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)

Mind the Heads: Topological Representation Alignment for Multimodal LLMs

Supervisor:
Prof. Lorenzo Baraldi
Prof. Marcella Cornia
Co-supervisor:
Alberto Compagnoni
Davide Caffagni

Candidate:
Federico Melis
Student ID 199842

ACADEMIC YEAR 2025/2026

Abstract

Mind the Heads: Topological Representation Alignment for Multimodal LLMs

Representation alignment has recently shown an effective strategy to enhance Multimodal Large Language Models (MLLMs) by encouraging their internal activations to follow the representational geometry of strong external vision encoders. Existing approaches, however, usually impose this constraint on a predefined layer of the language backbone, neglecting the finer organization induced by Transformer attention heads. In this thesis, we introduce **He**ad-Wise **R**epresentation **A**lignment (**HeRA**) [9], a training strategy that brings cross-modal alignment to the level of individual attention heads. Grounded in the Platonic Representation Hypothesis, our method targets the preservation of the *topological structure* of visual representations, namely their local neighborhood relations, across modalities. Building on the Mutual K-Nearest Neighbor (MKNN) alignment metric, we design a contrastive objective that provides a differentiable surrogate for matching such local structures. During multimodal training, HeRA applies this objective to selected attention heads of the LLM, chosen according to their MKNN-based alignment score. Surprisingly, we observe that regularizing the *least* aligned heads leads to the greatest improvements. Experiments on multiple MLLMs and 18 benchmarks show that HeRA consistently enhances performance on demanding vision-centric tasks, while also reducing visual hallucinations by limiting the model's tendency to rely excessively on linguistic priors.

Keywords: Multimodal LLMs, Representation Alignment, Multimodal Reasoning, Vision and Language, Large-scale models.

Contents

Abstract **2**

1 Introduction **17**

2 Background **19**

2.1 Deep Learning 19

2.1.1 Supervised Learning 20

2.1.2 Self-Supervised Learning 20

2.2 Attention 21

2.2.1 Scaled Dot-Product Attention 21

2.2.2 Matricial Attention 22

2.2.3 Multi-Head Attention 23

2.2.4 Cross-Attention 25

2.3 Transformer 26

2.3.1 Motivations 26

2.3.2 Architecture 29

2.3.3 Opportunities 32

2.3.4 Limitations 33

2.4 Large Language Models 33

2.4.1 Qwen 34

2.4.2 Llama 35

2.5 Transformers in Vision 36

2.5.1 Vision Transformers 36

2.5.2 CLIP 37

2.5.3 SigLIP 39

2.5.4 DINO 40

2.6 Multimodal Large Language Models 42
2.6.1 LLaVA 43
2.6.2 Multimodal Adapters 45
2.7 Platonic Representation Hypothesis 46
2.7.1 Latent Representation Space 46
2.7.2 Representation Similarity 47
2.7.3 The Platonic Representation Hypothesis 49
2.7.4 Cross-Modal Alignment 49
2.7.5 Mutual K-Nearest Neighbor 51
2.7.6 The Aristotelian Revision 52
2.8 Representation Alignment 53
2.8.1 REPA 54
2.8.2 VIRAL 55
2.8.3 ROSS 57
2.8.4 JARVIS 58
2.8.5 CMAR 59
2.8.6 Discussion 60

3 Method **62**
3.1 Preliminaries 62
3.2 Contrastive Learning Proxy for Representation Alignment 63
3.3 Head-Wise Representation Alignment (HeRA) 65

4 Experiments **68**
4.1 Experimental Settings 68
4.2 Evaluation Benchmarks 70
4.3 Experimental Results 72
4.3.1 Ablation Studies and Analyses 72
4.3.2 Main Experimental Results 73
4.3.3 Varying the Teacher Visual Encoder 77

4.3.4 Additional Ablation Studies and Results 78
4.3.5 Additional Analyses 80
4.3.6 Qualitative Results 83

5 Additional Experiments **87**
5.1 Feature Alignment 87
5.2 Projection Strategies 89
5.3 Bidirectional Attention over Image Tokens 92
5.4 Head Selection and Feature Alignment 93
5.5 Reusing the Projection as a Residual Signal 94
5.6 Patch Importance for Selective Alignment 96
5.7 CKA-Based Alignment 97
5.8 Contrastive Alignment Variants 98
5.9 Neighborhoods Ablation 100
5.10 Discussion 101

6 Conclusion **103**

References **109**

List of Figures

2.1 Self-attention mechanism visualization. One or multiple queries are compared with one or multiple keys to compute attention scores. Then, based on these scores, the values are combined to produce an enriched embedding representation. 22

2.2 Multi-head attention. Each projection is processed independently. The outputs from the different heads are then concatenated and projected back to the d_{model} dimension. 24

2.3 Convolution visualization. The kernel slides over the image, producing each output value as the sum of the input values weighted by the kernel. 26

2.4 Transformer architecture. The overall architecture is composed of two stacks of Transformer blocks. The encoder, shown on the left, encodes the input embeddings, while the decoder, shown on the right, autoregressively produces the outputs. 27

2.5 Qwen family of models. Qwen is provided in numerous versions, including models specialized in math and code. Base, instruction-tuned, and RLHF models are released. Figure from [2]. 34

2.6 LLaMA 2 model. It differs from standard decoder-only LLMs because of the use of pre-normalization with RMSNorm and Grouped Query Attention (GQA). 35

2.7 ViT architecture. The image is split into square patches that, after projection, pass through a Transformer encoder architecture. At the end, the CLS token is fed to an MLP that classifies the image. Figure from [20]. 36

2.8 The figure depicts the CLIP architecture (left), which connects a text encoder and a vision encoder and optimizes the InfoNCE loss. It also shows the CLIP zero-shot classification setup (right): given a set of class sentences, the predicted class is selected as the sentence with the highest similarity to the image. Figure from [61]. 38
2.9 DINO self-distillation framework: two augmented views are processed by a student and an EMA teacher. The teacher output is centered and detached through stop-gradient, while the student is trained to match it via cross-entropy. Figure from [15]. 41
2.10 Attention maps for the CLS token of DINO. The figure shows that, even without direct supervision on the classification task, representations of salient objects emerge. Figure from [15]. 42
2.11 LLaVA architecture. The image is processed by a frozen vision encoder and is then projected with an MLP adapter to the LLM feature space. Figure from [45]. 43
2.12 The figure shows several adapter architectures. MLP (bottom-left) simply projects features. Q-Former (bottom-center) learns feature representations using attention. XAttn (bottom-right) injects features into the model through attention. Figure from [11]. 44
2.13 Visualization of the Platonic Representation Hypothesis. The idea is that both textual descriptions and images describe the same underlying reality. Different models working with different modalities should therefore be able to converge toward a unified representation of reality. Figure from [35]. 48
2.14 Alignment to DINOv2 increases consistently with language-model performance across model families and scales. This suggests that stronger models converge toward a shared, modality-independent representational geometry. Figure from [35]. 50

2.15 Visualization of model convergence under the PRH. As architectures scale up, distinct hypothesis spaces increasingly overlap around a shared low-loss region. Figure from [35]. 52
2.16 REPA operates through the DiT layers by aligning hidden states with features from a pretrained vision encoder (left). This results in faster training convergence (right). Figure from [95]. 55
2.17 The figure shows the VIRAL approach. A layer around the middle of the MLLM is selected before training, and its image hidden states are aligned through feature matching with features from a vision encoder. Figure from [92]. 56
2.18 ROSS proposes several representation-alignment approaches. Its method enforces the hidden state to preserve useful features by using it as conditioning for a denoising process. Figure from [81]. . . . 58
2.19 JARVIS integrates a JEPA objective into MLLM training by predicting masked visual target tokens in latent space. Figure from [10]. 59

3.1 Standard representation alignment enforces strict vision-language feature matching (*left*), whereas HeRA (*center*) aligns cross-modal local neighbors, yielding stronger VQA results (*right*). 63
3.2 **Overview of HeRA.** In addition to the standard language modeling objective ($\mathcal{L}_{\text{LM}}$), HeRA uses a contrastive loss ($\mathcal{L}_{\text{HeRA}}$) to move representations from selected LLM attention heads closer to their k-nearest neighbors (Top-k), computed in the latent space of a frozen teacher vision encoder. 64
3.3 **Left:** Alignment with DINOv2-L, measured with the MKNN metric on each layer and attention head of Qwen2.5-3B. **Right:** MKNN scores of the Worst-5 and Top-5 heads, computed on *(i)* the base LLM; *(ii)* after LLaVA multimodal training; and *(iii)* after adding HeRA. 66

4.1 VQA results of HeRA with different teacher vision encoders. 77
4.2 Effect of LLaVA multimodal training and representation alignment methods on the Worst-5 and Top-5 heads. Worst-5 and Top-5 heads are selected according to the lowest and highest MKNN alignment score with DINOv2-L, computed on the Qwen2.5-3B LLM before multimodal training. 81
4.3 **Left:** Alignment with DINOv2-L, measured using the MKNN metric on each layer and attention head of Qwen2.5-7B. **Right:** MKNN scores of the Worst-5 and Top-5 heads, computed on *(i)* the base LLM; *(ii)* after LLaVA multimodal training; and *(iii)* after applying HeRA. 82
4.4 **Left:** Alignment with DINOv2-L, measured using the MKNN metric on each layer and attention head of Qwen2.5-14B. **Right:** MKNN scores of the Worst-5 and Top-5 heads, computed on *(i)* the base LLM; *(ii)* after LLaVA multimodal training; and *(iii)* after applying HeRA. 83
4.5 Comparison of MKNN scores of the Worst-5 and Top-5 heads after the second training stage performed with HeRA and our competitors (*i.e.*, LLaVA, VIRAL, JARVIS, ROSS, CMAR). 84
4.6 VQA results on Qwen3-VL-4B after fine-tuning, with and without the HeRA objective. 84
4.7 Qualitative comparison of LLaVA [44], ROSS [81], and HeRA using Qwen3-8B and SigLIP2. We show representative examples across all Cambrian categories: General, Knowledge, OCR, and Vision-Centric. 86
4.8 Failure cases of HeRA on VQA tasks. 86

5.1 The figure shows how well spatial features are organized while produced with a vision encoder (top) in comparison with how those features look after a projection phase. The MLP projection (middle) clearly alters the spatial quality of the features; instead, convolution (bottom) preserves well the feature organization. 91

List of Tables

4.1 Ablation study on the choice of *(i)* the objective for representation alignment (feature- vs. contrastive-based), and *(ii)* the granularity of the LLM representation to align (layer- vs. head-level). 69

4.2 Checkpoint reference and list of selected heads for each LLM. 70

4.3 VQA results of HeRA applied to the LLaVA training recipe on different LLMs. 74

4.4 Results of HeRA on visual hallucinations benchmarks. 75

4.5 VQA comparison of different representation alignment strategies for MLLMs. 76

4.6 Detailed VQA results of HeRA applied to the LLaVA training recipe on different LLMs. 79

4.7 Detailed VQA results of different representation alignment strategies for MLLMs. 80

4.8 VQA comparison between DINOv2 and SigLIP2 as the vision encoder for LLaVA. 85

4.9 Effects of $\mathcal{L}_{\text{HeRA}}$ when applied to the different training stages of LLaVA. 85

5.1 Results of feature matching alignment on Qwen2.5-3B with CLIP as the vision encoder. For this set of experiments, the projection from the MLLM to the teacher vision encoder is obtained by interpolating the MLLM features. 88

5.2 Results of feature matching alignment on Qwen2.5-3B with CLIP as the vision encoder. In this configuration, all models include the intermediate Stage 1.5, where only the projection is trained. We report a set of experiments with the target projection frozen during Stage 2, as well as experiments with a learnable projection. 89

5.3 Results of feature matching alignment on Qwen2.5-3B with CLIP as the vision encoder. The target projection is implemented as a multi-convolution module, with a different convolution for each head. "mConv" performs a projection from the MLLM space to the teacher space, while "mConvRev" performs the projection from the teacher space to the MLLM space. 90
5.4 Comparison between causal and bidirectional top-5 alignment, with and without the intermediate stage, on Qwen2.5-3B with CLIP as the vision encoder. 93
5.5 Results of feature matching alignment on Qwen2.5-3B with SigLIP2 as the vision encoder. In these experiments, we vary the teacher vision encoder across most DINO model sizes. We also report an experiment using the 10 most aligned heads. 94
5.6 Results of Gram-based feature matching alignment on Qwen2.5-3B with CLIP as the vision encoder. In the first set of experiments, the Gram matrix is used only as a training signal, since it is discarded at inference time. In the second set of experiments, we modify the inference procedure to inject the learned information through a residual stream. 95
5.7 Results of patch-importance feature matching alignment on Qwen2.5-3B with CLIP as the vision encoder. We report the percentage of used patches (P%). With only 10% of the patches used in both Stage 1 and Stage 2, we achieve performance comparable to the baseline. 96
5.8 Contrastive alignment experiments on Qwen2.5-3B with SigLIP2 as the vision encoder. We compare different alignment configurations, including causal and bidirectional masks, worst- and top-head selection, image- and EOS-token alignment, and cross-modal or unimodal contrastive losses. 99

5.9 Detailed VQA results for different neighborhood choices in Stage 1 and Stage 2. 100

1. Introduction

Recent progress in large language models (LLMs), multimodal large language models (MLLMs), and generative models has notably increased the range of tasks that can be addressed by machine learning systems. In particular, Multimodal Large Language Models (MLLMs) have become one of the most active research directions, because they combine language processing with visual context and thus enable a single architecture to interact with images, text, and other modalities. This integration has led to important results in vision-based tasks.

Despite this huge progress, MLLMs still exhibit weaknesses in visual perception. In particular, these models remain highly dependent on their strong linguistic priors, resulting in being prone to errors when the task strictly requires grounding in the visual context. This limitation is especially visible in tasks such as object counting, spatial reasoning, and OCR. As a consequence, quality improvements cannot rely only on scaling data and model size; it also requires interventions that directly affect how visual and textual representations interact with each other inside the model.

A promising line of research addresses this issue through *representation alignment*. The idea is to use visual features as a regularizer for the language backbone and to explicitly modify how the interaction between texts and images happens inside the model. In this view, the MLLM can be considered as a student model that learns from a vision teacher, transferring useful structural information from the visual domain into the multimodal system. *Representation alignment* has already been explored in generative models, where aligning hidden states with external visual representations can accelerate training convergence and improve the quality of learned features. More recently, the same principle has been extended to MLLMs, with several methods attempting to inject vision-centric supervision into the language backbone.

However, existing approaches are limited by some design choices. First, they typically align a fixed layer of the language model, such as a middle or last layer.

Second, they usually force direct feature matching. These choices may limit the capability and development of MLLMs. In particular, aligning a fixed layer can be regarded as a heuristic, as the optimal choice is highly sensitive to the architecture and internal organization of the model. Moreover, enforcing direct matching with external features can deteriorate the linguistic space, leading to suboptimal results or even being harmful for the overall MLLM capabilities.

This thesis proposes **He**ad-Wise **R**epresentation **A**lignment (**HeRA**) [9], a principled form of representation alignment for MLLMs. The central idea is to move from layer-wise feature matching to head-wise topological alignment. Instead of treating the layer as a monolithic block, HeRA operates at the level of individual attention heads, and then, rather than enforcing pointwise similarity between activations, it focuses on preserving local neighborhood relationships across modalities. In other words, it aims to align the topology of the latent spaces, instead of regressing on values.

2. Background

This chapter introduces the background required to understand the work presented in this thesis. It first reviews the fundamental concepts of deep learning, then discusses the Transformer architecture and its role in the development of Large Language Models. The chapter subsequently introduces Multimodal Large Language Models, with particular attention to LLaVA-style architectures, and concludes by presenting representation alignment methods and the Platonic Representation Hypothesis, which constitute the main research context of this work.

2.1 Deep Learning

Deep learning is a subfield of machine learning based on the use of artificial neural networks with multiple layers. While traditional machine learning methods usually employ manually designed features, deep learning models are characterized by the learning of such features directly from data. This is done within the training loop, while the model updates its parameters. This distinction highlights why deep learning techniques are more suitable for tasks related to computer vision and natural language processing; instead of relying on handcrafted features that intrinsically store human biases and errors, we let the model learn how to express the useful traits of the data. With deep learning, the feature extraction phase, which, with traditional methods, requires designing features, is done entirely in the model. From a theoretical perspective, the expressive power of neural networks is supported by the Universal Approximation Theorem. This theorem states that a neural network with at least one hidden layer and a non-linear activation function can approximate any continuous function with arbitrary precision, provided that the network has a sufficient number of neurons. The Universal Approximation Theorem just provides a theoretical justification for the expressive power of neural networks; later in the background section,

we will discuss how the theory should be combined with architectural choices to actually achieve working systems such as Transformers, Large Language Models, and Multimodal Large Language Models.

2.1.1 Supervised Learning

Supervised learning is one of the main paradigms in machine learning. In this setting, a model is trained using a dataset composed of input-output pairs, where each input is associated with a ground truth. Formally, given a training dataset

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}, \tag{2.1}$$

$x_i \in \mathcal{X}$ denotes an input sample, $y_i \in \mathcal{Y}$ denotes the corresponding target, and N is the number of training examples. The goal is to learn a function $f_\theta : \mathcal{X} \to \mathcal{Y}$ parameterized by θ, that maps samples from the input space to the output space. The parameters of the model are optimized by minimizing a loss function $\mathcal{L}$, which measures the difference between the model prediction $\hat{y}_i = f_\theta(x_i)$ and the target y_i. The training objective is commonly written as:

$$\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\big(\hat{y}_i, y_i\big). \tag{2.2}$$

In the context of deep learning, the function f_θ is usually identified by the neural network, and θ represents its parameters.

2.1.2 Self-Supervised Learning

Self-supervised learning is a learning paradigm in which the supervision signal comes directly from the data, instead of being manually annotated. The training objective is then constructed based solely on data. Common examples of self-supervised tasks are predicting masked tokens or parts of an image, or predicting the next token in a sequence.

This approach shows its strength when dealing with large amounts of unlabeled data. In language modeling, for example, language models usually learn from raw

text by predicting the next token, using the text itself as supervision. Similarly, in computer vision, self-supervised methods can learn visual representations directly from the image, without requiring a classification proxy objective.

2.2 Attention

Attention is the core mechanism of the Transformer architecture [80]. Its main idea is to update the representation of each element in a sequence by weighting the information coming from the other elements. Instead of relying only on local operations or sequential processing, attention allows each token to directly interact with all the others. This makes it possible to model long-range dependencies within a single operation.

2.2.1 Scaled Dot-Product Attention

Scaled dot-product attention is the specific attention operation used in the original Transformer architecture.

Let

$$X = [x_1, x_2, \ldots, x_n] \in \mathbb{R}^{n \times d_{\text{model}}} \tag{2.3}$$

be a sequence of n input token representations, where d_{model} is the hidden dimensionality of the model. In self-attention, each input vector is projected into three different vectors: a query, a key, and a value. These projections are obtained through linear transformations:

$$W^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, \qquad W^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, \qquad W^V \in \mathbb{R}^{d_{\text{model}} \times d_v}. \tag{2.4}$$

If we want to interpret what those projections express in the self-attention operator, we can say that the query vector defines what each token is looking for, the key vector describes what each token exposes, and the value vectors contain the information that will be aggregated to create the updated token.

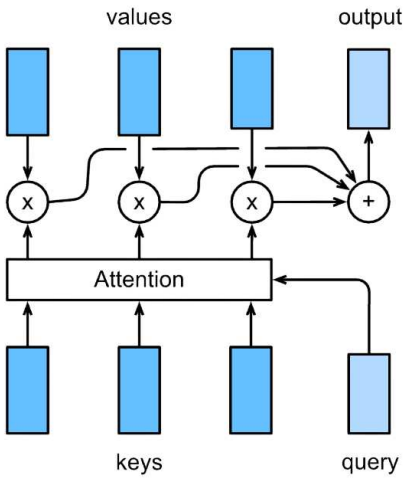

Figure 2.1: Self-attention mechanism visualization. One or multiple queries are compared with one or multiple keys to compute attention scores. Then, based on these scores, the values are combined to produce an enriched embedding representation.

Given a query vector q, a set of key vectors $\{k_i\}_{i=1}^n$, and value vectors $\{v_i\}_{i=1}^n$, the scaled dot-product attention for q is defined as

$$\text{Attention}(q, K, V) = \sum_{i=1}^{n} \alpha_i v_i, \tag{2.5}$$

where the attention weights α_i are computed as

$$\alpha_i = \frac{\exp\left(\frac{qk_i^\top}{\sqrt{d_k}}\right)}{\sum_{l=1}^{n} \exp\left(\frac{qk_l^\top}{\sqrt{d_k}}\right)}. \tag{2.6}$$

The dot product $qk_i^\top$ measures the similarity between the query and the key of token i. The scaling factor $\sqrt{d_k}$ is used to make the softmax operation more stable, avoiding large values. The scaled dot-product attention is depicted in Figure 2.1.

2.2.2 Matricial Attention

In matrix form, the same operation can be written compactly as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V. \tag{2.7}$$

The matrix $QK^\top \in \mathbb{R}^{n\times n}$ contains the pairwise similarities between all queries and all keys. After applying the softmax operation independently to each of the rows, we end up with n probability distributions. The final multiplication by V produces a weighted combination of the value vectors, so that each token representation is updated according to its attention weights.

2.2.3 Multi-Head Attention

Transformers do not use a single attention operation, but instead a multi-head attention mechanism. Instead of computing just one linear projection of the inputs, the model projects those inputs multiple times and applies attention independently to each of them.

Multi-head attention allows the model to attend to different types of relationships in parallel. Different heads can focus on different aspects of the sequence. This property is important in LLMs and MLLMs, where the model may need to encode different aspects of the data at the same time, such as linguistic structure, semantic content, spatial information, and multimodal relations.

For the h-th attention head, the projections are defined as

$$Q_h = XW_h^Q, \qquad K_h = XW_h^K, \qquad V_h = XW_h^V, \tag{2.8}$$

with

$$W_h^Q \in \mathbb{R}^{d_{\text{model}}\times d_k}, \qquad W_h^K \in \mathbb{R}^{d_{\text{model}}\times d_k}, \qquad W_h^V \in \mathbb{R}^{d_{\text{model}}\times d_v}. \tag{2.9}$$

Each head computes an independent attention output:

$$\text{head}_h = \text{Attention}(Q_h, K_h, V_h). \tag{2.10}$$

The outputs of all heads are then concatenated and projected back to the model dimension:

$$\text{MultiHead}(X) = \text{Concat}\left(\text{head}_1, \ldots, \text{head}_H\right) W^O, \tag{2.11}$$

where H is the number of attention heads and

$$W^O \in \mathbb{R}^{H\cdot d_h\times d_{\text{model}}}. \tag{2.12}$$

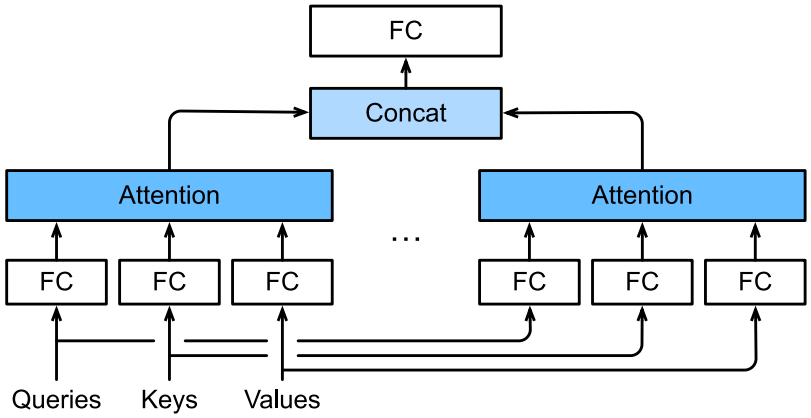

Figure 2.2: Multi-head attention. Each projection is processed independently. The outputs from the different heads are then concatenated and projected back to the d_{model} dimension.

Let the output of the h-th attention head be denoted as

$$\text{head}_h \in \mathbb{R}^{n \times d_h}, \tag{2.13}$$

where n is the sequence length and d_h is the dimensionality of each head output. The standard multi-head attention operation concatenates the outputs of all heads and applies a final output projection:

$$\text{MultiHead}(X) = \text{Concat}\left(\text{head}_1, \ldots, \text{head}_H\right) W^O, \tag{2.14}$$

where

$$W^O \in \mathbb{R}^{H \cdot d_h \times d_{\text{model}}}. \tag{2.15}$$

A visualization of multi-head attention is reported in Figure 2.2. Since the concatenated head representation is multiplied by W^O (*i.e.*, a linear projection), the output projection matrix can be partitioned into H slices:

$$W^O = \begin{bmatrix} W_1^O \\ W_2^O \\ \vdots \\ W_H^O \end{bmatrix}, \qquad W_h^O \in \mathbb{R}^{d_h \times d_{\text{model}}}. \tag{2.16}$$

Therefore, the multi-head attention output can be rewritten as the sum of the projected contributions of the individual heads:

$$\text{MultiHead}(X) = \sum_{h=1}^{H} \text{head}_h W_h^O. \tag{2.17}$$

This decomposition allows us to isolate the contribution of each attention head before all heads are summed into the final multi-head representation. In particular, the projected contribution of the h-th head can be defined as

$$c_h(X) = \text{head}_h W_h^O, \tag{2.18}$$

with

$$c_h(X) \in \mathbb{R}^{n \times d_{\text{model}}}. \tag{2.19}$$

2.2.4 Cross-Attention

Cross-attention can be seen as a variant of self-attention in which queries, keys, and values are not all computed from the same input sequence. In self-attention, the queries, keys, and values are obtained as different projections of the same input. In cross-attention, instead, the queries are produced from one sequence, while keys and values are produced from another source.

Let $X \in \mathbb{R}^{n \times d_x}$ be the sequence that provides the queries, and let $Z \in \mathbb{R}^{m \times d_z}$ be a second sequence that provides the information to be attended to (*i.e.*, the context). In general, these two inputs may come from different modalities or from different stages of processing. For instance, X may be a sequence of textual embeddings, while Z may be a sequence of visual features extracted by a vision encoder.

We can define two functions, $\mathcal{G}$ and $\mathcal{F}$, that produce the two sequences involved in the cross-attention operation. In the simplest case, the query sequence is used directly, so that $\mathcal{G}$ corresponds to the identity mapping, while $\mathcal{F}$ represents the processing function that produces the context sequence:

$$X = \mathcal{G}(X_{\text{in}}) = X_{\text{in}}, \tag{2.20}$$

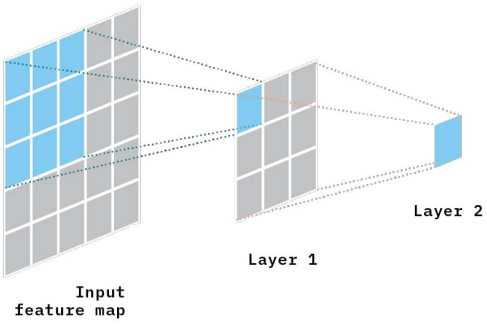

Figure 2.3: Convolution visualization. The kernel slides over the image, producing each output value as the sum of the input values weighted by the kernel.

$$Z = \mathcal{F}(Z_{\text{in}}). \tag{2.21}$$

Once the two sequences X and Z have been obtained, they are mapped through the usual linear projections, and finally, the attention is performed as defined in Equation 2.7.

2.3 Transformer

2.3.1 Motivations

The Transformer, introduced by Vaswani *et al.* [80], and represented in Figure 2.4, marked a major turning point in deep learning, becoming a fundamental building block for modern models in natural language processing, computer vision, and multimodal learning.

Before the rise of Transformers, computer vision was dominated by models based on Convolutional Neural Networks (CNNs) [37, 71, 66, 33]. These architectures process images through convolutional kernels, which can be interpreted as learnable feature extractors with a fixed spatial size. A kernel slides over the input feature map and computes each output feature as a weighted combination of values in a local neighborhood, as shown in Figure 2.3. By stacking multiple convolutional layers, CNNs progressively build hierarchical representations, while the first layers process

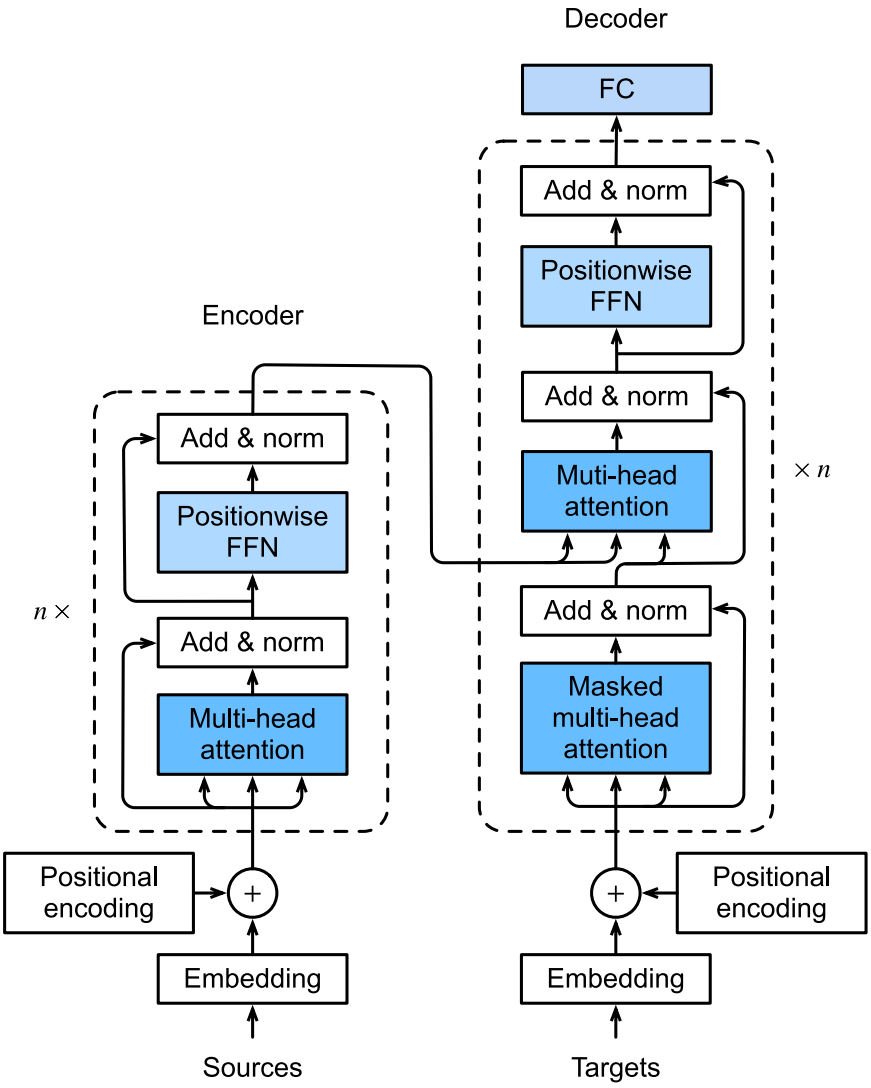

Figure 2.4: Transformer architecture. The overall architecture is composed of two stacks of Transformer blocks. The encoder, shown on the left, encodes the input embeddings, while the decoder, shown on the right, autoregressively produces the outputs.

low-level details and higher layers encode more general information.

This approach achieved outstanding results in several visual tasks, including image classification, object detection, semantic segmentation, and many others [63, 26, 27, 64, 37, 71, 66, 33].

However, CNNs also present some limitations. To understand these limitations well, we have to introduce the concept of receptive field, namely, the region of the input that can influence a given output activation. In a single convolutional layer, the receptive field is determined by the spatial size of the kernel. Larger kernels allow each output feature to aggregate information from a wider neighborhood, but this increases the number of parameters. Several works have tried to address this problem by relaxing the linear dependency through dilated convolutions [93] and by reducing the computational cost by limiting the interactions between input channels and kernel channels.

Transformers address this limitation in a different way. Instead of relying on convolutional kernels, they use scaled dot-product attention to compute similarity between elements of a sequence. In self-attention, each token can directly compare itself with every other token in the input, independently of their relative distance. This breaks the locality imposed by convolution and gives the attention operator an infinite receptive field over the input sequence.

Importantly, increasing the range of interactions in self-attention does not require increasing the number of parameters. The same projection matrices are used to compute queries, keys, and values for all tokens, while the pairwise interactions are obtained through similarity scores between queries and keys. As a consequence, the number of learnable parameters does not grow with the sequence length. This property is one of the main reasons why Transformers became effective backbones not only for language modeling, but also for vision and multimodal tasks.

2.3.2 Architecture

The Transformer can be described, at a high level, as the composition of two main blocks: an encoder and a decoder (see Figure 2.4). In its original application to machine translation, the encoder processes the input sequence to be translated and produces contextualized token representations. The decoder then generates the output sequence autoregressively, predicting each token conditioned on the previously generated tokens and on the representations produced by the encoder.

Nowadays, the applications of Transformers go far beyond machine translation. They are used as general-purpose architectures in language modeling, computer vision, and multimodal learning. For this reason, and to better understand the work presented in this thesis, it is useful to review the structure of the Transformer and its main building blocks.

2.3.2.1 Encoder

As discussed in the introduction, the encoder provides a way to process the information contained in a sequence of tokens, independently of the original modality, as long as the input can be represented as a token sequence. The encoder shares several architectural components with the decoder, but it differs in the way attention is applied. In particular, it is composed of a stack of attention layers, which will be discussed in detail in the following paragraphs.

Before describing the internal structure of the encoder, it is useful to introduce the main preprocessing steps required to transform raw input data into a sequence of vectors that can be processed by the Transformer.

Tokenization. When dealing with text, the input is usually provided as plain text. The first operation is performed by the tokenizer, which converts the text into a sequence of discrete tokens according to a fixed vocabulary. Early approaches represented text using word or character tokenization. Word tokenization is intuitive, but

it leads to very large vocabularies and has difficulties handling rare or unseen words. Character tokenization ensures that all possible combinations of characters can be handled without errors, but trades this robustness for increased sequence length and greater complexity in modeling dependencies between tokens.

Modern Transformers usually rely on subword tokenization methods, such as Byte Pair Encoding (BPE). BPE starts from a vocabulary made of single bytes and iteratively merges the most frequent adjacent bytes (single or groups) to form new tokens. In this way, frequent words can be represented as single tokens, while rare words or symbols remain representable, although using more tokens. This defines a tradeoff between the size of the vocabulary and the length of the resulting token sequence.

Embedding. After the text has been correctly tokenized, it is represented as a sequence of integer indices ranging from 0 to the vocabulary size ($|V|$). These discrete indices must then be converted into continuous vector representations. To do this, we use an embedding matrix of size $|V| \times d_{\text{model}}$. Each token index is used to look up the corresponding row of the embedding matrix, producing the embedding vector associated with that token.

Positional Encoding. Due to the permutation-equivariant nature of self-attention, Transformers do not have an intrinsic notion of token position. If no positional information is provided, the attention mechanism treats the input as a set of tokens rather than as an ordered sequence. To make Transformers suitable for processing ordered sequences, such as text tokens or image patches, we must inject positional information.

In the original Transformer architecture, Vaswani *et al.* [80] proposed to encode positions using fixed sinusoidal functions. Given a position pos and a dimension index i, the positional encoding is defined as

$$\text{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right), \tag{2.22}$$

$$\mathrm{PE}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right). \tag{2.23}$$

The resulting positional vector is added to the token embedding, so that the model receives information about both the content of the token and its position in the sequence.

Modern Transformer models use alternative strategies. One common approach is to learn positional embeddings while training [25]. In this case, to each position corresponds a trainable vector. Another common method is Rotary Positional Embedding (RoPE) [69]. Instead of adding a positional vector to the token embedding, RoPE injects positional information by applying a rotation based on the position to the query and key vectors. RoPE is commonly employed in LLMs.

Transformer Block. After the preliminary steps, the input sequence is processed by a stack of Transformer blocks. The first component of each block is a multi-head self-attention layer, where each token attends to each other token.

The attention output is combined with the input through a residual connection and then normalized using layer normalization:

$$\tilde{X} = \mathrm{LayerNorm}(X + \mathrm{MHA}(X)). \tag{2.24}$$

Layer normalization normalizes each token representation by its feature dimension. The normalized values are then scaled and shifted through learnable parameters α and β.

The second component is a position-wise feed-forward network (FFN), applied independently to each token of the sequence. It is composed of two fully connected layers with an activation function in between. The first layer enlarges the dimensionality of the representation, while the second brings it back to the original model dimension. Thanks to the non-linearity, we can increase the expressiveness of the block.

Finally, another residual connection and layer normalization are applied:

$$Y = \mathrm{LayerNorm}(\tilde{X} + \mathrm{FFN}(\tilde{X})). \tag{2.25}$$

The output Y is then passed to the next Transformer block.

2.3.2.2 Decoder

The decoder is the part of the Transformer that generates the output sequence. It relies on the representations produced by the encoder to condition the generation process. The decoder shares most of its components with the encoder, but differs in the way attention is applied.

Masked Multi-Head Self-Attention. The first attention layer in the decoder is a masked multi-head self-attention layer. Unlike the encoder, the decoder must preserve the autoregressive nature of generation. This means that, when predicting a token at position t, the model can only attend to tokens at positions smaller than or equal to t.

To mask some part of the attention, before applying the softmax, a very large negative value is added to the attention scores corresponding to the future. After the softmax, the masked positions receive approximately zero attention weight.

After masked self-attention, the decoder performs cross-attention over the encoder outputs. In this layer, the queries come from the decoder representations, while keys and values come from the encoder representations. This is a smart way of conditioning the generation of the decoder on the information extracted from the encoder.

2.3.3 Opportunities

The Transformer architecture provides an effective framework for modeling language as a sequence of textual tokens. Before Transformers, several approaches had been explored in Natural Language Processing (NLP), including recurrent sequence-to-sequence models [70]. However, Transformers removed the need for sequential recurrence by relying on self-attention, allowing tokens to interact directly with one another and making long-range dependencies easier to model in parallel. This property has been crucial for the development of Large Language Models, enabling effective scaling in terms of data, parameters, and computation.

Beyond language modeling, Transformers have become a general backbone for

foundation models across different modalities. Since text, images, retrieved documents, and visual features can all be represented as sequences of tokens or latent embeddings, the same architectural principles can be adapted to many settings. This has enabled research on diffusion-based generation and diffusion language modeling [19, 8, 79, 58, 6], retrieval-augmented generation systems [18, 12], and several applications of LLMs and MLLMs involving visual reasoning, evidence grounding, vector-graphics understanding, and hallucination mitigation [101, 53, 14, 17]. These directions show how the Transformer has evolved from an architecture for sequence modeling into a general framework for integrating heterogeneous sources of information within shared representational spaces.

2.3.4 Limitations

Despite their success, Transformers come with some limitations. First, in decoder-only models, generation is autoregressive: each token is generated conditioned on the previous ones, and the next step cannot be computed before the current token has been produced. This makes decoding inherently sequential and hard to parallelize. Several works try to mitigate this issue [13], but it remains an open problem.

A second limitation is the cost of self-attention. Given a sequence of length n, the attention matrix has size $n \times n$, since each token is compared with every other token. Therefore, memory and computational costs scale quadratically with the sequence length. Alternative architectures, such as Mamba [30] and LCM [5], aim to reduce or avoid this quadratic dependency.

2.4 Large Language Models

Large Language Models are systems based on the transformer architecture and allow models to deal with a sequence of textual tokens. Their objective is to model the probability distribution of language, usually by predicting the next token given the previous ones. These models are usually developed as decoder-only models, but

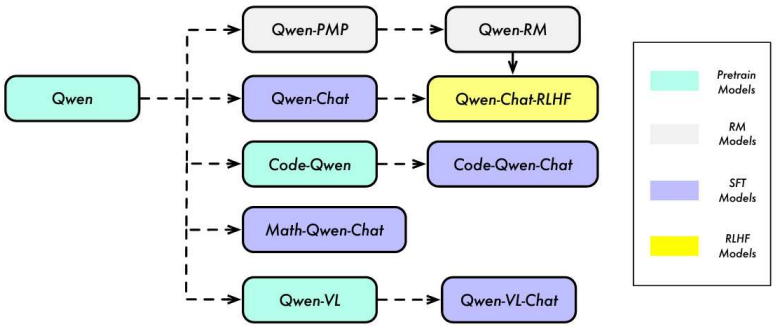

Figure 2.5: Qwen family of models. Qwen is provided in numerous versions, including models specialized in math and code. Base, instruction-tuned, and RLHF models are released. Figure from [2].

also exist in encoder-only and encoder-decoder versions. Before the LLMs, text was modeled with rule-based techniques that struggled in capturing the dependency of tokens.

A key aspect of LLMs is their pretraining procedure. They are commonly trained on large collections of raw text using self-supervised objectives, where the supervision signal is obtained directly from the text itself. Through this process, the model learns the structure of the language and the relations between tokens. After pretraining, LLMs can be adapted to specific tasks or instruction-following behavior through supervised fine-tuning and related alignment techniques.

2.4.1 Qwen

Qwen [59, 60, 4, 91, 90, 89, 2, 3] is a family of Large Language Models developed by Alibaba. Qwen models are decoder-only Transformer architectures and are released in different sizes. Qwen's family, depicted in Figure 2.5, is particularly suitable for multilingual applications, and it also provides models specialized in math and code. The Qwen family is particularly relevant for recent research because it provides open-weight models across multiple scales.

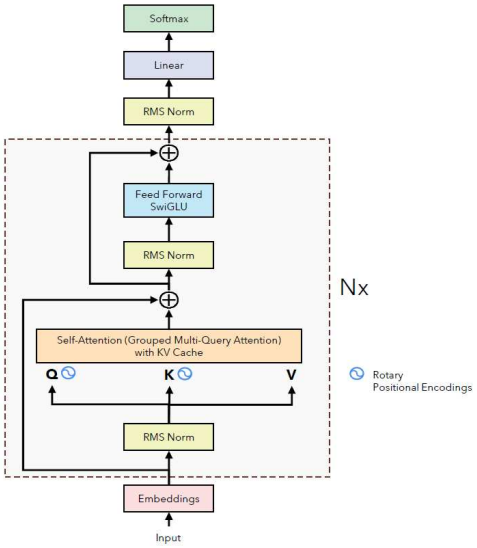

Figure 2.6: LLaMA 2 model. It differs from standard decoder-only LLMs because of the use of pre-normalization with RMSNorm and Grouped Query Attention (GQA).

.

2.4.2 Llama

Llama [76, 75, 28] is a family of open-weight Large Language Models developed by Meta. These models are based on the decoder-only Transformer architecture and are designed as general-purpose language models. Recent versions, such as Llama 3, have been trained to support a broad set of capabilities, including multilingual understanding, coding, reasoning, and tool use.

The relevance of Llama models comes not only from their performance, but also from their availability as open-weight models. This makes them widely used in research, where they can be analyzed, fine-tuned, and integrated into larger systems. In the context of Multimodal Large Language Models, Llama models are often used as language backbones and connected to visual encoders through projection modules or multimodal adapters.

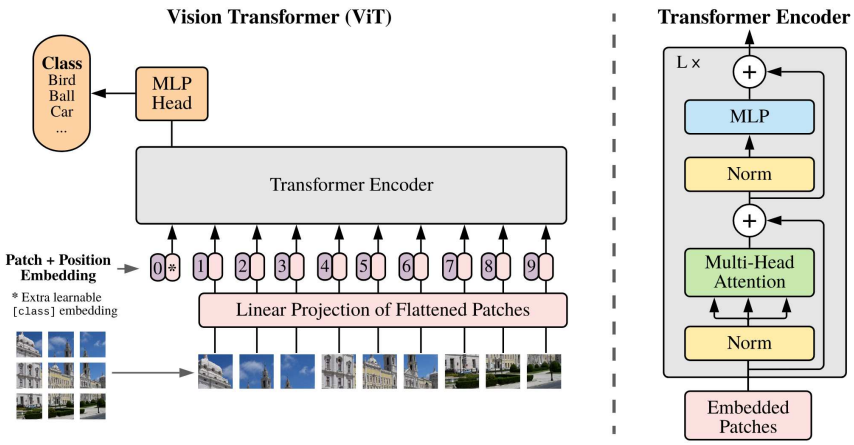

Figure 2.7: ViT architecture. The image is split into square patches that, after projection, pass through a Transformer encoder architecture. At the end, the CLS token is fed to an MLP that classifies the image. Figure from [20].

2.5 Transformers in Vision

While the effectiveness of Transformers was first demonstrated in natural language processing, their adoption in computer vision was more gradual. For several years, vision systems were dominated by Convolutional Neural Networks (CNNs).

The application of Transformers to vision required a change in perspective: instead of processing an image as a regular grid with convolutional kernels, the image can be represented as a sequence of visual tokens. This makes it possible to use the same attention-based approach, originally developed for text, allowing the model to capture interactions between distant regions of the image.

2.5.1 Vision Transformers

In 2020, Dosovitskiy *et al.* [20] showed that a Transformer encoder, when properly trained, can be used directly for image classification. The resulting architecture, called Vision Transformer (ViT) [20], adapts the standard Transformer encoder to pure visual inputs.

The idea is simple. Given an input image, ViT splits it into fixed-size square patches, for example 16×16 pixels. Each patch is flattened and projected through a linear layer into a vector with the same dimensionality used by the Transformer. In this way, the image is converted into a sequence of patch embeddings, which represent a direct parallelism with the field of language processing.

We then have to add the positional encoding because, as with the tokens, the Transformer has no information on the position of the tokens, and we are interested in expressing the position of the patches to allow the model to handle spatial relations. ViT also introduces a learnable classification token, called CLS token, which is prepended to the sequence. After the sequence is processed by the Transformer encoder, the final representation of the CLS token is passed to a classification head to predict the image class.

ViT opened the way to the use of Transformer-based architectures in computer vision [52] and later in multimodal models [9], where visual inputs are naturally represented as sequences of tokens. The entire architecture is represented in Figure 2.7.

2.5.2 CLIP

Before the emergence of large-scale vision-language models, visual representation learning was mostly based on supervised classification. In this setting, an image is associated with a single label chosen from a fixed set of classes. Although this approach has been extremely successful, it also presents important limitations. A single class label often cannot describe the full semantic content of an image. For example, an image may contain multiple objects, attributes, actions, and relations that cannot be expressed by assigning only one category. Moreover, supervised classification requires manually annotated datasets, whose construction is expensive and difficult to scale.

CLIP, introduced by Radford *et al.* [61], and represented in Figure 2.8, addresses this limitation by learning visual representations from natural language supervision. Instead of training on strict image-class pairs, CLIP is trained on image-text pairs

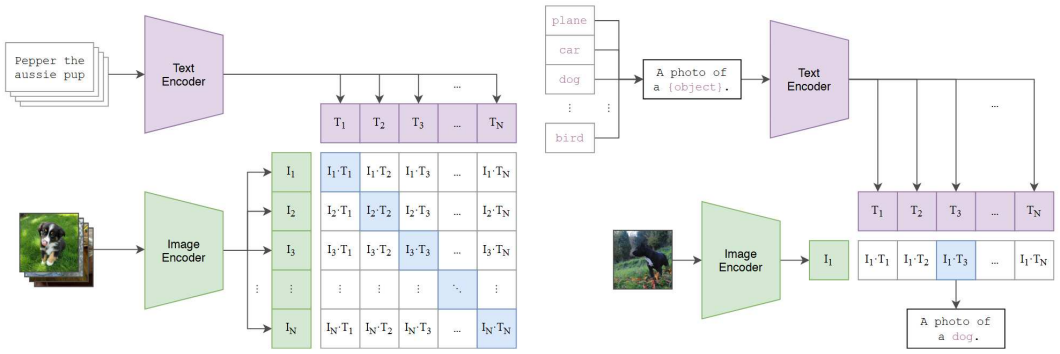

Figure 2.8: The figure depicts the CLIP architecture (left), which connects a text encoder and a vision encoder and optimizes the InfoNCE loss. It also shows the CLIP zero-shot classification setup (right): given a set of class sentences, the predicted class is selected as the sentence with the highest similarity to the image. Figure from [61].

collected at a large scale from the web. The key idea is that a natural language description can provide richer and more flexible supervision than a class label. In this way, images are not forced into a predefined set of categories, but are instead associated with textual descriptions that can express more detailed semantic information.

CLIP is composed of two encoders: an image encoder and a text encoder. Given an image I and a text description T, the image encoder maps the image into a vector representation, while the text encoder maps the text into the same embedding space:

$$f_\theta(I) \in \mathbb{R}^d, \qquad g_\phi(T) \in \mathbb{R}^d. \tag{2.26}$$

The model is trained so that the representation of an image is close to the representation of its corresponding text, and far from the representations of negative text examples. The similarity between image and text embeddings is computed with the cosine similarity:

$$\text{sim}(I,T) = \frac{f_\theta(I)^\top g_\phi(T)}{\|f_\theta(I)\| \, \|g_\phi(T)\|}. \tag{2.27}$$

During training, a large batch of N image-text pairs ($32,768$ in the paper's experiments) is processed by the two encoders. This produces N image embeddings and N text embeddings. The model then computes all pairwise similarities between

images and texts, obtaining a similarity matrix of size $N \times N$. The diagonal elements correspond to matching image-text pairs, while all the other elements correspond to negative pairs. CLIP is then trained with cross-entropy, maximizing for each element the similarity with the corresponding vector in the other modality while minimizing the similarity of the negative examples.

This training strategy maps images and texts into a shared semantic embedding space. As a result, CLIP can be used not only for representation learning, but also for zero-shot classification. Given a set of possible classes, each class name is converted into a textual prompt, such as "a photo of a {class}". The text encoder embeds all class prompts, while the image encoder embeds the input image. The predicted class is the one whose text embedding has the highest similarity with the image embedding.

From an architectural point of view, CLIP allows different choices for the image encoder. In the paper, the authors present two versions of CLIP, one with a ResNet [33] and one with ViT [20]. In the case of ViT, the image representation is usually obtained from the final representation of the CLS token.

Despite its effectiveness, CLIP also has limitations. Since the training data are collected from the web, image-text pairs can be noisy. Nevertheless, CLIP has become a fundamental component in modern vision-language systems and is widely used as a visual encoder or representation backbone in MLLMs.

2.5.3 SigLIP

In CLIP, each image has one positive text and $N-1$ negative texts within a batch of size N. Therefore, the loss depends on the full similarity matrix and requires comparing each sample with all the others. This makes training sensitive to the batch size and requires large batches and expensive distributed gathering operations.

SigLIP, Sigmoid Loss for Language-Image Pre-training [77], is a vision-language model that builds on the same general idea as CLIP and tries to address the aforementioned problem. The main difference lies in the training objective. While CLIP uses a softmax-based contrastive loss over all image-text pairs in a batch, SigLIP replaces

this objective with a pairwise sigmoid loss.

SigLIP addresses this issue by formulating image-text matching as a binary classification problem. Each image-text pair is evaluated independently: matching pairs are treated as positives, while non-matching pairs are treated as negatives. Given normalized image embeddings x_i and text embeddings y_j, the model computes a score

$$s_{ij} = \alpha x_i^\top y_j + b, \tag{2.28}$$

where α is a learnable scale parameter and b is a learnable bias. The label z_{ij} is set to 1 if the image and text form a positive pair, and to -1 otherwise. The sigmoid loss can then be written as

$$\mathcal{L} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \log\left(1 + \exp(-z_{ij} s_{ij})\right). \tag{2.29}$$

This objective encourages positive image-text pairs to have high similarity and negative pairs to have low similarity, without requiring a softmax normalization over the entire batch. As a result, SigLIP breaks the dependency between the loss and the global batch structure, making the training process much lighter.

2.5.4 DINO

DINO, [57, 65, 15] is a family self-supervised visual backbone that learns visual features without annotated data. The idea is to train a model to produce consistent representations for different augmented views of the same image. Instead of relying on class labels or explicit negative samples, DINO adopts a teacher-student paradigm.

In the first version of DINO [15], both teacher and student are implemented as ViTs. During training, different augmentations of the same image are given to the two networks. The student is trained to match the output distribution produced by the teacher.

A central element of DINO is the way the teacher is updated. The teacher is not trained directly through backpropagation. Instead, its parameters are updated as an

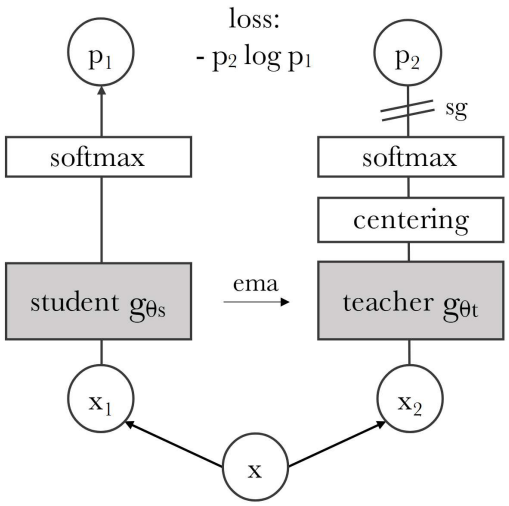

Figure 2.9: DINO self-distillation framework: two augmented views are processed by a student and an EMA teacher. The teacher output is centered and detached through stop-gradient, while the student is trained to match it via cross-entropy. Figure from [15].

exponential moving average (EMA) of the student parameters:

$$\theta_{\text{teacher}} \leftarrow m\theta_{\text{teacher}} + (1-m)\theta_{\text{student}}, \tag{2.30}$$

where m is a momentum coefficient close to 1. This means that the teacher updates its parameters more slowly than the student. The EMA update provides stable targets during training: the student learns from the teacher, while the teacher is continuously updated from a smoothed version of the student. This limits the magnitude of the update of the targets, stabilizing the training procedure. A schematic representation of DINO can be found in Figure 2.9.

DINOv2 [57] shared the same idea of DINO, but improves the training pipeline. In particular, DINOv2 improves the self-supervised objective inspired by masked image modeling, adding a prediction at patch level. At the image level, the student learns to match the global representation produced by the EMA teacher. At the patch level, the model is trained to recover information about masked regions using teacher features as targets. This additional objective helps in developing localized features. In figure

Figure 2.10: Attention maps for the CLS token of DINO. The figure shows that, even without direct supervision on the classification task, representations of salient objects emerge. Figure from [15].

Figure 2.10, we reported some examples of how those features organize together.

2.6 Multimodal Large Language Models

Multimodal Large Language Models (MLLMs) usually extend pretrained Large Language Models with support for processing inputs from multiple modalities, *e.g.*text and images.

Earlier attempts usually trained parts of the architecture from scratch. For instance, SimVLM [85] uses a CNN to process images and trains the language model from scratch. CoCa [94] trains an image encoder, a text decoder, and a multimodal decoder, optimizing the model with both cross-entropy and contrastive objectives. FLAVA [67] trains two unimodal encoders from scratch using masked image modeling and masked language modeling, and then bridges the two modalities through a multimodal encoder. Frozen [78] is among the first works to exploit a pretrained LLM for multimodal learning, while training from scratch a vision encoder that processes the images before feeding visual information to the language model.

Recent methods, instead, try to limit the trained from scratch components. The general idea is to combine a pretrained vision encoder, which extracts visual features from an image, with a pretrained LLM, which performs reasoning and text generation. Since visual features and textual embeddings usually belong to different representation

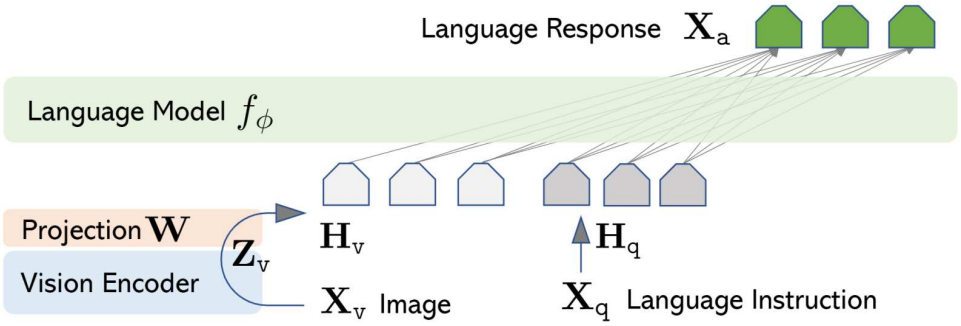

Figure 2.11: LLaVA architecture. The image is processed by a frozen vision encoder and is then projected with an MLP adapter to the LLM feature space. Figure from [45].

spaces, MLLMs require an intermediate module, called adapter or a projector, to map visual representations into the LLM representation space.

This formulation enables the language model to ground its generation in visual content, rather than relying just on textual context. As a result, MLLMs can be applied to tasks that require joint reasoning over language and images, such as visual question answering, image captioning, OCR-based reasoning, chart understanding, spatial reasoning, and multimodal instruction following.

2.6.1 LLaVA

LLaVA is one of the most influential works in the development of modern MLLMs [45, 44]. Its importance lies not only in the architecture itself, but also in the training recipe, which has been adopted by a big slice of works. LLaVA connects a pretrained vision encoder to a pretrained LLM through a simple projection module, an MLP. The role of this projector is to map the visual features produced by the image encoder into the input embedding space of the LLM. This process can be visualized in Figure 2.11.

The training procedure is composed of two main stages. The first stage performs feature alignment between the vision encoder and the LLM. In this stage, both the vision encoder and the LLM are kept frozen, and only the adapter is trained. The training data are image-caption pairs, and the objective is to make the projected visual

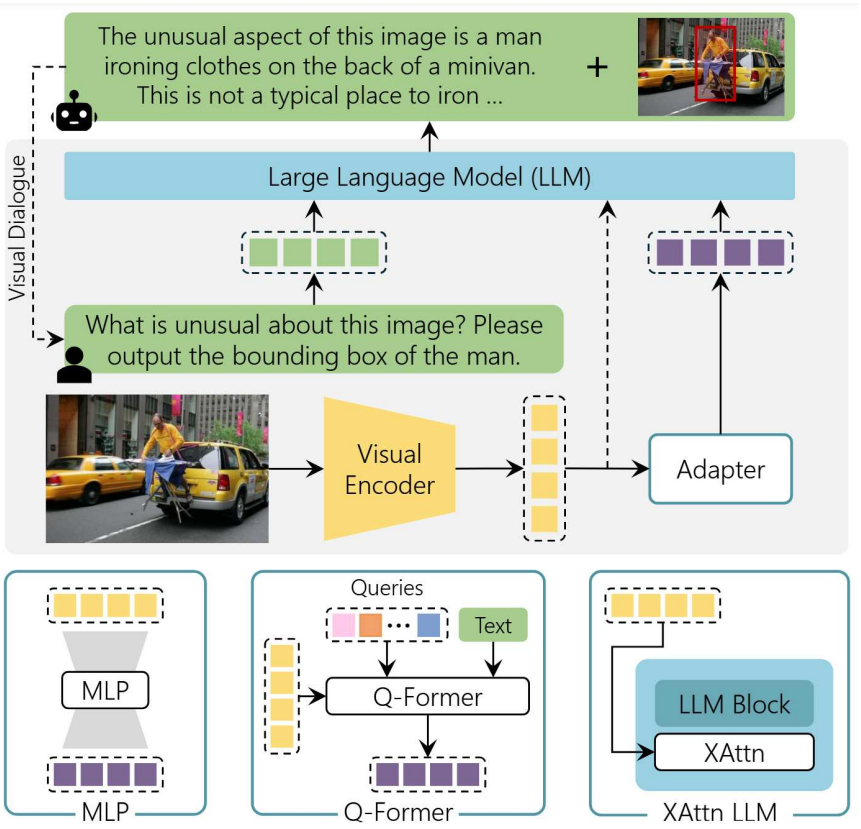

Figure 2.12: The figure shows several adapter architectures. MLP (bottom-left) simply projects features. Q-Former (bottom-center) learns feature representations using attention. XAttn (bottom-right) injects features into the model through attention. Figure from [11].

features compatible with the textual embedding space of the LLM.

The second stage is visual instruction tuning. In this phase, the model is trained on multimodal instruction-following data, where instructions usually require the visual content. The vision encoder is still kept frozen, while the projector and the LLM are trainable. This stage teaches the model to use visual information when answering natural language instructions, extending the instruction-following capabilities of the LLM to the multimodal domain.

This two-stage recipe is simple but effective. The first stage aligns visual and textual representations, while the second stage teaches the model how to use those

representations in conversational and instruction-following settings. For this reason, LLaVA-style training has become a common baseline for many MLLMs.

2.6.2 Multimodal Adapters

A central component of MLLMs is the adapter that connects the vision encoder to the LLM. Different models use different adapter designs, depending on specific needs. The adapter determines how visual information is transformed before being injected into the LLM. A comprehensive listing of multimodal adapters can be found in Figure 2.12.

2.6.2.1 MLP Projector

The simplest, and also the most effective, adapter is the MLP projector, as used in LLaVA-style architectures [45]. Given the visual tokens extracted by the image encoder, the MLP maps each of these into the embedding dimension of the LLM. The projected visual tokens are then concatenated with the textual tokens and processed by the LLM as part of the same input sequence. In this setting, we turn the visual tokens into a representation that the LLM can easily understand. This fact is supported by some mechanistic interpretability studies, where the authors analyze how the visual tokens are mapped in token space by the LM head.

This solution is computationally simple and easy to train. However, it assumes that a relatively small projection module is sufficient to bridge the gap between the visual representation space and the language model embedding space. Despite this simplicity, MLP projectors have proven effective across different MLLM architectures.

2.6.2.2 Q-Former

The Q-Former, introduced in BLIP-2 [41, 40], is more complicated than MLP. Instead of directly projecting all visual tokens into the LLM space, the Q-Former uses a set of learnable query tokens that attend to the visual features extracted by a frozen vision

encoder. The learnable queries (as in a perceiver) interact with the visual tokens through cross-attention.

The advantage of this design is that the number of visual tokens passed to the language model can be controlled with the number of learnable queries. This makes the adapter suitable for situations in which we do not have a fixed number of visual tokens (*e.g.*, processing of video).

2.6.2.3 Gated Cross-Attention

Another important strategy is gated cross-attention, introduced in Flamingo [1]. Instead of converting visual features and concatenate it with text, Flamingo introduces cross-attention layers inside the LLM. This is a more complex level of interaction, while visual conditioning and text interact directly within the model.

The cross-attention is controlled by gating mechanisms, which regulate how much visual information is injected into the LLM. This is useful when adapting a pre-trained LLM, since the model can incorporate visual information without completely disrupting its original linguistic capabilities.

2.7 Platonic Representation Hypothesis

2.7.1 Latent Representation Space

Modern models not only produce task-specific predictions but also build latent representations of the input data. A representation can be defined as an embedding function $f : \mathcal{X} \to \mathbb{R}^d$ that maps an input $x \in \mathcal{X}$ to a vector $f(x)$ in a latent space. The geometry of this space is not arbitrary: distances, similarities, and neighborhood relations between embedded samples encode information about how the model organizes the data and on the data itself. In this sense, a representation defines an implicit structure of the input domain, where samples that share geometrical properties are expected to share semantic or perceptual properties in the input domain.

This view is particularly interesting for vision and vision-language models. Contrastive models, such as CLIP, learn an image-text shared space, while self-supervised vision models such as DINO learn visual representations whose structure reflects semantic regularities in images. Even if these models are trained with different objectives, they can still be compared through the geometry of their embeddings. To compare how those different models represent data points, we do not pose constraint like the hidden dimension of the training strategy. What matters is the relational structure that each model imposes over a common set of data points. This observation provides the basis for studying representation alignment: two models are considered aligned when they organize the same data in similar ways within their respective latent spaces.

2.7.2 Representation Similarity

Representations can be interpreted as geometric objects with characterizing properties. Now we may be interested in how to quantify and measure their similarity. A first type of method evaluates global alignment, namely, whether two representation spaces share a similar overall geometry. Metrics such as CKA and SVCCA belong to this family and compare statistical properties of the representations. These measures are useful when the analysis is about the representation space as a whole.

However, based on the needs and on the focus of the analysis, there exists a set of different techniques to express representation similarity. A second family of methods focuses on local alignment, which asks if the same samples are close to each other across two representation spaces. Local metrics do not compare global features but focus on the geometrical relations of a limited neighborhood. This distinction is important because two models may represent the same semantic structure without preserving exact distances and directions. For instance, a vision encoder and an LLM may embed data with different dimensionalities and different coordinate systems, while still sharing the same idea on which samples are semantically nearest to one another.

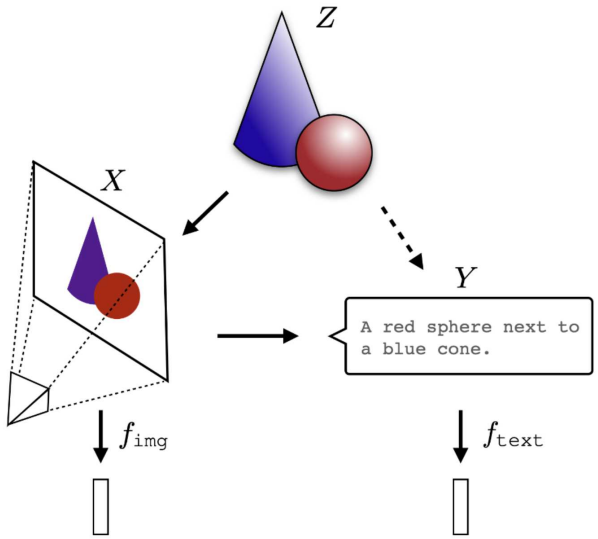

Figure 2.13: Visualization of the Platonic Representation Hypothesis. The idea is that both textual descriptions and images describe the same underlying reality. Different models working with different modalities should therefore be able to converge toward a unified representation of reality. Figure from [35].

The Platonic Representation Hypothesis is defined for the similarity structure of latent representations and not in terms of pointwise feature correspondence. In this setting, a representation is associated with a kernel or a similarity function that quantifies the level of agreement between two latent spaces. Huh *et al.*, study representational convergence by comparing latent structures and use mutual nearest-neighbor (MKNN) alignment as a metric for evaluating how much different models agree on the organization of data [35]. This raises a question: if representations can be compared through the relations they create between data points, can the same perspective be extended beyond the training objective and the main operative modality, and used to study whether different modalities tend to organize information in structurally similar ways?

2.7.3 The Platonic Representation Hypothesis

The Platonic Representation Hypothesis (PRH) states that *neural networks trained with different architectures, objectives, datasets, and modalities may converge toward structurally similar representations*. The hypothesis is motivated by the observation that increasingly capable models appear to organize data in similar ways, even when they are trained differently and on different modalities. Huh *et al.*, interpret this convergence as the tendency of models to approximate a shared model of reality: if different modalities are generated by the same underlying world, then sufficiently powerful models may recover compatible latent structures [35]. An helpful visualization of the PRH is reported in Figure 2.13.

The empirical evidence presented in the Platonic Representation Hypothesis paper supports this claim along two main directions. First, the authors report that models within the same modality become more aligned as their performance increases. Stronger vision models tend to construct more similar representation structures than weaker ones, suggesting that improved performance is associated with convergence toward an organization of the underlying data. Second, the paper extends this observation across modalities. By comparing image representations from vision models and text representations from language models on paired image-text data, the authors show that larger and more powerful models tend to induce increasingly similar representation structures [35].

2.7.4 Cross-Modal Alignment

The cross-modal setting makes the distinction between global and local alignment particularly important. Images and texts are different observations of the same underlying content, but they do not expose exactly the same information. A visual encoder may organize samples according to perceptual similarity, object configuration, or visual texture, while a language model may organize textual descriptions according to semantic and linguistic relations. Therefore, expecting the two modalities to share a

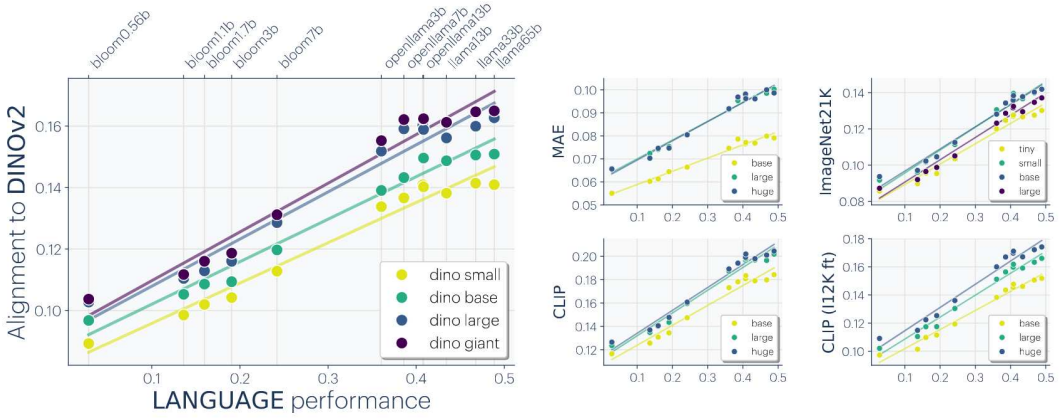

Figure 2.14: Alignment to DINOv2 increases consistently with language-model performance across model families and scales. This suggests that stronger models converge toward a shared, modality-independent representational geometry. Figure from [35].

single coordinate system is too restrictive.

A more appropriate requirement is that semantically corresponding samples preserve their local relations (extracted from their modality-specific space). Given a set of paired images and texts, an image representation and a text representation may be considered aligned if the neighbors of a sample in the visual space correspond to the neighbors of its paired textual description in the language space. For example, if an image of a dog near a car has visual neighbors that contain similar scenes, the corresponding text should be close to descriptions of similar scenes in the textual space. The absolute distances may differ, but the local organization of the data remains compatible. This kind of approach compares the logical organization of the space rather than magnitude, orientation, or channel values.

This form of alignment is topological rather than purely metric. It concerns the preservation of neighborhood structure, namely, which samples are near each other, instead of the exact numerical value of their distances. In the context of multimodal representation learning, this is natural because it allows different modalities to maintain their own local geometry while still sharing a semantic structure. The Platonic

Representation Hypothesis provides the motivation for expecting such cross-modal structure to emerge, while local neighborhood comparison provides a concrete way to measure it. They report in Figure 2.14 how the alignment levels scale with model dimensions and datasets, and an explanation in loss space in Figure 2.15.

2.7.5 Mutual K-Nearest Neighbor

A standard way to formalize local topological alignment is through Mutual K-Nearest Neighbor alignment. Let $D = \{x_i\}_{i=1}^{N}$ be a dataset of samples and let $f : \mathcal{X} \to \mathbb{R}^{d_f}$ be a representation function. For each sample x_i, the k-nearest neighbor set induced by f is denoted as $\mathcal{N}_k^f(i)$ and contains the indices of the k samples most similar to x_i in the representation space of f. Using dot-product similarity, this set can be written as

$$\mathcal{N}_k^f(x_i) = \operatorname{argmax}_{j \neq i}^{(k)} f(x_i)^\top f(x_j). \tag{2.31}$$

Given two representation functions f and g defined over corresponding samples, MKNN measures how much their local neighborhood structures agree. The alignment score is computed as the average normalized overlap between the neighbor sets induced by the two representations:

$$\text{MKNN}(f, g, D) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{k} \left| \mathcal{N}_k^f(i) \cap \mathcal{N}_k^g(i) \right|. \tag{2.32}$$

The score lies in $[0, 1]$. A value close to 0 indicates that the two representations induce largely different local neighborhoods, while a value close to 1 indicates that they preserve similar neighbor relations. In the multimodal case, f and g can correspond to representations extracted from different modalities, for instance, a visual encoder and a language model. MKNN is therefore well-suited to measure whether the local organization learned in one modality is preserved in another.

This metric is important for the method developed later in this thesis because it provides a quantitative way to assess local topological agreement between visual teacher representations and internal representations of a multimodal model. At this

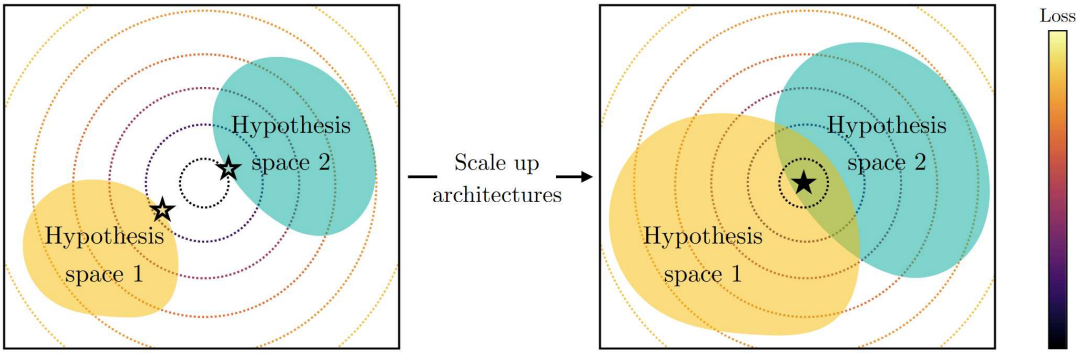

Figure 2.15: Visualization of model convergence under the PRH. As architectures scale up, distinct hypothesis spaces increasingly overlap around a shared low-loss region. Figure from [35].

stage, however, MKNN should be understood only as a background tool for measuring alignment. The training objective that uses this intuition is introduced separately in the Method chapter.

2.7.6 The Aristotelian Revision

A more recent line of work revisits the Platonic Representation Hypothesis by questioning whether all observed forms of representational convergence are equally reliable. The Aristotelian Representation Hypothesis paper argues that some similarity metrics used to support the Platonic view can be confounded by model scale [29]. In particular, increasing model width or depth may systematically inflate representation similarity scores, even when the apparent agreement is partly explained by statistical artifacts rather than meaningful convergence. This critique does not reject the general idea that representations may share structure, but it requires a more careful interpretation of what kind of structure is actually preserved.

To address these confounders, the Aristotelian paper introduces a permutation-based null calibration framework. The purpose of this calibration is to compare an observed similarity score against a null distribution obtained by breaking the correspondence between samples. This allows similarity measures to be interpreted relative

to what would be expected by chance under the same dimensionality, sample size, and comparison procedure. As a result, the calibrated score provides a more reliable estimate of whether the observed alignment reflects genuine shared structure [29].

After applying this calibration, the paper reports a more nuanced picture. Evidence for global spectral convergence becomes substantially weaker, suggesting that some previously observed global alignment may be explained by scale-related confounders. In contrast, local neighborhood similarity remains significant across modalities after calibration [29]. This finding refines the Platonic view: the most robust form of convergence may not be convergence toward a globally identical representation space, but convergence toward shared local neighborhood relationships.

This revision is directly relevant for multimodal representation alignment. If local neighborhoods are the aspect of representation geometry that remains stable after controlling for scale-related effects, then alignment methods should prioritize the preservation of local topology rather than enforce global feature matching. The theoretical and empirical background, therefore, points toward alignment objectives that respect modality-specific geometries while encouraging agreement in the semantic neighborhood structure of the data.

2.8 Representation Alignment

Representation alignment refers to a family of methods that regularize the internal representations of a model by making them closer to representations produced by another model, modality, or learning objective. The general idea is that a pretrained model may already encode useful semantic or perceptual structure, and that this structure can be transferred to another model by aligning their latent spaces during training.

This perspective is directly connected to the discussion on the Platonic Representation Hypothesis. If different models can organize data according to similar latent structures, then it becomes meaningful to use one representation space as a guide for

another. In this setting, alignment is not necessarily restricted to pointwise feature matching. It may also concern the preservation of neighborhood relations, similarity structures, or other geometric properties of the latent space.

In the context of Multimodal Large Language Models (MLLMs), representation alignment is particularly relevant because standard visual instruction tuning is mainly supervised through language modeling. Visual tokens are used as input to the LLM, but the loss is applied to textual outputs. As a consequence, the visual path receives only indirect supervision. This can encourage the model to retain only those visual features that are immediately useful for text prediction, while discarding other meaningful visual information that may be important for tasks such as spatial reasoning, object counting, or visual grounding.

A common way to introduce representation alignment is to augment the original training objective with an auxiliary regularization term:

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda \mathcal{L}_{\text{align}}, \tag{2.33}$$

where $\mathcal{L}_{\text{task}}$ is the original task loss, $\mathcal{L}_{\text{align}}$ encourages similarity between selected representations, and λ controls the strength of the alignment objective. Different methods differ in what they align, which teacher representation they use, and how similarity is measured.

2.8.1 REPA

REPA, or Representation Alignment [95], introduces representation alignment as a regularization strategy for training diffusion and flow-based Transformer models. Although REPA is not designed for MLLMs, it is useful in this context because it clearly shows the general principle behind representation alignment: an internal representation learned by a model can be improved by aligning it with a stronger external representation (see Figure 2.16).

In REPA, the hidden states of a denoising network are aligned with clean image representations extracted from pretrained visual encoders. The motivation is that

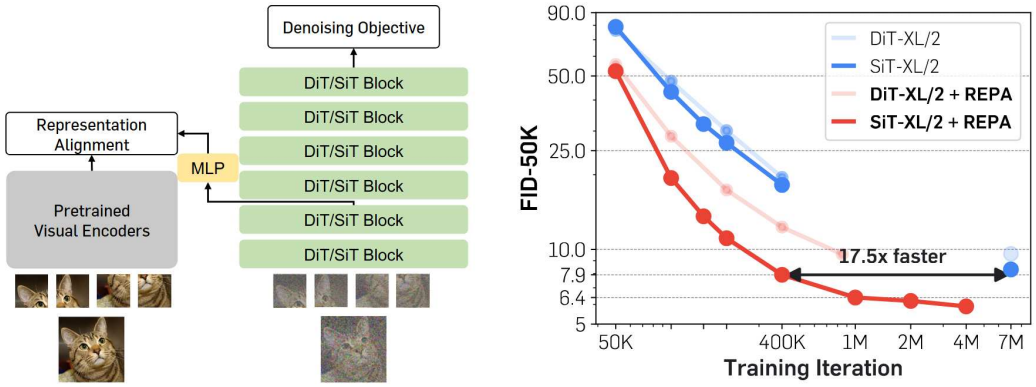

Figure 2.16: REPA operates through the DiT layers by aligning hidden states with features from a pretrained vision encoder (left). This results in faster training convergence (right). Figure from [95].

diffusion models learn meaningful representations during denoising, but these representations may emerge slowly and may be weaker than those produced by strong self-supervised visual encoders. By forcing intermediate hidden states to predict or match pretrained visual features, REPA injects semantic structure into the generative model during training.

The main contribution of REPA is therefore conceptual: representation alignment can make training easier by providing additional structure to the model's internal states. Instead of expecting the model to discover all useful representations from the primary objective alone, the model is guided toward an already informative latent space. This idea is later reflected in MLLM alignment methods, where internal visual representations are guided by vision foundation models.

2.8.2 VIRAL

VIRAL applies representation alignment directly to Multimodal Large Language Models [92]. The starting point of the method is the observation that visual instruction tuning is dominated by text-only supervision. During training, the MLLM receives image tokens, but the objective is to predict textual outputs. Therefore, the model may

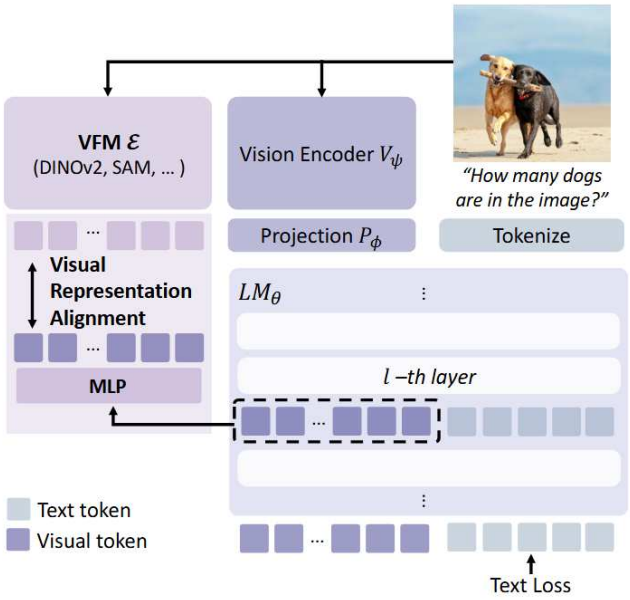

Figure 2.17: The figure shows the VIRAL approach. A layer around the middle of the MLLM is selected before training, and its image hidden states are aligned through feature matching with features from a vision encoder. Figure from [92].

learn to discard visual details that are not immediately useful for language generation, even if those details are necessary for vision-centric tasks.

To address this issue, VIRAL introduces an auxiliary visual representation alignment objective. The internal visual representations of the MLLM are aligned with representations produced by pretrained vision foundation models. These external visual models act as teachers and provide richer visual features than those indirectly induced by the language modeling loss. The method is depicted in Figure 2.17.

The goal is not only to preserve the information produced by the input vision encoder, but also to complement the MLLM with additional visual knowledge from stronger visual foundation models. In practice, VIRAL aligns selected intermediate visual representations using a feature-level similarity objective, such as cosine similarity. This encourages the MLLM to maintain fine-grained visual information throughout its internal layers, improving performance on tasks that require visual grounding, spatial understanding, and object-level reasoning.

VIRAL is important because it explicitly identifies visual representation loss as a limitation of standard MLLM training. It also shows that direct supervision on internal visual representations can improve the behavior of the model without changing the overall LLaVA-style architecture.

2.8.3 ROSS

ROSS, or Reconstructive Visual Instruction Tuning, addresses the same general limitation from a different direction [81]. Instead of aligning internal representations directly with a visual teacher through a feature-matching loss, ROSS introduces a vision-centric reconstructive objective. The motivation is that standard visual instruction tuning supervises only textual outputs, while the visual part of the sequence remains largely unsupervised.

ROSS asks the model to preserve visual information by reconstructing visual content from its internal representations. However, reconstructing raw RGB pixels is difficult and may force the model to focus on low-level details that are not useful for reasoning. For this reason, ROSS reconstructs latent visual tokens instead of raw image pixels. These latent targets provide a more compact and meaningful supervision signal (see Figure 2.18).

This reconstructive objective encourages the MLLM to retain image details that would otherwise be lost under text-only supervision. In this sense, ROSS can be interpreted as a form of representation alignment through reconstruction: rather than directly minimizing the distance between two feature spaces, it forces the internal representation to contain enough information to recover visual latent targets.

Compared with VIRAL, ROSS places more emphasis on visual output supervision. VIRAL aligns internal features with a pretrained visual representation, while ROSS supervises internal representations by requiring them to reconstruct latent visual information. Both methods share the same motivation: the visual pathway of MLLMs needs explicit supervision beyond language modeling.

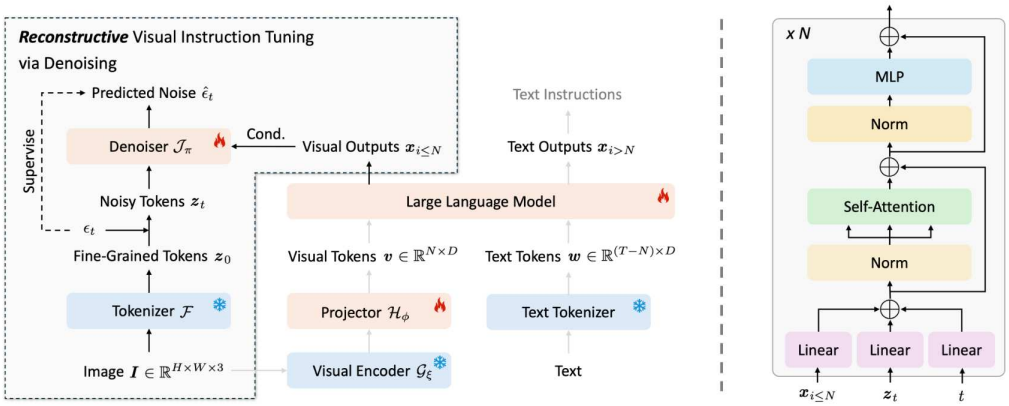

Figure 2.18: ROSS proposes several representation-alignment approaches. Its method enforces the hidden state to preserve useful features by using it as conditioning for a denoising process. Figure from [81].

2.8.4 JARVIS

JARVIS [10] extends this line of work by introducing a self-supervised visual learning objective into MLLM training. The method is inspired by JEPA-style representation learning, where the model learns by predicting missing information in latent space rather than reconstructing raw input data. You can find a representation of the methodology in Figure 2.19.

The motivation is again that textual supervision provides an incomplete learning signal for visual understanding. Captions and instructions describe only part of the visual content, and they often omit spatial relations, fine-grained attributes, or local structures. As a consequence, MLLMs may overfit language priors and fail to develop robust visual perception.

JARVIS addresses this issue by making the MLLM predict missing visual information in the latent space. Frozen vision foundation models are used as context and target encoders, while the trainable components learn to infer structural and semantic regularities from images. This provides a self-supervised visual signal that does not depend exclusively on language.

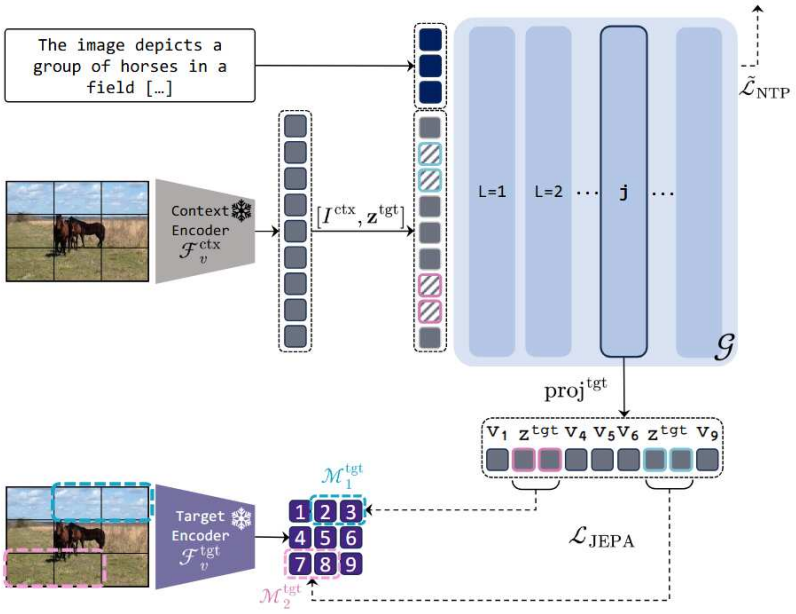

Figure 2.19: JARVIS integrates a JEPA objective into MLLM training by predicting masked visual target tokens in latent space. Figure from [10].

Compared with feature-level alignment methods, JARVIS emphasizes predictive visual learning. The model is not only encouraged to match a given representation, but also to infer hidden visual structure. This makes JARVIS particularly relevant as an example of how self-supervised visual objectives can be integrated into MLLM training to strengthen visual perception.

2.8.5 CMAR

CMAR, or Cross-Modal Alignment Regularization, studies representation alignment from a broader cross-modal perspective [24]. Instead of focusing only on MLLMs that process both images and text, CMAR investigates whether a language model can benefit from being aligned with representations produced by a vision model.

The method introduces an alignment regularization term during language model training. The language model representations are compared with the vision model representations at selected layers, and the training objective encourages the two repre-

sentation spaces to become more aligned. This is conceptually related to the Platonic Representation Hypothesis: if different modalities capture different observations of the same underlying world, then aligning their representations may help transfer useful structure from one modality to another.

An important aspect of CMAR is that it does not require the two models to have the same architecture or the same feature dimensionality. Instead of enforcing direct equality between feature vectors, the method relies on similarity-based alignment. This makes it suitable for cross-modal settings, where vision and language models usually have different architectures, tokenizations, and latent dimensions.

CMAR is therefore especially relevant for the theoretical motivation of this thesis. It supports the idea that alignment should not be understood only as direct feature regression, but also as the preservation of relational structure between representations. This connects naturally with local topological alignment measures such as MKNN.

2.8.6 Discussion

The methods discussed in this section show different ways of introducing representation alignment into modern neural architectures. REPA demonstrates that aligning intermediate representations with pretrained visual features can improve generative model training. VIRAL applies this idea to MLLMs by directly aligning internal visual representations with vision foundation models. ROSS introduces reconstructive visual supervision to preserve image details. JARVIS adds a JEPA-inspired self-supervised visual objective. CMAR shows that cross-modal alignment can improve language models by regularizing them with visual representations.

Despite their differences, these methods share a common motivation: the primary training objective may not be sufficient to induce the desired internal representations. In MLLMs, this issue is particularly important because language modeling supervision alone may not preserve the full visual structure needed for fine-grained multimodal reasoning.

Most existing methods apply alignment at the level of selected layers or global

internal representations. However, Transformer representations are not monolithic. As discussed in the previous sections, multi-head attention decomposes the model into several attention heads, each potentially capturing different types of relations. This suggests that representation alignment may be more effective if applied selectively to specific internal components rather than uniformly to an entire layer.

This observation motivates the direction studied in this thesis: instead of aligning only coarse intermediate representations, one can analyze and regularize the alignment of individual attention heads. Moreover, following the local alignment perspective introduced by MKNN, the objective is not necessarily to enforce exact feature equality, but to preserve the neighborhood structure induced by strong visual representations.

3. Method

3.1 Preliminaries

Multimodal Large Language Models (MLLMs). From an architectural perspective (refer to Fig. 3.2, *left*), an MLLM $\mathcal{M}$ is composed of *(i)* an LLM $\mathcal{G}$, which serves as the reasoning backbone and natural language interface of the model; *(ii)* a pre-trained vision encoder $\mathcal{V}$, used to process visual inputs; and *(iii)* a projector proj, which maps the output embedding space of $\mathcal{V}$ into the input embedding space of $\mathcal{G}$.

$\mathcal{M}$ ingests and generates text x as a sequence of tokens $\mathbf{x}_{1,\dots,T}$ converted into latent vectors by the embedding matrix of $\mathcal{G}$. Visual inputs I, instead, are first processed by the vision encoder $\mathcal{V}$, then mapped into the input embedding space of $\mathcal{G}$ through the projector: $\mathbf{v}_{1,\dots,V} = \text{proj}(\mathcal{V}(I))$, and finally concatenated with the sequence of text embeddings. We train $\mathcal{M}$ by minimizing the negative log-likelihood of generating token $\mathbf{x}_j$ conditioned on the image I and on the preceding text $\mathbf{x}_{1,\dots,j-1}$:

$$\mathcal{L}_{\text{LM}}(I_i, x_i, \mathcal{M}) = -\sum_{j=1}^{T} \log P\left(\mathbf{x}_j | \mathbf{v}_{1,\dots,V}, \mathbf{x}_{1,\dots,j-1}; \mathcal{M}\right). \tag{3.1}$$

Mutual K-Nearest Neighbor (MKNN) Alignment Metric. MKNN [35] is a kernel alignment metric that allows different representation functions to be compared. In this work, we use it to quantify the alignment between textual and visual representations associated with the same data point. Given an image-text pair $(I, x)_i \in \mathcal{D}$, where $\mathcal{D}$ is a dataset of aligned image-text pairs, we denote by $\mathcal{G}(x_i) \in \mathbb{R}^{d_\mathcal{G}}$ a representation of the text extracted from the language model, and by $\mathcal{V}^t(I_i) \in \mathbb{R}^{d_{\mathcal{V}^t}}$ representation of the corresponding image from a *teacher* vision encoder. Here, $\mathcal{G}(x_i)$ refers to an internal representation (*e.g.*, from intermediate layers or attention heads).

For a dataset $\mathcal{D}$, MKNN estimates how well the local neighborhood structures induced by the two representation spaces agree, by computing the average intersection of their k-nearest neighbor sets. We denote by $\mathcal{N}_k^{\mathcal{F}}(\cdot)$ the operator returning the

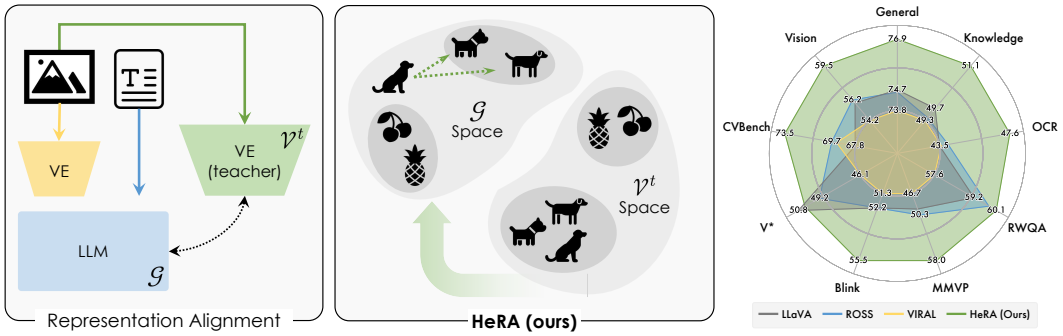

Figure 3.1: Standard representation alignment enforces strict vision-language feature matching (*left*), whereas HeRA (*center*) aligns cross-modal local neighbors, yielding stronger VQA results (*right*).

k-nearest neighbors according to maximum dot product similarity in the latent space of the embedding function $\mathcal{F}$. For instance, in the language space, where we average pool the output embeddings, it is formally defined as follows:

$$\mathcal{N}_k^{\mathcal{G}}(x_i) = \text{argmax}_{j \neq i}^{(k)} \mathcal{G}(x_i)^\top \mathcal{G}(x_j). \tag{3.2}$$

In the visual space, we pool by taking the `CLS` embeddings at the output of the vision encoder $\mathcal{V}^t$. The MKNN alignment metric between $\mathcal{G}$ and $\mathcal{V}^t$ is thus defined as:

$$m_{k\text{NN}}(\mathcal{G}, \mathcal{V}^t, \mathcal{D}) = \mathop{\mathbb{E}}_{(I,x)_i \in \mathcal{D}} \left[\frac{1}{k} \left| \mathcal{N}_k^{\mathcal{G}}(x_i) \cap \mathcal{N}_k^{\mathcal{V}^t}(I_i) \right| \right] \in [0, 1], \tag{3.3}$$

where $|\cdot|$ denotes set cardinality.

High scores in Eq. 3.3 indicate that the *local* topological latent structure produced by $\mathcal{G}$ is retained in the latent space of $\mathcal{V}^t$.

3.2 Contrastive Learning Proxy for Representation Alignment

For a fixed vision encoder $\mathcal{V}^t$, $m_{k\text{NN}}$ has been positively correlated with better performance on language modeling tasks [35]. We aim to investigate whether a similar

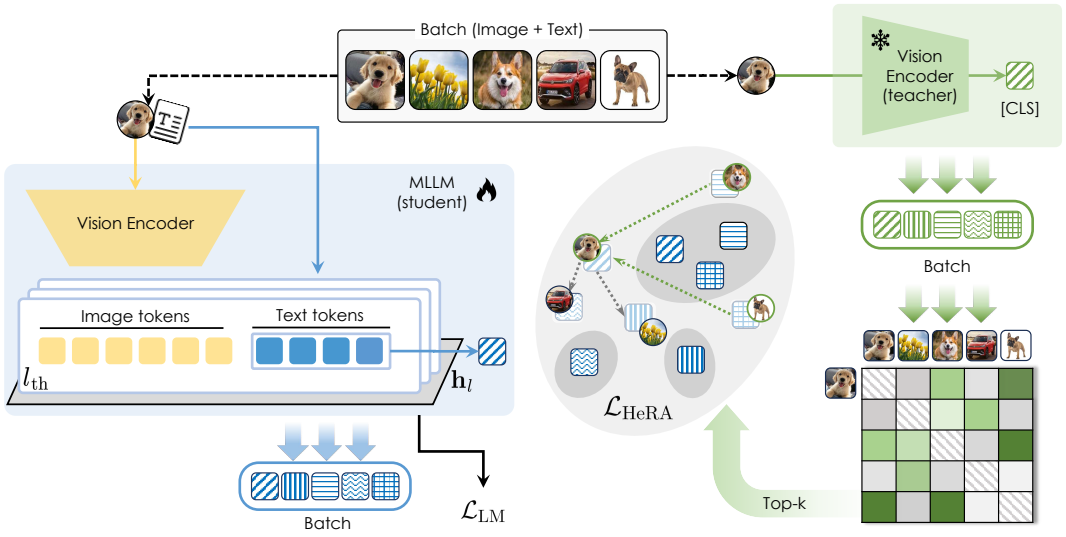

Figure 3.2: **Overview of HeRA.** In addition to the standard language modeling objective ($\mathcal{L}_{\text{LM}}$), HeRA uses a contrastive loss ($\mathcal{L}_{\text{HeRA}}$) to move representations from selected LLM attention heads closer to their k-nearest neighbors (Top-k), computed in the latent space of a frozen teacher vision encoder.

causal effect may also arise in a multimodal setting: *can we train a better MLLM by explicitly enforcing alignment with the visual domain?*

A direct solution would be to maximize $m_{k\text{NN}}$ during training. However, Eq. 3.3 depends on discrete neighbor indices and is therefore not differentiable. To overcome this limitation, we introduce a contrastive objective that encourages the multimodal representations produced by $\mathcal{M}$ to reproduce the local neighborhood structure induced by the teacher vision encoder.

Given a batch $\mathcal{B} = \{(I, x)_i\}$, let $\mathcal{M}(I_i, x_i) \in \mathbb{R}^{d_{\mathcal{G}}}$ denote the multimodal representation obtained by average pooling the text embeddings. For each sample i, we first identify a set of target neighbors $\mathcal{N}_k^{\mathcal{V}^t}(I_i)$, corresponding to the k nearest neighbors of I_i in the teacher vision space. We then optimize $\mathcal{M}$ so that its representation $\mathcal{M}(I_i, x_i)$ is close to the representations of these neighbors, while remaining separated from the rest of the batch. Formally, this can be achieved via a multi-target variant of the

InfoNCE [56] loss:

$$\mathcal{L}_{\text{RA}}(I_i, x_i, \mathcal{M}) = -\frac{1}{k} \sum_{j \in \mathcal{N}_k^{\mathcal{V}^t}(I_i)} \log \frac{\exp\left(\frac{\mathcal{M}(I_i,x_i)^\top \mathcal{M}(I_j,x_j)}{\tau}\right)}{\sum_{z \in \mathcal{B}, z \neq i} \exp\left(\frac{\mathcal{M}(I_i,x_i)^\top \mathcal{M}(I_z,x_z)}{\tau}\right)}, \tag{3.4}$$

where τ is a learnable scalar governing the sharpness of the distribution. Minimizing Eq. 3.4 encourages the *student* model $\mathcal{M}$ to produce multimodal representations that share the same *local* neighborhood as the corresponding visual representations extracted from the *teacher* model $\mathcal{V}^t$, which is exactly the property measured by the $m_{k\text{NN}}$ metric.

3.3 Head-Wise Representation Alignment (HeRA)

While the contrastive objective in Eq. 3.4 imposes alignment on a single pooled representation, it does not explicitly consider the internal structure of the language backbone. Since $\mathcal{G}$ processes the entire multimodal sequence, alignment can be controlled more precisely by acting directly on its internal representations.

In principle, $\mathcal{G}$ produces several representations for the same input. In particular, we can extract a representation from each Transformer layer of the language backbone. For language modeling (*i.e.*, Eq. 3.1), the last layer is the one used to sample the next token after the unembedding matrix. Conversely, representation alignment methods [38, 95] typically rely on intermediate layers. However, the choice of which layer(s) to use is usually treated as a fixed hyperparameter (*e.g.*, selecting the middle layer [92]), and therefore does not adapt to the specific structure of a given model.

In this work, we instead analyze finer-grained representations within the language model, focusing on the individual attention heads in each multi-head self-attention layer of $\mathcal{G}$. Since different attention heads specialize in different roles within an LLM [54, 55, 83], operating at the head level enables a more localized intervention on the language model, reducing potential conflicts between language modeling and representation alignment.

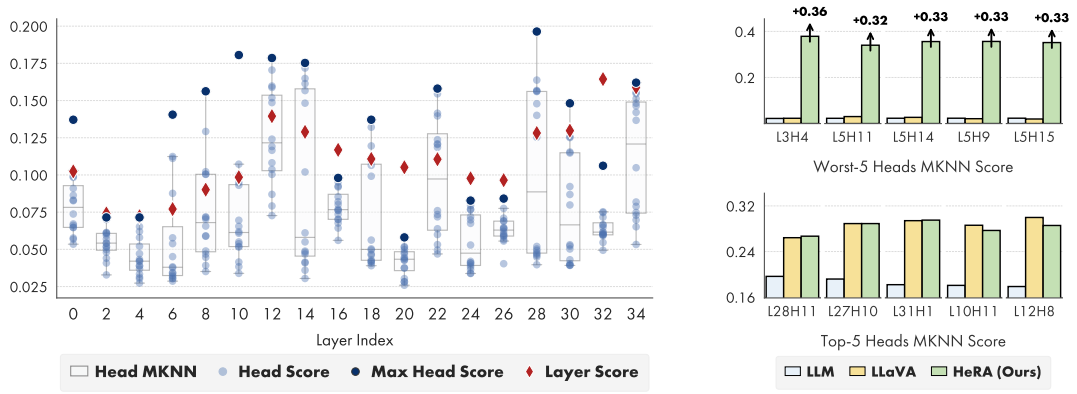

Figure 3.3: **Left:** Alignment with DINOv2-L, measured with the MKNN metric on each layer and attention head of Qwen2.5-3B. **Right:** MKNN scores of the Worst-5 and Top-5 heads, computed on *(i)* the base LLM; *(ii)* after LLaVA multimodal training; and *(iii)* after adding HeRA.

Head-level Representations. In standard multi-head attention, the final output of a layer is obtained by concatenating the outputs of the individual heads and multiplying them by an output projection matrix $\mathbf{W}_O \in \mathbb{R}^{d_\mathcal{G} \times d_\mathcal{G}}$. Let $\mathbf{h}_{l,h} \in \mathbb{R}^{d_{head}}$ be the output of the h-th attention head in layer l, where $d_{head} = d_\mathcal{G}/H$. The output projection can be written as:

$$\text{MultiHead}(\cdot) = [\mathbf{h}_{l,1}(\cdot), \ldots, \mathbf{h}_{l,H}(\cdot)]\mathbf{W}_O. \tag{3.5}$$

Because matrix multiplication is a linear operator, we can decompose $\mathbf{W}_O$ into H distinct blocks along its row dimension, such that $\mathbf{W}_O = [\mathbf{W}_{O,1}^\top, \ldots, \mathbf{W}_{O,H}^\top]^\top$, with each $\mathbf{W}_{O,h} \in \mathbb{R}^{d_{head} \times d_\mathcal{G}}$. The multi-head attention output can then be equivalently written as:

$$\text{MultiHead}(\cdot) = \sum_{h=1}^{H} \mathbf{h}_{l,h}(\cdot)\mathbf{W}_{O,h}. \tag{3.6}$$

This decomposition makes it possible to isolate the projected contribution of each head before it is added to the shared residual stream. Given a multimodal input (I_i, x_i), we define:

$$\mathcal{M}^{l,h}(I_i, x_i) = \mathbf{h}_{l,h}(I_i, x_i)\mathbf{W}_{O,h} \in \mathbb{R}^{d_\mathcal{G}}, \tag{3.7}$$

where $\mathcal{M}^{l,h}$ denotes the representation extracted from the h-th attentive head in the

l-th layer of $\mathcal{M}$ during the multimodal forward pass.

Head-wise Alignment Objective. We apply the contrastive alignment loss of Eq. 3.4 independently to selected head-level representations. For a set of layer-head indices $\mathcal{H} = \{(l, h)\}$, we define our head-wise representation alignment loss as the average alignment loss over the selected heads:

$$\mathcal{L}_{\text{HeRA}}(I_i, x_i, \mathcal{M}, \mathcal{H}) = \mathop{\mathbb{E}}_{(l,h)\in\mathcal{H}} \left[\mathcal{L}_{\text{RA}}(I_i, x_i, \mathcal{M}^{l,h})\right]. \tag{3.8}$$

The final training objective (illustrated in Fig. 3.2) is given by the sum of the language modeling and head-wise representation alignment losses:

$$\mathcal{L}(I_i, x_i, \mathcal{M}, \mathcal{H}) = \mathcal{L}_{\text{LM}}(I_i, x_i, \mathcal{M}) + \lambda\mathcal{L}_{\text{HeRA}}(I_i, x_i, \mathcal{M}, \mathcal{H}), \tag{3.9}$$

where λ is a fixed hyperparameter to balance the two contributions.

Heads Selection. For an LLM with L layers and H attention heads, the total number of heads is $L \times H$. In practice, this number is in the order of hundreds, making an extensive search for the optimal $\mathcal{H}$ unfeasible. To avoid this issue, we use the $m_{k\text{NN}}$ metric of Eq. 3.3 to rank the heads according to their alignment score with the vision encoder $\mathcal{V}^t$. We compute this rank using the language model $\mathcal{G}$ *before* the multimodal training, so that its representations are purely textual. Surprisingly, we observe that there always exists a set of heads whose alignment score is substantially higher than that of any full layer in the same model (see Fig. 3.3, left). Once the alignment rank is computed, we select a subset of m heads according to two reasonable strategies. Specifically, we select either *(i)* the best aligned heads (*i.e.*, $\mathcal{H}_m^{\text{top}}$), since enforcing alignment may be easier when they already start from a partially aligned latent space, or *(ii)* the least aligned heads (*i.e.*, $\mathcal{H}_m^{\text{worst}}$), in order to strengthen the components of the model that are farther from the visual domain. Empirically, we find that choosing $\mathcal{H}_m^{\text{worst}}$ works best: it improves the alignment of poorly aligned heads, while preserving the alignment of the strongest heads, as displayed in Fig. 3.3, right.

4. Experiments

4.1 Experimental Settings

Training Details. We train all models using the two-stage LLaVA-1.5 pipeline [44], keeping both the training data and the training protocol unchanged across experiments. As vision encoder $\mathcal{V}$, we use SigLIP2 ViT-SO400M/14@384 [77] in all settings. For the LLM $\mathcal{G}$, we evaluate a set of architectures, including Vicuna [16], Llama3 [28], Qwen2.5 [59], and Qwen3 [91], with model sizes ranging from 3B to 14B.

Following the LLaVA-1.5 recipe, in the first stage, we train only the projector proj, implemented as a two-layer MLP, on 558k image-caption pairs, while keeping the language model $\mathcal{G}$ frozen. In the second stage, we jointly optimize $\mathcal{G}$ and proj for visual instruction tuning on the LLaVA-Instruct-665k dataset. The training configuration (*e.g.*, optimizer, learning rate, and batch size) is kept identical to LLaVA-1.5. We apply the HeRA loss (cf. Equation 3.8) during both training stages, setting λ equal to 0.01, and k equal to 10. Unless otherwise stated, DINOv2 ViT-L [57] is used as the teacher vision encoder $\mathcal{V}^t$.

All experiments are run on AMD MI250x devices, each equipped with 2 GPUs and 64GB of VRAM. The first training stage uses 16 GPUs and requires up to 6 hours, depending on the size of the LLM. The second training stage uses 32 GPUs and runs for up to 14 hours. We observe no noticeable increase in training time when adding $\mathcal{L}_{\text{HeRA}}$. The MKNN alignment scores can be computed offline efficiently. For example, with a 7B LLM and DINOv2-L, the computation takes less than one hour on a single GPU.

Head Selection and LLM Details. Head selection is carried out *before* multimodal training. More precisely, we compute the $m_{k\text{NN}}$ alignment score (Equation 3.3) for every attention head using the LLM $\mathcal{G}$, prior to any multimodal finetuning. This yields a ranking of heads according to their alignment with the visual domain. The scores are

Table 4.1: Ablation study on the choice of *(i)* the objective for representation alignment (feature- vs. contrastive-based), and *(ii)* the granularity of the LLM representation to align (layer- vs. head-level).

Representation Alignment			**LLM:** Qwen2.5-3B					**LLM:** Qwen3-4B				
Objective	Granularity	Selection	**General**	**Knowledge**	**OCR**	**Vision**	**All**	**General**	**Knowledge**	**OCR**	**Vision**	**All**
-	-	-	73.5	46.7	42.2	50.5	54.2	75.6	49.6	43.8	56.3	57.4
Feature	Layer	Middle	72.6	46.4	42.3	50.1	53.8	75.0	48.6	42.8	53.4	56.0
Feature	Head	Worst (5)	74.0	46.5	42.8	51.6	54.7	76.0	49.3	44.7	55.9	57.6
Contrastive	Layer	Middle	72.8	45.7	41.6	51.1	53.8	75.8	49.5	43.7	57.2	57.6
Contrastive	Layer	Worst (5)	73.1	46.2	40.9	51.7	54.0	73.2	47.9	41.2	52.1	54.6
Contrastive	Head	Random (5)	73.7	46.6	42.1	49.7	54.0	**76.1**	49.7	44.0	56.6	57.7
Contrastive	Head	Top (5)	73.8	46.5	42.8	50.2	54.3	75.9	49.9	**45.2**	56.3	57.8
Contrastive	Head	Worst (1)	73.5	46.9	42.9	51.2	54.6	76.0	49.7	44.1	57.0	57.8
Contrastive	Head	Worst (3)	73.9	**47.6**	43.6	51.0	55.0	75.8	49.9	44.7	57.0	57.9
Contrastive	Head	Worst (10)	62.6	45.5	34.6	38.9	46.0	75.8	49.7	44.7	56.3	57.7
Contrastive	**Head**	**Worst (5)**	**74.5**	47.5	**43.8**	**52.9**	**55.7**	76.0	**50.1**	44.5	**58.5**	**58.4**

estimated on 1,000 samples from the GranD dataset [62], whose dense captions make it suitable for reliably estimating cross-modal neighborhood structure. We select the $m = 5$ least aligned heads and apply HeRA only to this subset throughout training.

Table 4.2 reports the exact checkpoints of each LLM used in this work, all publicly available on the Hugging Face Hub. The table also lists the specific attention heads used to compute $\mathcal{L}_{\text{HeRA}}$ for each LLM, ordered from left to right by increasing MKNN alignment score. We denote with `LXHY` the index of the `Y`-th head in layer `X`.

Contrastive Learning Details. To obtain the supervision signal for each batch, we extract the `[CLS]` token representations from the teacher vision encoder $\mathcal{V}^t$. We then compute all pairwise dot products among these representations, obtaining a similarity matrix (*i.e.*, a Gram matrix) that encodes the visual neighborhood structure within the batch. For each sample, we select its top-k nearest neighbors and use them to define multi-positive contrastive targets. This is done by assigning a uniform probability of $\frac{1}{k}$ to the selected k neighbors, and zero probability to all remaining samples.

For the student model, we extract head-wise representations from the selected set

Table 4.2: Checkpoint reference and list of selected heads for each LLM.

		Selected Heads ($\mathcal{H}$)				
LLM	**Hugging Face Page**	H1	H2	H3	H4	H5
Qwen2.5-3B [59]	`Qwen/Qwen2.5-3B-Instruct`	L3H4	L5H11	L5H14	L5H9	L5H15
Qwen3-4B [91]	`Qwen/Qwen3-4B-Instruct-2507`	L7H6	L13H12	L3H6	L3H5	L13H13
Vicuna-7B [16]	`lmsys/vicuna-7b-v1.5`	L0H14	L1H1	L0H1	L0H3	L0H30
Llama3-8B [28]	`meta-llama/Meta-Llama-3-8B-Instruct`	L0H31	L2H21	L0H29	L2H23	L2H25
Qwen2.5-7B [59]	`Qwen/Qwen2.5-7B-Instruct`	L14H5	L4H7	L14H4	L4H10	L4H13
Qwen3-8B [91]	`Qwen/Qwen3-8B`	L13H12	L7H6	L3H5	L13H13	L13H15
Vicuna-13B [16]	`lmsys/vicuna-13b-v1.5`	L0H38	L1H0	L0H11	L0H10	L0H26
Qwen2.5-14B [59]	`Qwen/Qwen2.5-14B-Instruct`	L13H27	L18H13	L0H9	L0H7	L3H8
Qwen3-14B [91]	`Qwen/Qwen3-14B`	L14H24	L9H36	L9H32	L1H37	L1H36

of heads ($\mathcal{H}$) and average-pool the embeddings associated with text tokens. In the first training stage, the text tokens correspond to the caption of the input image, so we use all of them. In the second training stage, the text tokens correspond to multi-turn dialogs, and we pool only over the tokens belonging to the `<ASSISTANT>` turn. These are the same tokens that contribute to the language modeling loss in Equation 3.1.

The temperature parameter τ of Equation 3.4 is learned on a logarithmic scale, and it is initialized as 0.07.

Evaluation Benchmarks. We mainly assess our method on the Cambrian comprehensive benchmark suite [73], which covers General, Knowledge, OCR, and Vision tasks. We further evaluate hallucination robustness on CHAIR-MSCOCO [97], AMBER [82], and HallusionBench [31]. Complete details about the evaluation datasets are provided in section 4.2.

4.2 Evaluation Benchmarks

Cambrian Evaluation Suite [73].[1] It consists of a comprehensive benchmark suite designed to assess a broad range of MLLM capabilities, including general perception,

[1] `https://github.com/cambrian-mllm/cambrian`

knowledge reasoning, OCR, and chart understanding, and core visual abilities. Following this organization, the benchmarks are divided into four categories: General, Knowledge, OCR, and Vision. In our experiments, we use 18 benchmarks: GQA [34], POPE [42], MME [22], MMBench (MMB) [46], and SEED-Bench (SEED) [39] for the General category; ScienceQA (SQA) [49], MMMU [96], MathVista [50], and AI2D [36] for Knowledge; ChartQA [51], OCRBench (OCRB) [47], TextVQA [68], and VizWiz [32] for OCR; and MMVP [74], RealWorldQA (RWQA) [88], Blink [23], V* [87], and CVBench [73] for Vision. When computing average scores, we normalize MME by dividing it by 20, so that its scale is consistent with the other benchmarks.

Hallucination Datasets. We evaluate hallucination behavior on three commonly used benchmarks: AMBER [82], CHAIR-MSCOCO [97], and HallusionBench [31]. CHAIR-MSCOCO measures object-level and sentence-level hallucination rates (*i.e.*, CHAIR_i and CHAIR_s) on model-generated captions for 500 images sampled from the MSCOCO [43] validation set. The AMBER generative task further reports *cognition* (Cog), which measures the overlap between objects hallucinated by the model and by humans, and *coverage* (Cover), which measures object-level recall. In addition, the AMBER discriminative task evaluates a wider range of hallucination types, including attribute and relation hallucinations, besides object existence, using a ground-truth set of 1,004 manually annotated images. For CHAIR-MSCOCO and the AMBER generative task, we use greedy decoding with a maximum generation length of 512 tokens. For the AMBER discriminative task, we append the instruction *"Answer only with Yes or No. Use exactly one word. Do not use commas, periods, or symbols."* to each query to enforce binary (Yes/No) answers. We evaluate hallucination robustness on HallusionBench [31] with an exact-match protocol. Specifically, we constrain the model to produce unambiguous responses by appending the following instruction to each query: *"Answer the question using a single word or phrase: Yes or No."* We report three standard metrics. *qAcc* (Question Pair Accuracy) evaluates consistency at the group level. A prediction is considered correct under qAcc only when the model answers all questions within the same group correctly. We also report *Easy* accuracy,

computed on the original questions, and *Hard* accuracy, computed on adversarially modified or misleading variants designed to trigger hallucinations.

4.3 Experimental Results

4.3.1 Ablation Studies and Analyses

We first report a set of ablation studies in Table 4.1, aimed at analyzing how the main architectural and objective design choices affect the behavior of HeRA. In these experiments, we use Qwen2.5-3B [59] and Qwen3-4B [91] as the underlying LLMs. As baseline, we consider a LLaVA model trained on the same LLMs without any alignment regularization (*first* row). All configurations follow the same training settings adopted by our method, as described in section 4.1.

Objective and Granularity. We first evaluate a standard representation alignment strategy [92, 95], where the LLM is optimized to minimize the cosine similarity between visual-token features extracted at the middle layer and those produced by the teacher vision encoder (*second* row), using a trainable projector. This configuration is ineffective on both LLMs. In contrast, replacing it with our contrastive learning objective while keeping the same representation granularity (*fourth* row) produces encouraging improvements on vision tasks.

Head-Level Alignment and Selection. Keeping the contrastive objective fixed, we then move to a finer level of granularity by considering textual representations extracted from specific attentive heads in the LLM. Head selection is based on the MKNN alignment score with the vision encoder: we select either the top-5 (*seventh* row) or worst-5 (*last* row) heads according to this ranking. On both LLMs, the results show a clear advantage for aligning the worst-5 heads. For example, Qwen2.5-3B improves its performance on vision-centric tasks by +1.4 points, while Qwen3-4B obtains a +2.3 points gain. As control experiments, we also apply the contrastive alignment loss to a random subset of 5 heads (*sixth* row), which does not lead to a

consistent improvement, and test the "worst-5" selection strategy at the layer level (*fifth* row), which instead causes a mild regression on Qwen2.5-3B and a severe degradation on Qwen3-4B.

Connection to the Platonic Representation Hypothesis. The effectiveness of the worst-5 strategy supports the positive relation between alignment and performance discussed by the PRH [35]. As reported in Figure 3.3 (*right*), after training with HeRA, the worst-5 heads substantially improve their vision-language alignment, without reducing the alignment of the top-5 heads. In contrast, explicitly aligning the top-5 heads does not produce a meaningful collateral improvement on the worst-5 heads. Interestingly, aligning visual features extracted from the worst-5 heads (*third* row) is not effective and only marginally affects their alignment scores (see the plot in Figure 4.2).

Number of Heads to Align. Finally, we study the effect of varying the number of aligned heads (Table 4.1, *bottom*). Aligning 3 heads, or even a single head, yields moderate improvements on both LLMs, whereas the best overall results are obtained with 5 heads, especially on the most challenging vision benchmarks. On the other hand, increasing the number of aligned heads to 10 leads to a performance drop, particularly for Qwen2.5-3B, suggesting that aligning too many heads can start interfering with the primary language modeling objective.

4.3.2 Main Experimental Results

Results on Cambrian Benchmarks. To evaluate the generality and scalability of the proposed representation alignment regularization, we apply HeRA to a diverse set of language models, as shown in Table 4.3. We intentionally include models from different architectural generations, so that the analysis is not limited to a single model family or design. This selection includes established baselines such as the Vicuna family, together with recent state-of-the-art open-source models such as Qwen3. We also vary the scale of the LLM backbone, moving from compact models (3B and 4B) to larger reasoning engines (13B and 14B). We refer to subsection 4.3.4 for the full

Table 4.3: VQA results of HeRA applied to the LLaVA training recipe on different LLMs.

	General	Knowledge	OCR	Vision						
Model	Avg	Avg	Avg	RWQA	MMVP	Blink	V*	CVBench	Avg	
Qwen2.5-3B	73.5	46.7	42.2	55.2	46.0	46.8	44.5	**60.2**	50.5	
+ HeRA (Ours)	**74.5**	**47.5**	**43.8**	**56.3**	**48.0**	**49.1**	**51.3**	59.6	**52.9**	Δ+2.4
Qwen3-4B	75.6	49.6	43.8	59.9	48.0	**55.1**	50.3	68.3	56.3	
+ HeRA (Ours)	**76.0**	**50.1**	**44.5**	**61.3**	**56.0**	53.7	**52.4**	**69.1**	**58.5**	Δ+2.2
Vicuna-7B	**72.2**	44.3	**45.7**	56.5	38.7	46.8	44.5	**62.1**	49.7	
+ HeRA (Ours)	72.1	**44.5**	**45.7**	**57.8**	**42.7**	**47.9**	**49.7**	61.9	**52.0**	Δ+2.3
LLama3-8B	73.3	45.0	43.0	60.1	46.0	49.2	44.0	**69.5**	53.8	
+ HeRA (Ours)	**74.6**	**46.3**	**44.7**	**60.4**	**46.7**	**50.2**	**51.8**	66.4	**55.1**	Δ+1.3
Qwen2.5-7B	76.2	50.2	47.9	59.6	51.3	**51.7**	**50.8**	70.2	56.7	
+ HeRA (Ours)	**76.5**	**50.5**	**48.6**	**61.3**	**54.0**	50.2	50.3	**71.3**	**57.4**	Δ+0.7
Qwen3-8B	74.7	49.7	43.5	59.2	49.3	52.2	**50.8**	67.8	55.9	
+ HeRA (Ours)	**76.9**	**51.1**	**47.6**	**60.1**	**58.0**	**55.5**	50.3	**73.5**	**59.5**	Δ+3.6
Vicuna-13B	73.4	45.5	**47.7**	**58.7**	**44.7**	50.6	46.1	63.7	52.7	
+ HeRA (Ours)	**73.6**	**45.7**	47.6	57.3	44.0	**52.3**	**49.2**	**66.5**	**53.9**	Δ+1.2
Qwen2.5-14B	75.6	50.7	44.8	**60.1**	47.3	52.7	46.6	67.9	54.9	
+ HeRA (Ours)	**77.4**	**52.8**	**49.3**	**60.1**	**52.0**	**55.2**	**52.4**	**71.6**	**58.3**	Δ+3.4
Qwen3-14B	77.4	**52.8**	46.1	60.3	57.3	**52.6**	50.3	**70.8**	58.2	
+ HeRA (Ours)	**77.7**	52.6	**47.8**	**62.5**	**58.0**	52.0	**51.8**	70.2	**58.9**	Δ+0.7

breakdown of the General, Knowledge, and OCR categories.

A common concern when modifying the internal representations of a pre-trained LLM is the possible degradation of its linguistic and reasoning priors. However, the results show that our head-wise alignment preserves, and often improves, the core capabilities of the model. Across the General, Knowledge, and OCR categories, adding HeRA consistently leads to stable or higher average scores compared to the standard LLaVA training recipe. For instance, Qwen2.5-14B increases its General average from 75.6 to 77.4, while also improving on Knowledge and OCR. This suggests that restricting the alignment objective to a carefully selected subset of attention heads limits catastrophic interference with the main language modeling objective.

The strongest impact of HeRA emerges on Vision-Centric benchmarks, which

Table 4.4: Results of HeRA on visual hallucinations benchmarks.

	MSCOCO		AMBER (Generative)				AMBER (Discriminative)				HallusionBench		
	$CHAIR_s \downarrow$	$CHAIR_i \downarrow$	$CHAIR_i \downarrow$	Cover ↑	HalRate ↓	Cog ↓	Acc ↑	Prec ↑	Rec ↑	F1 ↑	qAcc ↑	Easy ↑	Hard ↑
Qwen2.5-3B	44.2	12.6	5.8	52.0	30.1	3.3	83.6	83.9	**93.2**	88.3	23.7	60.7	55.2
+ HeRA (Ours)	**44.0**	**11.8**	**5.2**	**52.3**	**27.6**	**2.8**	**85.6**	**86.5**	92.8	**89.5**	**24.4**	**61.3**	**60.2**
Qwen3-4B	**42.0**	**11.7**	**5.5**	**53.0**	28.2	3.0	**86.5**	89.9	**89.6**	**89.7**	**22.2**	**63.1**	**54.6**
+ HeRA (Ours)	43.8	11.9	**5.5**	52.5	**27.6**	**2.8**	**86.5**	**90.1**	89.4	**89.7**	18.9	60.7	50.3
Vicuna-7B	**46.6**	**12.6**	6.3	52.2	**30.8**	**3.3**	84.2	90.0	85.8	87.8	15.6	**55.4**	45.7
+ HeRA (Ours)	47.6	12.9	**6.1**	**52.6**	31.9	3.5	**85.9**	**90.8**	**87.7**	**89.2**	**16.3**	55.0	**49.6**
LLama3-8B	44.8	13.0	**5.5**	51.5	27.5	2.9	**85.9**	**89.2**	89.6	89.4	16.9	**58.7**	**49.9**
+ HeRA (Ours)	**40.2**	**11.7**	5.6	**51.7**	**26.6**	**2.7**	**85.9**	88.3	**90.8**	**89.5**	**17.4**	58.0	49.3
Qwen2.5-7B	45.2	**11.9**	5.3	52.3	26.7	**2.7**	87.6	**89.6**	91.9	90.7	21.5	**66.8**	49.4
+ HeRA (Ours)	**44.8**	12.4	**4.9**	**52.5**	**25.1**	**2.7**	**87.7**	89.1	**92.8**	**90.9**	**23.3**	66.6	**50.9**
Qwen3-8B	42.0	12.3	5.8	52.8	28.9	3.2	87.3	90.3	90.6	90.4	21.1	61.1	53.3
+ HeRA (Ours)	**39.2**	**11.0**	**5.2**	**53.1**	**27.9**	**3.0**	**89.0**	**92.4**	**90.9**	**91.6**	**25.7**	**66.2**	**54.5**
Vicuna-13B	**43.0**	**11.9**	**6.1**	52.5	**28.4**	3.3	84.8	**93.9**	82.4	87.8	13.0	55.2	43.9
+ HeRA (Ours)	47.4	12.2	**6.1**	**52.9**	30.3	**3.2**	**85.7**	91.7	**86.2**	**88.9**	**14.7**	**56.0**	**45.6**
Qwen2.5-14B	42.6	12.2	6.1	**52.3**	29.4	3.6	86.2	87.9	91.8	89.8	23.7	61.5	**56.5**
+ HeRA (Ours)	**39.0**	**10.6**	**5.4**	51.9	**28.0**	**3.1**	**89.0**	**90.4**	**93.4**	**91.9**	**25.9**	**66.2**	56.4
Qwen3-14B	43.2	11.7	5.5	**53.1**	28.8	3.2	88.8	**91.6**	91.5	91.5	23.7	66.2	53.0
+ HeRA (Ours)	**41.2**	**11.4**	**5.0**	53.0	**26.7**	**3.0**	**88.9**	90.7	**92.7**	**91.7**	**27.9**	**69.0**	**56.8**

directly assess visual perception, spatial reasoning, and multimodal grounding. Across different architectures and release periods, our method consistently improves visual performance. Older models such as Vicuna-7B obtain a solid +2.3 point gain, showing that effective representation alignment can also benefit legacy architectures. At the same time, newer models with stronger textual priors also benefit substantially; in particular, Qwen3-8B achieves the largest individual improvement, with a +3.6 average gain.

The same trend remains visible when scaling the LLM backbone. On the largest models considered in our experiments, representation alignment remains effective: Qwen2.5-14B obtains a +3.4 point increase, while Qwen3-14B reaches the highest Vision-Centric average, equal to 58.9.

Results on Hallucination Benchmarks. Although hallucination mitigation remains an open problem in MLLMs and is not directly optimized by our contrastive rep-

Table 4.5: VQA comparison of different representation alignment strategies for MLLMs.

	General	Knowledge	OCR	Vision						
Alignment	Avg	Avg	Avg	RWQA	MMVP	Blink	V*	CVBench	Avg	
-	74.7	49.7	43.5	59.2	49.3	52.2	50.8	67.8	55.9	
ROSS [81]	74.6	49.4	44.0	59.8	50.3	52.2	49.2	69.7	56.2	Δ+0.3
VIRAL [92]	73.8	49.3	43.6	57.6	46.7	51.3	46.1	69.3	54.2	Δ-1.7
JARVIS [10]	76.8	49.9	46.2	59.2	54.7	54.2	**55.5**	69.9	58.7	Δ+2.8
CMAR [24]	76.4	51.0	46.0	58.8	50.0	52.1	52.9	70.9	56.9	Δ+1.0
HeRA (Ours)	**76.9**	**51.1**	**47.6**	**60.1**	**58.0**	**55.5**	50.3	**73.5**	**59.5**	Δ+3.6

resentation alignment loss, HeRA still has a positive effect on hallucination robustness, as reported in Table 4.4. On both CHAIR-MSCOCO and AMBER generative benchmarks, models trained with HeRA consistently reduce hallucination rates (*e.g.*, CHAIR_s and CHAIR_i). Importantly, on AMBER, this reduction is obtained while also improving, or at least preserving, the cognition (Cog) score. This is a non-trivial outcome, since models penalized for hallucinations often become excessively conservative.

In discriminative evaluations, HeRA produces consistent gains in accuracy and F1 score on AMBER, suggesting stronger visual grounding. On HallusionBench, the method improves almost all models, with clear gains in qAcc, Easy, and Hard metrics. The only exception is Qwen3-4B, which drops on this specific benchmark; however, this decrease is largely compensated by the stronger improvements obtained on standard VQA and Vision-Centric benchmarks, as shown in Table 4.3. Overall, explicitly aligning internal LLM representations with the visual domain reduces the tendency to rely excessively on language priors, leading to more faithful vision-language generation.

Comparison with Previous Representation Alignment Methods. In Table 4.5, we compare HeRA with recent representation alignment methods: ROSS [81] trains an auxiliary denoising network conditioned on LLM visual features to reconstruct visual tokens; VIRAL [92] aligns visual features extracted from the middle layer of the

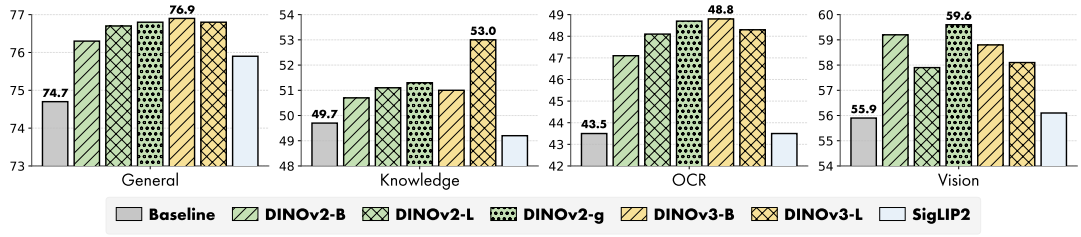

Figure 4.1: VQA results of HeRA with different teacher vision encoders.

LLM during instruction tuning; JARVIS [10] reconstructs masked image latents using representations from one quarter of the LLM depth; and CMAR [24] optimizes the CKA alignment metric between textual features from the penultimate LLM layer and the teacher encoder. We run each method according to its official implementation. All methods are trained on the same LLaVA [44] dataset, use Qwen3-8B [4] as the LLM, SigLIP2 ViT-SO400M/14@384 [77] as the vision encoder, and DINOv2-L [57] as the alignment teacher, except for ROSS.

Compared with these fixed-layer methods, our targeted head-wise alignment is substantially more effective. In particular, the strict pointwise feature alignment used by VIRAL is the only method that underperforms LLaVA (*first* row). Moreover, although CMAR shares with our method the objective of topological alignment between representation spaces from different modalities, the CKA metric enforces *global* pointwise relationships to match those from the vision encoder. By contrast, HeRA focuses exclusively on preserving *local* neighborhood structures, without imposing rigid distance constraints between samples. As a result, on the challenging Vision-Centric benchmarks, HeRA achieves a +3.6 point average improvement, outperforming the second-best method, JARVIS (+2.8), and obtains the highest overall scores on General, Knowledge, and OCR tasks. Full per-category results are reported in subsection 4.3.4.

4.3.3 Varying the Teacher Visual Encoder

In Figure 4.1, we analyze the effect of using different teacher vision encoders to define the targets for the HeRA contrastive loss. All models are trained with Qwen3-8B as

the language backbone and SigLIP2 as the primary vision encoder. We find that using SigLIP2 itself as a teacher is largely ineffective, consistent with concurrent work [92, 95] showing that unsupervised vision encoders provide better representation teachers than encoders trained with language supervision. In contrast, DINO-based [57, 65] teachers lead to strong and stable improvements, even when using base models (DINOv2-B and DINOv3-B). However, we do not observe clear additional benefits from the larger 1B-parameter DINOv2-g, suggesting that base vision encoders may already be sufficient for topological alignment.

4.3.4 Additional Ablation Studies and Results

DINOv2 as Vision Encoder. In this work, we show the effectiveness of using a teacher vision encoder, *e.g.*, DINOv2-L [57], as a source of supervision for topological representation alignment. A natural question is what happens if we remove representation alignment entirely and directly use DINOv2-L as the vision encoder of an MLLM. Table 4.8 addresses this question by comparing DINOv2-L against SigLIP2 [77], showing that DINOv2-L is ineffective when used alone as the vision encoder for MLLMs. For a fair comparison, DINOv2-L receives images at the same resolution of 384×384 pixels used for SigLIP2. Even under this setting, DINOv2-L shows severe limitations, especially on OCR tasks. These findings are consistent with prior work [21, 73], which shows that unsupervised visual encoders alone underperform language-supervised encoders on VQA benchmarks. At the same time, as shown in Figure 4.1, language-supervised encoders are not effective representation teachers. This motivates representation alignment methods such as HeRA, where MLLMs benefit from the complementary properties of language-supervised and unsupervised visual encoders.

HeRA in Different Training Stages. We apply $\mathcal{L}_{\text{HeRA}}$ to both training stages of LLaVA. However, there are clear differences between the two stages that are worth analyzing. In the first training stage (*i.e.*, St.1), the MLLM is trained on images and their corresponding captions, which form aligned image-text pairs, *i.e.*, the same

Table 4.6: Detailed VQA results of HeRA applied to the LLaVA training recipe on different LLMs.

	General						Knowledge					OCR					Vision					
	Avg	*GQA*	*POPE*	*MME*	*MMB*	*SEED*	**Avg**	*SQA*	*MMMU*	*MathV*	*AI2D*	**Avg**	*ChartQA*	*OCRB*	*TextVQA*	*VizWiz*	**Avg**	*MMVP*	*RWQA*	*Blink*	*V**	*CVBench*
Qwen2.5-3B	73.5	63.8	87.7	1484.0	70.9	71.2	46.7	75.4	40.9	7.7	62.9	42.2	17.3	37.1	60.6	53.8	50.5	46.0	55.2	46.8	44.5	**60.2**
+ HeRA (Ours)	**74.5**	**64.1**	**87.9**	**1525.5**	**72.0**	**72.0**	**47.5**	**75.5**	**41.3**	**7.9**	**65.1**	**43.8**	**20.2**	**39.1**	**61.7**	**54.3**	**52.9**	**48.0**	**56.3**	**49.1**	**51.3**	59.6
Qwen3-4B	75.6	64.6	87.3	1528.4	75.4	74.1	49.6	77.2	44.6	**8.4**	68.4	43.8	23.3	**42.5**	**65.6**	44.0	56.3	48.0	59.9	**55.1**	50.3	68.3
+ HeRA (Ours)	**76.0**	**64.9**	**87.9**	**1544.6**	**75.6**	**74.3**	**50.1**	**78.8**	**44.7**	8.1	**68.9**	**44.5**	**24.8**	40.5	**65.6**	**47.2**	**58.5**	**56.0**	**61.3**	53.7	**52.4**	**69.1**
Vicuna-7B	**72.2**	64.9	87.5	**1469.0**	**64.6**	70.8	44.3	70.3	**36.3**	11.1	59.4	**45.7**	20.0	**40.6**	64.3	**58.0**	49.7	38.7	56.5	46.8	44.5	**62.1**
+ HeRA (Ours)	72.1	**65.1**	**87.8**	1453.4	64.0	**70.9**	**44.5**	**70.9**	35.0	**12.4**	**59.5**	**45.7**	**20.6**	40.3	**64.4**	57.6	**52.0**	**42.7**	**57.8**	**47.9**	**49.7**	61.9
LLama3-8B	73.3	64.8	87.5	**1506.0**	67.4	71.5	45.0	72.5	37.9	7.2	62.5	43.0	18.8	**40.1**	63.8	49.5	53.8	46.0	60.1	49.2	44.0	**69.5**
+ HeRA (Ours)	**74.6**	**65.6**	**87.6**	1503.3	**71.7**	**72.7**	**46.3**	**75.9**	**38.0**	**8.4**	**63.1**	**44.7**	**19.9**	39.9	**64.6**	**54.2**	**55.1**	**46.7**	**60.4**	**50.2**	**51.8**	66.4
Qwen2.5-7B	76.2	64.9	88.2	**1582.2**	75.3	73.7	50.2	**77.9**	45.1	**8.8**	68.8	47.9	24.2	39.9	64.9	62.5	56.7	51.3	59.6	**51.7**	**50.8**	70.2
+ HeRA (Ours)	**76.5**	**65.3**	**88.3**	1574.8	**76.0**	**74.1**	**50.5**	**77.9**	**46.9**	7.6	**69.6**	**48.6**	**23.9**	**40.7**	**65.6**	**64.1**	**57.4**	**54.0**	**61.3**	50.2	50.3	**71.3**
Qwen3-8B	74.7	64.8	86.5	1472.8	75.4	73.1	49.7	77.5	**47.0**	7.3	67.2	43.5	20.2	37.4	64.3	51.9	55.9	49.3	59.2	52.2	**50.8**	67.8
+ HeRA (Ours)	**76.9**	**66.1**	**87.6**	**1562.3**	**77.6**	**74.9**	**51.1**	**78.8**	46.2	**8.6**	**70.8**	**47.6**	**27.6**	**41.4**	**67.9**	**53.6**	**59.5**	**58.0**	**60.1**	**55.5**	50.3	**73.5**
Vicuna-13B	73.4	65.5	87.7	1491.9	67.1	**72.2**	45.5	72.0	**38.0**	9.9	**62.0**	**47.7**	**22.2**	41.6	**67.2**	**59.5**	52.7	**44.7**	**58.7**	50.6	46.1	63.7
+ HeRA (Ours)	**73.6**	**65.8**	**88.0**	**1504.7**	**67.2**	71.9	**45.7**	**72.2**	36.6	**13.2**	60.8	47.6	22.0	**41.8**	67.1	59.4	**53.9**	44.0	57.3	**52.3**	**49.2**	**66.5**
Qwen2.5-14B	75.6	64.2	**88.0**	1548.8	75.5	72.7	50.7	76.4	48.2	8.3	69.7	44.8	19.6	33.9	63.7	61.9	54.9	47.3	**60.1**	52.7	46.6	67.9
+ HeRA (Ours)	**77.4**	**66.1**	**88.0**	**1600.9**	**77.5**	**75.5**	**52.8**	**78.6**	**49.1**	**9.6**	**74.1**	**49.3**	**27.4**	**39.0**	**67.9**	**62.9**	**58.3**	**52.0**	**60.1**	**55.2**	**52.4**	**71.6**
Qwen3-14B	77.4	65.6	87.6	1609.5	**78.5**	74.6	**52.8**	**79.6**	**49.6**	**10.2**	71.9	46.1	26.8	41.1	69.7	47.0	58.2	57.3	60.3	**52.6**	50.3	**70.8**
+ HeRA (Ours)	**77.7**	**66.3**	**88.0**	**1632.5**	78.0	**74.9**	52.6	79.4	49.1	9.2	**72.6**	**47.8**	**28.1**	**41.8**	**69.9**	**51.4**	**58.9**	**58.0**	**62.5**	52.0	**51.8**	70.2

underlying concept is expressed through two modalities. This makes St.1 a natural setting for representation alignment, since the student MLLM and the teacher vision encoder process semantically matched inputs. By contrast, image-text pairs in the second training stage (*i.e.*, St.2) are not aligned in the same way: the text consists of a multi-turn dialog between a user and the assistant, centered on the image but not equivalent to a direct caption of its visual content. From this perspective, if one had to choose a single stage for $\mathcal{L}_{\text{HeRA}}$, one might expect St.1 to be more beneficial than St.2. However, Table 4.9 shows that this is not always the case: for Qwen2.5-3B, $\mathcal{L}_{\text{HeRA}}$ is more effective when applied during St.1, whereas for Qwen3-4B the opposite trend holds. Overall, applying $\mathcal{L}_{\text{HeRA}}$ to both training stages, as in our original design, achieves the best results on both models.

Extended Results on All Benchmarks. Since our goal is to improve MLLMs, we focus on strengthening visual perception, which is especially stressed by vision-

Table 4.7: Detailed VQA results of different representation alignment strategies for MLLMs.

	General						Knowledge					OCR					Vision					
Alignment	**Avg**	*GQA*	*POPE*	*MME*	*MMB*	*SEED*	**Avg**	*SQA*	*MMMU*	*MathV*	*AI2D*	**Avg**	*ChartQA*	*OCRB*	*TextVQA*	*VizWiz*	**Avg**	*RWQA*	*MMVP*	*Blink*	*V**	*CVBench*
-	74.7	64.8	86.5	1472.7	75.4	73.1	49.7	77.5	**47.0**	7.3	67.2	43.5	20.2	37.4	64.3	51.9	55.9	59.2	49.3	52.2	50.8	67.8
ROSS [81]	74.6	64.5	87.2	1472.5	74.9	72.7	49.4	76.4	46.3	7.4	67.4	44.0	17.8	36.8	64.8	**56.8**	56.2	59.8	50.3	52.2	49.2	69.7
VIRAL [92]	73.8	64.2	87.3	1396.6	74.9	72.6	49.3	76.4	45.7	8.0	67.1	43.6	19.3	36.7	64.2	54.2	54.2	57.6	46.7	51.3	46.1	69.3
JARVIS [10]	76.8	64.6	**88.2**	**1605.1**	76.7	74.2	49.9	77.0	45.4	8.4	68.6	46.2	27.1	40.5	66.0	51.2	58.7	59.2	54.7	54.2	**55.5**	69.9
CMAR [24]	76.4	65.6	87.4	1556.6	76.5	74.7	51.0	78.3	45.9	**8.6**	**71.0**	46.0	23.3	**41.6**	67.6	51.6	56.9	58.8	50.0	52.1	52.9	70.9
HeRA (Ours)	**76.9**	**66.1**	87.6	1562.3	**77.6**	**74.9**	**51.1**	**78.8**	46.2	**8.6**	70.8	**47.6**	**27.6**	41.4	**67.9**	53.6	**59.5**	**60.1**	**58.0**	**55.5**	50.3	**73.5**

centric benchmarks. Accordingly, in the main paper, we reported detailed scores on specific vision-centric VQA datasets, including RealWorldQA, MMVP, Blink, V*, and CVBench, while only reporting average scores for the General, Knowledge, and OCR categories. Here, we provide the complete results over all 18 VQA benchmarks considered in our study. In particular, Table 4.6 reports the results for different LLM families, while Table 4.7 provides a detailed comparison among representation alignment methods using Qwen3-8B as the LLM.

4.3.5 Additional Analyses

Additional MKNN Head-Wise Analysis. In Figure 4.2, we provide a detailed comparison of MKNN alignment scores for the Worst-5 (*upper* half) and Top-5 (*bottom* half) attention heads under different training strategies. These heads are selected by computing MKNN alignment scores on the base Qwen2.5-3B LLM before any multimodal training. Interestingly, the relative alignment of these heads is largely preserved after standard multimodal training (*i.e.*, LLaVA): heads that are naturally well aligned in the base LLM remain well aligned, whereas poorly aligned heads remain poorly aligned.

When HeRA is applied to the Top-5 heads, their alignment scores increase further, but the intervention has no effect on the Worst-5 heads. As shown in Table 4.1 (*seventh* row), this corresponds to suboptimal downstream performance. Conversely,

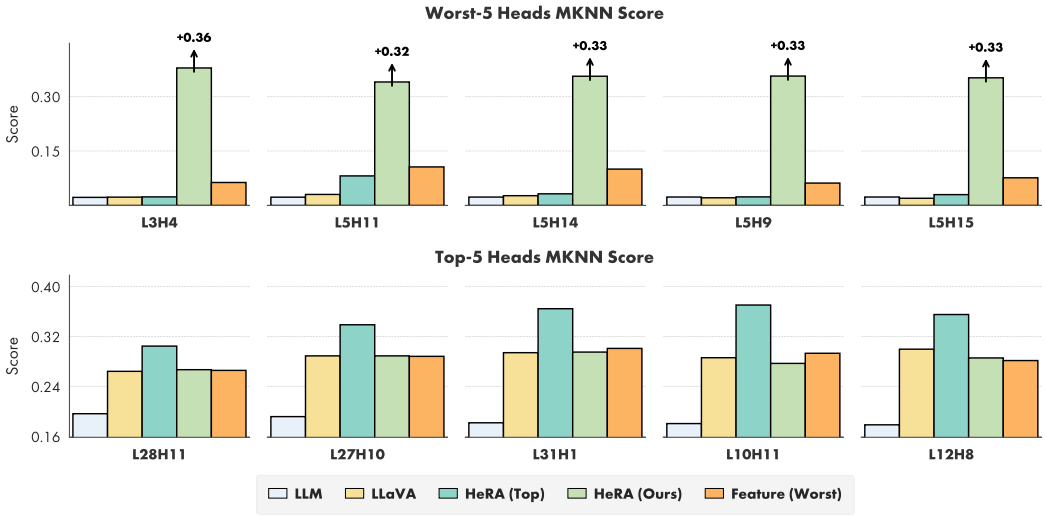

Figure 4.2: Effect of LLaVA multimodal training and representation alignment methods on the Worst-5 and Top-5 heads. Worst-5 and Top-5 heads are selected according to the lowest and highest MKNN alignment score with DINOv2-L, computed on the Qwen2.5-3B LLM before multimodal training.

our proposed strategy, namely applying HeRA to the Worst-5 heads, strongly increases the alignment of the targeted components. Importantly, this large improvement does not compromise the Top-5 heads, whose MKNN alignment scores remain very close to those of the LLaVA baseline.

Finally, we observe that feature-level representation alignment, implemented through cosine similarity maximization between MLLM visual features and the teacher vision encoder, does not effectively modify the local topological structure, nor does it improve performance, as shown in Table 4.1 (*third* row). As illustrated in the plot, this strategy has only a limited effect on the MKNN alignment scores of the targeted Worst-5 heads, further emphasizing the specific role of our topology-aware contrastive objective.

MKNN Alignment With Larger Qwen2.5 Models. In Figure 4.3 and Figure 4.4, we extend the MKNN alignment analysis of Figure 3.3 to larger LLMs from the same family, namely Qwen2.5-7B and Qwen2.5-14B, using DINOv2-L as teacher. The left parts confirm that the same behavior observed before persists at larger scales:

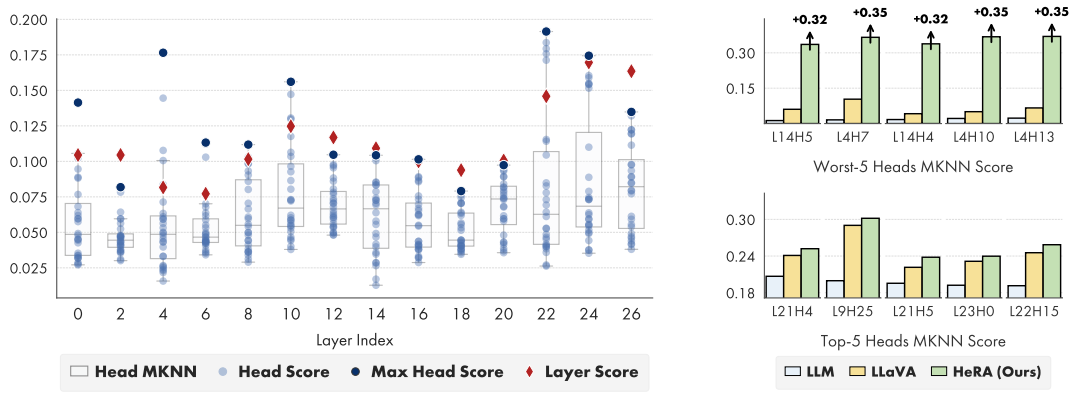

Figure 4.3: **Left:** Alignment with DINOv2-L, measured using the MKNN metric on each layer and attention head of Qwen2.5-7B. **Right:** MKNN scores of the Worst-5 and Top-5 heads, computed on *(i)* the base LLM; *(ii)* after LLaVA multimodal training; and *(iii)* after applying HeRA.

representations extracted from specific individual attention heads consistently show much stronger natural alignment with the visual domain than representations obtained from any full layer in the same model. The right parts further emphasize the atomic nature of our intervention. Applying HeRA to the Worst-5 heads produces a strong increase in their cross-modal alignment, without disrupting the structural alignment of the Top-5 heads, which are already naturally aligned in the base LLM.

MKNN Analysis with Different Representation Alignment Methods. In Figure 4.5, we compare the MKNN alignment scores obtained by HeRA and by existing representation alignment strategies. In this analysis, all methods are trained with Qwen3-8B as the LLM and SigLIP2 as the vision encoder, while the MKNN alignment metric is computed with respect to the DINOv2-L teacher. We refer to Table 4.5 for the quantitative comparison on VQA benchmarks.

In line with our previous observations, all methods increase the alignment of the Top-5 heads. However, this effect is largely a natural consequence of multimodal training itself, since most methods produce scores that closely resemble the unregularized LLaVA baseline. The only method with a more distinct impact on the Top-5

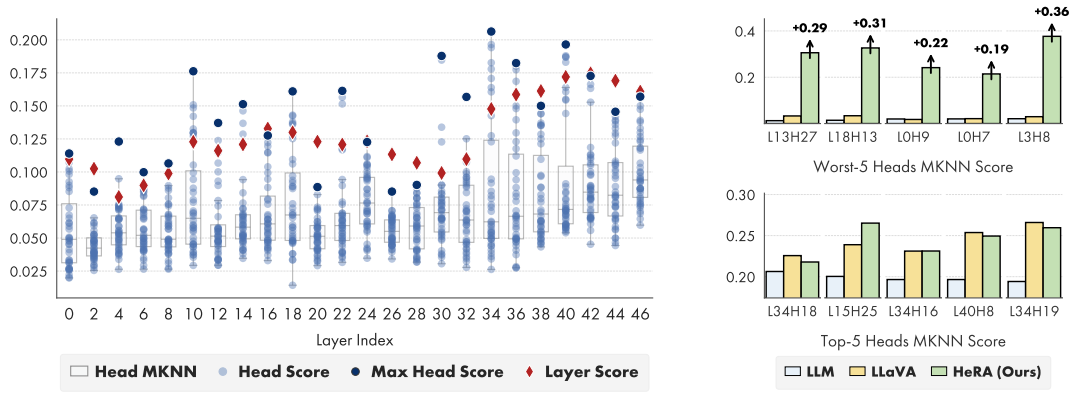

Figure 4.4: **Left:** Alignment with DINOv2-L, measured using the MKNN metric on each layer and attention head of Qwen2.5-14B. **Right:** MKNN scores of the Worst-5 and Top-5 heads, computed on *(i)* the base LLM; *(ii)* after LLaVA multimodal training; and *(iii)* after applying HeRA.

heads is CMAR. CMAR is conceptually related to HeRA because it also encourages cross-modal topological alignment rather than strict feature-level visual matching. The main difference lies in the scope of the alignment: CMAR uses the CKA metric to match *global* pairwise relationships across all samples in a training batch, whereas HeRA focuses specifically on preserving *local* neighborhood consistency.

For the Worst-5 heads, both CMAR and feature-matching methods fail to induce meaningful structural changes. In contrast, HeRA is the *only* method that substantially increases the cross-modal alignment of these initially poorly aligned heads.

4.3.6 Qualitative Results

In Figure 4.7, we present a qualitative comparison between the LLaVA [44] baseline, ROSS [81], and HeRA, using Qwen3-8B and SigLIP2. The selected examples show that HeRA produces more accurate and better-grounded answers across all evaluated categories (General, Knowledge, OCR, and Vision-Centric), often correcting perceptual errors made by the baselines.

Despite these improvements, Figure 4.8 reports a few failure cases in which our

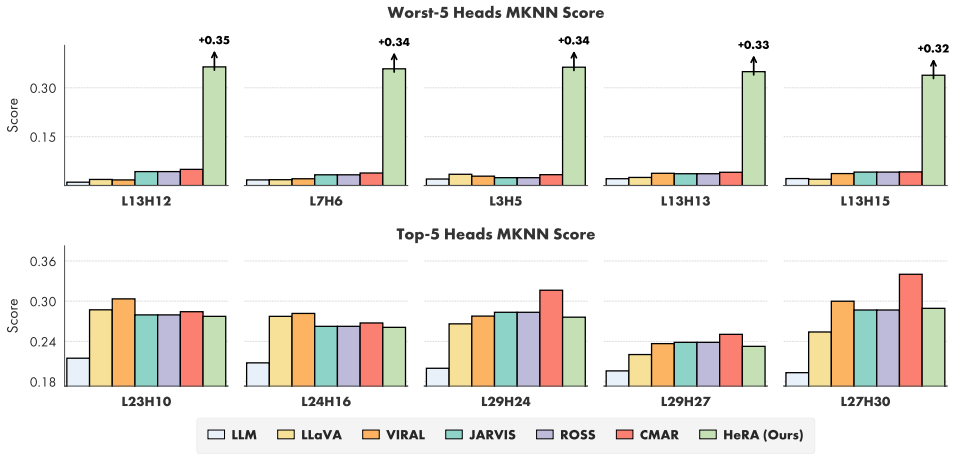

Figure 4.5: Comparison of MKNN scores of the Worst-5 and Top-5 heads after the second training stage performed with HeRA and our competitors (*i.e.*, LLaVA, VIRAL, JARVIS, ROSS, CMAR).

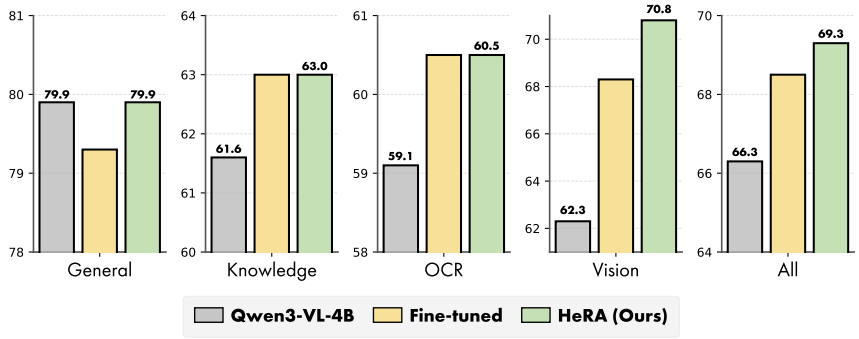

Figure 4.6: VQA results on Qwen3-VL-4B after fine-tuning, with and without the HeRA objective.

model still struggles. In particular, HeRA may occasionally misinterpret fine-grained visual details, such as counting multiple small instances, recognizing ambiguous materials and shapes, or inferring precise spatial relations in cluttered scenes.

Table 4.8: VQA comparison between DINOv2 and SigLIP2 as the vision encoder for LLaVA.

		General	**Knowledge**	**OCR**	**Vision**					
LLM	**Vision Encoder**	Avg	Avg	Avg	RWQA	MMVP	Blink	V*	CVBench	Avg
Qwen2.5-3B	DINOv2	67.1	43.3	26.3	51.5	30.0	47.8	46.1	58.0	46.7
Qwen2.5-3B	SigLIP2	73.5	46.7	42.2	55.2	46.0	46.8	44.5	**60.2**	50.5
+ HeRA (Ours)	SigLIP2	**74.5**	**47.5**	**43.8**	**56.3**	**48.0**	**49.1**	**51.3**	59.6	**52.9**
Qwen3-4B	DINOv2	70.5	46.1	24.2	54.9	38.7	49.2	49.2	63.1	51.0
Qwen3-4B	SigLIP2	75.6	49.6	43.8	59.9	48.0	**55.1**	50.3	68.3	56.3
+ HeRA (Ours)	SigLIP2	**76.0**	**50.1**	**44.5**	**61.3**	**56.0**	53.7	**52.4**	**69.1**	**58.5**

Table 4.9: Effects of $\mathcal{L}_{\text{HeRA}}$ when applied to the different training stages of LLaVA.

	$\mathcal{L}_{\text{HeRA}}$		**General**	**Knowledge**	**OCR**	**Vision**					
Model	St.1	St.2	Avg	Avg	Avg	RWQA	MMVP	Blink	V*	CVBench	Avg
Qwen2.5-3B	-	-	73.5	46.7	42.2	55.2	46.0	46.8	44.5	60.2	50.5
+ HeRA	✓	-	74.2	**47.5**	**44.2**	**57.4**	44.0	46.9	**51.3**	**60.7**	52.1
+ HeRA	-	✓	73.5	46.4	43.0	54.5	40.0	48.8	44.5	60.5	49.7
+ HeRA (Ours)	✓	✓	**74.5**	**47.5**	43.8	56.3	**48.0**	**49.1**	**51.3**	59.6	**52.9**
Qwen3-4B	-	-	75.6	49.6	43.8	59.9	48.0	**55.1**	50.3	68.3	56.3
+ HeRA	✓	-	75.6	49.7	43.4	60.7	51.3	53.3	46.1	66.2	55.5
+ HeRA	-	✓	75.7	50.0	**44.6**	59.6	54.7	52.7	49.2	67.8	56.8
+ HeRA (Ours)	✓	✓	**76.0**	**50.1**	44.5	**61.3**	**56.0**	53.7	**52.4**	**69.1**	**58.5**

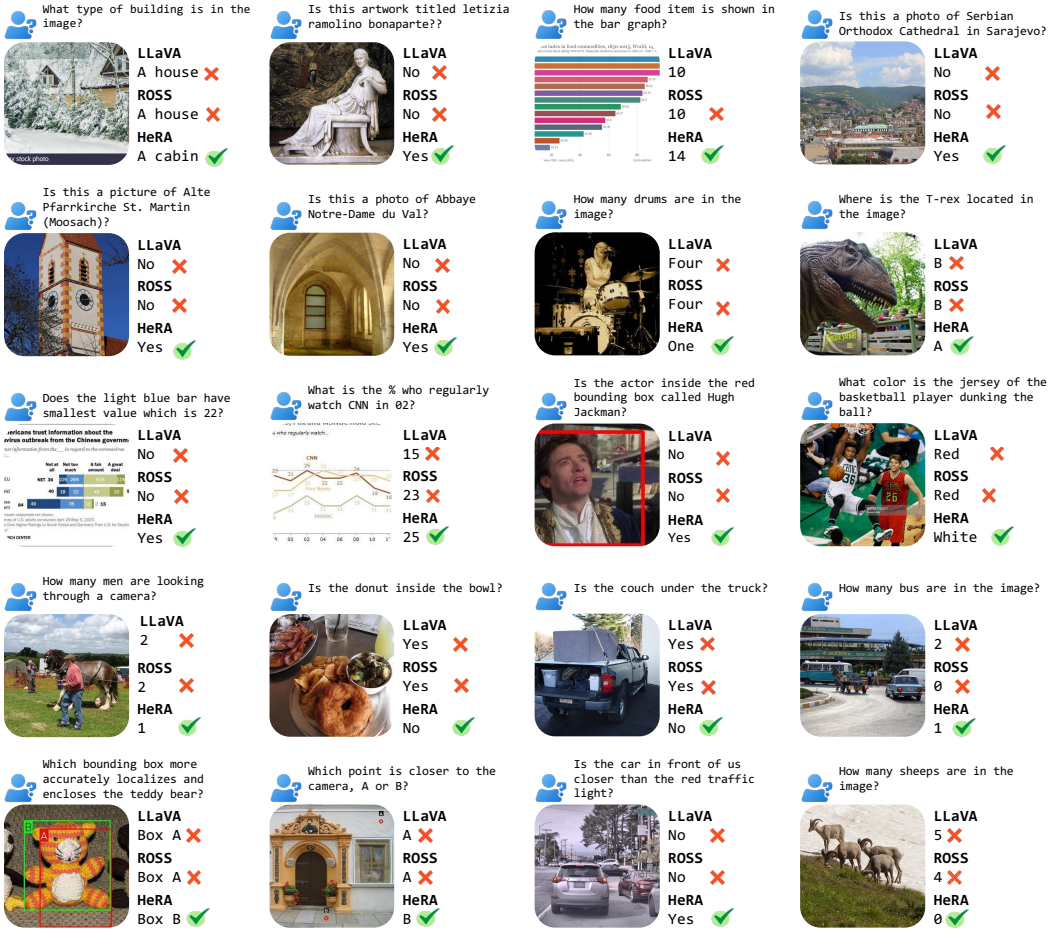

Figure 4.7: Qualitative comparison of LLaVA [44], ROSS [81], and HeRA using Qwen3-8B and SigLIP2. We show representative examples across all Cambrian categories: General, Knowledge, OCR, and Vision-Centric.

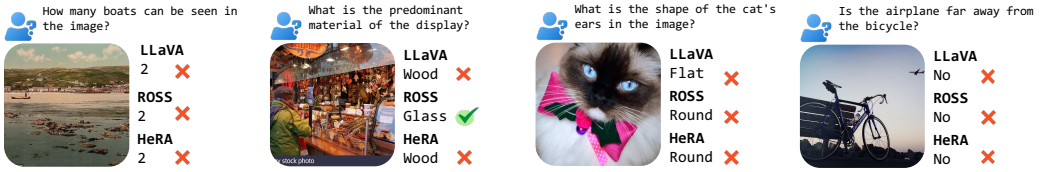

Figure 4.8: Failure cases of HeRA on VQA tasks.

5. Additional Experiments

This chapter presents a set of parallel experiments conducted during the development of HeRA. While the previous chapter focuses on the final formulation of the method and on its main experimental validation, here we analyze alternative design choices and research lines that were explored before reaching the final approach. These experiments are useful and meaningful to expose: their purpose is not to introduce independent methods, but to clarify why the choices made in the final definition of HeRA [9].

In particular, we investigate direct feature alignment with visual teacher representations, different strategies to bridge the dimensionality discrepancy between LLM hidden states and vision encoder features, architectural variants involving residual injections, selective patch alignment, CKA-based alignment, and alternative contrastive targets. All these directions share the same underlying goal: improving the visual grounding of the multimodal model by regularizing its internal representations with respect to strong frozen visual encoders.

Unless otherwise specified, the experiments are conducted on Qwen2.5-3B as the language backbone. We chose this model because experiments results cheap. The visual teacher is always kept frozen, and we consider different teacher encoders, including DINOv2-B, DINOv2-L, DINOv2-G, DINOv3-B, DINOv3-L, and DINOv3-H. The standard training protocol follows the LLaVA-style two-stage procedure, except for those experiments that specifically introduce different training pipelines.

5.1 Feature Alignment

The most direct way to perform representation alignment is to force the internal representations of the model to match the features produced by a frozen vision encoder. This strategy follows the intuition that a strong self-supervised visual backbone, such

Table 5.1: Results of feature matching alignment on Qwen2.5-3B with CLIP as the vision encoder. For this set of experiments, the projection from the MLLM to the teacher vision encoder is obtained by interpolating the MLLM features.

		General						Knowledge					OCR					Vision					
LLM	**Teacher**	***Avg***	*GQA*	*POPE*	*MME*	*MMB*	*SEED*	***Avg***	*SQA*	*MMMU*	*MathV*	*AI2D*	***Avg***	*ChartQA*	*OCRB*	*TextVQA*	*VizWiz*	***Avg***	*MMVP*	*RWQA*	*Blink*	*V**	*CVBench*
Qwen2.5-3B	-	**73.5**	**63.8**	**87.7**	**1484.0**	**70.9**	**71.2**	**46.7**	**75.4**	**40.9**	**7.7**	**62.9**	**42.2**	**17.3**	**37.1**	**60.6**	**53.8**	**50.5**	**46.0**	**55.2**	46.8	44.5	**60.2**
Qwen2.5-3B	DINOv2-B	36.1	44.7	72.9	839.9	29.8	40.0	42.9	69.4	38.2	4.8	59.0	18.0	10.3	2.5	41.2	35.9	35.3	2.7	44.7	41.2	38.2	44.9
Qwen2.5-3B	DINOv2-L	67.6	61.8	87.2	1464.2	68.3	68.0	45.4	73.3	39.2	6.8	62.4	32.9	15.3	29.5	53.7	51.8	50.4	27.3	54.4	46.9	47.1	57.5
Qwen2.5-3B	DINOv2-G	67.0	61.6	86.9	1434.6	67.4	68.1	45.1	73.1	38.7	6.2	62.5	33.3	16.2	29.6	54.1	52.8	50.3	29.3	53.9	47.3	44.0	57.0
Qwen2.5-3B	DINOv3-B	65.8	60.8	87.0	1416.6	65.9	67.5	44.4	71.8	38.2	7.3	60.1	31.7	14.5	27.5	53.0	49.7	49.0	27.3	52.9	**47.5**	45.0	55.6
Qwen2.5-3B	DINOv3-L	65.1	61.3	87.4	1368.2	65.4	67.3	45.1	73.2	38.9	6.6	61.9	31.3	14.2	26.8	52.8	46.4	50.0	29.3	52.7	46.3	**47.6**	57.8
Qwen2.5-3B	DINOv3-H	36.7	45.3	73.8	788.7	29.6	40.0	43.0	70.4	37.4	5.5	58.7	18.0	10.7	3.0	40.1	33.9	35.6	0.7	42.4	42.4	40.3	46.6

as DINOv2 or DINOv3, already provides rich visual representations. Therefore, aligning the LLM internal states to these features could transfer part of this visual structure into the multimodal model. This research pathway is followed by several works [92].

Let $v_i \in \mathbb{R}^{d_{\text{vision}}}$ be the representation extracted by the frozen visual teacher for the i-th sample, and let $h_i \in \mathbb{R}^{d_{\text{model}}}$ be the corresponding representation extracted from the LLM. Since the two representations usually have different dimensionalities, we introduce a linear projection function p_h that maps the LLM representation into the visual feature space:

$$z_i = p_h(h_i), \qquad z_i \in \mathbb{R}^{d_{\text{vision}}}. \tag{5.1}$$

The feature matching objective is then defined using the negative cosine similarity:

$$\mathcal{L}_{\text{feat}} = -\frac{1}{N}\sum_{i=1}^{N}\frac{z_i^\top v_i}{\|z_i\|\,\|v_i\|}. \tag{5.2}$$

The representations are taken in their raw form before the cosine computation, without additional normalization steps outside the loss.

This objective encourages the projected LLM representation to point in the same direction as the teacher visual representation. Compared to the contrastive objective used in HeRA, this formulation is simpler and more direct: each sample is aligned only with its corresponding teacher feature. However, this also makes the objective

Table 5.2: Results of feature matching alignment on Qwen2.5-3B with CLIP as the vision encoder. In this configuration, all models include the intermediate Stage 1.5, where only the projection is trained. We report a set of experiments with the target projection frozen during Stage 2, as well as experiments with a learnable projection.

				General						Knowledge					OCR					Vision					
LLM	**Teacher**	**St 1.5**	**T. Proj**	***Avg***	*GQA*	*POPE*	*MME*	*MMB*	*SEED*	***Avg***	*SQA*	*MMMU*	*MathV*	*AI2D*	***Avg***	*ChartQA*	*OCRB*	*TextVQA*	*VizWiz*	***Avg***	*MMVP*	*RWQA*	*Blink*	*V**	*CVBench*
Qwen2.5-3B	-	×	×	**73.5**	**63.8**	**87.7**	**1484.0**	**70.9**	**71.2**	**46.7**	**75.4**	**40.9**	**7.7**	62.9	**42.2**	**17.3**	**37.1**	**60.6**	53.8	50.5	**46.0**	**55.2**	46.8	44.5	**60.2**
Qwen2.5-3B	DINOv2-B	✓	×	67.2	61.7	87.3	1410.5	68.9	68.7	45.5	73.8	38.7	6.7	63.0	33.8	16.6	31.2	53.8	52.6	50.8	32.7	54.8	46.4	**49.2**	57.4
Qwen2.5-3B	DINOv2-L	✓	×	68.0	61.8	87.4	1447.5	69.5	68.7	45.7	73.2	39.7	6.7	**63.1**	33.8	16.4	31.6	53.3	52.3	51.4	30.7	54.8	47.7	48.2	59.0
Qwen2.5-3B	DINOv2-G	✓	×	67.6	61.7	87.3	1418.2	69.5	68.6	45.7	73.7	39.0	7.1	62.9	33.8	16.6	31.5	53.2	53.9	**51.9**	36.7	52.9	47.4	47.6	58.7
Qwen2.5-3B	DINOv3-B	✓	×	68.1	62.0	87.5	1456.1	70.0	68.5	45.8	74.2	38.7	7.6	62.9	33.9	16.4	30.7	54.4	52.0	50.9	30.0	53.7	47.4	45.6	58.1
Qwen2.5-3B	DINOv3-L	✓	×	67.5	62.1	87.6	1420.2	69.5	68.5	45.4	74.1	38.1	6.3	63.0	33.8	16.2	31.1	53.9	**56.0**	50.7	30.0	54.4	47.8	45.6	57.0
Qwen2.5-3B	DINOv3-H	✓	×	67.9	61.6	87.6	1463.6	69.5	68.4	45.3	74.1	37.9	6.7	62.6	33.9	16.3	31.0	54.3	54.3	50.7	28.7	53.9	47.2	45.0	58.5
Qwen2.5-3B	DINOv2-B	✓	✓	67.8	62.0	87.4	1431.6	69.6	68.6	45.3	72.9	39.1	6.4	62.8	33.8	16.6	31.4	53.4	52.2	51.4	32.0	53.9	47.8	47.6	58.5
Qwen2.5-3B	DINOv2-L	✓	✓	67.6	61.8	87.2	1436.3	68.5	68.4	45.1	72.7	38.4	6.4	62.7	33.6	16.6	30.9	53.4	54.6	50.4	28.0	53.9	46.4	47.6	58.3
Qwen2.5-3B	DINOv2-G	✓	✓	67.5	61.7	87.2	1415.3	68.8	68.4	45.6	73.7	39.1	6.6	62.8	34.0	17.0	31.5	53.3	51.4	50.8	29.3	54.9	47.5	46.1	58.4
Qwen2.5-3B	DINOv3-B	✓	✓	67.8	61.9	87.6	1428.7	69.6	68.7	45.4	73.1	39.2	6.5	62.8	33.9	16.6	31.2	53.8	52.4	51.2	28.7	54.6	48.2	44.0	58.4
Qwen2.5-3B	DINOv3-L	✓	✓	67.8	61.9	**87.7**	1452.5	69.2	68.5	45.9	74.2	39.9	6.5	**63.1**	33.8	16.4	31.1	53.8	52.9	51.3	31.3	**55.2**	47.3	47.1	57.9
Qwen2.5-3B	DINOv3-H	✓	✓	67.9	61.6	**87.7**	1447.0	70.0	68.6	45.5	73.9	39.4	6.4	62.3	33.7	16.3	30.8	53.9	53.9	51.3	29.3	53.9	**48.9**	47.6	58.7

more restrictive. It assumes that the LLM representation should match the teacher representation pointwise, even though the two models may organize information in different coordinate systems and with different modality-specific biases.

This direct feature matching baseline is useful because it tests the simplest version of visual regularization. If matching teacher features directly were sufficient, then no topological or contrastive proxy would be necessary. However, as discussed in the following sections, the design of the projection module, the train-inference mismatch, and the rigidity of pointwise matching all introduce practical limitations.

5.2 Projection Strategies

A central issue in feature-level alignment is the dimensionality gap between the language model and the visual teacher. The LLM hidden states live in a space of dimension d_{model}, while the visual teacher features live in a space of dimension d_{vision}. Therefore, feature matching cannot be applied directly unless the two representations

Table 5.3: Results of feature matching alignment on Qwen2.5-3B with CLIP as the vision encoder. The target projection is implemented as a multi-convolution module, with a different convolution for each head. "mConv" performs a projection from the MLLM space to the teacher space, while "mConvRev" performs the projection from the teacher space to the MLLM space.

		General						**Knowledge**					**OCR**					**Vision**					
LLM	**Projection**	**Avg**	*GQA*	*POPE*	*MME*	*MMB*	*SEED*	**Avg**	*SQA*	*MMMU*	*MathV*	*AI2D*	**Avg**	*ChartQA*	*OCRB*	*TextVQA*	*VizWiz*	**Avg**	*MMVP*	*RWQA*	*Blink*	*V**	*CVBench*
Qwen2.5-3B	-	**73.5**	**63.8**	**87.7**	1484.0	**70.9**	**71.2**	**46.7**	**75.4**	**40.9**	**7.7**	62.9	**42.2**	**17.3**	**37.1**	**60.6**	**53.8**	50.5	**46.0**	**55.2**	**46.8**	44.5	**60.2**
Qwen2.5-3B	mConv	67.6	61.7	87.4	1432.8	69.2	68.0	45.6	73.9	39.0	6.0	**63.3**	33.0	15.8	29.6	53.5	48.0	50.3	32.0	53.3	**46.8**	**46.1**	56.7
Qwen2.5-3B	mConvRev	69.0	61.5	87.0	**1521.5**	70.0	68.4	45.5	74.4	38.1	7.2	62.5	33.8	16.2	31.4	53.8	49.1	**50.7**	31.3	53.3	46.7	45.6	58.1

are first mapped to a common dimensionality.

In our experiments, the standard solution is to introduce a learnable projection:

$$p_h : \mathbb{R}^{d_{\text{model}}} \longrightarrow \mathbb{R}^{d_{\text{vision}}}. \tag{5.3}$$

In most cases, p_h is implemented as a linear layer and is defined independently for each selected attention head. Given the contribution $c_h(X)$ of the h-th attention head, the projection produces:

$$z_h(X) = p_h(c_h(X)). \tag{5.4}$$

The resulting vector can then be compared with the teacher representation using cosine similarity.

Although this solution is straightforward, it introduces an auxiliary module whose role is only to enable the alignment loss. At inference time, the teacher vision encoder is not used, and there is no need to match its representation. As a consequence, the learned LLM-to-vision projection is not part of the effective inference computation. This creates a mismatch between training and evaluation: during training, the model is optimized through a projection that disappears at inference.

To reduce this mismatch, we also explored non-parametric alternatives. In particular, we replaced the learnable projection with feature interpolation. Instead of learning an additional mapping, interpolation adapts the representation shape so that

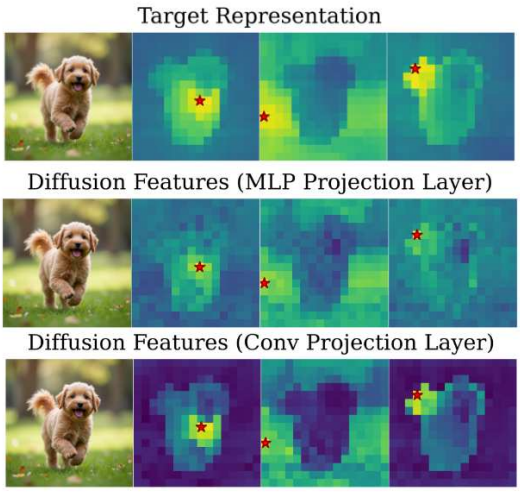

Figure 5.1: The figure shows how well spatial features are organized while produced with a vision encoder (top) in comparison with how those features look after a projection phase. The MLP projection (middle) clearly alters the spatial quality of the features; instead, convolution (bottom) preserves well the feature organization.

the alignment loss can be computed without introducing parameters that are discarded at inference. The motivation is to make the alignment signal more directly tied to the original model representation. The results are reported in Table 5.1.

A second alternative consists of introducing an intermediate training stage, referred to as Stage 1.5. In this setting, the standard LLaVA Stage 1 is first performed. Then, before stage 2, an additional stage is introduced in which only the LLM-to-vision projection is trained, using the same image-caption data employed in Stage 1. The LLM and the teacher vision encoder are kept frozen. After this intermediate phase, the projection is frozen during Stage 2. See Table 5.2 for results.

The purpose of Stage 1.5 is to decouple the learning of the projection from the main multimodal training process. Instead of letting the projection change jointly with the model representations, this setup first learns a stable mapping between the LLM hidden space and the teacher visual space. This can make the feature matching objective more stable, although it also increases the complexity of the training pipeline.

Finally, we explored a convolutional projection. In this variant, the sequence of

image-related representations is reshaped into a two-dimensional grid following the spatial layout of the image patches. A two-dimensional convolution, with kernel size 3×3, is then used to project the representation toward the teacher visual space. We reported results in Table 5.3. The motivation is that visual tokens preserve a spatial organization, and a convolutional mapping can exploit local relationships between neighboring patches more naturally than an independent linear projection. A representation of visual features after some projection is reported in Figure 5.1.

5.3 Bidirectional Attention over Image Tokens

Decoder-only MLLMs usually inherit the causal attention structure of language models. This is necessary for text generation, since each textual token must only attend to previous tokens. However, image tokens are different from text tokens: they represent an input that is fully available from the beginning. Therefore, applying a strictly causal mask to image tokens may unnecessarily restrict the exchange of visual information between patches.

To test this hypothesis, we explored a masking variant in which attention among image tokens is made bidirectional. The causal structure is preserved for text tokens, while image-to-image attention is allowed in both directions. In other words, visual tokens can attend to all other visual tokens, independently of their position in the sequence. This modification is applied both during training and during inference.

Since the image is not generated autoregressively, there is no causal constraint requiring patch i to ignore patch j for $j > i$. A bidirectional image-token mask may therefore improve the internal organization of visual information before it is used by the language model.

At the same time, this modification changes the attention pattern of the decoder-only backbone. The experiment is useful to understand whether part of the limitation of MLLMs comes from the causal treatment of image tokens. Results on an exhaustive set of configurations are reported in Table 5.4.

Table 5.4: Comparison between causal and bidirectional top-5 alignment, with and without the intermediate stage, on Qwen2.5-3B with CLIP as the vision encoder.

				General						Knowledge					OCR					Vision					
LLM	**Mask**	***Stage 1.5***	**Teacher**	***Avg***	*GQA*	*POPE*	*MME*	*MMB*	*SEED*	***Avg***	*SQA*	*MMMU*	*MathV*	*AI2D*	***Avg***	*ChartQA*	*OCRB*	*TextVQA*	*VizWiz*	***Avg***	*MMVP*	*RWQA*	*Blink*	*V**	*CVBench*
Qwen2.5-3B	Causal	×	-	64.6	59.4	86.8	1412.3	65.4	64.8	44.7	73.2	39.6	6.8	59.4	28.6	13.0	23.2	49.7	46.7	47.5	26.7	51.8	46.3	45.0	54.5
Qwen2.5-3B	Causal	×	DINOv2-B	66.4	61.4	86.7	1443.4	66.6	67.5	45.1	73.1	39.0	6.9	61.5	32.1	14.8	28.9	52.7	47.5	50.6	32.7	52.6	46.5	42.9	**58.2**
Qwen2.5-3B	Causal	×	DINOv2-G	66.3	61.6	87.2	1441.4	67.3	67.7	45.7	74.0	**40.4**	7.4	60.9	32.5	15.2	29.3	53.1	48.3	50.5	**33.3**	52.7	46.7	49.2	57.4
Qwen2.5-3B	Causal	×	DINOv2-L	65.7	61.0	**87.6**	1411.5	65.3	67.7	44.9	72.5	39.2	6.6	61.1	31.3	14.0	27.8	52.2	43.9	49.2	28.7	52.3	46.8	51.8	56.9
Qwen2.5-3B	Causal	×	DINOv3-B	**67.9**	61.9	87.3	1449.7	69.2	68.5	45.7	74.3	39.7	6.4	62.3	**33.7**	**16.3**	**31.1**	**53.8**	**50.8**	**51.4**	32.7	54.4	**50.2**	46.1	57.9
Qwen2.5-3B	Causal	×	DINOv3-H	67.3	61.5	87.1	1439.3	68.3	67.9	45.4	73.8	38.8	6.8	62.3	32.8	15.5	29.3	53.5	49.5	49.3	28.0	54.0	48.9	46.1	54.8
Qwen2.5-3B	Causal	×	DINOv3-L	66.9	61.6	87.1	1417.1	67.8	**68.6**	45.4	74.2	38.8	6.8	61.9	33.1	16.0	29.9	53.5	49.1	50.3	30.0	54.4	47.7	49.2	57.4
Qwen2.5-3B	Causal	✓	DINOv2-B	67.2	61.7	86.8	1442.7	67.9	67.8	45.6	74.5	39.1	6.4	62.2	33.1	16.1	29.6	53.7	46.9	49.0	27.3	53.6	48.0	**52.4**	55.5
Qwen2.5-3B	Causal	✓	DINOv2-G	67.8	61.8	86.9	**1468.3**	68.9	68.1	45.9	74.1	39.3	**7.5**	62.5	33.4	16.1	30.8	53.4	44.0	50.3	30.7	53.5	47.2	47.1	57.3
Qwen2.5-3B	Causal	✓	DINOv2-L	67.5	61.8	86.8	1436.3	**69.3**	68.0	45.8	74.4	39.2	7.1	62.5	33.3	15.9	30.4	53.7	45.9	48.9	25.3	**54.6**	46.3	50.3	56.0
Qwen2.5-3B	Causal	✓	DINOv3-B	67.7	62.0	86.7	1444.8	69.1	67.9	**46.1**	**75.2**	39.7	7.1	62.6	33.4	15.7	30.8	53.7	46.4	49.4	29.3	54.4	47.0	45.0	55.7
Qwen2.5-3B	Causal	✓	DINOv3-H	67.2	**62.1**	86.7	1409.0	68.8	67.9	45.7	74.5	38.8	7.1	62.4	32.9	15.7	29.4	53.6	46.3	49.1	28.7	53.3	47.3	46.1	55.1
Qwen2.5-3B	Causal	✓	DINOv3-L	67.4	61.8	86.5	1428.8	68.3	68.1	45.9	74.7	39.3	6.4	**63.3**	33.1	15.8	29.7	**53.8**	47.0	49.0	29.3	54.0	47.7	48.2	54.6
Qwen2.5-3B	Bidir	×	-	**68.4**	61.9	87.1	1446.9	**70.1**	68.4	45.7	73.2	39.0	6.8	**63.9**	**34.0**	**17.0**	31.3	53.9	52.0	50.6	28.0	52.9	49.1	47.6	58.2
Qwen2.5-3B	Bidir	×	DINOv2-B	66.5	60.9	87.0	1458.9	66.7	67.6	45.0	72.2	39.4	7.0	61.1	31.9	14.4	28.0	53.2	47.5	49.6	27.3	51.2	**49.7**	49.7	56.5
Qwen2.5-3B	Bidir	×	DINOv2-G	66.1	60.9	**87.6**	1442.8	65.2	67.2	45.1	72.0	**40.7**	6.8	60.8	31.5	14.3	27.3	52.8	44.9	50.5	30.0	53.5	48.1	**50.8**	57.5
Qwen2.5-3B	Bidir	×	DINOv2-L	66.9	61.4	87.3	1444.5	67.2	68.0	45.0	72.7	38.7	6.2	62.6	33.1	15.8	29.9	53.5	49.0	50.1	29.3	52.8	47.4	48.2	57.7
Qwen2.5-3B	Bidir	×	DINOv3-B	64.8	60.2	86.9	1388.4	64.2	66.3	45.4	73.3	40.6	7.2	60.7	30.2	13.4	25.8	51.4	46.3	49.3	24.7	52.4	46.0	45.6	58.3
Qwen2.5-3B	Bidir	×	DINOv3-H	36.5	45.2	71.0	800.6	28.3	40.5	42.3	69.4	37.0	5.1	57.6	17.5	10.3	2.3	39.8	39.1	35.1	2.7	41.8	42.1	42.4	45.3
Qwen2.5-3B	Bidir	×	DINOv3-L	67.2	61.7	87.2	1445.6	68.7	68.2	45.5	74.0	38.1	6.6	63.2	33.7	16.8	29.8	54.3	52.5	50.7	32.0	53.1	46.7	**50.8**	58.7
Qwen2.5-3B	Bidir	✓	DINOv2-B	67.2	61.7	87.3	1410.5	68.9	**68.7**	45.5	73.8	38.7	6.7	63.0	33.8	16.6	31.2	53.8	52.6	50.8	32.7	**54.8**	46.4	49.2	57.4
Qwen2.5-3B	Bidir	✓	DINOv2-G	67.6	61.7	87.3	1418.2	69.5	68.6	45.7	73.7	39.0	7.1	62.9	33.8	16.6	31.5	53.2	53.9	**51.9**	**36.7**	52.9	47.4	47.6	58.7
Qwen2.5-3B	Bidir	✓	DINOv2-L	68.0	61.8	87.4	1447.5	69.5	**68.7**	45.7	73.2	39.7	6.7	63.1	33.8	16.4	**31.6**	53.3	52.3	51.4	30.7	**54.8**	47.7	48.2	**59.0**
Qwen2.5-3B	Bidir	✓	DINOv3-B	68.1	62.0	87.5	1456.1	70.0	68.5	**45.8**	**74.2**	38.7	**7.6**	62.9	33.9	16.4	30.7	**54.4**	52.0	50.9	30.0	53.7	47.4	45.6	58.1
Qwen2.5-3B	Bidir	✓	DINOv3-H	67.9	61.6	**87.6**	**1463.6**	69.5	68.4	45.3	74.1	37.9	6.7	62.6	33.9	16.3	31.0	54.3	54.3	50.7	28.7	53.9	47.2	45.0	58.5
Qwen2.5-3B	Bidir	✓	DINOv3-L	67.5	**62.1**	**87.6**	1420.2	69.5	68.5	45.4	74.1	38.1	6.3	63.0	33.8	16.2	31.1	53.9	**56.0**	50.7	30.0	54.4	47.8	45.6	57.0

5.4 Head Selection and Feature Alignment

The final version of HeRA [9] aligns the least aligned attention heads according to the MKNN metric. This choice is motivated by the observation that different heads encode different structures, and that the heads with lower initial visual alignment may benefit more from targeted regularization. However, an alternative hypothesis is also possible: heads that are already well aligned with the visual teacher may be easier to improve further.

To test this, we performed feature-level alignment on the top aligned heads. The heads are selected according to their MKNN alignment score with the visual teacher, but instead of applying the contrastive topological objective used by HeRA, we apply

Table 5.5: Results of feature matching alignment on Qwen2.5-3B with SigLIP2 as the vision encoder. In these experiments, we vary the teacher vision encoder across most DINO model sizes. We also report an experiment using the 10 most aligned heads.

		General						Knowledge					OCR					Vision					
LLM	**Teacher / Top H**	***Avg***	*GQA*	*POPE*	*MME*	*MMB*	*SEED*	***Avg***	*SQA*	*MMMU*	*MathV*	*AI2D*	***Avg***	*ChartQA*	*OCRB*	*TextVQA*	*VizWiz*	***Avg***	*MMVP*	*RWQA*	*Blink*	*V**	*CVBench*
Qwen2.5-3B	-	**70.1**	63.7	87.9	**1482.6**	**71.2**	71.3	**46.6**	**75.7**	40.1	**7.9**	**62.9**	38.3	17.6	36.9	60.6	53.5	55.2	47.3	55.3	46.7	45.0	**60.6**
Qwen2.5-3B	DINOv2-L 10	68.7	63.4	**88.3**	1425.2	70.0	71.3	46.4	74.9	40.6	**7.9**	62.1	38.3	**18.8**	35.7	60.3	51.2	53.9	41.3	**56.7**	48.1	44.5	59.2
Qwen2.5-3B	DINOv2-B 5	68.5	63.6	88.0	1405.5	68.9	71.5	45.8	73.6	39.4	7.2	62.7	38.0	18.3	35.6	60.2	**56.0**	55.0	47.3	54.0	48.4	45.6	59.8
Qwen2.5-3B	DINOv2-L 5	69.6	**63.9**	88.2	1458.2	70.4	71.6	45.9	73.8	39.4	7.7	62.5	**39.3**	18.6	**37.3**	**62.1**	55.1	**55.3**	47.3	55.8	**49.3**	**47.1**	59.1
Qwen2.5-3B	DINOv2-G 5	68.8	63.5	88.1	1409.2	69.5	**71.7**	46.2	74.1	**41.0**	7.0	62.8	37.8	17.7	36.1	59.7	55.2	55.2	**48.7**	55.0	48.7	46.1	59.6
Qwen2.5-3B	DINOv3-B 5	56.0	52.6	82.5	1188.0	55.6	55.4	44.4	72.1	39.9	5.7	60.1	21.5	11.2	9.7	43.8	47.8	42.0	10.7	45.9	44.8	38.7	51.6
Qwen2.5-3B	DINOv3-L 5	56.7	52.4	81.4	1181.2	57.6	55.3	44.8	72.5	40.4	5.4	60.9	20.9	10.9	8.6	43.2	44.3	40.6	6.0	44.2	44.0	34.6	51.2
Qwen2.5-3B	DINOv3-H 5	68.9	63.5	87.9	1441.2	68.8	71.4	45.9	73.7	39.9	**7.9**	62.3	36.9	15.9	34.5	60.1	56.0	55.1	44.7	56.0	47.5	46.1	60.1
Qwen2.5-3B	SigLIP2 5	55.6	51.7	79.7	1207.5	55.8	53.5	44.4	70.6	40.2	6.1	60.8	19.2	10.7	3.6	43.3	42.1	40.8	8.7	45.8	43.0	37.7	50.4

the direct feature matching objective introduced earlier. In this way, the experiment isolates two design choices: the head selection strategy and the type of alignment loss. Results are reported in Table 5.5.

This comparison helps clarify whether the benefit of representation alignment comes from reinforcing already aligned components or from correcting poorly aligned ones. If the top heads were the best target for alignment, then the final method should focus on preserving and strengthening the heads that already share structure with the visual teacher. On the other hand, if the least aligned heads provide larger improvements, this supports the design choice adopted in HeRA [9].

5.5 Reusing the Projection as a Residual Signal

The projection introduced for feature matching contains information about how to map LLM representations toward the visual teacher space. In the previous experiments, this projection is used only to compute the alignment loss and discarded at inference time. However, since it is learned during training, it is natural to ask whether it contains useful information that can be reused at test time.

To investigate this possibility, we construct a residual transformation from the

Table 5.6: Results of Gram-based feature matching alignment on Qwen2.5-3B with CLIP as the vision encoder. In the first set of experiments, the Gram matrix is used only as a training signal, since it is discarded at inference time. In the second set of experiments, we modify the inference procedure to inject the learned information through a residual stream.

			General						Knowledge					OCR					Vision					
LLM	***Gram at inf.***	**Teacher**	***Avg***	*GQA*	*POPE*	*MME*	*MMB*	*SEED*	***Avg***	*SQA*	*MMMU*	*MathV*	*AI2D*	***Avg***	*ChartQA*	*OCRB*	*TextVQA*	*VizWiz*	***Avg***	*MMVP*	*RWQA*	*Blink*	*V**	*CVBench*
Qwen2.5-3B	-	-	**73.5**	**63.8**	**87.7**	**1484.0**	**70.9**	**71.2**	**46.7**	**75.4**	**40.9**	**7.7**	**62.9**	**42.2**	**17.3**	**37.1**	**60.6**	53.8	**50.5**	**46.0**	**55.2**	46.8	44.5	**60.2**
Qwen2.5-3B	×	DINOv2-B	60.4	56.0	83.9	1244.1	60.9	63.2	43.8	71.1	39.3	5.7	59.1	27.5	12.6	20.4	49.5	47.3	45.2	22.7	46.9	47.0	39.8	52.4
Qwen2.5-3B	×	DINOv2-L	63.1	59.3	86.4	1303.1	64.3	64.6	44.7	73.5	37.9	6.8	60.4	31.0	14.6	25.8	52.5	53.5	47.0	26.0	51.5	46.8	48.2	53.7
Qwen2.5-3B	×	DINOv3-B	59.6	56.2	83.7	1244.5	58.9	62.1	44.1	71.7	39.6	5.8	59.3	25.4	12.4	15.5	48.3	47.7	44.8	18.0	48.8	45.4	39.8	53.8
Qwen2.5-3B	×	DINOv3-H	31.0	39.8	60.3	687.0	20.6	37.3	41.7	68.4	38.0	4.5	55.9	16.2	9.9	2.1	36.7	41.1	35.7	6.7	42.8	40.7	33.0	44.9
Qwen2.5-3B	×	DINOv3-L	60.2	56.6	84.8	1232.2	60.4	62.8	44.4	72.5	38.2	7.0	60.0	29.6	14.6	24.2	50.1	50.6	43.3	19.3	45.8	45.0	37.2	51.4
Qwen2.5-3B	✓	DINOv2-B	63.0	59.7	86.9	1317.0	62.4	66.1	44.2	70.5	40.3	6.1	59.9	30.2	13.0	25.8	51.9	47.8	48.0	24.7	50.1	46.2	47.6	56.5
Qwen2.5-3B	✓	DINOv2-G	64.4	60.1	87.3	1370.2	65.0	66.0	44.6	72.4	39.9	6.1	59.9	29.8	13.3	24.5	51.7	47.6	47.3	26.0	51.1	46.2	**49.7**	53.6
Qwen2.5-3B	✓	DINOv2-L	64.6	61.0	87.0	1340.8	64.9	66.9	45.1	72.8	39.4	6.0	61.0	31.4	14.5	26.9	52.7	**54.3**	48.9	26.7	53.7	46.4	48.2	55.5
Qwen2.5-3B	✓	DINOv3-B	64.0	59.5	87.4	1402.1	63.0	65.3	45.1	72.7	**40.9**	7.0	59.6	29.5	13.0	24.3	51.1	48.4	48.7	24.0	53.6	**47.4**	46.1	56.8
Qwen2.5-3B	✓	DINOv3-H	24.1	39.8	60.3	0.0	20.6	37.3	41.7	68.4	38.0	4.5	55.9	16.2	9.9	2.1	36.7	41.1	27.8	0.0	42.8	0.0	0.0	44.9
Qwen2.5-3B	✓	DINOv3-L	64.3	61.0	87.3	1354.0	63.7	66.5	44.3	71.3	38.9	6.7	60.4	31.2	15.0	26.8	51.7	50.0	48.9	28.0	51.0	46.3	44.5	56.7

projection matrix itself. Let $X \in \mathbb{R}^{d_{\text{model}} \times d_{\text{vision}}}$ be the learned projection matrix from the LLM hidden space to the visual feature space. We compute:

$$R = XX^{\top}, \tag{5.5}$$

where $R \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$. This matrix defines a transformation that operates directly in the LLM hidden space. Given the projected contribution $c_h(X)$ of an aligned head, the residual variant modifies it as:

$$c'_h(X) = c_h(X) + c_h(X)R. \tag{5.6}$$

The intuition is that R captures directions in the LLM hidden space that are relevant for matching the visual teacher. Adding this term as a residual connection injects information derived from the alignment projection back into the model stream. Unlike the standard feature matching setup, where the projection is used only for the loss, this variant allows the learned alignment structure to influence the forward computation.

Table 5.7: Results of patch-importance feature matching alignment on Qwen2.5-3B with CLIP as the vision encoder. We report the percentage of used patches (P%). With only 10% of the patches used in both Stage 1 and Stage 2, we achieve performance comparable to the baseline.

			General						Knowledge					OCR					Vision					
LLM	**S1 P%**	**S2 P%**	**Avg**	*GQA*	*POPE*	*MME*	*MMB*	*SEED*	**Avg**	*SQA*	*MMMU*	*MathV*	*AI2D*	**Avg**	*ChartQA*	*OCRB*	*TextVQA*	*VizWiz*	**Avg**	*MMVP*	*RWQA*	*Blink*	*V**	*CVBench*
Qwen2.5-3B	-	-	**73.5**	**63.8**	**87.7**	1484.0	**70.9**	**71.2**	**46.7**	**75.4**	**40.9**	**7.7**	**62.9**	**42.2**	**17.3**	**37.1**	**60.6**	**53.8**	**50.5**	**46.0**	**55.2**	46.8	44.5	**60.2**
Qwen2.5-3B	0.1	0.1	67.3	61.5	87.5	1445.7	68.5	68.0	46.0	74.0	40.1	**7.7**	62.0	33.0	16.4	29.2	53.5	52.7	50.3	27.3	54.4	**47.4**	49.7	58.0
Qwen2.5-3B	0.1	0.2	67.7	61.4	87.4	**1486.2**	68.9	67.8	46.1	74.1	40.8	7.2	62.3	33.1	16.8	28.9	53.5	51.5	49.8	26.0	54.4	45.9	**50.3**	57.8
Qwen2.5-3B	0.1	0.5	67.4	61.5	87.6	1459.9	69.0	67.6	46.0	73.7	40.3	7.2	62.6	33.2	16.6	29.1	54.0	52.5	50.1	30.0	53.9	46.7	**50.3**	56.9
Qwen2.5-3B	0.4	0.1	60.7	59.0	86.5	1256.4	60.2	62.8	43.5	71.2	38.4	5.4	59.0	27.3	12.8	20.3	48.8	45.5	45.3	18.7	48.4	45.0	44.5	54.4
Qwen2.5-3B	0.4	0.4	60.2	59.0	86.6	1240.3	59.1	62.9	43.7	70.6	39.3	5.8	59.2	27.5	13.1	20.0	49.4	44.7	46.6	20.7	47.8	45.1	47.1	56.9

This residual branch is introduced during training. We then evaluate two settings: one in which the residual branch is also kept active at inference, and one in which it is removed during evaluation. This comparison tests whether the residual transformation provides useful representational information or whether it mainly acts as a training-time regularizer. See Table 5.6 for results.

5.6 Patch Importance for Selective Alignment

Another question concerns the granularity of visual alignment. In the standard setup, all image patches are treated uniformly. However, not all patches contribute equally to the model prediction. Some patches may correspond to salient objects or relevant visual regions, while others may contain background or redundant information. Aligning all patches may therefore introduce unnecessary noise into the training objective.

To study this, we perform a patch importance analysis. The importance of each image patch is estimated from the attention assigned by the model to image tokens. Specifically, attention scores over image tokens are averaged across layers and heads, producing a global importance ranking of patches over the dataset. Given a ratio ρ, we select only the top-ρ most important patches and apply the alignment loss only to

them:

$$\mathcal{P}_\rho = \mathrm{Top}_\rho \left(\mathrm{Attn}_{\mathrm{image}} \right), \tag{5.7}$$

where $\mathcal{P}_\rho$ denotes the selected set of image patches and ρ ranges from 10% to 90%.

The ranking is computed on the dataset corresponding to the stage under analysis. For Stage 1 experiments, patch importance is computed over the LLaVA Stage 1 data. For Stage 2 experiments, it is recomputed on the Stage 2 data. This avoids using a patch ranking obtained from a different data distribution. Results in Table 5.7.

This experiment is diagnostic because it tests whether visual alignment should be applied uniformly or selectively. If only a small subset of highly attended patches is sufficient, then the useful alignment signal may be concentrated in salient regions. Conversely, if performance improves only when a large fraction of patches is aligned, this suggests that the model benefits from preserving a broader visual structure.

5.7 CKA-Based Alignment

The alignment objective used in HeRA [9] is motivated by local neighborhood preservation. However, another possible way to compare representations is to use a global similarity metric. Centered Kernel Alignment (CKA) is commonly used to measure similarity between representation spaces by comparing their kernel matrices. In this sense, CKA provides a global view of representational similarity.

We test kernel CKA as a training loss. Let Z be the matrix of model representations and V the matrix of teacher visual representations for a batch of samples. The alignment loss is defined as:

$$\mathcal{L}_{\mathrm{CKA}} = -\mathrm{CKA}(Z, V). \tag{5.8}$$

Minimizing this objective encourages the model representation to become globally similar to the teacher representation according to the CKA measure.

This experiment compares two different notions of alignment. CKA encourages global agreement between representation spaces, while HeRA focuses on local topo-

logical structure through a contrastive proxy for neighborhood preservation. The comparison is important because the theoretical motivation of HeRA is not that the LLM should reproduce the exact teacher space, but that it should preserve useful local relations induced by the visual teacher.

5.8 Contrastive Alignment Variants

After observing the limitations of direct feature matching, we explored several variants based on contrastive alignment. These variants are closer to the final HeRA objective, since they do not require pointwise regression to the teacher features. Instead, they use the teacher visual encoder to define positive and negative relations between samples.

In all cases, nearest neighbors are computed in the representation space of the frozen visual teacher. The alignment objective then encourages the selected model representation to move closer to the teacher neighbors. The main difference between the variants lies in the choice of the model representation to align.

The first variant applies the contrastive objective to image tokens. This is a natural choice because the teacher is a visual encoder, and image tokens are part of the MLLM representation that is most directly connected to the visual input. The goal is to improve the visual grounding of the model by acting directly on the representations associated with image patches.

The second variant applies the contrastive objective to the EOS token of the entire sequence. The motivation is that the final EOS token can be interpreted as a compact summary of the multimodal context. If this token aggregates information from both image and text, aligning it with the teacher-induced structure may regularize the global representation used by the model.

A third variant applies alignment jointly to image and text representations. In this case, the loss is written as:

$$\mathcal{L}_{\text{joint}} = \lambda_{\text{img}}\mathcal{L}_{\text{img}} + \lambda_{\text{text}}\mathcal{L}_{\text{text}}, \tag{5.9}$$

Table 5.8: Contrastive alignment experiments on Qwen2.5-3B with SigLIP2 as the vision encoder. We compare different alignment configurations, including causal and bidirectional masks, worst- and top-head selection, image- and EOS-token alignment, and cross-modal or unimodal contrastive losses.

	General						Knowledge					OCR					Vision					
Model	***Avg***	*GQA*	*POPE*	*MME*	*MMB*	*SEED*	***Avg***	*SQA*	*MMMU*	*MathV*	*AI2D*	***Avg***	*ChartQA*	*OCRB*	*TextVQA*	*VizWiz*	***Avg***	*MMVP*	*RWQA*	*Blink*	*V**	*CVBench*
Qwen2.5-3B	**73.5**	63.8	87.7	1484.0	70.9	71.2	46.7	75.4	40.9	7.7	62.9	**42.2**	17.3	37.1	60.6	53.8	50.5	46.0	55.2	46.8	44.5	60.2
Qwen2.5-3B$^{\circ,\clubsuit,\dagger,\star}$	69.3	63.4	87.5	1471.1	69.9	71.2	45.9	74.4	39.3	7.0	62.9	36.7	16.8	35.0	58.2	55.0	52.8	42.0	54.4	46.7	41.9	57.8
Qwen2.5-3B$^{\circ,\clubsuit,\dagger,\blacklozenge}$	69.4	63.3	87.8	1459.3	70.0	71.1	46.2	74.2	40.4	7.1	63.2	37.4	17.0	34.8	60.4	56.0	54.1	41.3	55.3	47.4	46.6	60.4
Qwen2.5-3B$^{\circ,\clubsuit,\dagger,\blacklozenge}$	71.0	64.1	88.3	1504.1	72.1	72.2	47.0	75.5	40.8	7.8	64.0	40.0	20.0	38.4	61.5	56.5	**55.0**	41.3	55.2	**50.3**	47.1	61.2
Qwen2.5-3B$^{\circ,\clubsuit,\dagger,\star}$	71.1	63.8	88.0	**1535.1**	72.1	72.1	47.1	74.8	41.2	8.1	64.4	39.9	19.4	38.3	61.9	52.7	53.7	40.0	54.1	46.8	47.6	60.8
Qwen2.5-3B$^{\circ,\wr,\clubsuit,\dagger}$	70.8	63.8	88.2	1487.0	**73.1**	72.0	47.3	76.3	40.8	8.2	64.0	39.6	**20.4**	36.9	61.5	56.9	54.2	**46.7**	**56.2**	46.2	43.5	58.5
Qwen2.5-3B$^{\circ,\ddagger,\clubsuit,\dagger}$	69.4	63.8	88.1	1443.0	70.3	71.6	46.5	74.8	39.9	7.9	63.4	38.9	18.2	37.3	61.1	54.3	53.8	46.0	55.7	48.3	49.2	56.9
Qwen2.5-3B$^{\circ,\ddagger,\clubsuit,\dagger}$	70.1	64.0	87.5	1450.4	70.9	71.8	46.5	75.0	40.8	7.3	62.9	39.3	18.7	38.0	61.2	56.6	54.8	41.3	54.3	47.5	43.5	**62.2**
Qwen2.5-3B$^{\circ,\clubsuit,\dagger,*}$	71.1	**64.2**	**88.5**	1487.4	72.7	**72.4**	47.7	75.6	42.1	7.8	65.2	40.8	19.9	**39.6**	**62.8**	55.0	53.3	44.0	54.1	46.0	45.0	58.3
Qwen2.5-3B$^{\circ,\clubsuit,\clubsuit,\dagger,*}$	70.8	63.9	88.1	1473.0	73.0	72.3	**48.1**	76.2	**42.6**	**8.3**	**65.3**	40.2	19.2	39.0	62.4	54.5	54.0	44.7	54.9	47.4	45.0	58.5
Qwen2.5-3B$^{\circ,\clubsuit,\dagger,*}$	70.9	64.0	88.2	1477.4	72.5	71.9	47.8	**76.4**	41.6	8.0	**65.3**	40.4	19.6	39.4	62.1	54.3	54.3	**46.7**	55.6	46.3	**50.3**	58.5
Qwen2.5-3B$^{\circ,\clubsuit,\dagger,*}$	70.7	64.1	88.3	1487.4	72.2	72.1	47.4	75.2	41.8	8.0	64.8	40.1	19.3	38.4	62.5	**57.1**	**55.0**	44.0	56.0	45.9	48.2	60.8

$\wr$: causal mask; ‡: bidirectional mask; ♠: worst-head selection; ♣: top-head selection; ♦: image-token alignment; ⋆: EOS-token alignment; †: cross-modal contrastive loss; ∗: unimodal contrastive loss; ○: stage 1 with the proposed method and vanilla stage 2; ⋄: stage 2 with the proposed method.

where $\mathcal{L}_{\text{img}}$ is applied to the mean image-token representation and $\mathcal{L}_{\text{text}}$ is applied to the mean text-token representation. This setup tests whether alignment should regularize only the linguistic stream, only the visual stream, or both.

We also tested variants with separate logit scales for the image and text alignment losses. Since image and text representations may have different similarity distributions, using a shared logit scale can impose an artificial constraint on the two modalities. Separate logit scales allow each alignment term to adapt its own temperature. A full set of experiments is reported in Table 5.8.

These experiments help identify which internal representation is the most effective target for contrastive alignment. They also clarify why the final formulation of HeRA focuses on selected attention heads rather than applying the same objective uniformly to all tokens or to a single global sequence representation.

Table 5.9: Detailed VQA results for different neighborhood choices in Stage 1 and Stage 2.

	$\mathcal{N}_{\text{HeRA}}$		General						Knowledge					OCR					Vision					
Model	St.1	St.2	**Avg**	*GQA*	*POPE*	*MME*	*MMB*	*SEED*	**Avg**	*SQA*	*MMMU*	*MathV*	*AI2D*	**Avg**	*ChartQA*	*OCRB*	*TextVQA*	*VizWiz*	**Avg**	*MMVP*	*RWQA*	*Blink*	*V**	*CVBench*
Qwen2.5-3B	-	-	73.5	63.8	87.7	1484.0	70.9	71.2	46.7	75.4	40.9	7.7	62.9	42.2	17.3	37.1	60.6	53.8	50.5	46.0	55.2	46.8	44.5	**60.2**
+ HeRA	Worst	Worst	69.2	59.9	87.2	1374.0	64.4	65.8	45.4	74.6	39.3	6.6	61.0	36.6	13.4	27.6	52.2	53.1	44.7	31.3	51.8	45.6	39.3	55.6
+ HeRA	Worst	Random	69.0	59.3	86.9	1364.6	65.4	65.1	46.0	75.2	40.4	7.1	61.2	36.5	14.3	27.1	53.1	51.6	44.3	25.3	50.8	46.7	42.4	56.2
+ HeRA	Worst	Top	68.3	59.3	87.3	1318.9	63.9	64.8	45.5	73.9	40.6	6.0	61.3	34.3	13.3	25.4	50.9	47.5	43.5	26.7	49.8	45.6	38.7	56.8
+ HeRA	Random	Worst	73.8	64.0	**88.3**	1503.6	70.1	71.6	46.5	74.5	39.9	8.5	63.1	43.2	18.0	37.3	61.4	56.2	50.6	44.7	55.4	47.9	47.6	57.4
+ HeRA	Random	Random	73.3	63.8	87.9	1459.0	70.3	71.6	46.7	74.9	40.0	8.7	63.3	42.7	18.0	37.4	61.2	54.1	49.7	42.0	54.8	47.4	46.1	58.0
+ HeRA	Random	Top	73.7	63.7	88.1	1500.7	70.0	71.9	46.8	75.1	40.3	**9.0**	62.9	43.2	18.2	37.7	61.5	55.3	50.5	44.0	54.2	47.9	48.2	58.3
+ HeRA	Top	Worst	74.4	**64.3**	88.0	1510.3	**72.2**	**72.0**	**47.6**	75.2	**41.9**	8.0	**65.3**	**44.8**	19.7	**39.1**	**62.1**	**58.3**	52.3	47.3	56.5	48.4	**51.3**	58.0
+ HeRA	Top	Random	74.3	**64.3**	88.1	1508.5	72.1	71.9	47.5	**76.1**	41.2	7.9	64.6	44.4	20.1	38.6	**62.1**	56.9	52.4	**50.0**	**56.6**	47.9	48.7	58.8
+ HeRA (Ours)	**Top**	**Top**	**74.5**	64.1	87.9	**1525.5**	72.0	**72.0**	47.5	75.5	41.3	7.9	65.1	43.8	**20.2**	**39.1**	61.7	54.3	**52.9**	48.0	56.3	**49.1**	**51.3**	59.6

5.9 Neighborhoods Ablation

To better understand how the quality of the neighborhood signal affects the training dynamics of HeRA, we perform an ablation on the neighborhoods used in the two training stages. Differently from the previous experiments, where the same neighborhood construction is used throughout training, here we independently vary the neighborhood selection strategy in Stage 1 and Stage 2.

More precisely, for each stage we compare three alternatives. The *Top* configuration uses the nearest neighbors computed in the teacher vision-encoder space, and corresponds to the standard formulation of HeRA. The *Random* configuration replaces these neighbors with randomly sampled examples, removing the geometric structure induced by the teacher. Finally, the *Worst* configuration uses the least similar samples according to the teacher representation space, thus providing an intentionally adverse neighborhood signal.

We evaluate all pairwise combinations of these strategies across the two stages. Therefore, each Stage 1 configuration, namely *Top*, *Random*, and *Worst*, is followed by a Stage 2 training phase using again one of the same three neighborhood definitions. This results in a total of nine configurations. All other training details are kept fixed,

so that differences in performance can be attributed to the interaction between the neighborhood signal used during the initial alignment phase and the one used during visual instruction tuning.

This analysis allows us to disentangle two aspects of the method. First, it tests whether a meaningful neighborhood structure is already necessary during Stage 1, where the multimodal projector is adapted to connect visual and textual features. Second, it evaluates whether Stage 2 can recover from a noisy or adversarial Stage 1 signal, or whether the quality of the early alignment phase has a persistent effect on the final MLLM. A consistent advantage of configurations involving *Top* neighborhoods would support the hypothesis that HeRA benefits from preserving the local topology of the teacher representation space, rather than from the mere presence of an auxiliary contrastive regularization term. Results are reported in Table 5.9.

5.10 Discussion

The experiments presented in this chapter explore several alternatives to the final formulation of HeRA [9]. They are diagnostic experiments, designed to understand which design choices are most appropriate for representation alignment in MLLMs.

Direct feature matching provides the simplest form of alignment, but it is also the most restrictive. It forces the model representation to match the teacher feature direction for each sample, without explicitly accounting for the fact that the LLM and the vision encoder may organize information differently. This motivates the use of alignment objectives that preserve relational structure rather than exact feature values.

The projection experiments show that bridging the dimensionality gap between the LLM and the visual teacher is a non-trivial component of feature-level alignment. Learnable projections make feature matching possible, but they also introduce auxiliary parameters that are not used at inference. Interpolation, Stage 1.5 training, and convolutional projections all address this issue from different perspectives, but they also increase the complexity of the pipeline.

Patch importance analysis shows that the visual alignment signal is not necessarily uniform across patches. By ranking patches according to the attention they receive and aligning only the most relevant ones, we can study whether visual grounding is concentrated in salient regions or requires a broader preservation of visual structure. This analysis is particularly useful for understanding how much of the visual input actually contributes to alignment.

Finally, the CKA and contrastive variants compare different notions of representation similarity. CKA captures global similarity between representation spaces, while HeRA is based on preserving local neighborhood structure. The contrastive variants further show that the choice of alignment target matters: image tokens, EOS representations, and joint image-text averages encode different aspects of the multimodal input.

Overall, these experiments support the final design of HeRA [9]. Instead of relying on direct feature regression, HeRA uses a contrastive objective as a proxy for local topological alignment. Instead of aligning an entire layer or a generic sequence representation, it operates on selected attention heads. This makes the alignment more targeted and more consistent with the idea that different heads encode different aspects of the multimodal representation.

6. Conclusion

In this thesis, we presented **He**ad-Wise **R**epresentation **A**lignment (**HeRA**) [9], a method for improving Multimodal Large Language Models through targeted topological representation alignment. The motivation behind this work is that, despite their strong linguistic and reasoning abilities, current MLLMs still struggle when tasks require precise visual grounding. In these cases, models often rely too heavily on language priors, leading to errors in visual perception, spatial reasoning, OCR, and hallucination-sensitive scenarios.

The main contribution of this thesis is to rethink representation alignment as a topological regularization problem, rather than as a direct feature matching objective. Instead of forcing LLM activations to reproduce the exact features of a vision encoder, HeRA focuses on preserving local neighborhood relationships in the representation space. This choice is inspired by the Platonic Representation Hypothesis and by the MKNN alignment metric, which measures whether two representation spaces organize samples according to similar local structures. In this way, the visual teacher provides a structural supervision signal without requiring the language model to match the exact coordinates of the visual feature space.

A second central aspect of the method is the move from layer-wise to head-wise alignment. Previous representation alignment approaches usually apply supervision to a fixed layer of the language backbone, treating the layer as a single monolithic representation. In contrast, HeRA operates on individual attention heads, exploiting the finer organization of Transformer models. By decomposing the contribution of each head before it is merged into the residual stream, the method can apply visual regularization in a more localized and controlled way.

This head-wise formulation also makes it possible to use MKNN alignment scores as a diagnostic tool for deciding where the alignment objective should be applied. By computing these scores before multimodal training, we identify which heads are

already close to the visual domain and which ones are poorly aligned. Surprisingly, the best results are obtained by regularizing the *least* aligned heads. This suggests that these heads provide useful capacity for improving visual grounding, while the already aligned components of the model can be preserved.

The experimental results show that HeRA consistently improves MLLM performance across different architectures and model sizes. The gains are especially clear on vision-centric benchmarks, where stronger visual grounding is required. At the same time, the method preserves, and in several cases improves, general knowledge and OCR capabilities. This indicates that the alignment objective does not simply trade linguistic ability for visual performance but provides a targeted regularization signal that improves the multimodal interaction inside the model.

The hallucination evaluations further support this interpretation. Although HeRA is not explicitly optimized with a hallucination loss, models trained with our method show reduced hallucination rates on multiple benchmarks. This suggests that improving the internal visual alignment of the language backbone can reduce the tendency of MLLMs to answer from language priors alone, making their generations more faithful to the image content.

The ablation studies confirm the importance of the main design choices. Direct feature matching is less effective than topological alignment, layer-level alignment is weaker than head-level alignment, and random head selection does not provide the same benefits as MKNN-based selection. Moreover, aligning too many heads can reduce performance, showing that representation alignment should be applied selectively rather than indiscriminately.

Overall, this thesis shows that improving multimodal grounding does not necessarily require modifying the inference-time architecture or increasing the computational cost of the deployed model. HeRA introduces its additional components only during training: the teacher encoder, the MKNN ranking, and the contrastive alignment objective are not required at inference. The final model keeps the same structure as the baseline MLLM, while benefiting from a better organization of its internal

representations.

Future work could investigate dynamic head selection during training, the extension of head-wise topological alignment to other modalities such as video, or identify which modality is converging towards the other [98]. More broadly, this thesis supports the idea that representation analysis can be used not only to interpret MLLMs but also to guide their optimization in a more precise and structured way.

Publications

The research activity carried out in the context of the realization of this thesis led to the realization of a paper that has been submitted to the Fortieth Annual Conference on Neural Information Processing Systems (NeurIPS 2026).

Bibliography

[1] Jean-Baptiste Alayrac et al. “Flamingo: a visual language model for few-shot learning”. In: *NeurIPS*. 2022.

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

[3] Shuai Bai et al. *Qwen3-VL Technical Report*. 2025.

[4] Shuai Bai et al. “Qwen3-VL Technical Report”. In: *arXiv preprint arXiv:2511.21631* (2025).

[5] Loïc Barrault et al. “Large concept models: Language modeling in a sentence representation space”. In: *arXiv preprint arXiv:2412.08821* (2024).

[6] Thomas Bertolani et al. “Diffusion Language Models: An Experimental Analysis”. In: *arXiv preprint arXiv:2606.19475* (2026).

[7] Garrick Brazil et al. “Omni3D: A Large Benchmark and Model for 3D Object Detection in the Wild”. In: *CVPR*. 2023.

[8] Davide Bucciarelli et al. “Tiny Inference-Time Scaling with Latent Verifiers”. In: *CVPR*. 2026.

[9] Davide Caffagni et al. “Mind the Heads: Topological Representation Alignment for Multimodal LLMs”. In: *arXiv preprint arXiv:2606.23885* (2026).

[10] Davide Caffagni et al. “Seeing Beyond Words: Self-Supervised Visual Learning for Multimodal Large Language Models”. In: *arXiv preprint arXiv:2512.15885* (2025).

[11] Davide Caffagni et al. “The revolution of multimodal large language models: A survey”. In: *ACL*. 2024.

[12] Davide Caffagni et al. “Wiki-llava: Hierarchical retrieval-augmented generation for multimodal llms”. In: *CVPR*. 2024.

[13] Tianle Cai et al. “Medusa: Simple llm inference acceleration framework with multiple decoding heads”. In: *ICML* (2024).

[14] Silvia Cappelletti et al. “Improving LLM First-Token Predictions in Multiple-Choice Question Answering via Prefilling Attack”. In: *arXiv preprint arXiv:2505.15323* (2025).

[15] Mathilde Caron et al. “Emerging properties in self-supervised vision transformers”. In: *CVPR*. 2021.

[16] Wei-Lin Chiang et al. *Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality*. 2023.

[17] Alberto Compagnoni et al. “Mitigating hallucinations in multimodal llms via object-aware preference optimization”. In: *BMVC*. 2025.

[18] Alberto Compagnoni et al. “Reag: Reasoning-augmented generation for knowledge-based visual question answering”. In: *CVPR*. 2026.

[19] Fabio D’Oronzio et al. “GramSR: Visual Feature Conditioning for Diffusion-Based Super-Resolution”. In: *ICPR*. 2026.

[20] Alexey Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *ICLR*. 2021.

[21] David Fan et al. “Scaling Language-Free Visual Representation Learning”. In: *ICCV*. 2025.

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

[23] Xingyu Fu et al. “BLINK: Multimodal Large Language Models Can See But Not Perceive”. In: *ECCV*. 2024.

[24] Yulu Gan, Kaiya Ivy Zhao, and Phillip Isola. “Cross-Modal Alignment Regularization: Enhancing Language Models with Vision Model Representations”. In: *ICLR Workshops*. 2025.

[25] Jonas Gehring et al. “Convolutional sequence to sequence learning”. In: *ICML*. 2017.

[26] Ross Girshick. “Fast r-cnn”. In: *ICCV*. 2015.

[27] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *CVPR*. 2014.

[28] Aaron Grattafiori et al. “The Llama 3 Herd of Models”. In: *arXiv preprint arXiv:2407.21783* (2024).

[29] Fabian Gröger, Shuo Wen, and Maria Brbić. “Revisiting the Platonic Representation Hypothesis: An Aristotelian View”. In: *ICML*. 2026.

[30] Albert Gu and Tri Dao. “Mamba: Linear-time sequence modeling with selective state spaces”. In: *arXiv preprint arXiv:2312.00752* (2023).

[31] Tianrui Guan et al. “HallusionBench: An Advanced Diagnostic Suite for Entangled Language Hallucination and Visual Illusion in Large Vision-Language Models”. In: *CVPR*. 2024.

[32] Danna Gurari et al. “VizWiz Grand Challenge: Answering Visual Questions From Blind People”. In: *CVPR*. 2018.

[33] Kaiming He et al. “Deep residual learning for image recognition”. In: *CVPR*. 2016.

[34] Drew A Hudson and Christopher D Manning. “GQA: A New Dataset for Real-World Visual Reasoning and Compositional Question Answering”. In: *CVPR*. 2019.

[35] Minyoung Huh et al. “The Platonic Representation Hypothesis”. In: *ICML*. 2024.

[36] Aniruddha Kembhavi et al. “A diagram is worth a dozen images”. In: *ECCV*. 2016.

[37] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *NeurIPS* (2012).

[38] Xingjian Leng et al. “REPA-E: Unlocking VAE for End-to-End Tuning with Latent Diffusion Transformers”. In: *ICCV*. 2025.

[39] Bohao Li et al. “SEED-Bench: Benchmarking Multimodal LLMs with Generative Comprehension”. In: *arXiv preprint arXiv:2307.16125* (2023).

[40] Junnan Li et al. “Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models”. In: *ICML*. 2023.

[41] Junnan Li et al. “Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation”. In: *ICML*. 2022.

[42] Yifan Li et al. “Evaluating Object Hallucination in Large Vision-Language Models”. In: *EMNLP*. 2023.

[43] Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *ECCV*. 2014.

[44] Haotian Liu et al. “Improved Baselines with Visual Instruction Tuning”. In: *CVPR*. 2024.

[45] Haotian Liu et al. “Visual instruction tuning”. In: *NeurIPS*. 2023.

[46] Yuan Liu et al. “MMBench: Is Your Multi-modal Model an All-around Player?” In: *ECCV*. 2024.

[47] Yuliang Liu et al. “OCRBench: On the Hidden Mystery of OCR in Large Multimodal Models”. In: *Sci China Inf Sci* 67.12 (2024), p. 220102.

[48] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization”. In: *ICLR*. 2019.

[49] Pan Lu et al. “Learn to Explain: Multimodal Reasoning via Thought Chains for Science Question Answering”. In: *NeurIPS*. 2022.

[50] Pan Lu et al. “MathVista: Evaluating Mathematical Reasoning of Foundation Models in Visual Contexts”. In: *ICLR*. 2024.

[51] Ahmed Masry et al. “ChartQA: A Benchmark for Question Answering about Charts with Visual and Logical Reasoning”. In: *ACL*. 2022.

[52] Federico Melis et al. “Harnessing Self-Supervised Features for Art Classification”. In: *arXiv preprint arXiv:2605.18974* (2026).

[53] Marco Morini et al. “Look Twice: Training-Free Evidence Highlighting in Multimodal Large Language Models”. In: *arXiv preprint arXiv:2604.01280* (2026).

[54] Andrew Joohun Nam et al. “Causal Head Gating: A Framework for Interpreting Roles of Attention Heads in Transformers”. In: *NeurIPS*. 2025.

[55] Catherine Olsson et al. “In-Context Learning and Induction Heads”. In: *arXiv preprint arXiv:2209.11895* (2022).

[56] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. “Representation Learning with Contrastive Predictive Coding”. In: *arXiv preprint arXiv:1807.03748* (2018).

[57] Maxime Oquab et al. “DINOv2: Learning Robust Visual Features without Supervision”. In: *TMLR* (2024).

[58] Tobia Poppi et al. “Do Models Share Safety Representations? Cross-Model Steering for Safe Visual Generation”. In: *arXiv preprint arXiv:2606.05290* (2026).

[59] Qwen Team. “Qwen2.5 Technical Report”. In: *arXiv preprint arXiv:2412.15115* (2024).

[60] Qwen Team. *Qwen3.5: Towards Native Multimodal Agents*. 2026. URL: https://qwen.ai/blog?id=qwen3.5.

[61] Alec Radford et al. “Learning transferable visual models from natural language supervision”. In: *ICML*. 2021.

[62] Hanoona Rasheed et al. “GLaMM: Pixel Grounding Large Multimodal Model”. In: *CVPR*. 2024.

[63] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *CVPR*. 2016.

[64] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *NeurIPS* (2015).

[65] Oriane Siméoni et al. “DINOv3”. In: *arXiv preprint arXiv:2508.10104* (2025).

[66] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[67] Amanpreet Singh et al. “Flava: A foundational language and vision alignment model”. In: *CVPR*. 2022.

[68] Amanpreet Singh et al. “Towards VQA Models That Can Read”. In: *CVPR*. 2019.

[69] Jianlin Su et al. “Roformer: Enhanced transformer with rotary position embedding”. In: *Neurocomputing* (2024).

[70] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. “Sequence to sequence learning with neural networks”. In: *NeurIPS* (2014).

[71] Christian Szegedy et al. “Going deeper with convolutions”. In: *CVPR*. 2015.

[72] Shengbang Tong et al. “Beyond Language Modeling: An Exploration of Multimodal Pretraining”. In: *arXiv preprint arXiv:2603.03276* (2026).

[73] Shengbang Tong et al. “Cambrian-1: A Fully Open, Vision-Centric Exploration of Multimodal LLMs”. In: *NeurIPS*. 2024.

[74] Shengbang Tong et al. “Eyes Wide Shut? Exploring the Visual Shortcomings of Multimodal LLMs”. In: *CVPR*. 2024.

[75] Hugo Touvron et al. “Llama 2: Open Foundation and Fine-Tuned Chat Models”. In: *arXiv preprint arXiv:2307.09288* (2023).

[76] Hugo Touvron et al. “LLaMA: Open and Efficient Foundation Language Models”. In: *arXiv preprint arXiv:2302.13971* (2023).

[77] Michael Tschannen et al. “SigLIP 2: Multilingual Vision-Language Encoders with Improved Semantic Understanding, Localization, and Dense Features”. In: *arXiv preprint arXiv:2502.14786* (2025).

[78] Maria Tsimpoukelli et al. “Multimodal few-shot learning with frozen language models”. In: *NeurIPS*. 2021.

[79] Evelyn Turri et al. “Few Channels Draw The Whole Picture: Revealing Massive Activations in Diffusion Transformers”. In: *arXiv preprint arXiv:2605.13974* (2026).

[80] Ashish Vaswani et al. “Attention is all you need”. In: *NeurIPS* (2017).

[81] Haochen Wang et al. “Reconstructive Visual Instruction Tuning”. In: *ICLR*. 2025.

[82] Junyang Wang et al. “AMBER: An LLM-free Multi-dimensional Benchmark for MLLMs Hallucination Evaluation”. In: *arXiv preprint arXiv:2311.07397* (2023).

[83] Kevin Ro Wang et al. “Interpretability in the Wild: a Circuit for Indirect Object Identification in GPT-2 Small”. In: *ICLR*. 2023.

[84] Weiyun Wang et al. “InternVL3.5: Advancing Open-Source Multimodal Models in Versatility, Reasoning, and Efficiency”. In: *arXiv preprint arXiv:2508.18265* (2025).

[85] Zirui Wang et al. “SimVLM: Simple Visual Language Model Pretraining with Weak Supervision”. In: *ICLR*. 2022.

[86] Luis Wiedmann et al. “FineVision: Open Data Is All You Need”. In: *arXiv preprint arXiv:2510.17269* (2025).

[87] Penghao Wu and Saining Xie. “V*: Guided Visual Search as a Core Mechanism in Multimodal LLMs”. In: *CVPR*. 2024.

[88] xAI. *Grok*. 2024. URL: https://x.ai/blog/grok-1.5v.

[89] An Yang et al. *Qwen2 Technical Report*. 2024.

[90] An Yang et al. *Qwen3 Technical Report*. 2025.

[91] An Yang et al. "Qwen3 Technical Report". In: *arXiv preprint arXiv:2505.09388* (2025).

[92] Heeji Yoon et al. "Visual Representation Alignment for Multimodal Large Language Models". In: *arXiv preprint arXiv:2509.07979* (2025).

[93] Fisher Yu and Vladlen Koltun. "Multi-scale context aggregation by dilated convolutions". In: *arXiv preprint arXiv:1511.07122* (2015).

[94] Jiahui Yu et al. "CoCa: Contrastive Captioners are Image-Text Foundation Models". In: *TMLR* (2022).

[95] Sihyun Yu et al. "Representation Alignment for Generation: Training Diffusion Transformers Is Easier Than You Think". In: *ICLR*. 2025.

[96] Xiang Yue et al. "MMMU: A Massive Multi-discipline Multimodal Understanding and Reasoning Benchmark for Expert AGI". In: *CVPR*. 2024.

[97] Zihao Yue, Liang Zhang, and Qin Jin. "Less is More: Mitigating Multimodal Hallucination from an EOS Decision Perspective". In: *ACL*. 2024.

[98] Zhaoyang Zhang et al. "The Wittgensteinian Representation Hypothesis: Is Language the Attractor of Multimodal Convergence?" In: *arXiv preprint arXiv:2605.09352* (2026).

[99] Lianmin Zheng et al. "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena". In: *NeurIPS*. 2023.

[100] Bolei Zhou et al. "Semantic Understanding of Scenes through the ADE20K Dataset". In: *IJCV* 127.3 (2019), pp. 302–321.

[101] Leonardo Zini et al. "vHector and HeisenVec: Scalable Vector Graphics Generation Through Large Language Models". In: *NeurIPS* (2026).