Exercise Session 8

Multi-Modal Architectures

Prepared by Lukas Klein and Benedikt von Querfurth

Overview

Optional Background Information. This exercise session is about Contrastive Language-Image Pre-Training (CLIP, Paper) and Large Language and Vision Assistant (LLaVA, Paper). We will discuss how such encoder and decoder-based multi-modal architectures work, how they are connected, and how multi-modal architectures beyond images and text could look like.

It is not required to read the respective papers, and it is encouraged to first write down your thoughts for the "understanding the concepts" questions before doing so.

Task 1. CLIP: Understanding the Concepts.

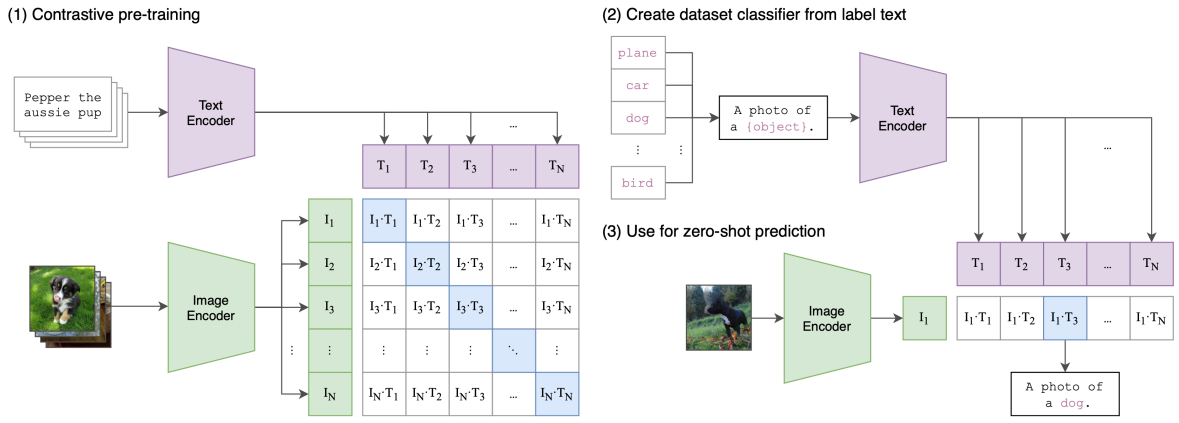

Figure 1: Overview of CLIP during training (1) and inference (2, 3).

CLIP works by jointly training two encoders, a text encoder and an image encoder, to map corresponding text-image pairs into a shared embedding space. During pre-training (Figure 1, step 1), the model learns to maximize the similarity between correct image-text pairs (along the diagonal) while minimizing similarity between mismatched pairs, creating aligned representations. Once trained, CLIP can be used as a zero-shot classifier by encoding candidate class

labels as text embeddings (step 2) and comparing them with an input image's embedding—the class whose text embedding has the highest similarity to the image embedding is predicted as the result (step 3). This approach enables flexible classification without requiring task-specific training data, as shown where the dog image is correctly matched to the "dog" label.

(a) Why does CLIP need to train on mismatched image-text pairs (the off-diagonal elements) during contrastive pre-training? What would happen if the model only learned from correct pairs along the diagonal?

(b) Explain why CLIP can classify images into categories it has never explicitly been trained on (zero-shot learning). How is this different from a traditional supervised image classifier trained on ImageNet?

(c) You want to build a CLIP-based classifier to distinguish between 'happy dogs', 'sad dogs', and 'angry dogs'. How would you design the text prompts for the best performance, and why might simply using the labels 'happy', 'sad', and 'angry' not work well?

(d) What applications of CLIP beyond natural images and text can you think of?

Task 2. CLIP: Zero-shot Classification and Evaluation of CLIP (Coding).

See IPython Notebook.

Task 3. LLaVA: Understanding the Concepts.

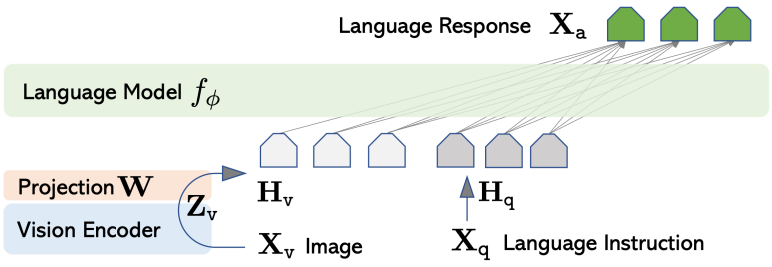

Figure 2: Overview of the LLaVA architecture.

LLaVA works by combining a pre-trained vision encoder with a pre-trained language model to enable multimodal understanding (see Figure 2). The vision encoder processes the input image $\mathbf{X}_v$ into visual features $\mathbf{Z}_v$, which are then transformed through a trainable projection layer $\mathbf{W}$ into visual token embeddings $\mathbf{H}_v$ that match the language model's input space. These visual tokens are concatenated with text instruction embeddings $\mathbf{H}_q$ (derived from the language instruction $\mathbf{X}_q$) and fed into the language model f_ϕ as a unified sequence. The language model then processes this combined visual-linguistic input and generates an appropriate text response $\mathbf{X}_a$, treating the projected image features as if they were additional "tokens" in its vocabulary. This

architecture allows LLaVA to perform visual question answering and instruction-following tasks by leveraging the capabilities of both frozen pre-trained components, with only the projection layer **W** needing to be trained to align the visual and language representations.[1]

(a) Both CLIP and LLaVA connect vision and language, but they serve different purposes. Explain the key architectural and functional differences between CLIP and LLaVA. Specifically, why can LLaVA generate free-form text responses while CLIP is limited to computing similarity scores?

(b) Why does LLaVA need a trainable projection layer W between the vision encoder and the language model? Why are the visual features Z_v in most cases not directly fed into the language model (ignoring that you could also fine-tune the full vision model), and what would happen if we tried to do so?

(c) In LLaVA pre-training, typically only the projection layer W is trained (with the vision encoder and language model remaining frozen) before any end-to-end fine-tuning of the language model. Explain the advantages and potential limitations of this training approach compared to fine-tuning all components directly end-to-end.

(d) *Bonus Question*: CellWhisperer (Paper) is an approach to translate the LLaVA architecture to the biological domain to "chat" with your single cell RNA sequence data. How do you think they modified and trained CLIP and LLaVA for chat-based interrogation of transcriptome (RNA) data in English language?

Task 4. LLaVA: Attention Analysis (Coding).

See IPython notebook.

---

[1] Audio-To-Text, Vision-to-Vision, Audio-To-Vision, etc. multimodal LLMs are all based on the similar to LLaVA concept.