4	39.1	48.5	46.3	31.3	42.4
Qwen2.5-7B	SigLIP	Baseline	34.3	42.7	56.5	49.2	48.7	46.6
Qwen2.5-7B	SigLIP	5-Layer	33.6	44.1	56.9	51.6	48.0	43.5
Llama-3-8B	CLIP	Baseline	31.4	37.4	57.1	49.3	35.3	51.8
Llama-3-8B	CLIP	5-Layer	31.3	37.3	58.6	49.7	36.0	52.4

Table A.15: Sequence-level layer distillation: fine-grained and OCR-oriented benchmarks.