Lecture 5.2 - Vision Transformer and CLIP

Generative AI Teaching Kit

This lecture

- Transformers Review
- Vision Transformers
- Contrastive Language-Image Pretraining (CLIP)
- Bootstrapping Language-Image Pre-training (BLIP)

Transformers: A quick recap

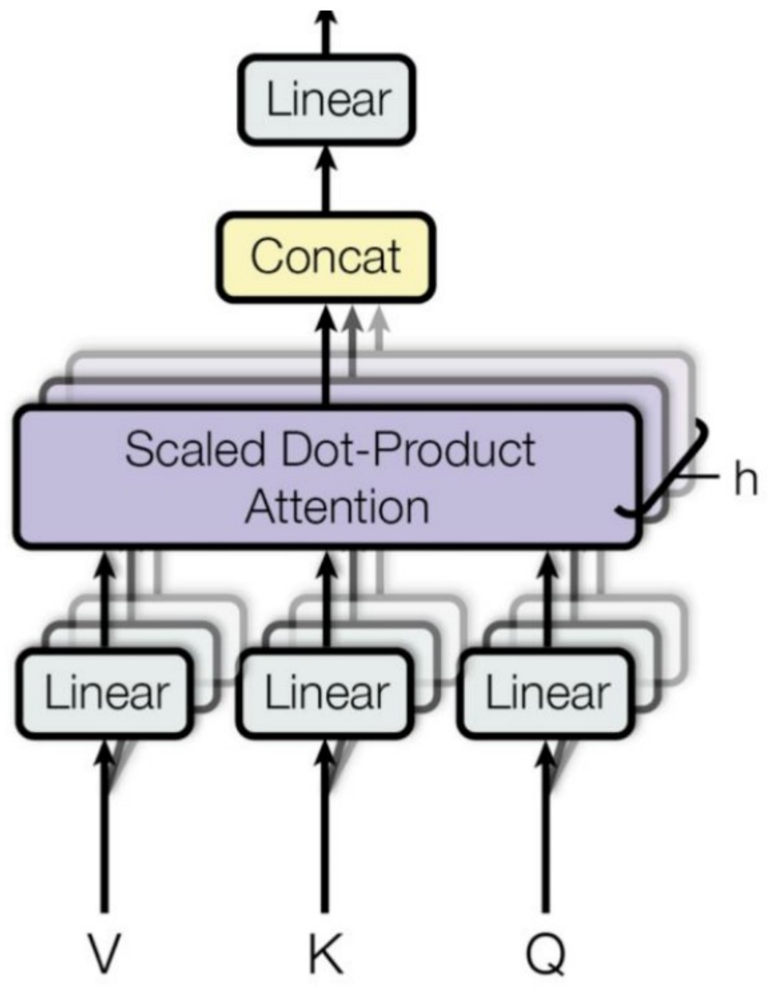
Recap on Transformers – Self Attention

Core Idea of Self-Attention
Each token in a sequence attends to every other word, learning contextual relationships. This allows the model to understand dependencies regardless of distance within the sequence.

How Self-Attention Works
Query (Q): What this word (or token) is “asking about” in the context of other words.
Key (K): The “properties” of other words that might be important.
Value (V): The actual content from each word that contributes to the final output.

Attention Calculation

- Scaled Dot-Product: Computes similarity between Q and K for each pair, highlighting key relationships.
- Softmax Function: Normalizes these scores into probabilities for how much focus to place on each word.
- Weighted Sum: Combines these scores with V to produce the attention-weighted representation.



Recap on Transformers – Encoder/Decoder

Two-Part Structure: Encoder and Decoder

- Encoder: Processes the input sequence (e.g., a sentence in English).
- Decoder: Generates the output sequence (e.g., a translated sentence in French).

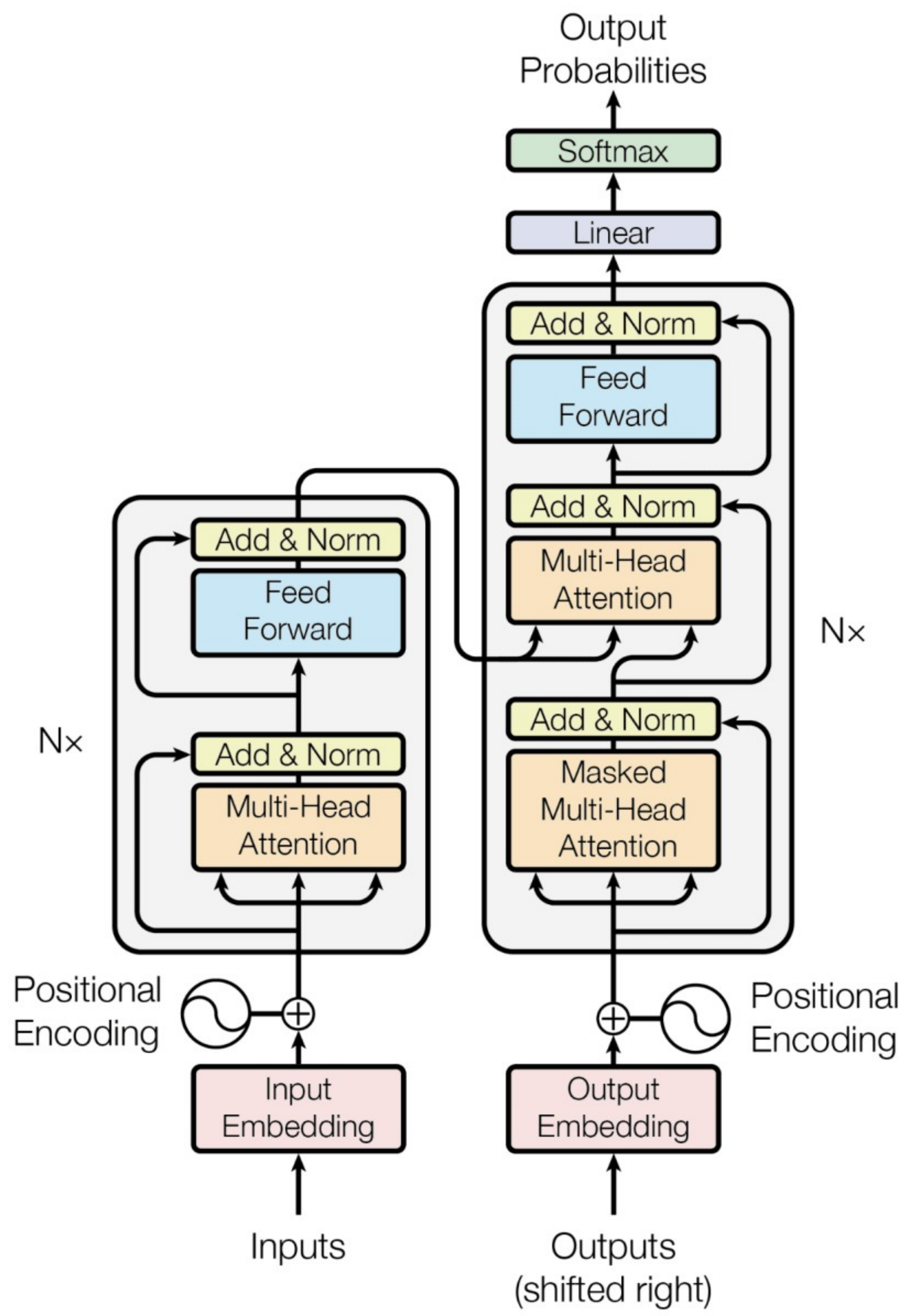
This separation allows for effective processing of complex language tasks.

Encoder Breakdown

- Self-Attention Layer: Each word attends to every other word in the input sequence, capturing relationships and context.
- Feedforward Layer: Applies transformations to each position independently adding nonlinear depth.
- Positional Encoding: Adds information about word order since self-attentior is order-agnostic.

Decoder Breakdown

- Masked Self-Attention: Each word only attends to previous words in the sequence, preventing it from "seeing" future tokens.
- Encoder-Decoder Attention: Allows each position in the output sequence tc attend to all positions in the encoder's output, linking input and output sequences.
- Feedforward Layer: Applies non-linear transformations, helping refine the final output.



Vision Transformers

Learning to see with attention

Vision Transformers – Image Patches as Tokens

Patching Concept

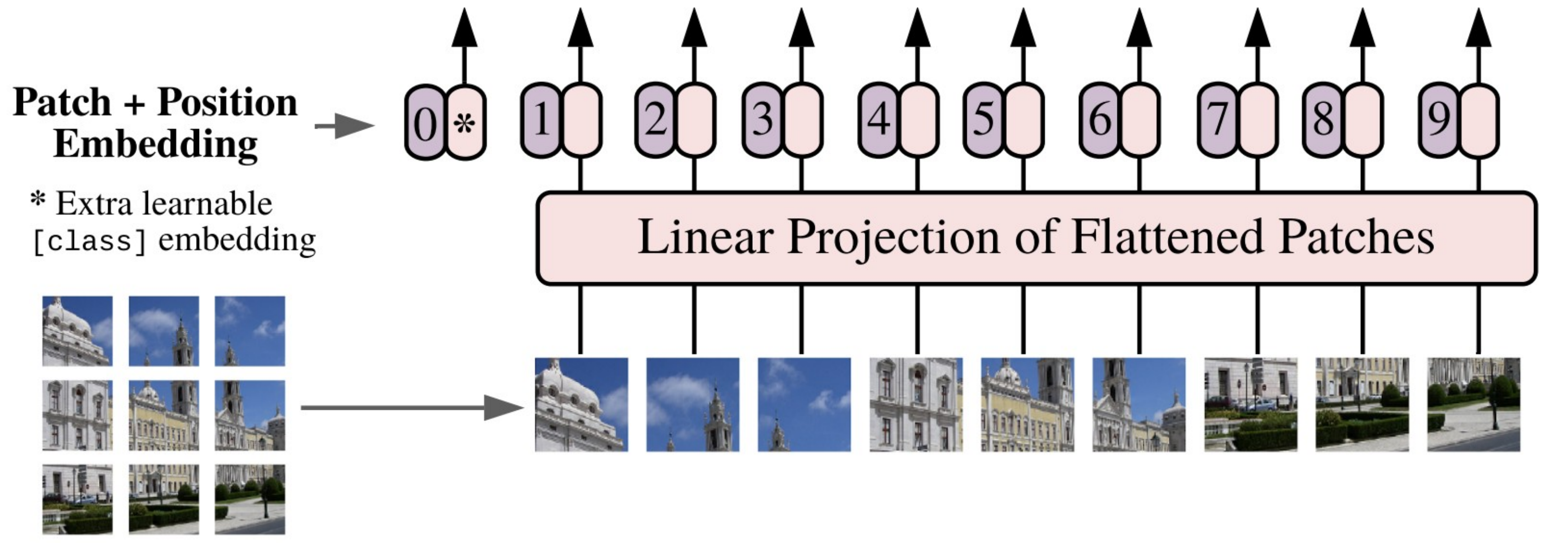
Vision Transformers (ViTs) introduce the idea of “patches” by dividing an image into smaller, fixed-size patches (e.g., 16x16 pixels). These patches are then treated as tokens, similar to words in an NLP transformer.

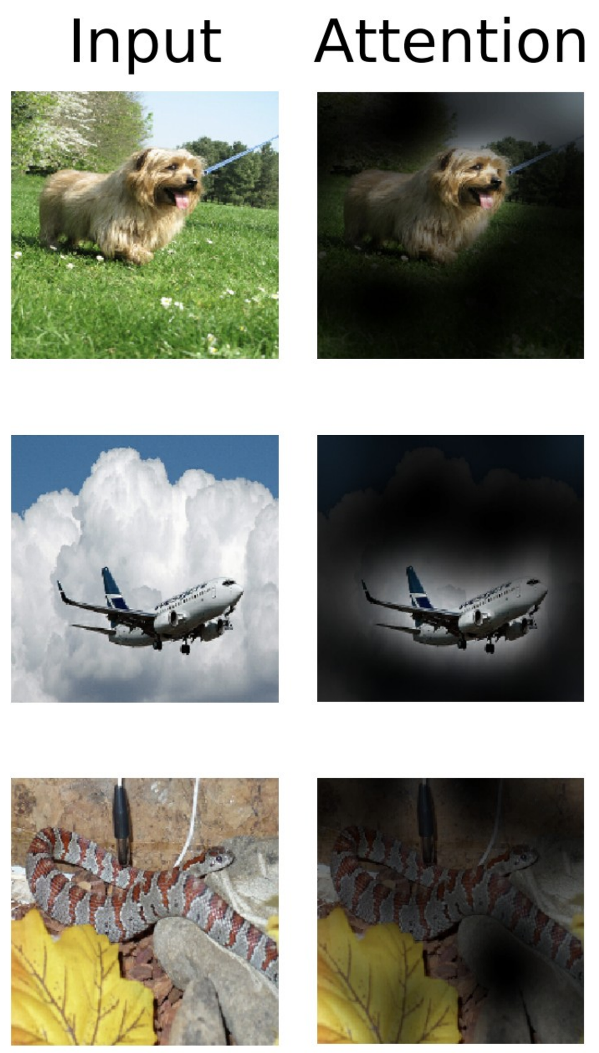
Why Patching is Crucial

- By using patches, ViTs can avoid convolutional layers, making them simpler and potentially more versatile than CNN-based models.
- This approach enables the model to capture both local and global context within an image through the self-attention mechanism, enhancing its ability to understand complex visual relationships.

Comparison with CNNs

CNNs use local receptive fields, which focus on parts of the image incrementally, whereas transformers with global attention can analyze the entire image context at once.


Patch + Position Embedding
* Extra learnable [class] embedding
0 *
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches



Vision Transformers – Architecture

Patch Embeddings

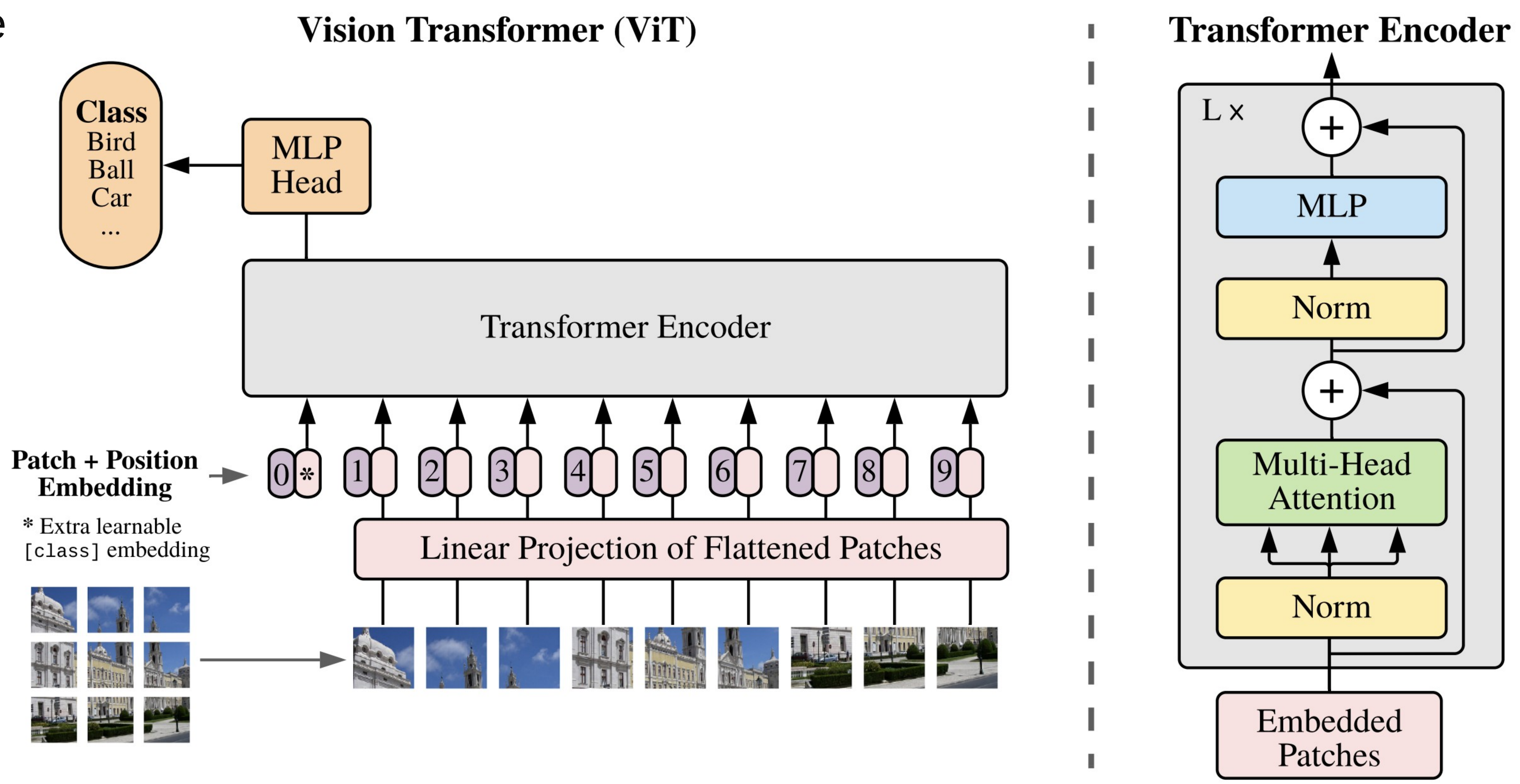
Each patch is linearly embedded into a fixed-size vector, making it compatible with the transformer's processing layers.

Positional Encoding

Since transformers lack a natural understanding of spatial arrangement, positional encodings are added to each patch embedding to provide spatial context.

Attention Mechanism in Vision

The multi-headed self-attention mechanism enables the model to assess relationships between patches across the entire image.



DARTMOUTH ENGINEERING | NVIDIA

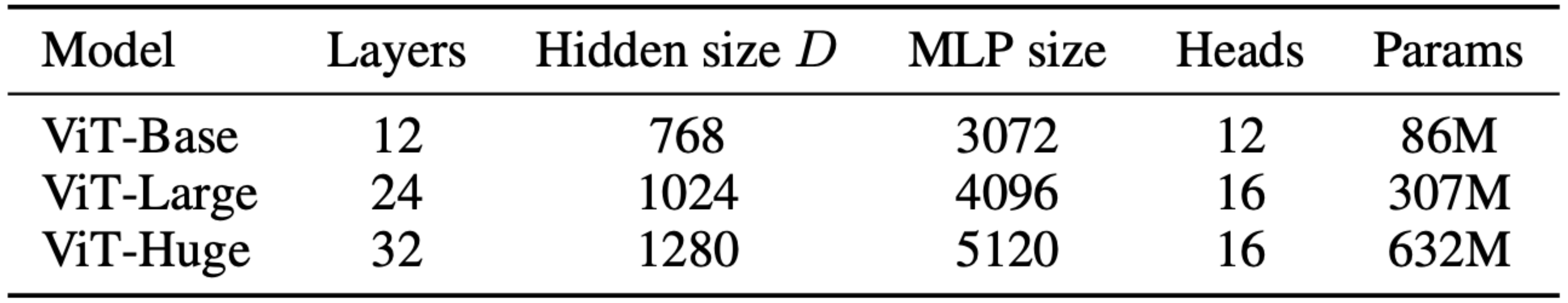
Vision Transformers – Training

Data Requirements

- Vision transformers, unlike CNNs, generally require large amounts of data to perform well, especially for visual recognition tasks.
- ViTs benefit from transfer learning, where they are pretrained on large datasets and then fine-tuned for specific tasks.

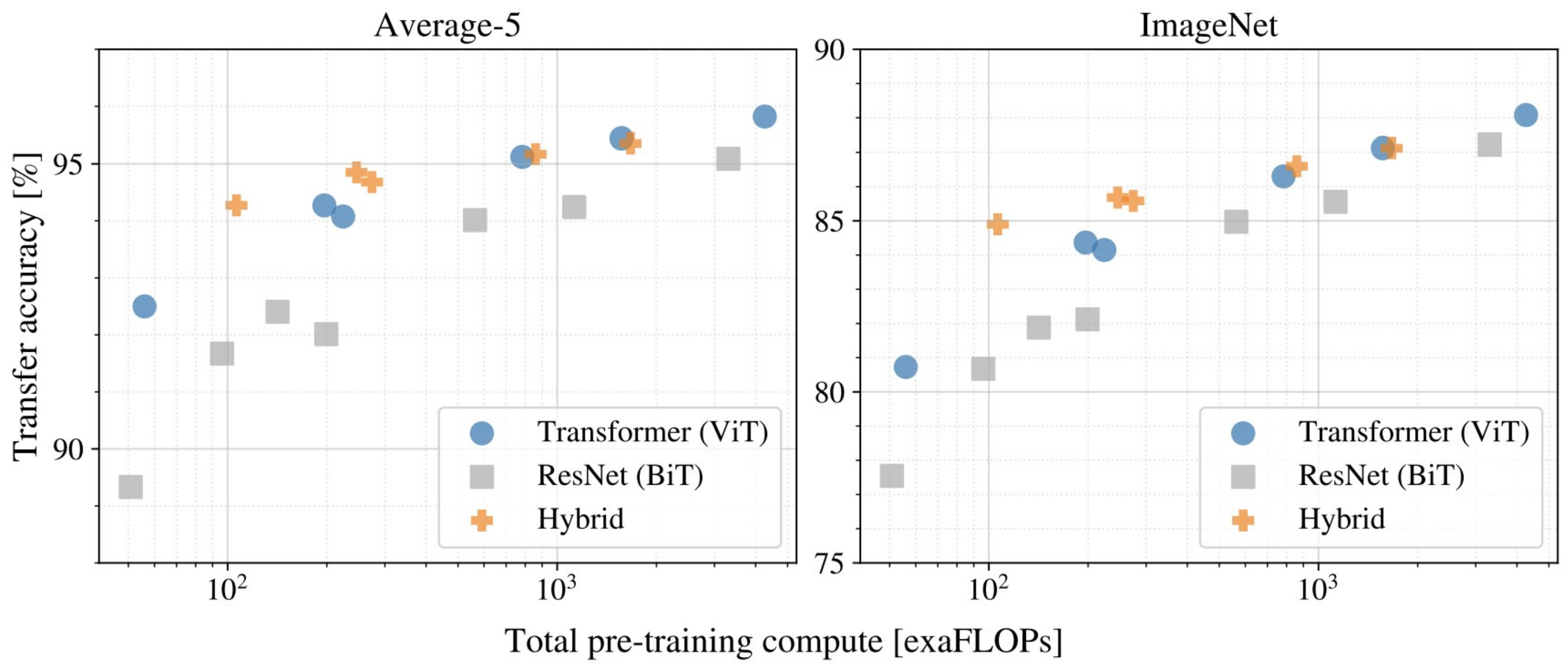
Computational Demands

ViTs are computationally intensive due to their use of self-attention, which scales quadratically with the numbe of patches. This can make them more demanding than CNNs for large images or high-resolution inputs.



Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

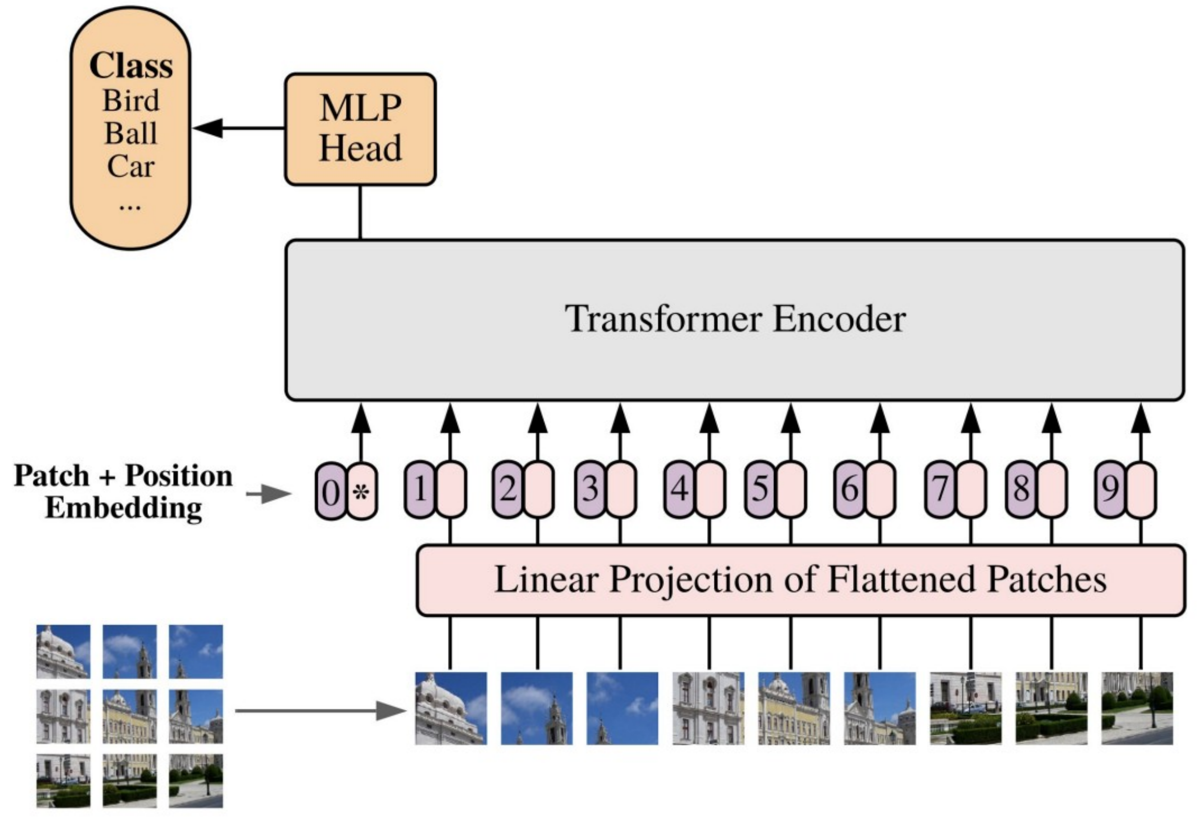
Table 1: Details of Vision Transformer model variants.

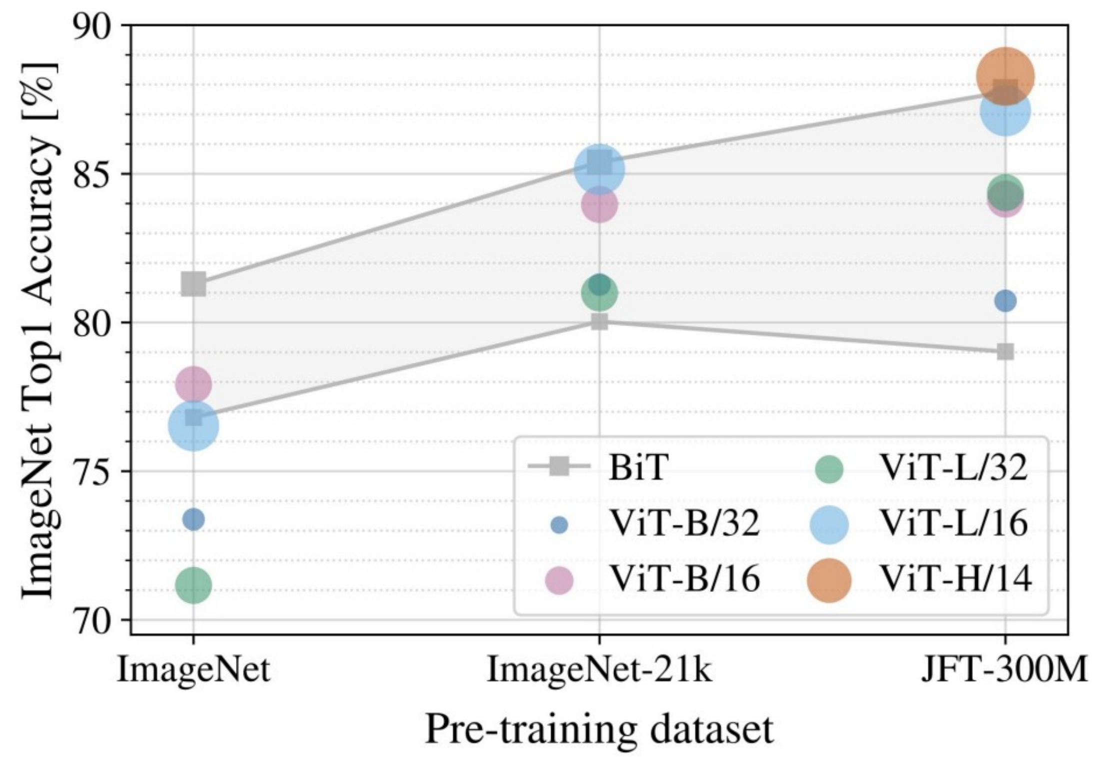


Vision Transformers – Usage as a classifier

Usage of ViT

- On their own, ViTs are similar to encoder LLMs like BERT
- During training, the ViT is attached to a classification head which allows the transformer to use the encoded vectors for some task
- ViTs **don’t create images** by generating tokens, this is typically done with Diffusion models, which leverage ViTs as an encoder backbone
- As with most transformer-based models, while simpler architectures can do better on smaller datasets, ViTs scale well with data and outperform traditional vision models on classification tasks





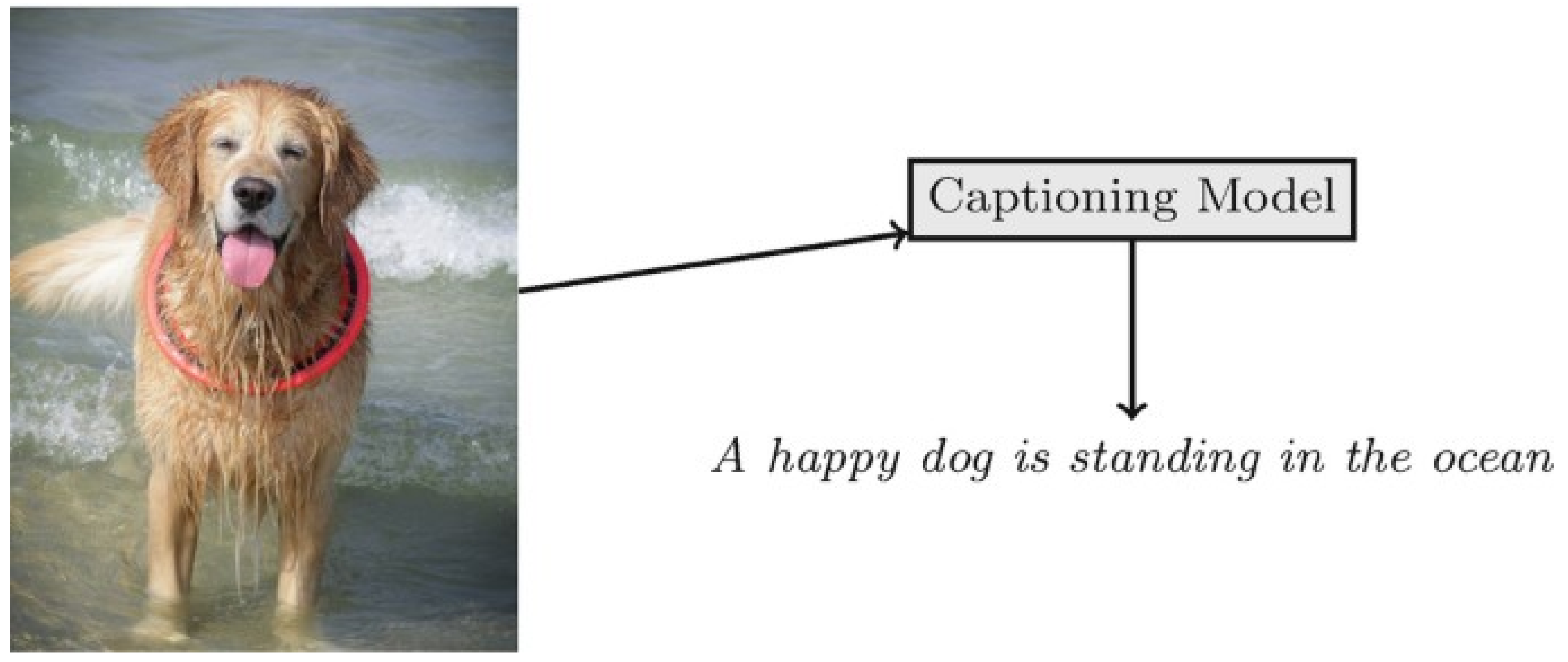
Contrastive Language-Image Pretraining (CLIP)

Say what you see

Image Captioning – A multimodal task

Image Captioning requires understanding objects, actions, and context in the image. ViTs are excellent at recognizing patterns and identifying objects but lack the ability to generate descriptive language based on these patterns.

- ViTs focus on visual features without a language model to interpret relationships (e.g., “dog sitting on grass”). They can identify objects, but they don’t naturally “understand” how these objects interact or relate in a way that makes sense in language.
- Language has a sequential structure (grammar, syntax, etc.), which ViTs aren’t equipped to handle. Captioning needs a model that understands not only what’s in an image but also how to order words in a coherent way—something beyond the scope of ViTs alone.



One approach to solving this problem is to make use of an image encoder **and** a text encoder…

CLIP – Dual Encoder Design

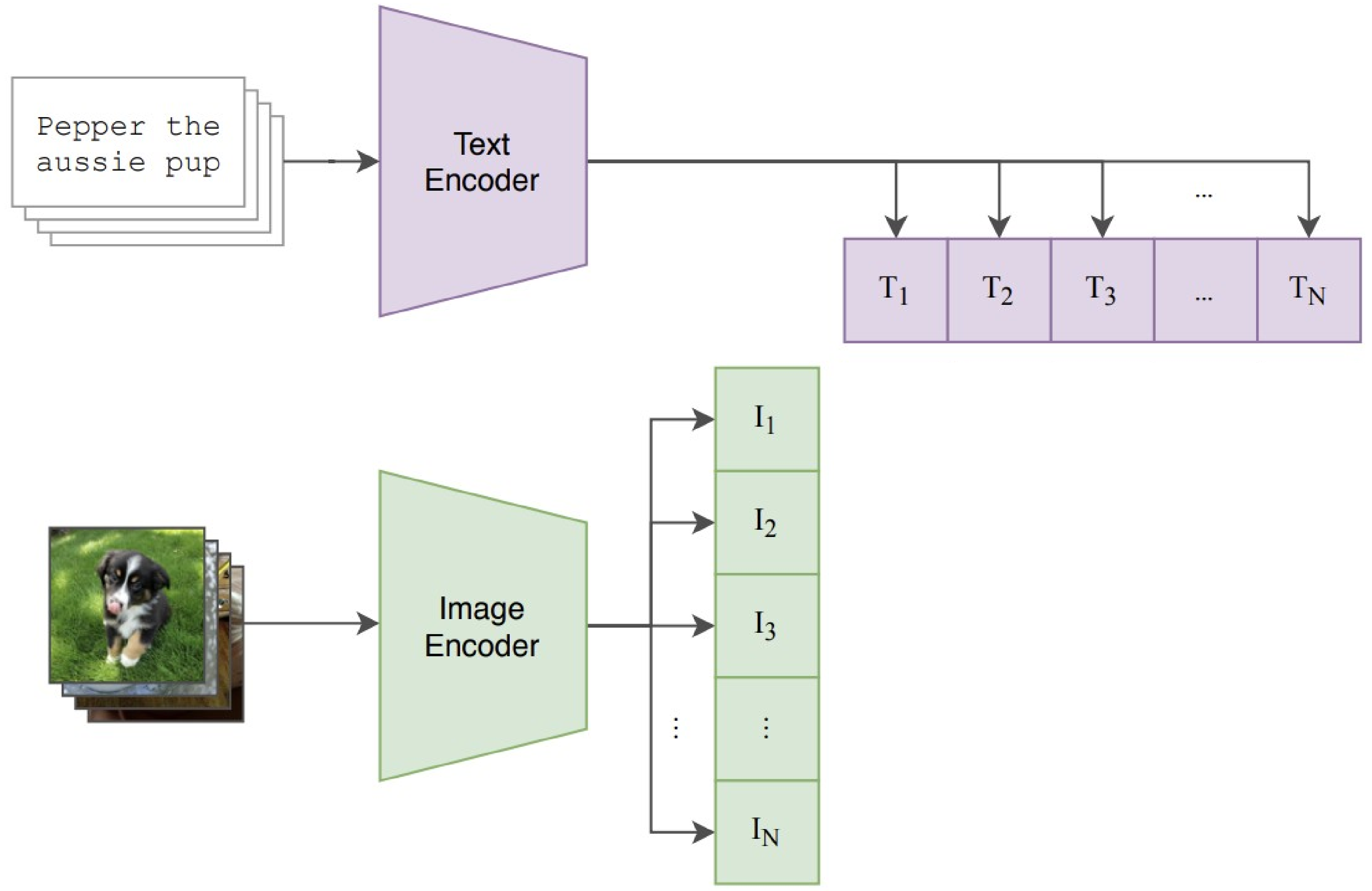
CLIP uses two encoders: one for **text** and one for **images**. Each encoder independently processes its respective modality, mapping both into a shared latent space.

Text and Image Encoders

The image encoder is typically a Vision Transformer or ResNet, and the text encoder is a transformer-based model.

Why Dual Encoders?

- These two encoders allow CLIP to compare the similarity of text-image pairs without requiring direct interactions between modalities during encoding.
- Enables efficient retrieval tasks, as embeddings can be precomputed and indexed for fast similarity searches.



CLIP – Contrastive Learning

Contrastive Loss Function

- CLIP uses a contrastive loss to maximize similarity between matching text-image pairs while minimizing it between non-matching pairs.
- This training method aligns embeddings in the shared latent space, ensuring that relevant text-image pairs are close together.

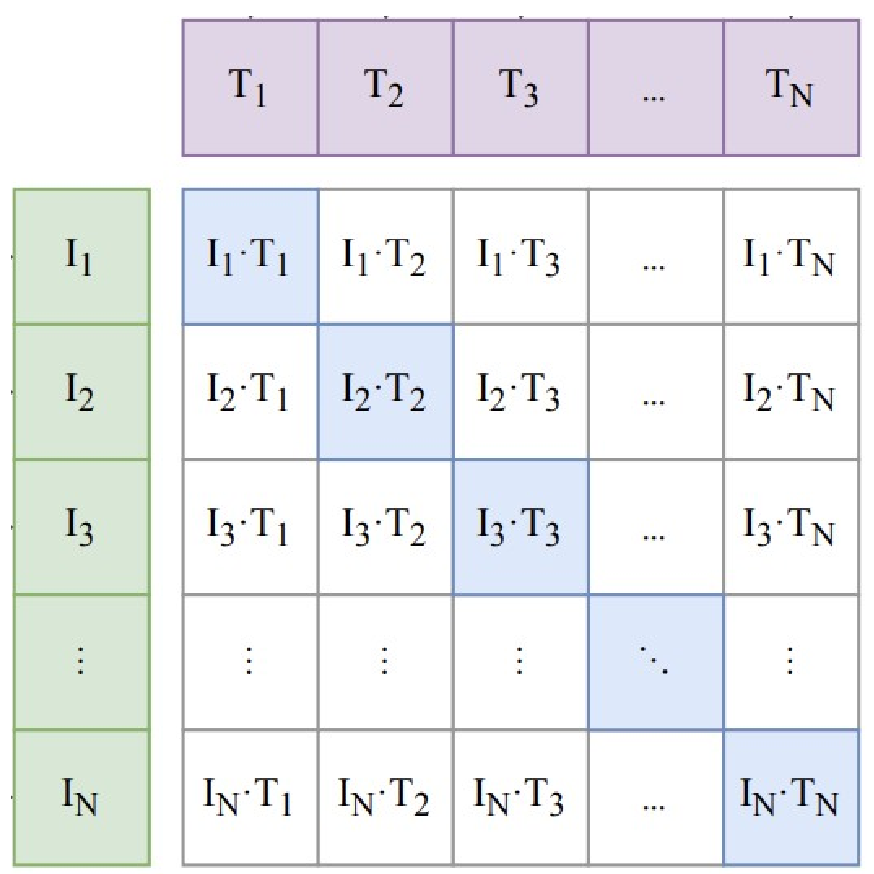
$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} {}_{[k\neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)},$$

Batchwise Contrastive Training

- During training, a batch of images and texts is used where each image is paired with a corresponding text description, and the contrastive loss is applied across the batch.
- Discuss the importance of large batch sizes to capture a wide range of text-image relationships.

Impact on Zero-Shot Learning

The contrastive training approach allows CLIP to generalize well to unseen classes, which contributes to its zero-shot capabilities.



T_1 T_2 T_3 ... T_N
I_1 $I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
I_2 $I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ ... $I_2 \cdot T_N$
I_3 $I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ ... $I_3 \cdot T_N$
$\vdots$ $\vdots$ $\vdots$ $\vdots$ $\ddots$ $\vdots$
I_N $I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ ... $I_N \cdot T_N$

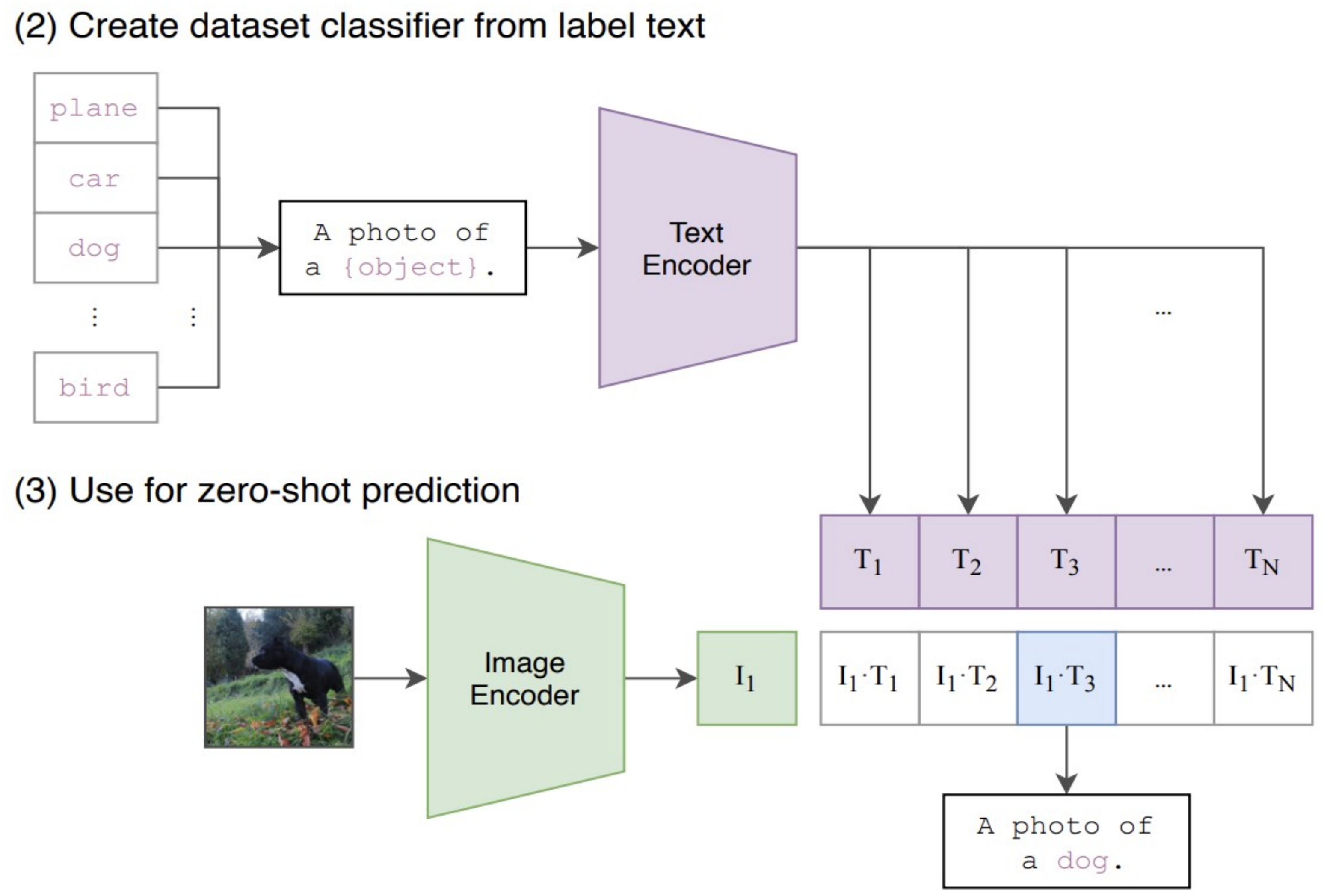
CLIP and Zero Shot

What is Zero-Shot Learning?

CLIP is pre-trained to predict if an image and a text snippet are paired together in its dataset. To perform zero-shot classification, this capability can be reused. For each dataset, the names of all the classes in the dataset is used as the set of potential text pairings and predict the most probable (image, text) pair according to CLIP.

Generalization Across Domains

Unlike traditional supervised models that require task-specific fine-tuning, CLIP can generalize to diverse tasks like image classification, object detection, and more with simple text prompts.



Building CLIP

Scaling Properties

CLIP's performance improves with more data, larger models, and higher-quality embeddings, following scaling laws similar to those observed in language models.

Data and Compute Challenges

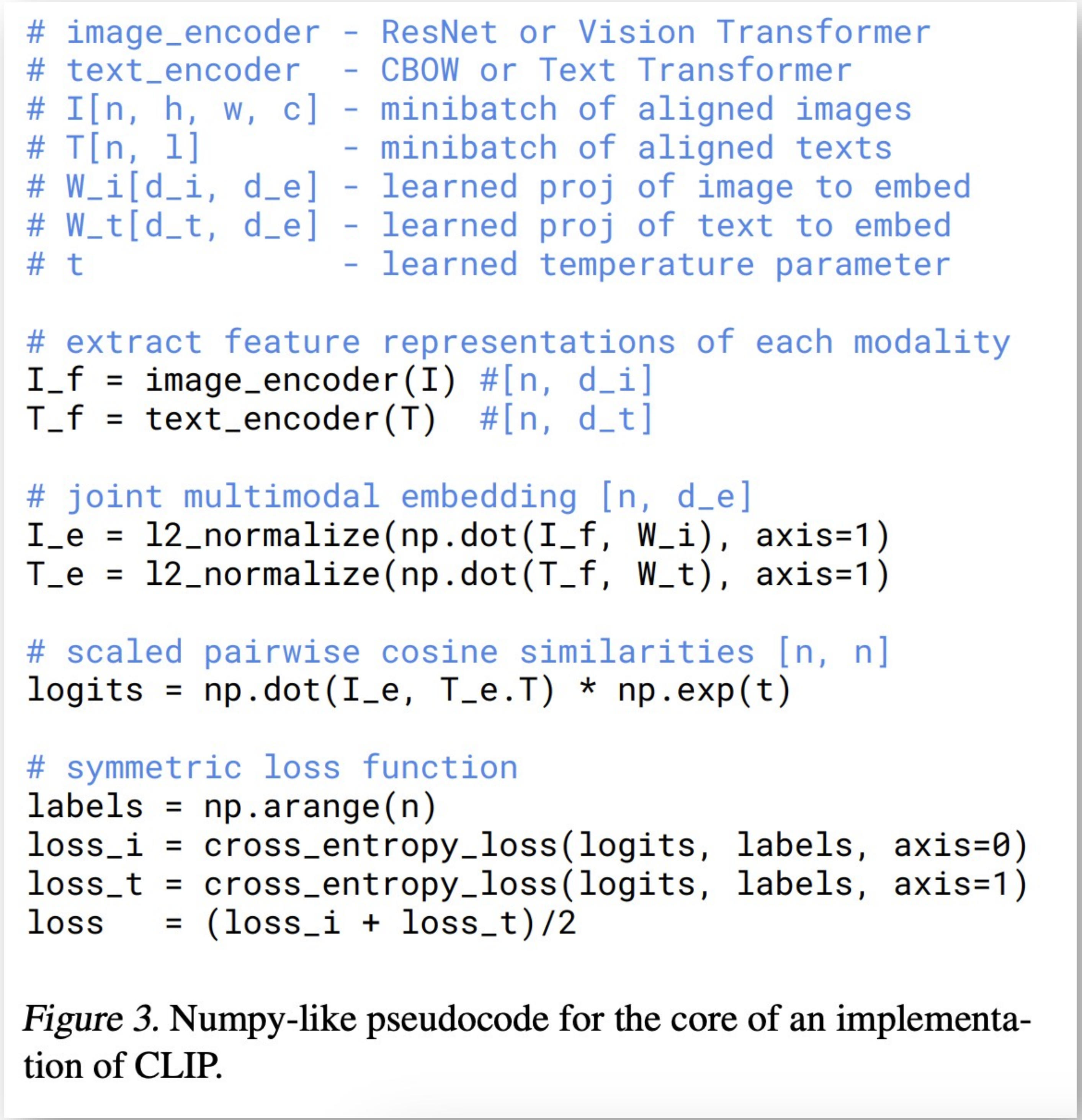
Training CLIP requires extensive computational resources and large multimodal datasets. Briefly address the implications for research and commercial use.

Limitations and Future Directions

While CLIP is fantastic for pairing images and text in a static way, it's limited in generating or interpreting more detailed language about an image (e.g., complex descriptions or answering nuanced questions).

CLIP's focus on joint embeddings of image-text pairs is somewhat rigid, as it lacks a generative language capability, making it less suitable for tasks like captioning, visual question answering (VQA), and interactive applications that require deeper understanding and flexibility.

Next we will see a new method, BLIP which addresses these issues.



```
# image_encoder - ResNet or Vision Transformer
# text_encoder  - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

Figure 3. Numpy-like pseudocode for the core of an implementation of CLIP.

Bootstrapping Language-Image Pre-training (BLIP)

Filtering out the noise

What is the Bootstrapping Language-Image Pretraining model?

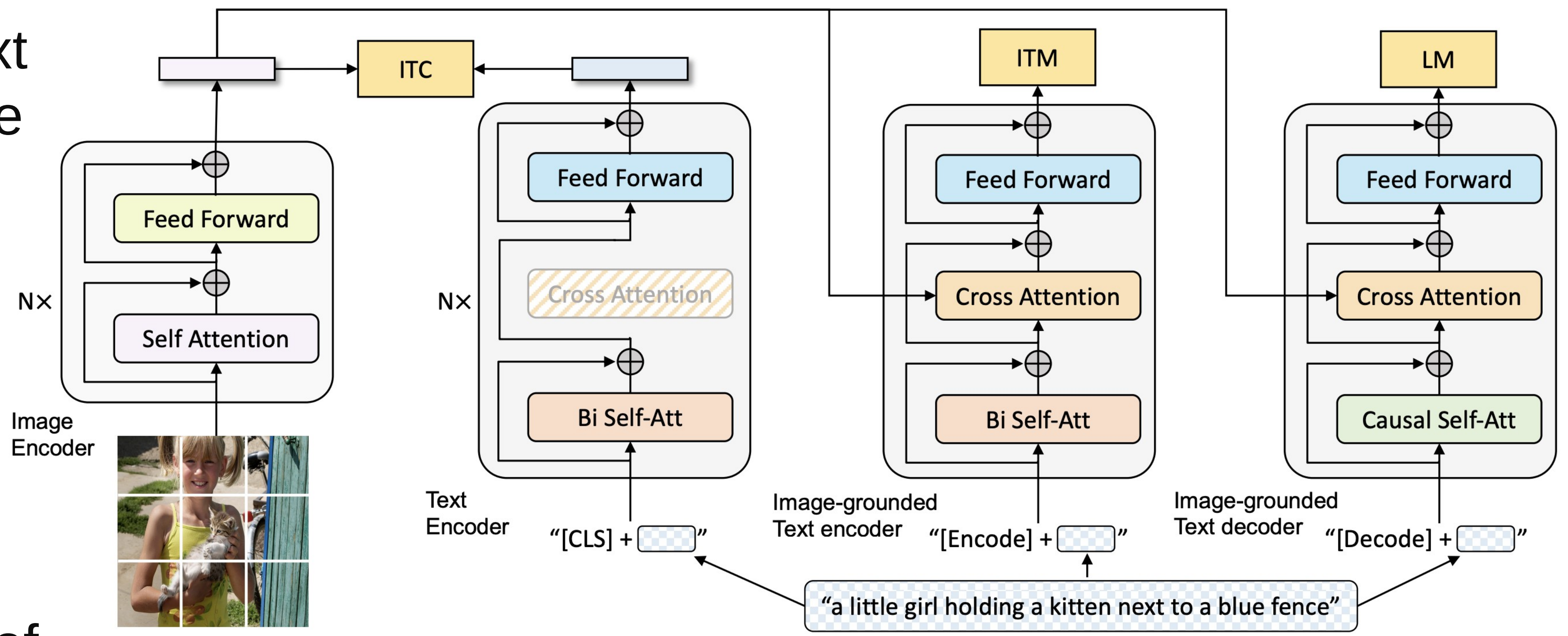
While **CLIP** is fantastic for pairing images and text in a static way, it's limited in generating or interpreting more detailed language about an image (e.g., complex descriptions or answering nuanced questions).

CLIP's focus on joint embeddings of image-text pairs is somewhat rigid, as it lacks a generative language capability, making it less suitable for tasks like captioning, visual question answering (VQA), and interactive applications that require deeper understanding and flexibility.

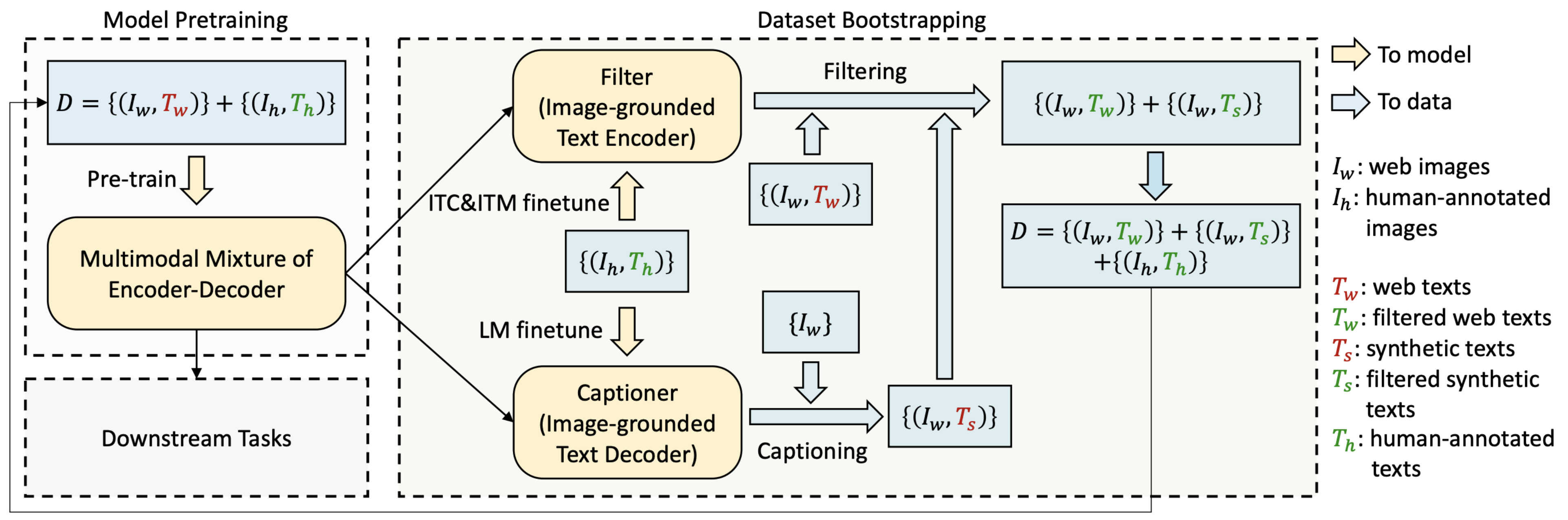
BLIP as a Solution:
BLIP extends beyond CLIP by not just aligning images and text but integrating them in a way that allows the model to generate rich text responses based on visual inputs.

BLIP's encoder-decoder framework allows it to not only match but also generate text, enabling tasks like open-ended captioning and visual question answering.

BLIP also introduces a training method that bootstraps noisy image-text pairs, which helps it learn from a more diverse set of real-world data, improving its generalization to new scenarios.



Training BLIP



Model Pretraining
Dataset Bootstrapping
$D = \{(I_w, T_w)\} + \{(I_h, T_h)\}$
Pre-train
Multimodal Mixture of Encoder-Decoder
Downstream Tasks
ITC&ITM finetune
LM finetune
Filter (Image-grounded Text Encoder)
Captioner (Image-grounded Text Decoder)
Filtering
Captioning
$\{(I_h, T_h)\}$
$\{(I_w, T_w)\}$
$\{I_w\}$
$\{(I_w, T_s)\}$
$\{(I_w, T_w)\} + \{(I_w, T_s)\}$
$D = \{(I_w, T_w)\} + \{(I_w, T_s)\} + \{(I_h, T_h)\}$
To model
To data
I_w: web images
I_h: human-annotated images
T_w: web texts
T_w: filtered web texts
T_s: synthetic texts
T_s: filtered synthetic texts
T_h: human-annotated texts

Figure 3. Learning framework of BLIP. We introduce a captioner to produce synthetic captions for web images, and a filter to remove noisy image-text pairs. The captioner and filter are initialized from the same pre-trained model and finetuned individually on a small-scale human-annotated dataset. The bootstrapped dataset is used to pre-train a new model.

Image-Text Pairing with Multimodal Encoders:

- BLIP uses a Vision Transformer (ViT) for images and a language model for text. The image is processed to extract features that represent visual aspects like objects, textures, or spatial layouts, while the language model represents the text in a way that can be aligned with these visual features.
- The core goal in this step is to build representations that allow the image and text to be understood in relation to each other. This multimodal encoder-decoder setup lets the model go beyond simple matching, making it possible to learn more complex relationships, like context or narrative details within an image.

Bootstrapping with Noisy Data:

- To improve generalization, BLIP includes “noisy” or imperfect image-text pairs in training. This can mean using mismatched, weakly aligned, or less clear pairs, such as an image and a roughly related caption.
- Over the course of training, BLIP refines these pairs, which helps the model learn to handle real-world data variability (e.g., social media images with informal captions).
- This bootstrapping process is iterative, where BLIP gradually strengthens its alignment between image and text representations by learning to identify which pairs make sense together and which don’t.

BLIP2 – Improving Efficiency in multimodal training

BLIP2 focuses on making the model faster and more efficient by changing how it processes images and text.

- **Separate Pathways**: BLIP2 processes the image and text separately at first, unlike BLIP1 which combines them early on. For the image, it uses a lighter, more efficient model to extract basic features. For the text, it still uses a language model.
- **Combining Later**: After it processes both image and text separately, it combines their information in a shared space later in the process. This makes it faster and more efficient because it doesn't have to fully integrate the image and text information immediately.
- **Training on Large Datasets**: This separate pathway approach allows BLIP2 to handle larger datasets more cost-effectively, improving performance on tasks like zero-shot learning (where the model sees new data it hasn't been explicitly trained on).

BLIP2's main advantage is efficiency—it can process image-text information faster and requires fewer resources than BLIP1, making it suitable for large-scale, real-time tasks.

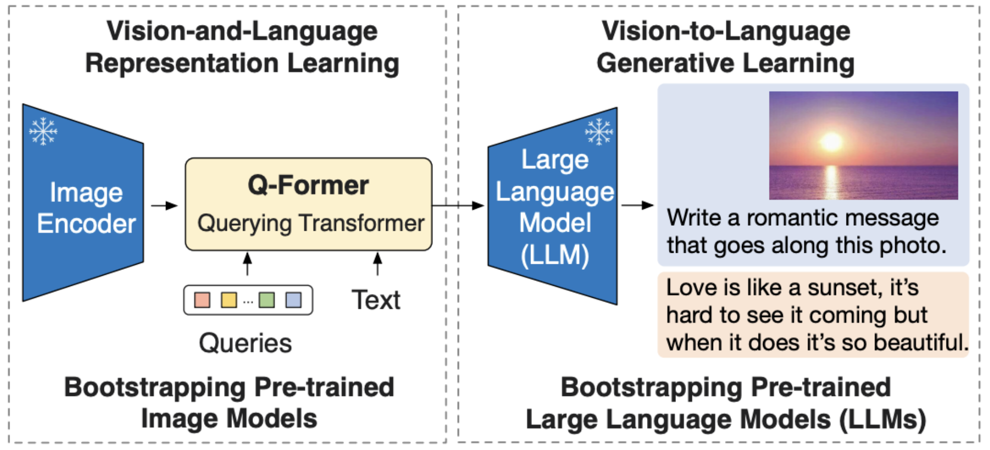
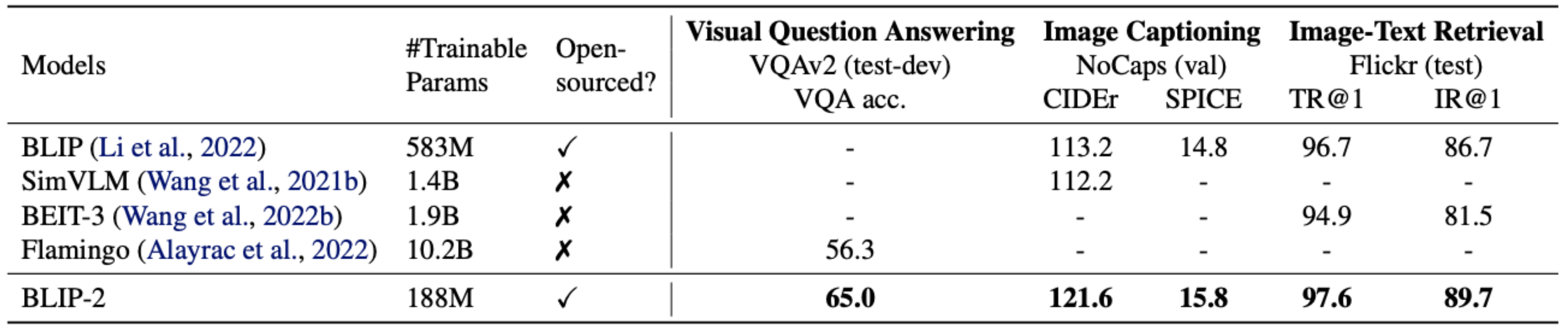
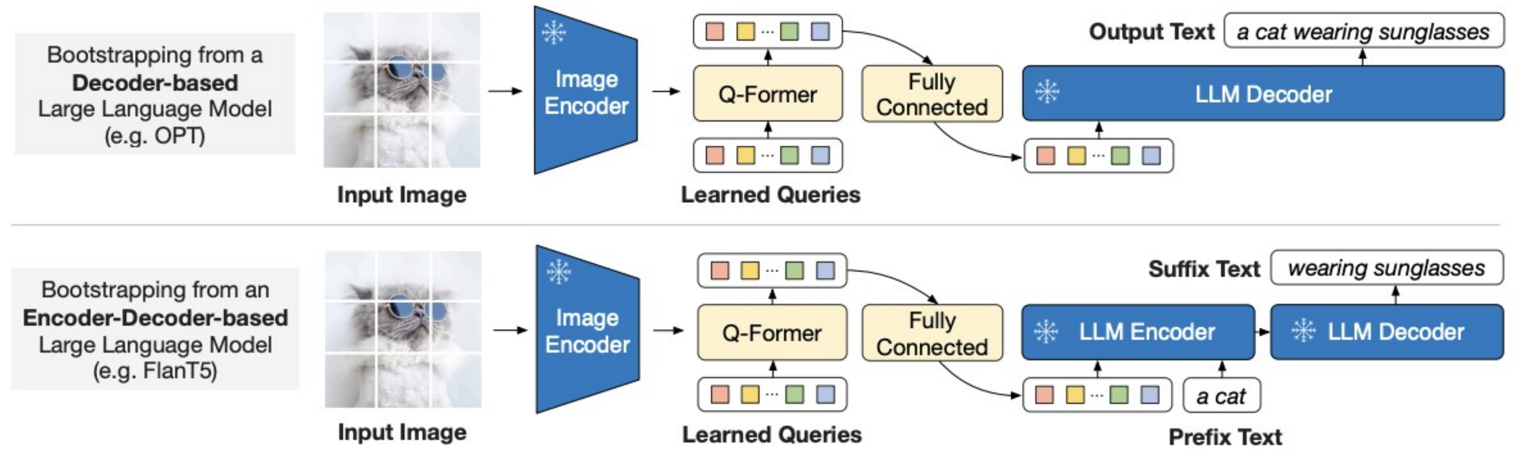

Vision-and-Language Representation Learning
Image Encoder
Q-Former
Querying Transformer
Queries
Text
Bootstrapping Pre-trained Image Models
Vision-to-Language Generative Learning
Large Language Model (LLM)
Write a romantic message that goes along this photo.
Love is like a sunset, it's hard to see it coming but when it does it's so beautiful.
Bootstrapping Pre-trained Large Language Models (LLMs)

Figure 1. Overview of BLIP-2's framework. We pre-train a lightweight Querying Transformer following a two-stage strategy to bridge the modality gap. The first stage bootstraps vision-language representation learning from a frozen image encoder. The second stage bootstraps vision-to-language generative learning from a frozen LLM, which enables zero-shot instructed image-to-text generation (see Figure 4 for more examples).



Models	#Trainable Params	Open-sourced?	Visual Question Answering VQAv2 (test-dev) VQA acc.	Image Captioning NoCaps (val) CIDEr	Image Captioning NoCaps (val) SPICE	Image-Text Retrieval Flickr (test) TR@1	Image-Text Retrieval Flickr (test) IR@1
BLIP (Li et al., 2022)	583M	✓	-	113.2	14.8	96.7	86.7
SimVLM (Wang et al., 2021b)	1.4B	✗	-	112.2	-	-	-
BEIT-3 (Wang et al., 2022b)	1.9B	✗	-	-	-	94.9	81.5
Flamingo (Alayrac et al., 2022)	10.2B	✗	56.3	-	-	-	-
BLIP-2	188M	✓	**65.0**	**121.6**	**15.8**	**97.6**	**89.7**

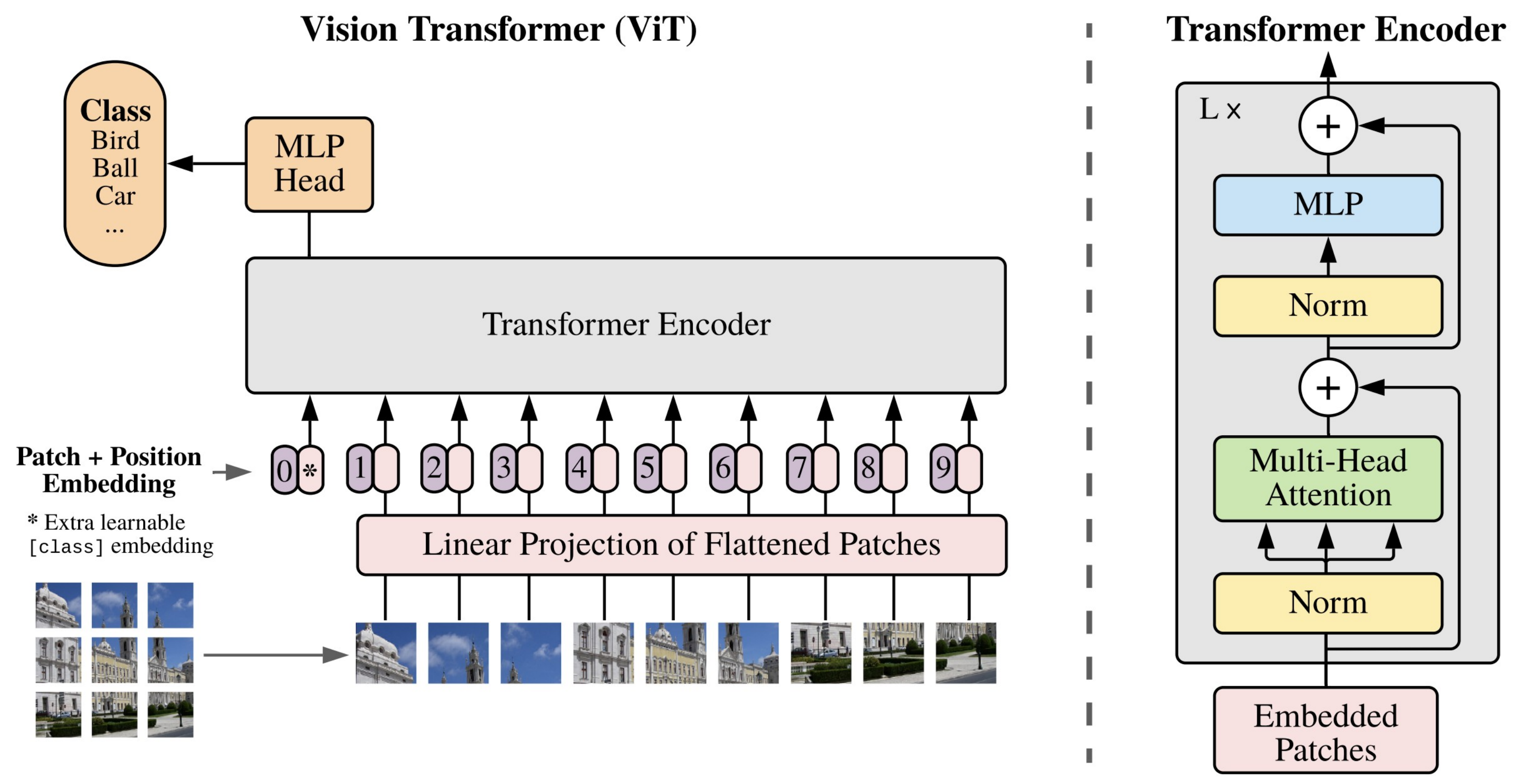


Wrap Up

Vision Transformer and CLIP

- Today we looked at a number of multimodal transformer-based models.
- We saw how Vision Transformers process images similarly to text by cutting images into patches and then processing them in the familiar transformer layers.
- CLIP was introduced as a means to connect ViTs and text to solve true multimodal problems such as image classification and captioning
- An extension to CLIP, BLIP which introduced a more complex but robust architecture which allowed it to perform well on problems such as visual QA, captioning, and many other multimodal tasks

----



NVIDIA® | DARTMOUTH ENGINEERING 1867

Thank you!