Multimodal Foundation Models and Efficient Adaptation Techniques

October 22, 2025

Contents

1 CLIP, Flamingo, GPT-4, LLaVA

Multimodal foundation models align vision and language representations at scale, enabling zero-shot recognition, image-conditioned generation, and interactive reasoning. Figure **??** compares embedding spaces learned by representative architectures.

1.1 Contrastive Language-Image Pretraining (CLIP)

CLIP pairs an image encoder f_θ and text encoder g_ϕ trained on 400M image-text pairs. Given batch size B, embeddings $\mathbf{v}_i = f_\theta(\mathbf{x}_i)$ and $\mathbf{t}_i = g_\phi(\mathbf{y}_i)$ are normalized and optimized via symmetric cross-entropy:

$$\ell_{\text{img}} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp(\mathbf{v}_i^\top \mathbf{t}_i/\tau)}{\sum_{j=1}^{B} \exp(\mathbf{v}_i^\top \mathbf{t}_j/\tau)}, \tag{1}$$

$$\ell_{\text{text}} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp(\mathbf{t}_i^\top \mathbf{v}_i/\tau)}{\sum_{j=1}^{B} \exp(\mathbf{t}_i^\top \mathbf{v}_j/\tau)}, \tag{2}$$

$$\mathcal{L}_{\text{CLIP}} = \frac{1}{2}(\ell_{\text{img}} + \ell_{\text{text}}). \tag{3}$$

Temperature τ is learned, improving sharpness of similarity scores. Zero-shot classification replaces linear classifiers with text prompts, $\hat{y} = \arg\max_k \mathbf{v}_{\text{test}}^\top \mathbf{t}_k$, where $\mathbf{t}_k$ encodes "a photo of a {label}".

1.2 Flamingo: Perceiver-based Vision-Language Model

Flamingo combines a frozen vision encoder and language model via gated cross-attention layers (Gated XAttn-Dense). Given token sequence $\mathbf{h}_{\text{LM}}$ and visual features $\mathbf{h}_{\text{vis}}$, a layer computes

$$\mathbf{z} = \text{MultiHeadQK}(\mathbf{h}_{\text{LM}}, \mathbf{h}_{\text{vis}}), \tag{4}$$

$$\mathbf{m} = \sigma(\mathbf{W}_g[\mathbf{h}_{\text{LM}}, \mathbf{z}]) \odot \mathbf{W}_m\mathbf{z}, \tag{5}$$

$$\mathbf{h}'_{\text{LM}} = \mathbf{h}_{\text{LM}} + \mathbf{m}, \tag{6}$$

where σ is sigmoid gating. The Perceiver Resampler distills variable-length visual tokens into a fixed set of latent vectors, enabling few-shot multimodal learning with minimal task-specific tuning.

1.3 GPT-4 and Multimodal Extensions

GPT-4 integrates visual understanding by fusing image embeddings through multimodal adapters. While architectural details remain proprietary, public descriptions emphasize:

- Vision encoder producing patch tokens passed to a projection layer aligning with transformer embeddings.
- Joint positional encodings allowing interleaved text and visual tokens.
- Reinforcement learning from human feedback (RLHF) on multimodal conversations.

The model excels at reasoning over charts, diagrams, and complex instructions, marking the shift toward generalist assistants.

1.4 LLaVA: Large Language and Vision Assistant

LLaVA fine-tunes a frozen CLIP vision encoder and Vicuna language model with visual instruction data. A projection matrix W_{proj} maps pooled visual features $\mathbf{v}$ to language hidden size d:

$$\tilde{\mathbf{v}} = W_{\text{proj}}\mathbf{v}, \quad \mathbf{H}_0 = [\text{BOS}, \tilde{\mathbf{v}}, \text{Text tokens}]. \tag{7}$$

Training uses a two-stage approach:

1. **Visual instruction tuning:** SFT on GPT-generated dialogues describing images.
2. **Alignment refinement:** Preference optimization (e.g., DPO) to align responses with human preferences.

The model demonstrates strong performance on benchmarks such as ScienceQA and VizWiz.

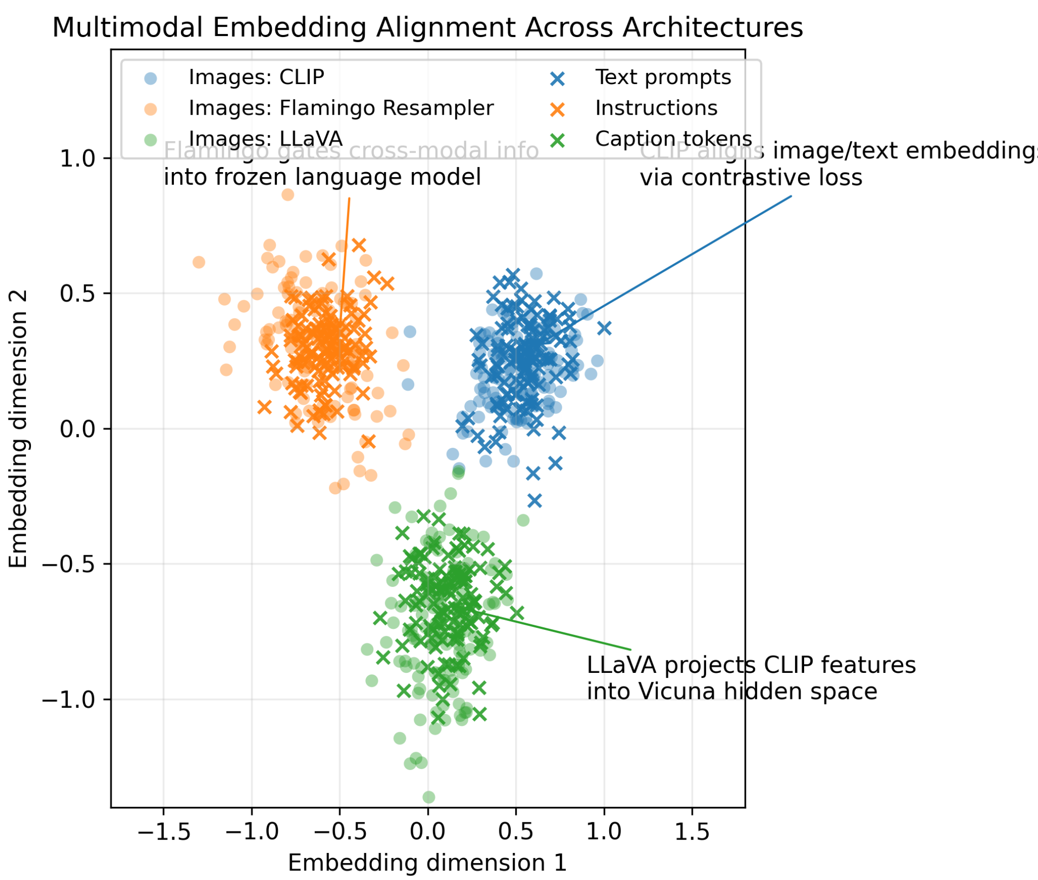

Figure 1: Embedding geometry for CLIP, Flamingo, GPT-4, and LLaVA. CLIP learns aligned spaces; Flamingo and LLaVA bridge visual features into language models.

2 Training and Fine-tuning Large Models: LoRA, PEFT, RAG

Scaling foundation models imposes prohibitive costs for full fine-tuning. Parameter-efficient fine-tuning (PEFT) techniques adapt pre-trained backbones with lightweight modules, while retrieval-augmented generation (RAG) grounds responses in external knowledge. Figures **??** and **??** illustrate key mechanisms.

2.1 Low-Rank Adaptation (LoRA)

LoRA freezes original weights $\mathbf{W}_0$ and learns low-rank updates $\Delta\mathbf{W} = \mathbf{BA}$ with rank $r \ll d$:

$$\mathbf{W} = \mathbf{W}_0 + \frac{\alpha}{r}\mathbf{BA}, \quad \mathbf{A} \in \mathbb{R}^{r \times d_{\text{in}}},\ \mathbf{B} \in \mathbb{R}^{d_{\text{out}} \times r}. \tag{8}$$

During forward pass for hidden states $\mathbf{h}$:

$$\mathbf{y} = \mathbf{W}_0\mathbf{h} + \frac{\alpha}{r}\mathbf{B}(\mathbf{Ah}). \tag{9}$$

Only $\mathbf{A}, \mathbf{B}$ are trained, reducing parameter count by $\mathcal{O}(r(d_{\text{in}} + d_{\text{out}}))$. Rank selection balances expressivity and storage; common values are $r \in \{4, 8, 16\}$.

2.2 Prefix/Prompt Tuning and AdapterFusion

PEFT encompasses multiple strategies:

- **Prefix tuning:** Optimizes virtual tokens prepended to each layer's key/value matrices.
- **Prompt tuning:** Adjusts continuous prompt embeddings at the input layer only.
- **Adapters:** Inserts bottleneck MLPs with residual connections. AdapterFusion learns task-specific mixtures of previously trained adapters, enabling multi-task reuse.

Libraries such as Hugging Face PEFT unify these approaches, allowing composition (e.g., LoRA + prompt tuning).

2.3 Retrieval-Augmented Generation (RAG)

RAG mitigates hallucinations by retrieving documents $\{\mathbf{d}_k\}_{k=1}^{K}$ from index $\mathcal{D}$ conditioned on query $\mathbf{q}$:

$$\mathbf{d}_k = \text{Retrieve}(\mathbf{q}, \mathcal{D}), \qquad \mathbf{y} \sim p_\theta(\mathbf{y} \mid \mathbf{q}, \mathbf{d}_{1:K}). \tag{10}$$

Dense retrievers (DPR, Contriever) encode queries/documents via bi-encoders trained with contrastive loss. Generation integrates retrieved passages via:

- **Fusion-in-decoder (FiD):** Concatenate encoder outputs per passage before cross-attention.
- **RAG-token/RAG-sequence:** Marginalize over retrieved documents during autoregressive decoding.

Adaptive retrieval refreshes indexes periodically, while caching strategies reduce latency in production.

2.4 Implementation Example

Listing 1: LoRA fine-tuning with retrieval-augmented prompting (Hugging Face PEFT).

```python
from peft import LoraConfig, get_peft_model
from transformers import AutoModelForCausalLM, AutoTokenizer
from rag_pipeline import DenseRetriever, format_context

base_model = AutoModelForCausalLM.from_pretrained("meta-llama/Llama-2-7b-hf",
    device_map="auto")
```

```python
tokenizer = AutoTokenizer.from_pretrained("meta-llama/Llama-2-7b-hf")

lora_config = LoraConfig(
    r=8,
    lora_alpha=32,
    lora_dropout=0.1,
    target_modules=["q_proj", "v_proj"],
)
model = get_peft_model(base_model, lora_config)

retriever = DenseRetriever(index_path="faiss.index")

def generate_with_rag(question: str):
    docs = retriever.search(question, top_k=5)
    prompt = format_context(question, docs)
    inputs = tokenizer(prompt, return_tensors="pt").to(model.device)
    outputs = model.generate(**inputs, max_new_tokens=256, temperature=0.7)
    return tokenizer.decode(outputs[0], skip_special_tokens=True)
```

2.5 Production Considerations

- **Memory footprint:** LoRA weights stored separately ($\sim$ tens of MB) support rapid model switching.
- **Evaluation:** Align with human preference tests; offline Rouge/BLEU insufficient for conversational agents.
- **Safety:** Retrieval filters and response vetting prevent leakage of undesired content.

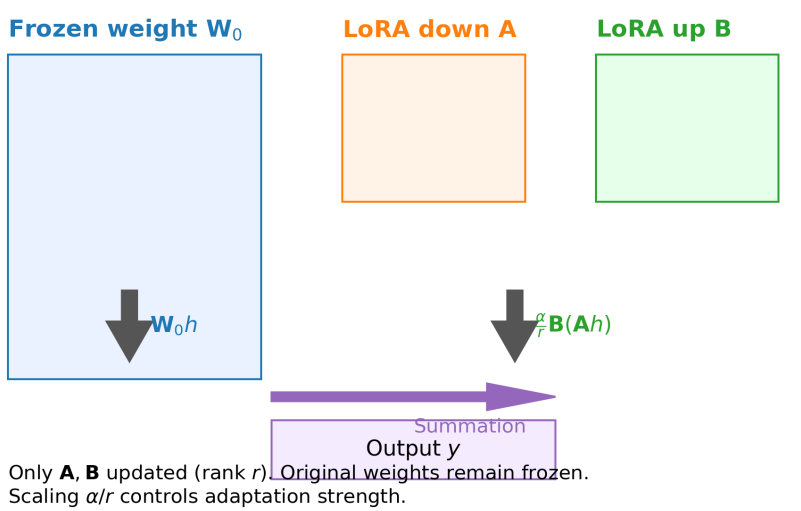

Figure 2: LoRA inserts low-rank adapters into attention projections. Rank and scaling determine adaptation capacity.

Figure 3: Retrieval-augmented generation pipeline with dense retriever, chunked index, and generator fusion strategies.

Further Reading

- Alec Radford et al. “Learning Transferable Visual Models From Natural Language Supervision.” ICML 2021.
- Jean-Baptiste Alayrac et al. “Flamingo: A Visual Language Model for Few-Shot Learning.” NeurIPS 2022.
- Jonathan Ho et al. “Scaling Instruction-Finetuned Language Models.” 2022.
- Edward Raffel et al. “Scaling Instruction-Finetuned Language Models.” 2023.
- Tianyi Zhang et al. “PEFT: Parameter-Efficient Fine-Tuning of Transformers.” 2022.