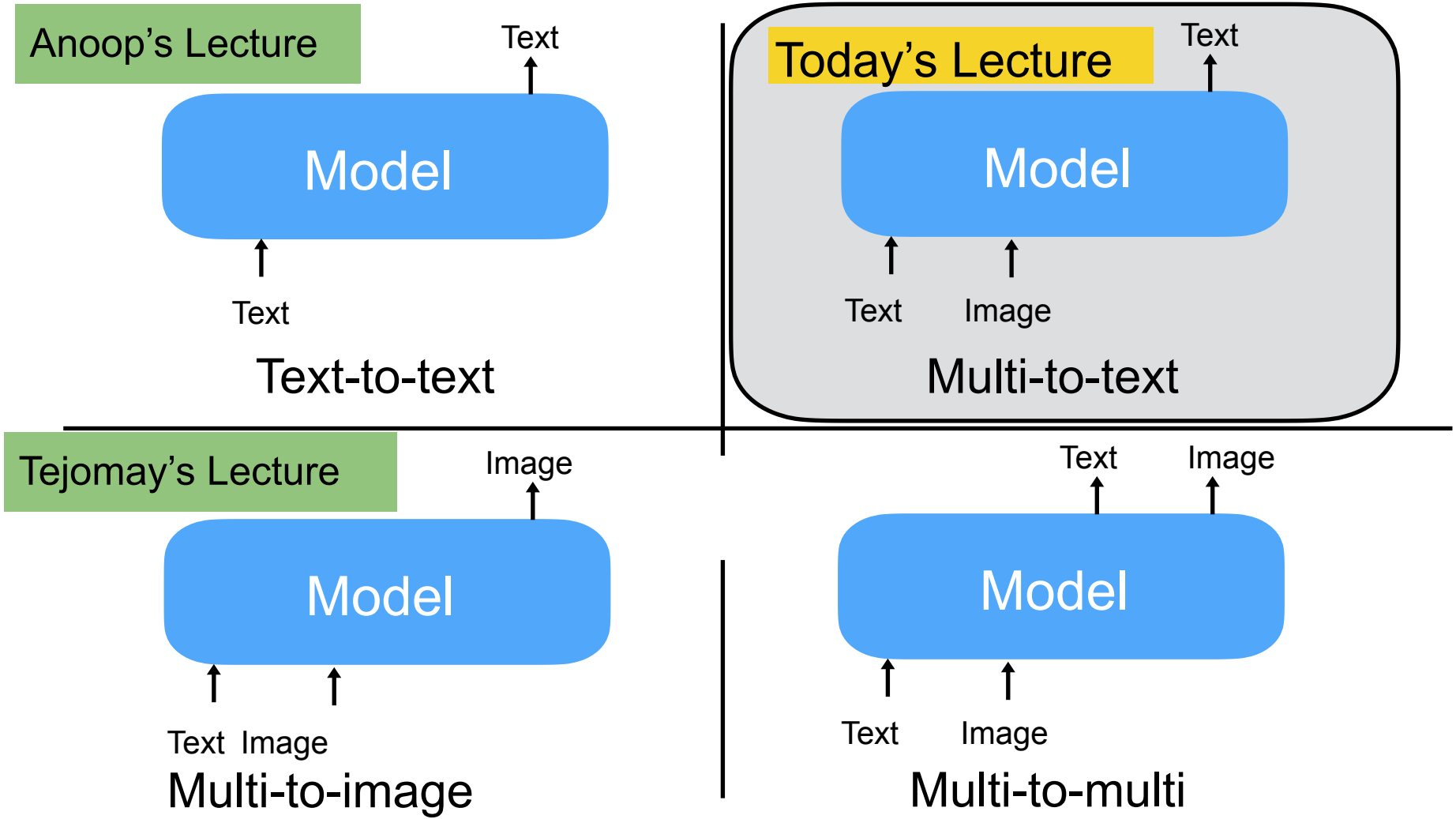
CS772-DLNLP

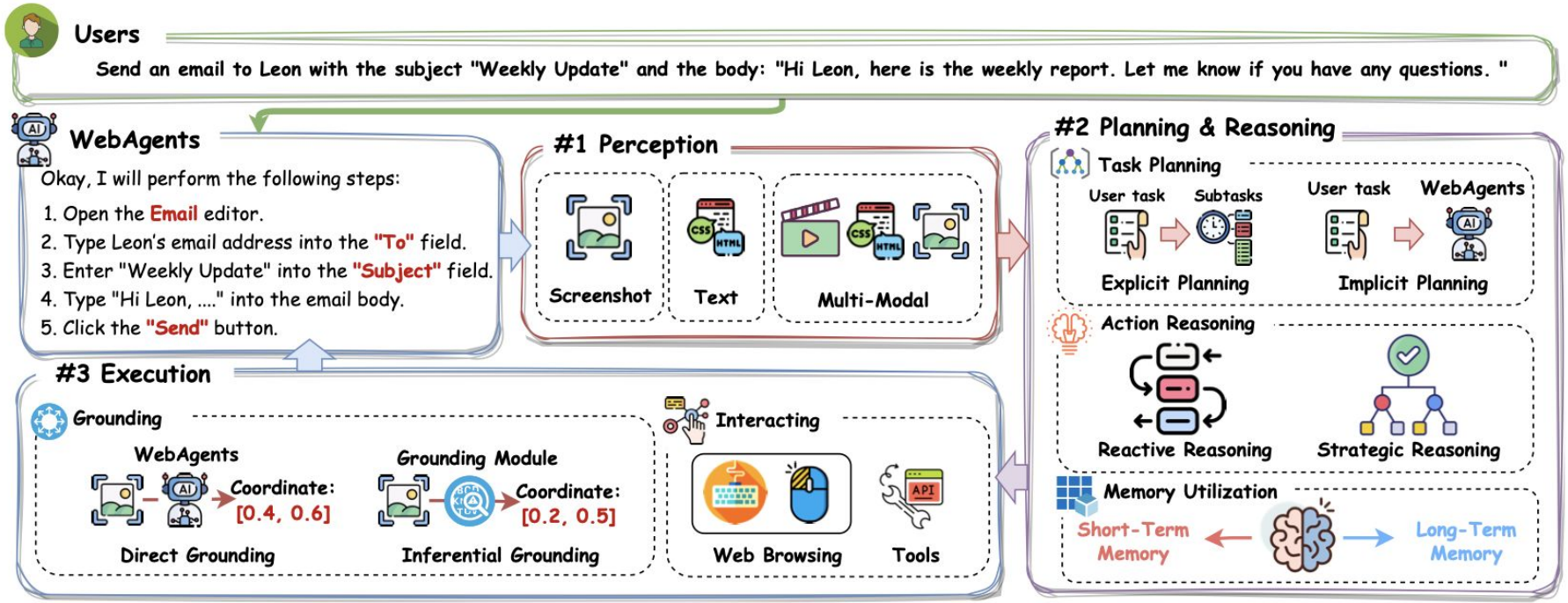
Topic: Intro to VLMS
Presenter: Sravanthi


Anoop's Lecture
Text
Model
Text
Text-to-text
Today's Lecture
Text
Model
Text
Image
Multi-to-text
Tejomay's Lecture
Image
Model
Text
Image
Multi-to-image
Text
Image
Model
Text
Image
Multi-to-multi

Describe this image

The image features two animated characters from the famous cartoon "Tom and Jerry." The blue ca t, Tom, is in the foreground with a mischievous grin, appearing to be running. On top of Tom's head s its the brown mouse, Jerry, who is also smiling broadly. Both characters are depicted in a playful and dynamic pose, capturing their classic comedic interaction.

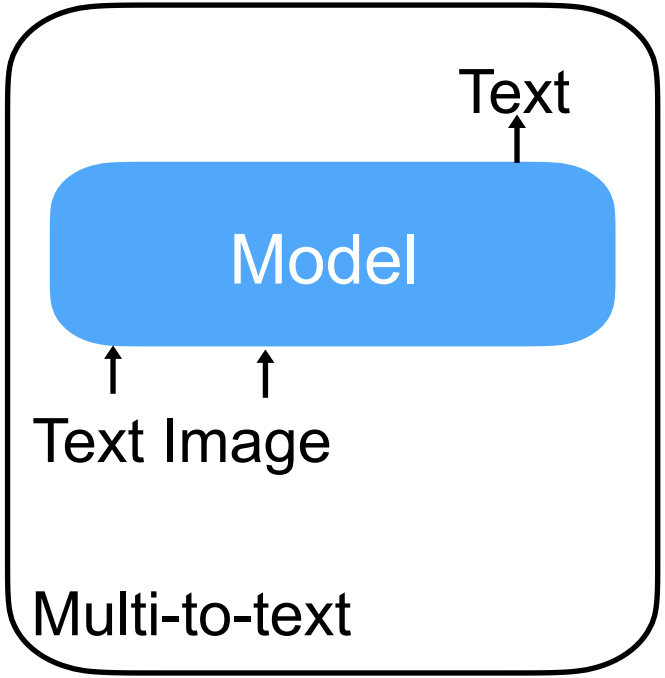
Example chat from Intern-S1 model



Example: Web agents

Overview

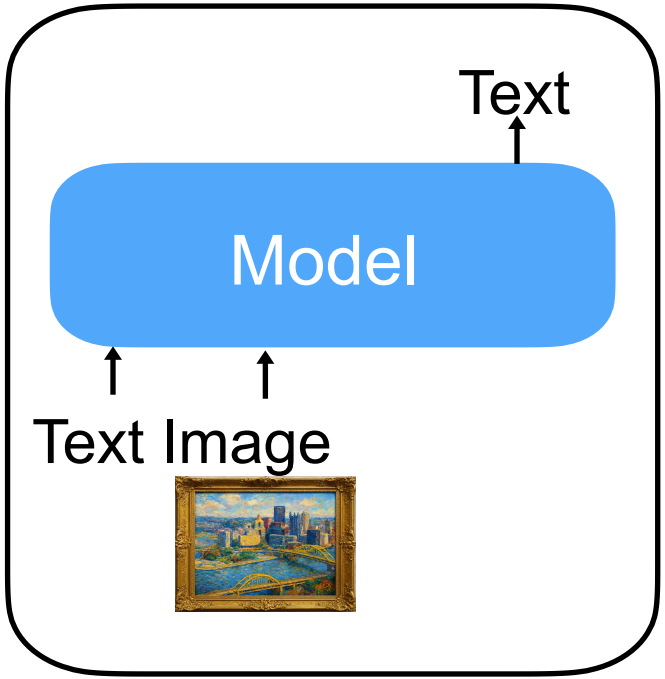
- Vision architecture basics
  - ViT [Dosovitskiy et al 2020]
- Learning image representations
  - CLIP [Radford et al 2021]
- Combining with a language model
  - Llava [Liu et al 2023]



Key problem: Representing Images

- We represent text as a sequence of vectors (token embeddings)
- We want to also represent an image as a sequence of vectors

$$f_{\text{enc}}(x_{\text{image}}) \rightarrow z_1, \ldots, z_L$$

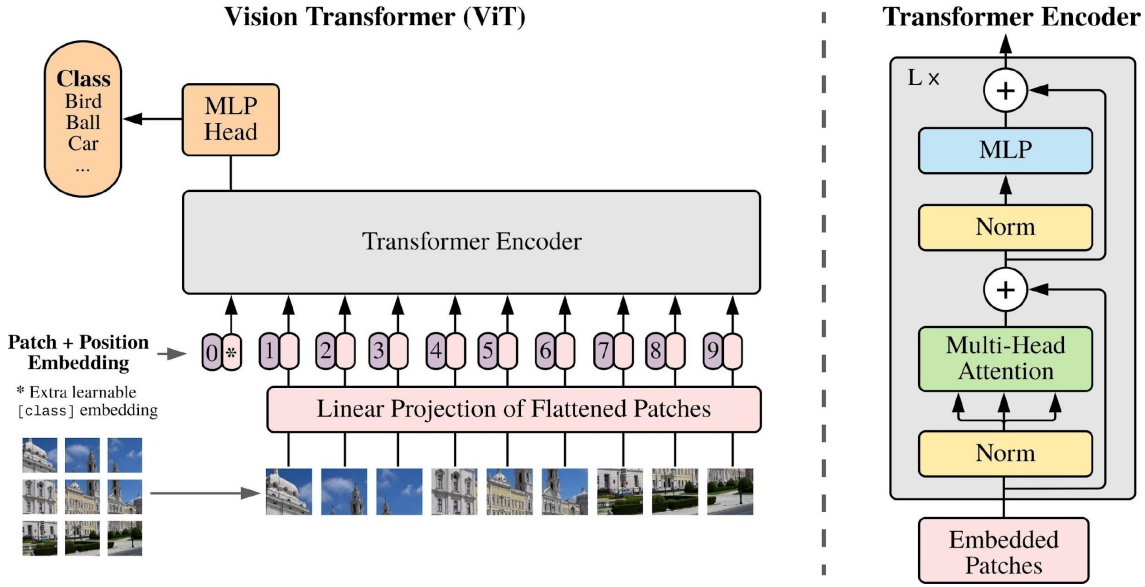


Need:

1. Neural network architecture
2. Algorithm for learning good vectors

Vision Transformer (ViT)

- Idea: Divide an image into patches, flatten the patches into vectors, use a standard transformer



How to build ViT in a Nutshell

- Break images into patches
- Flatten those patches into vectors
- Add some information about where in the image the chunks came from (positional encoding)
- Pass those vectors through a transformer
- Take the output, put it into a dense neural network, and predict what is in the image.

Vision Transformer (ViT)

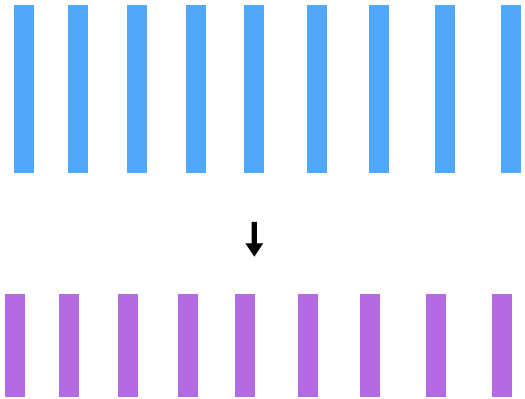
$x\text{image} \in \mathbb{R}^{H \times W \times C}$

$x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$

$x \in \mathbb{R}^{N \times D}$

$x = Wx_p$

$W^e \in \mathbb{R}^{D \times (P^2 \cdot C)}$



"Patch embeddings"

+ position embeddings

120 x 120 x 3

Patch: 40 x 40

$N = \frac{(120)(120)}{(40)(40)} = 9$ patches

$P^2 \cdot C = (40)(40)(3) = 4800$

$\Rightarrow x^p \in \mathbb{R}^{9 \times 4800}$

$W \in \mathbb{R}^{1024 \times 4800}$

$\Rightarrow x \in \mathbb{R}^{9 \times 1024}$

CLS token

- One *feature* introduced to transformers with the popular BERT models was the use of a [CLS] (or *“classification”*) token. The [CLS] token was a *“special token”* prepended to every sentence fed into BERT.
- This [CLS] token is converted into a token embedding and passed through several encoding layers.
- Two things make [CLS] embeddings special. First, it does *not* represent an actual token, meaning it begins as a “blank slate” for each sentence. Second, the final output from the [CLS] embedding is used as the input into a classification head during pretraining.
- Using a “blank slate” token as the sole input to a classification head pushes the transformer to learn to encode a *“general representation”* of the entire sentence into that embedding. The model *must* do this to enable accurate classifier predictions.
- ViT applies the same logic by adding a *“learnable embedding”*. This learnable embedding is the same as the [CLS] token used by BERT.
- The preferred pretraining function of ViT is based solely on classification, unlike BERT, which uses masked language modeling. Based on that, this learning embedding is *even more* important to the successful pretraining of ViT.

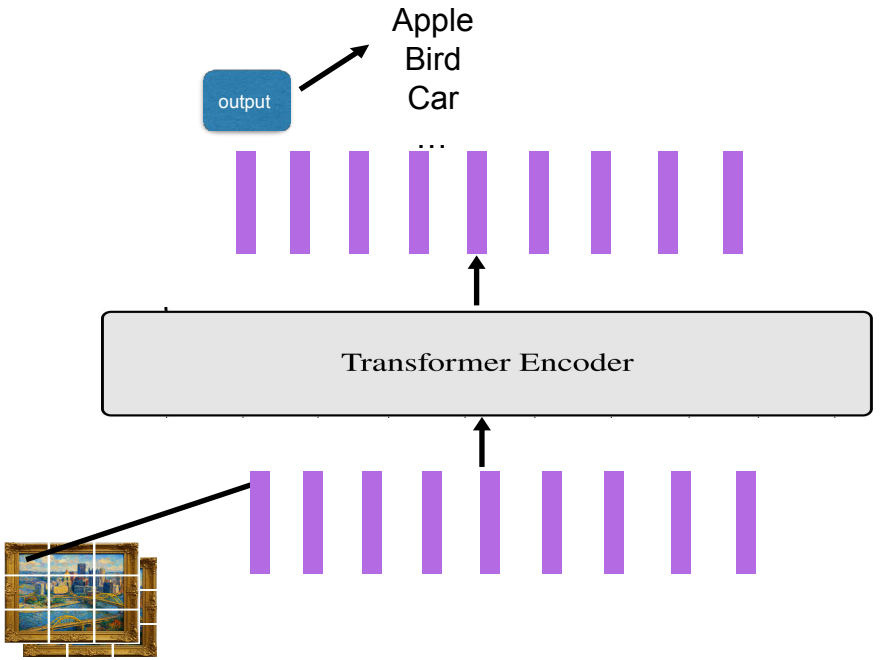
Position Embeddings

- Transformers do *not* have any default mechanism that considers the “order” of token or patch embeddings.
- We enable order with *positional embeddings*. For ViT, these positional embeddings are learned vectors with the same dimensionality as our patch embeddings.
- After creating the patch embeddings and prepending the “class” embedding, we sum them all with positional embeddings.
- These positional embeddings are learned during pretraining and (sometimes) during fine-tuning.

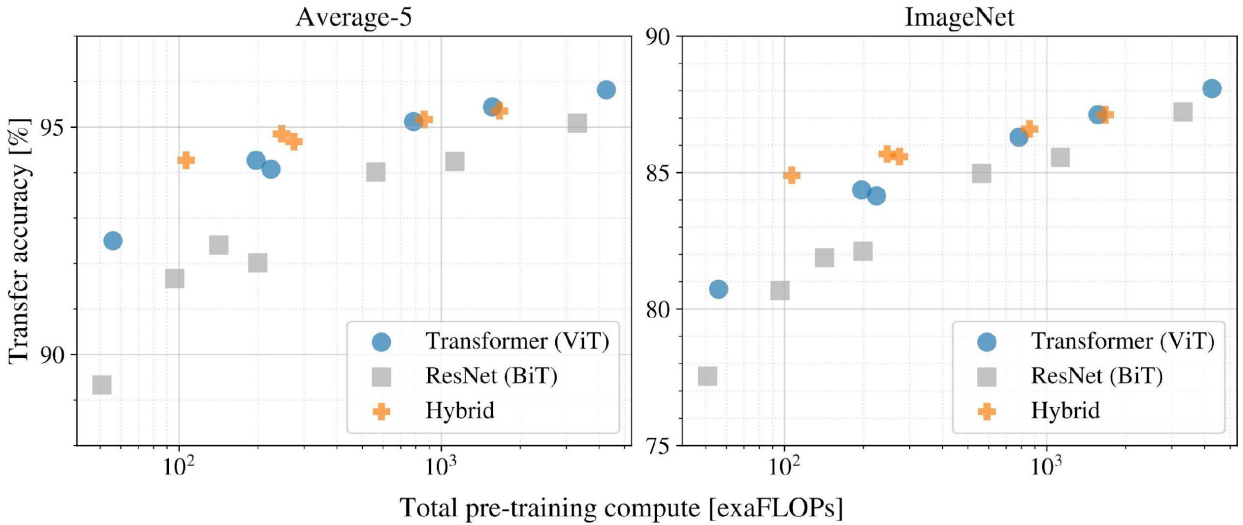
Vision Transformer (ViT)

- The transformer transforms the patch embeddings into vector representations

  $z_1, \ldots, z_N$
- We can train the model to perform a task such as classification



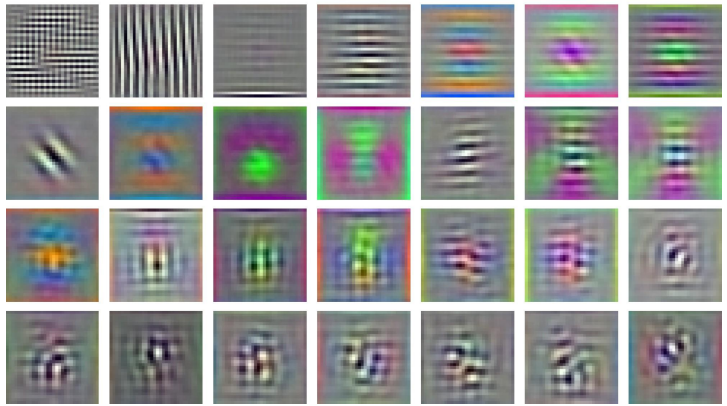
Vision Transformer (ViT)



Performance versus pre-training compute for different architectures

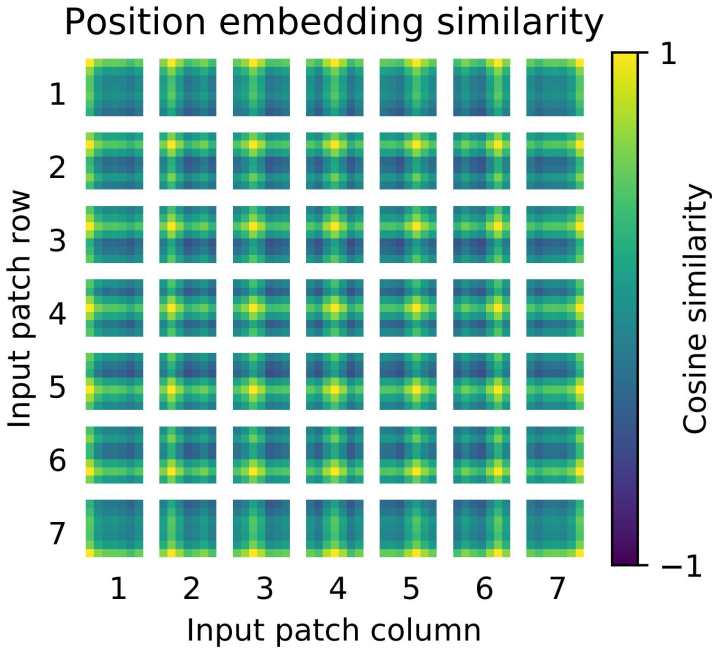
Vision Transformer (ViT)

RGB embedding filters
(first 28 principal components)



Reshape rows into P x P, visualize principal components of learned embedding filters. The components resemble plausible basis functions for a low-dimensional representation of the fine structure within each patch

Vision Transformer (ViT)

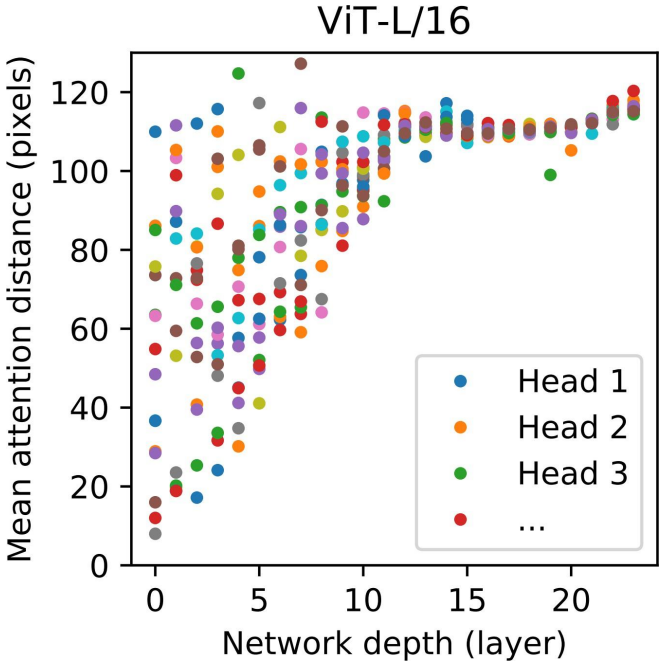


Cosine similarity between the position embedding of the patch with the indicated row and column and the position embeddings of all other patches

Vision Transformer (ViT)

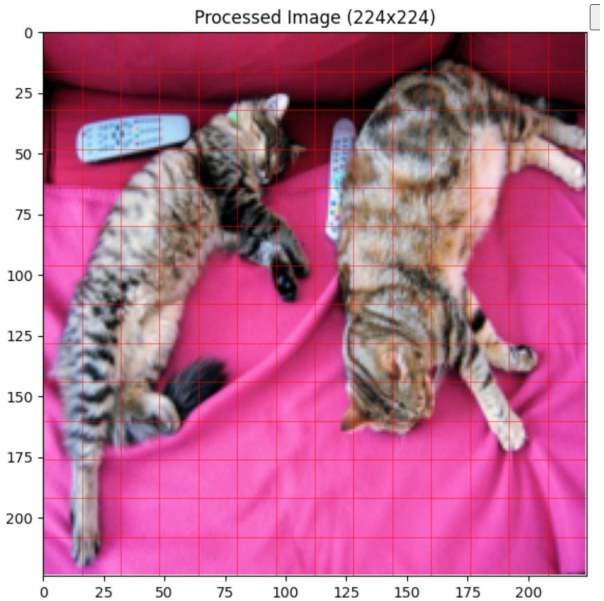
Attention from o/p tokens to i/p space. Can attend to regions that are salient for the task (here classification)

Vision Transformer (ViT)



Early layers either attend to large regions or narrow regions; later layers generally attend to larger regions

Code example




```
Patch embedding process:
1. Input image: torch.Size([1, 3, 224, 224])
   [batch, channels=3, height=224, width=224]

2. After patch projection: torch.Size([1, 196, 768])
   [batch, num_patches=196, hidden_size=768]

3. Each 16x16x3 patch → 768-dim vector
```

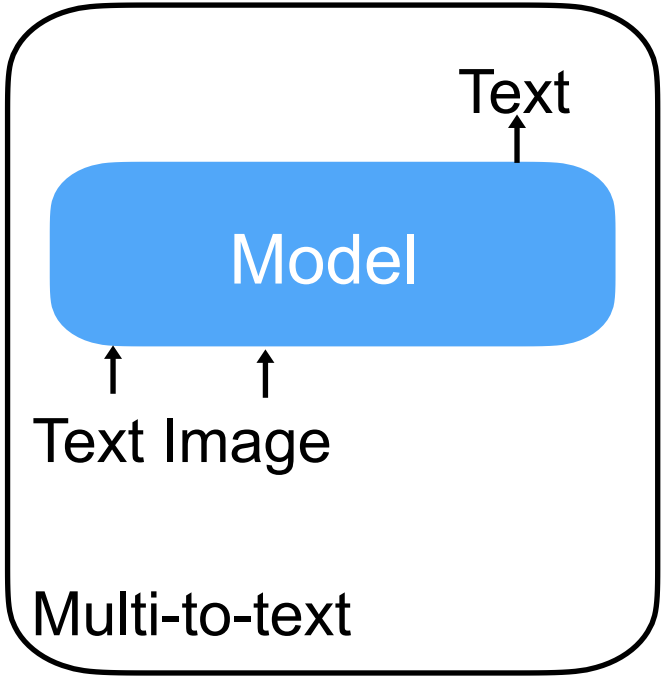
```
Model architecture:
Number of layers: 12
Number of attention heads: 12
Hidden size: 768
Head dimension: 64

Output shape: torch.Size([1, 197, 768])

CLS token representation shape: torch.Size([1, 768])
  (Used for classification tasks)
```

```
Top 5 predictions:
========================================
1. Egyptian cat: 93.74%
2. tabby, tabby cat: 3.84%
3. tiger cat: 1.44%
4. lynx, catamount: 0.33%
5. Siamese cat, Siamese: 0.07%
```

https://github.com/cmu-l3/anlp-fall2025-code

Overview

- Vision architecture basics
  - ViT
- **Learning image representations**
  - **CLIP**
- Combining with a language model
  - Llava

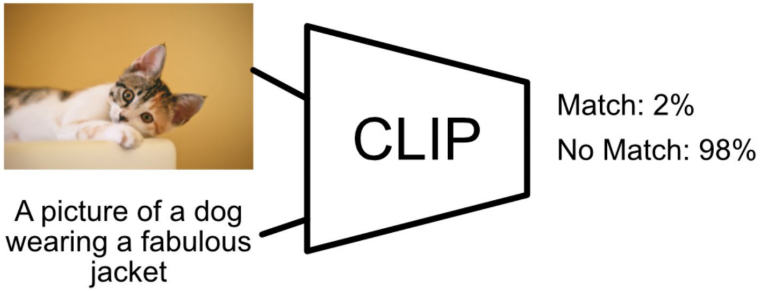


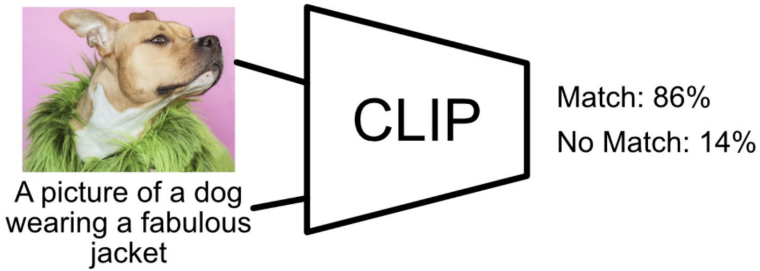
Contrastive Language-Image Pre-training (CLIP)

Goal: Pre-training objective for learning image representations

- Learn from text
  - At the time, models were pre-trained on classification: the only textual supervision was from the class label.
  - A textual description of an image provides much more information than one class label.
- Scalable
  - At the time, image pre-training was largely limited to hand labeled data.
  - Want to have the property of improving by adding more compute.

What is CLIP capable of ?





CLIP

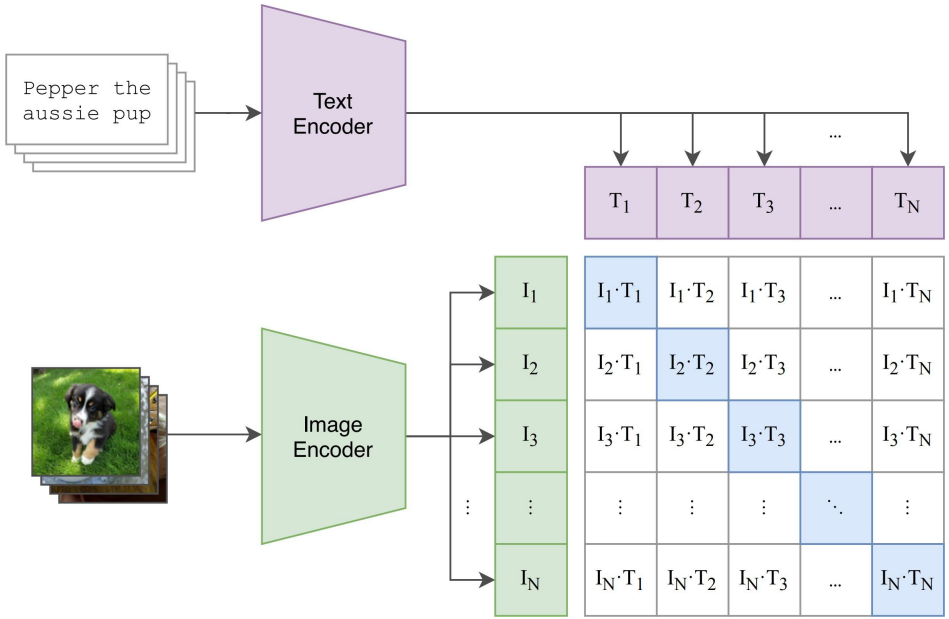
- Idea: learn image and text representations jointly in a shared embedding space
  - Learn an image encoder $f_I(x) \rightarrow z_I$
  - Learn a text encoder $f_T(y) \rightarrow z_T$
  - The representations for a paired image and its text should be close together.
  - The representations for an unpaired image and text should be far apart.
- Apply the method over a large dataset of (image, text) pairs

Key Components in CLIP

- Data: pairs of (image, text)
  - E.g. 400 million web images with their text descriptions
- Image encoder $f_I(x) \rightarrow z_I$
  - E.g. vision transformer
- Text encoder $f_T(y) \rightarrow z_T$
  - E.g. transformer

CLIP Architecture

- Basic idea: Given *N* (image, text) pairs, classify which image is paired with which text



Text Encoder

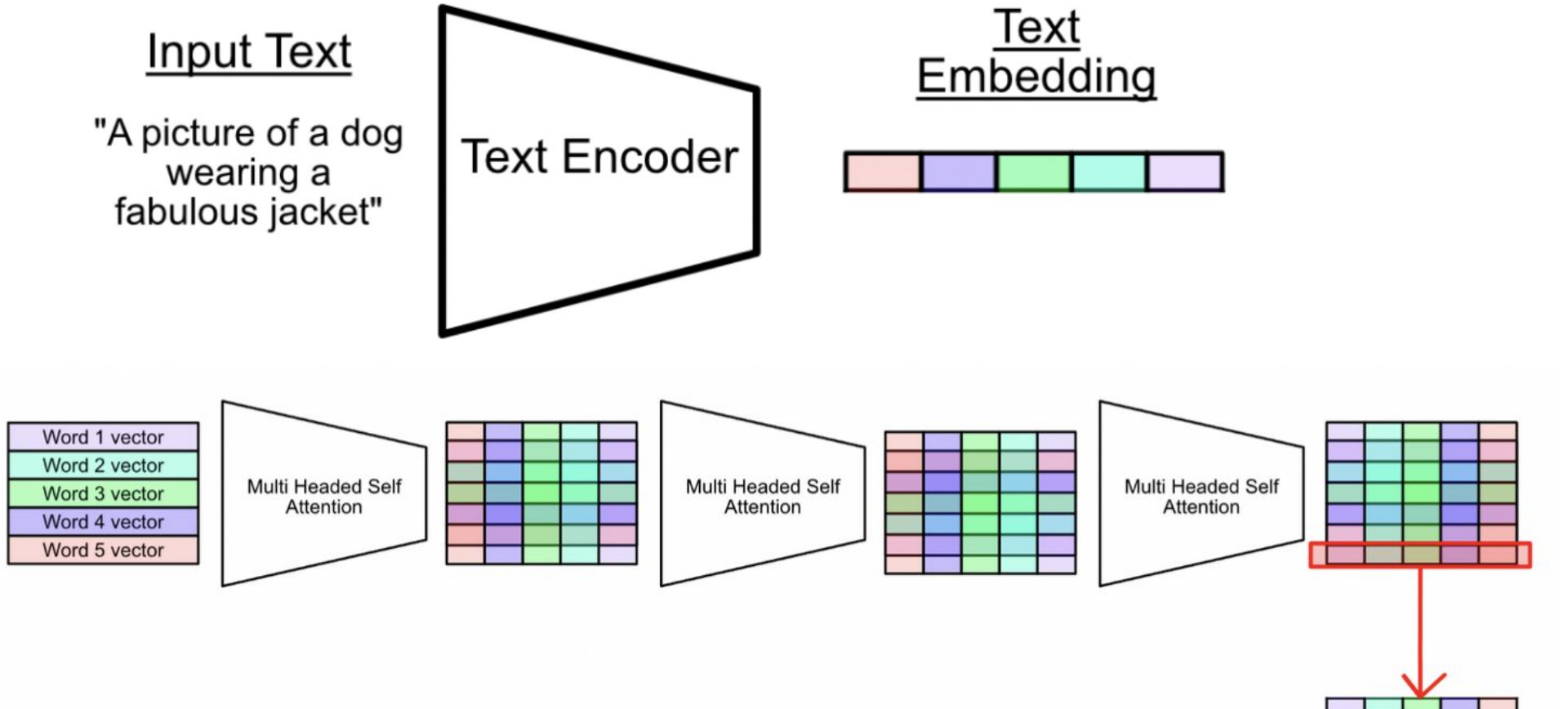


Image Encoder

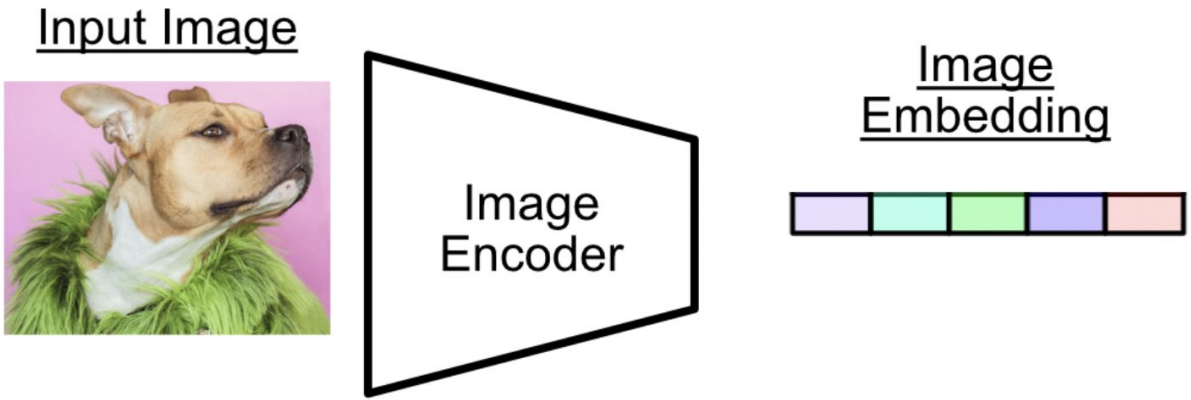
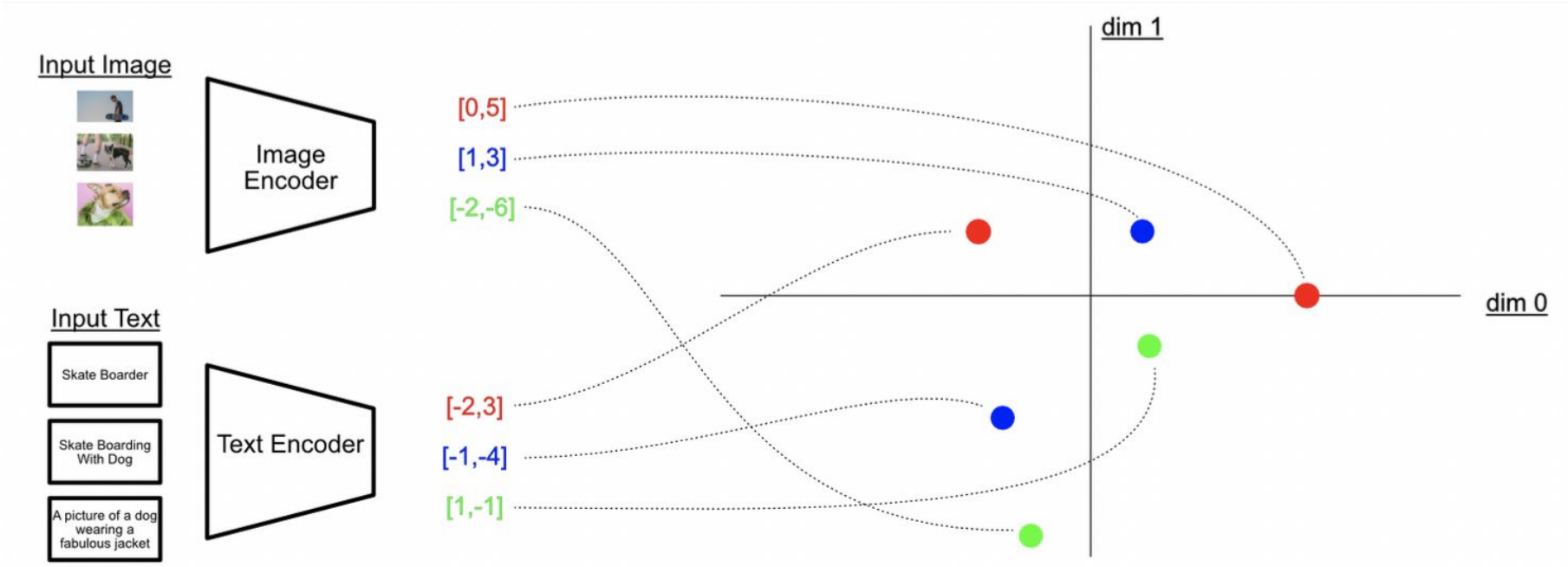
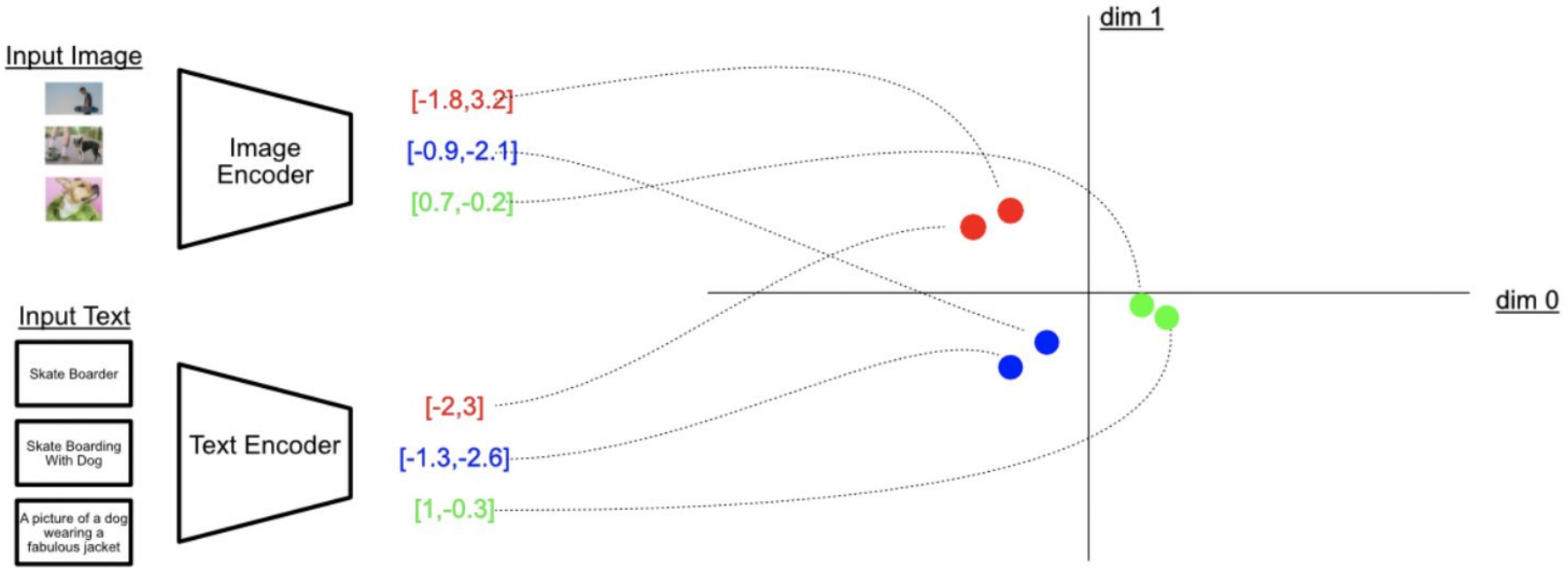


Image encoder can be either ResNet-50 or ViT

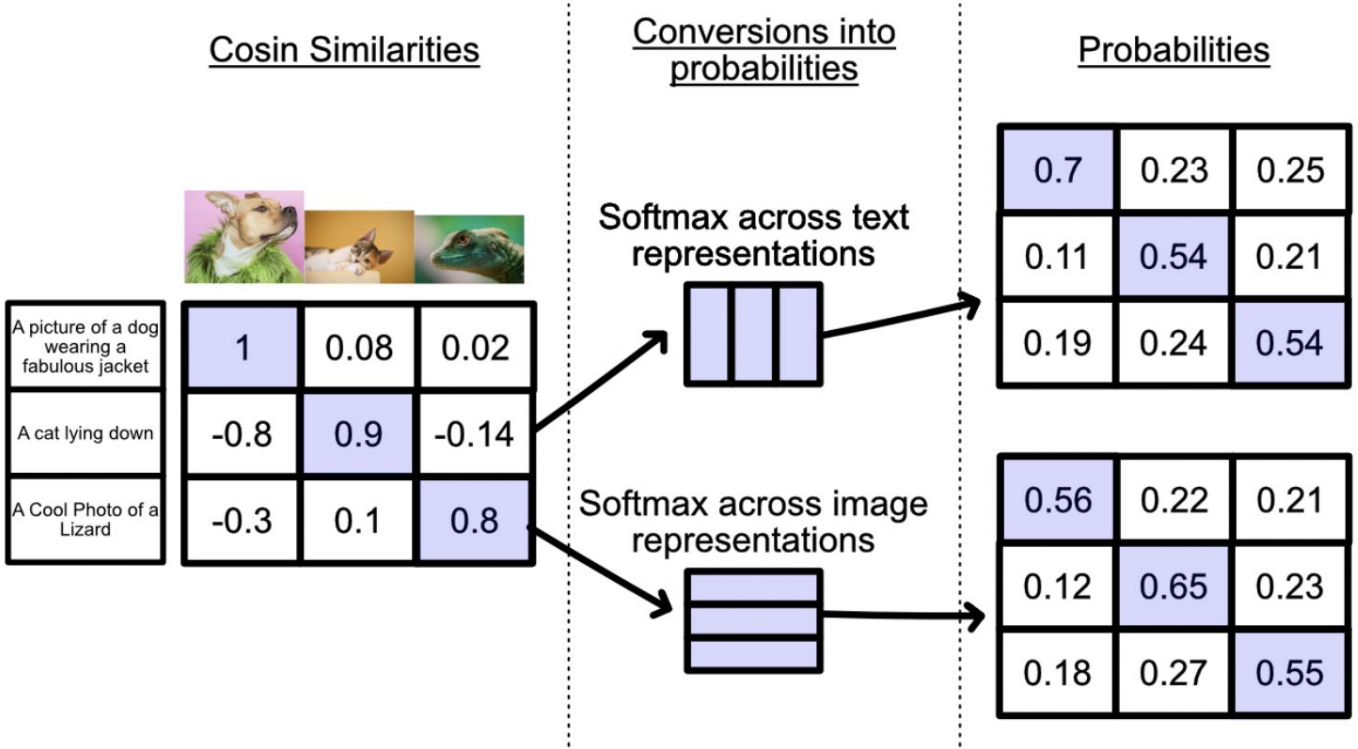
Before Training



After Training



How to Calculate Probabilities?



Contrastive Loss

$$L((x_1, y_1), \ldots, (x_N, y_N)) =$$

Push up dot product of the correct pair

$$-\frac{1}{2}\sum_{n=1}^{N}\left[\log\frac{\exp\left(f_I(x_n)^\top f_T(y_n)\right)}{\sum_j \exp\left(f_I(x_j)^\top f_T(y_n)\right)} + \log\frac{\exp\left(f_I(x_n)^\top f_T(y_n)\right)}{\sum_j \exp\left(f_I(x_n)^\top f_T(y_j)\right)}\right]$$

Push down dot product of other pairs

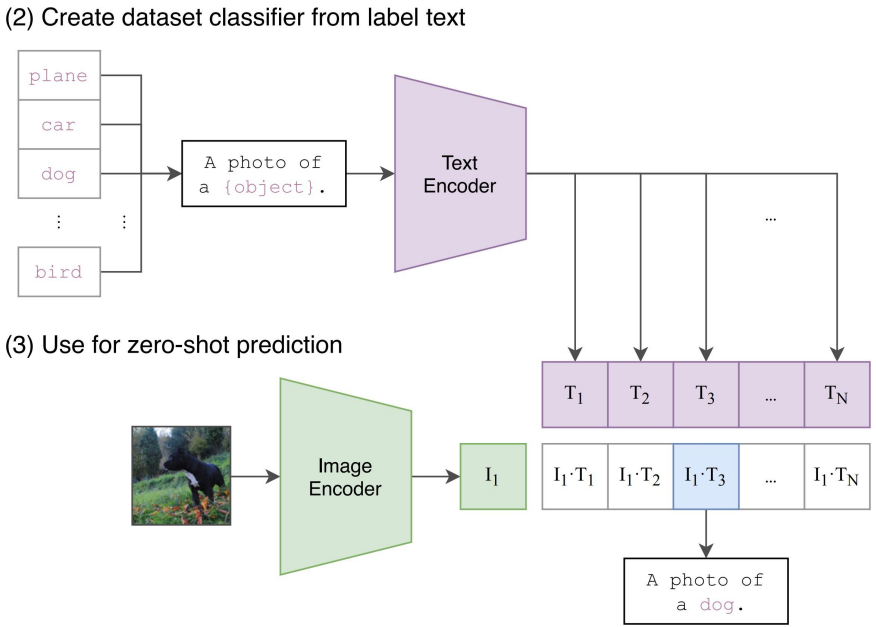
Softmax over images

Softmax over text

```
# 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
```

CLIP

- Example “zero-shot” usage



CLIP

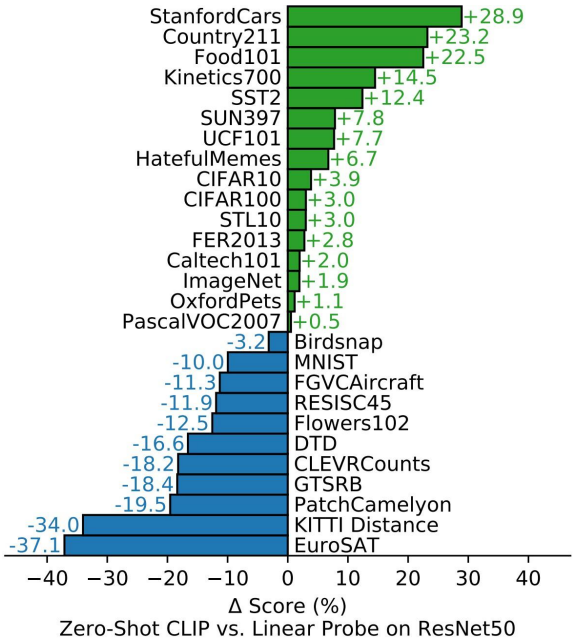
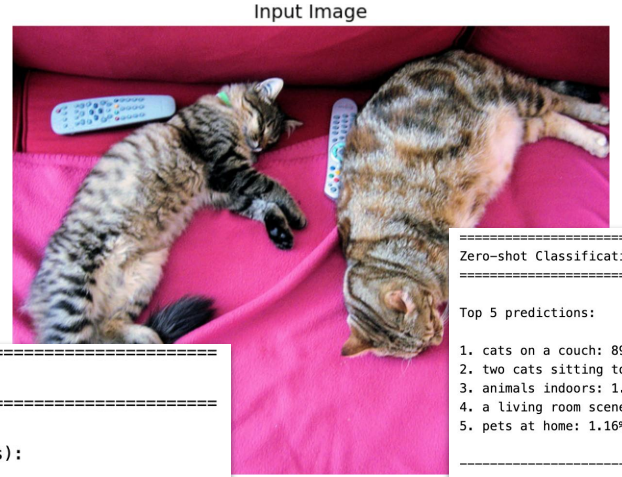


Figure 5. **Zero-shot CLIP is competitive with a fully supervised baseline.** Across a 27 dataset eval suite, a zero-shot CLIP classifier outperforms a fully supervised linear classifier fitted on ResNet-50 features on 16 datasets, including ImageNet.

Code example



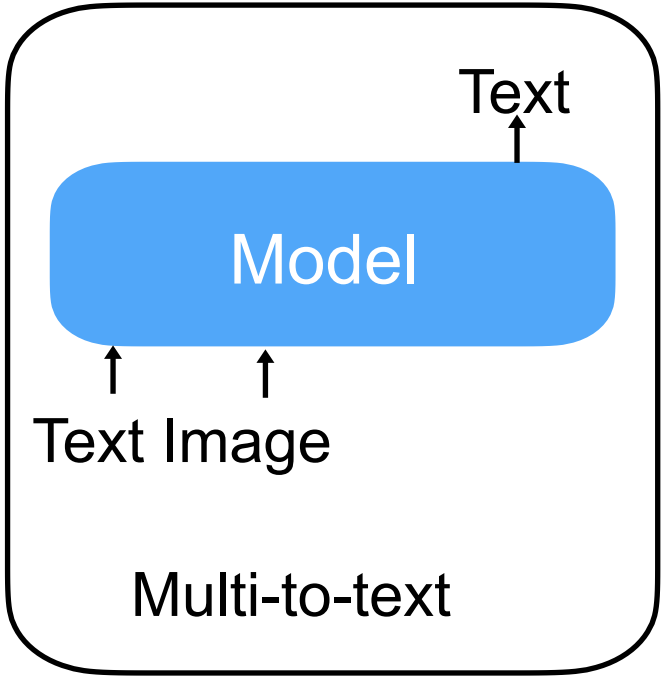

```
==================================================
Cat vs Dog Classification
==================================================

Raw similarity scores (logits):
  a photo of a cat: 18.904
  a photo of a dog: 11.716

Probabilities:
  a photo of a cat: 99.9%
  a photo of a dog: 0.1%

Prediction: a photo of a cat (99.9%)
```

```
============================================================
Zero-shot Classification Results
============================================================

Top 5 predictions:

1. cats on a couch: 89.08%
2. two cats sitting together: 5.39%
3. animals indoors: 1.97%
4. a living room scene: 1.62%
5. pets at home: 1.16%

----------------------------------------

All predictions:

cats on a couch                89.08%
two cats sitting together       5.39%
animals indoors                 1.97%
a living room scene             1.62%
pets at home                    1.16%
a photo of furniture            0.35%
a photo of a cat                0.33%
a photo of a kitten             0.10%
a photo of a person             0.00%
a photo of a dog                0.00%
```

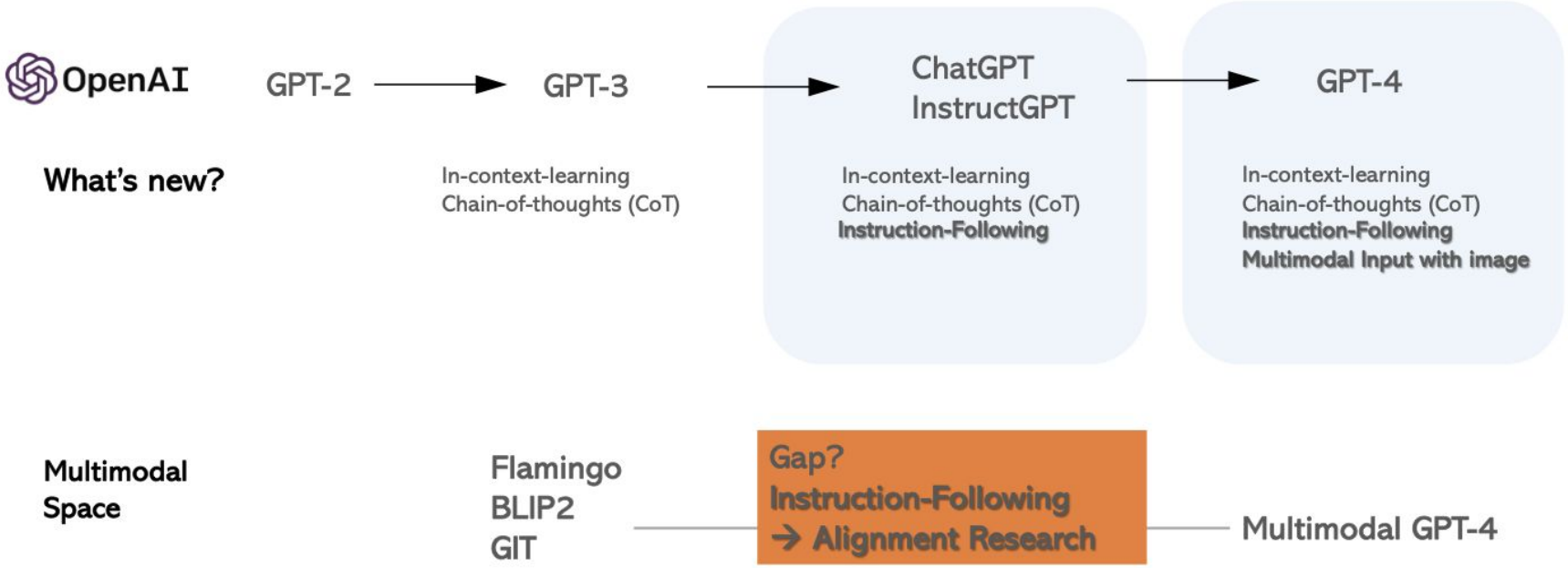
https://github.com/cmu-l3/anlp-fall2025-code

Overview

- Vision architecture basics
  - ViT
- Learning image representations
  - CLIP
- **Combining with a language model**
  - **Llava**



Research Gap



Llava

Source: https://www.barnorama.com/wp-content/uploads/2016/12/03-Confusing-Pictures.jpg

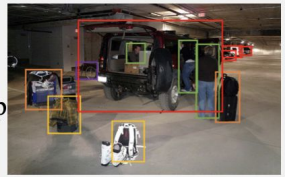
User What is unusual about this image?

LLaVA The unusual aspect of this image is a man ironing clothes on the back of a minivan or van. This is not a typical place to perform this activity, as one would usually iron clothes in a more stationary and safe location, such as a home, using a regular ironing board. The scene depicted in the image is peculiar as it involves a makeshift ironing setup on a vehicle, which can be both unsafe and unconventional. Additionally, it is not clear how the man is able to maintain balance and stability while ironing clothes in such an unstable environment.

Data Creation

Context type 1: Captions

A group of people standing outside of a black vehicle with various luggage.
Luggage surrounds a vehicle in an underground parking area
People try to fit all of their luggage in an SUV.
The sport utility vehicle is parked in the public garage, being packed for a trip
Some people with luggage near a van that is transporting it.



Context type 2: Boxes

person: [0.681, 0.242, 0.774, 0.694], backpack: [0.384, 0.696, 0.485, 0.914], suitcase: ...<omitted>

Response type 1: conversation

Question: What type of vehicle is featured in the image?
Answer: The image features a black sport utility vehicle (SUV) ...<omitted>

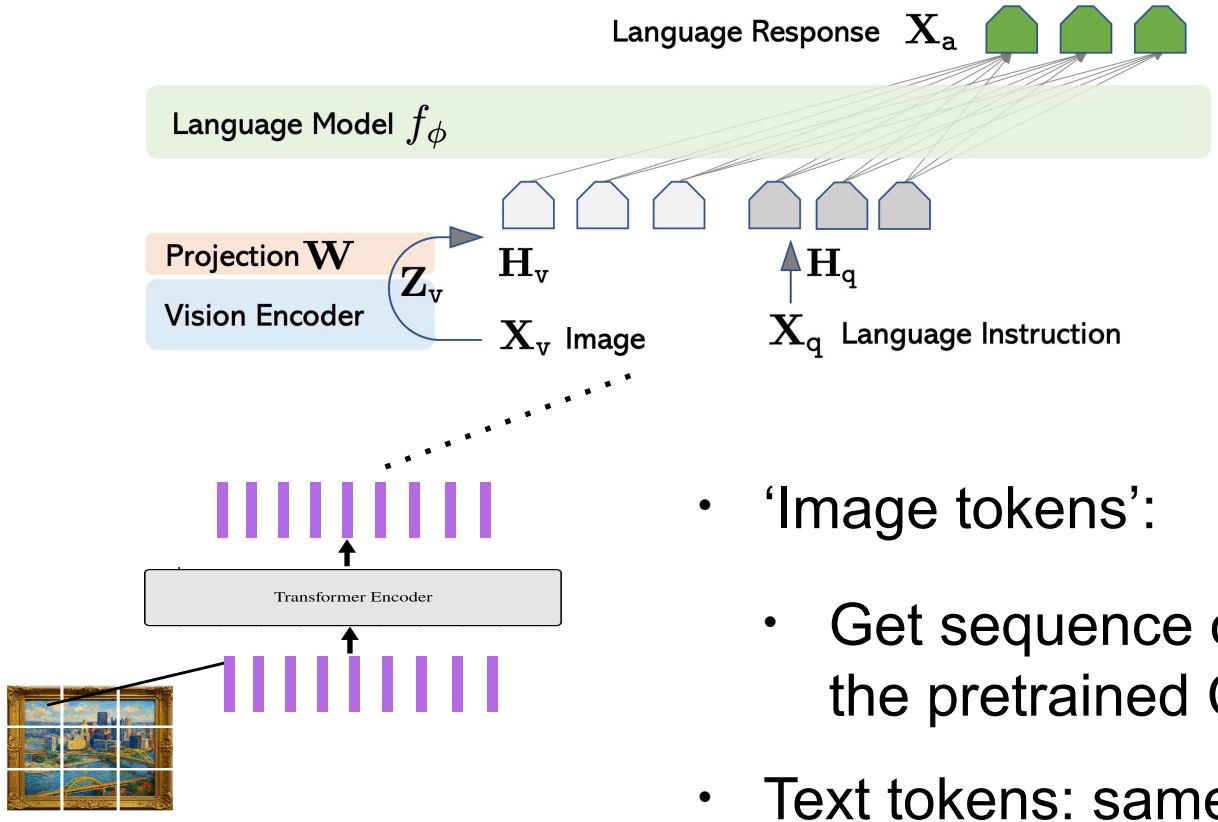
Response type 2: detailed description

The image is an underground parking area with a black sport utility vehicle (SUV) parked. There are three people in the scene, with one person standing closer to the left side of the vehicle, another person in the middle, and the third person on the right side. They are all working together to pack their luggage into the SUV for a trip. ...<omitted>

Response type 3: complex reasoning

Question: What challenges do these people face?
Answer: In the image, a group of people is standing outside a black SUV in a parking area, surrounded by various pieces of luggage, including suitcases and backpacks. They are facing the challenge of fitting all their luggage into the black SUV. There are multiple suitcases and backpacks to be packed, which suggests that the group has a significant amount of belongings ...<omitted>

Llava



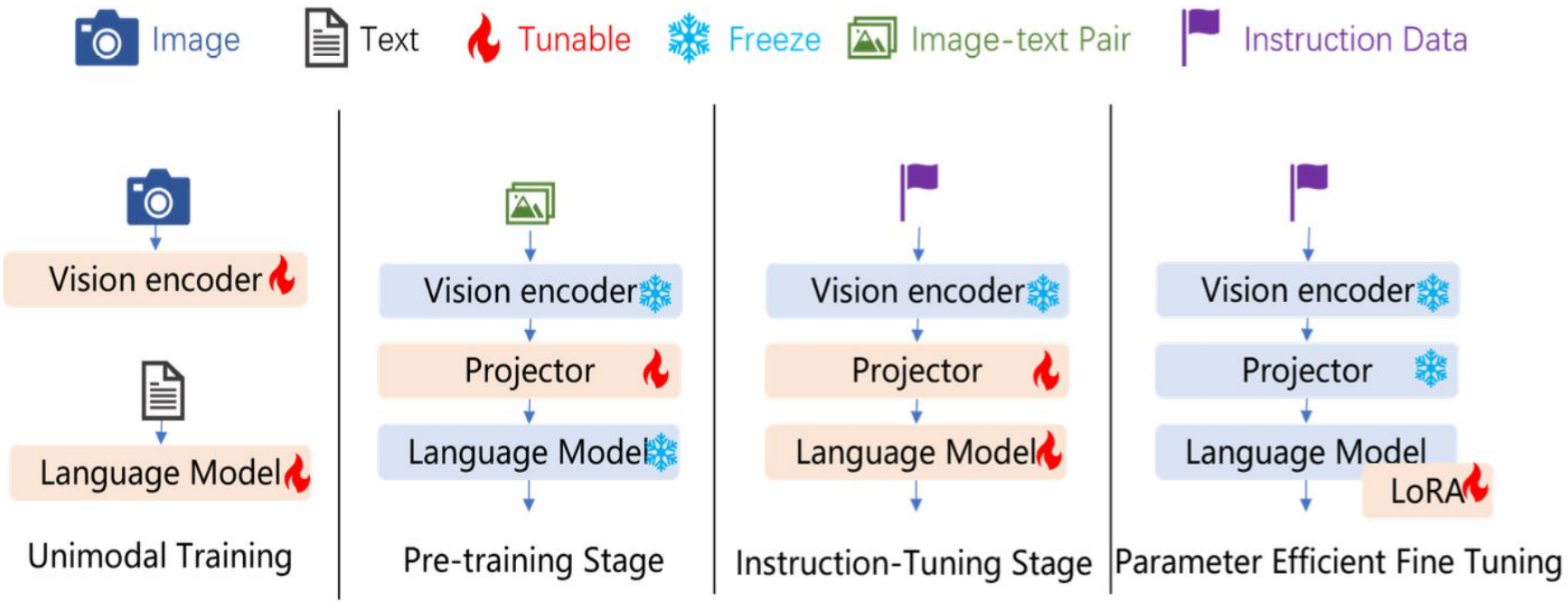
- ‘Image tokens’:
  - Get sequence of vectors from the pretrained CLIP ViT
- Text tokens: same as usual

Training Paradigm

Stage 1- Pre-training for Feature Alignment : 595K image-text pairs are filtered from CC3M dataset. Only projection layer is trained.

Stage 2- Fine-tuning End-to-End : Fine-tuning on the 158K language-image instruction-following data. Both projection layer and LLM are trained in this stage.

Stages of Training



General pipeline

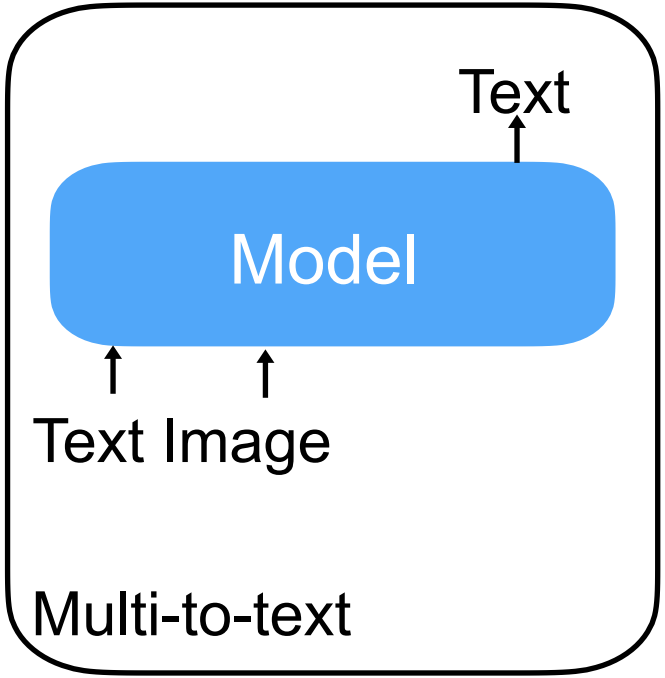
- Image preprocessing
  - E.g. split into patches and vectorize
- Image encoding
  - E.g. use a pre-existing CLIP model, get the ViT vectors from the last layer
- Provide the encodings to a LLM
  - E.g. linearly transform the vectors to be the model's embedding dimension
- Train/fine-tune on data that has text and images

Other Architectures

- Alternatives for CLIP: SiGLiP's pairwise sigmoid loss, Dino's (V2,V3) self supervised vision model.
- Flamingo is a family of VLMs that take as input visual data interleaved with text and produce free-form text as output.
- Similar to LLaVA: InstructBlip, Qwen VL, InternVL, TinyLLaVA, many more

Summary

- Vision architecture basics
  - ViT
- Learning image representations
  - CLIP
- Combining with a language model
  - Llava



Questions?

References

- The material of this lecture is inspired by CMU CS 11-711 Advanced NLP course.
- I have taken some images from the blog https://towardsdatascience.com/clip-intuitively-and-exhaustively-explained-1d02c07dbf40/

Thank You!!