UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Large-Scale Vision & Language Models: Architectures and Parallelism

About this training event

- Introduction to the MINERVA project
- Introduction to Large-Scale models in Vision, Language and Multimedia
    - SoTA Large-Scale Deep Learning architectures
    - For language, vision and multimodal AI
    - Scaling laws
- Tutorial on data and model parallelism

Most importantly…

- Not just give you information or pointers, but also practical channels and instructions to leverage HPC resources and/or assistance on how to get and use them.
- Your Swiss knife to HPC resources: **MINERVA**

OBJECTIVES

MINERVA is a **distributed, European-wide HPC-enabled AI application support service (AISC)**.

- It brings together EuroHPC Hosting Entities and partners representing major European stakeholders in AI.
- MINERVA acts as a central hub for cutting-edge European competences in large-scale ML/AI research and development.
- It started in January 2025, and the project's duration is 36 months

What MINERVA Aims To Achieve

- Establish and operate a Europe-wide Support Centre.
- **Interact with AI communities** through a User Advisory Board and Community Hub to identify needs and update the MINERVA service portfolio.
- Offer a **rich service portfolio** covering more levels of support, aligned with the European need to rely on open-source foundation models.
- Ensure models are developed according to ethical and responsible AI regulations.

How?

- **Knowledge transfer**: Publishing best practice guides and user guidelines.
- **Benchmarking**: Evaluating model performance on different supercomputers.
- **Data Access**: Providing information on access to public datasets.
- **Training Programs**: Providing training programs shaped by end-user feedback.
- **Community Hub**: Supporting large-scale open-source ML/AI research and development on HPC.

Before starting…

We would like to know a bit more about you and your needs to use HPC.

Fill in the MINERVA survey at
https://docs.google.com/forms/d/e/1FAIpQLSeWdowBcyc1d9bMsh6BGcolZQj_pECPpoCsF9Mdyf60Fcr68w/viewform

This data will be used to organize and tailor other training events!

MINERVA

Large-Scale Vision & Language Models: Architectures and Parallelism

Lorenzo Baraldi, Silvia Cascianelli, Sara Sarto

University of Modena and Reggio Emilia

THE ATTENTION OPERATOR

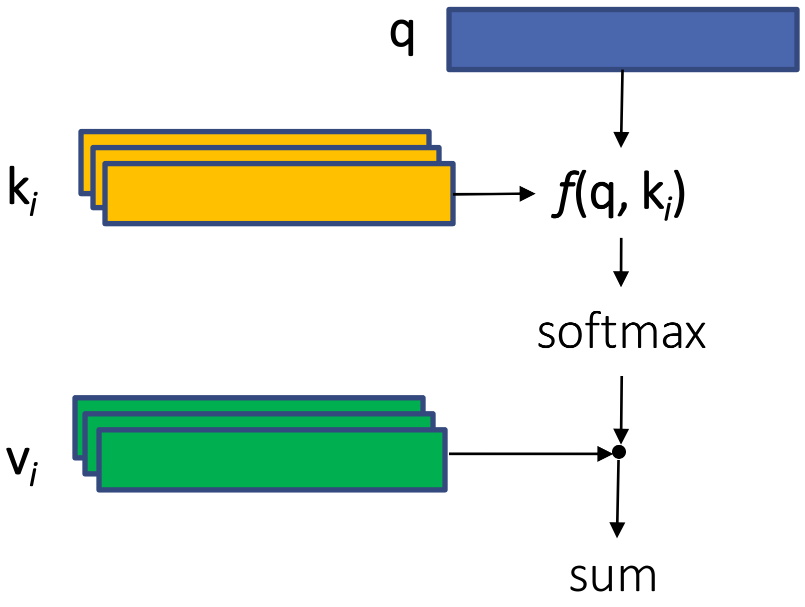
A new operator

Attention

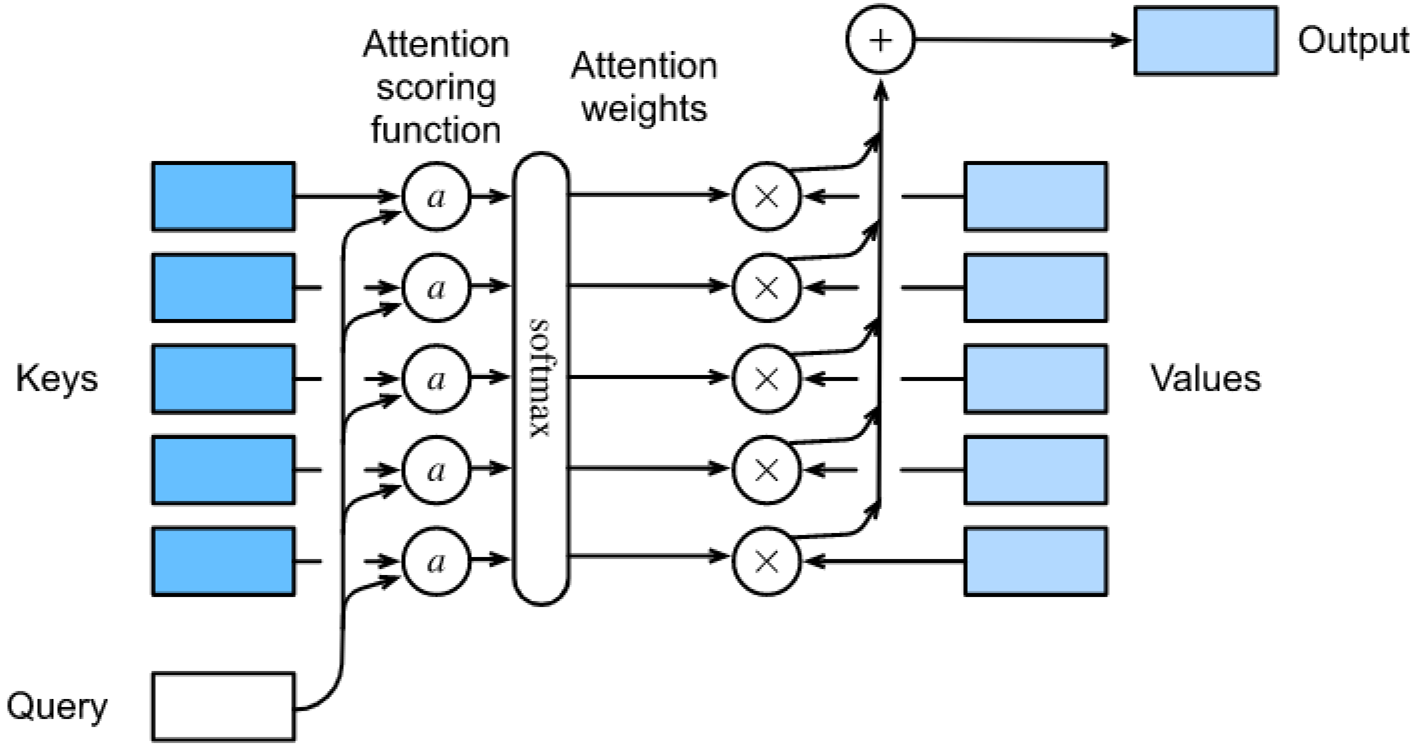
Provides a way to focus on part of an input set.

Given a query and pairs of keys and values,

- Compute similarities between queries and keys
- Normalizes similarities via softmax to obtain attention scores
- Multiplies values by the scores



Attention



Mathematically, suppose that we have a query $\mathbf{q} \in \mathbb{R}^q$ and m key-value pairs $(\mathbf{k}_1, \mathbf{v}_1), \ldots, (\mathbf{k}_m, \mathbf{v}_m)$, where any $\mathbf{k}_i \in \mathbb{R}^k$ and any $\mathbf{v}_i \in \mathbb{R}^v$. The attention pooling f is instantiated as a weighted sum of the values:

$$f(\mathbf{q}, (\mathbf{k}_1, \mathbf{v}_1), \ldots, (\mathbf{k}_m, \mathbf{v}_m)) = \sum_{i=1}^{m} \alpha(\mathbf{q}, \mathbf{k}_i)\mathbf{v}_i \in \mathbb{R}^v, \tag{10.3.1}$$

where the attention weight (scalar) for the query $\mathbf{q}$ and key $\mathbf{k}_i$ is computed by the softmax operation of an attention scoring function a that maps two vectors to a scalar:

$$\alpha(\mathbf{q}, \mathbf{k}_i) = \mathrm{softmax}(a(\mathbf{q}, \mathbf{k}_i)) = \frac{\exp(a(\mathbf{q}, \mathbf{k}_i))}{\sum_{j=1}^{m} \exp(a(\mathbf{q}, \mathbf{k}_j))} \in \mathbb{R}. \tag{10.3.2}$$

Attention

Attention

Provides a way to focus on part of an input set.

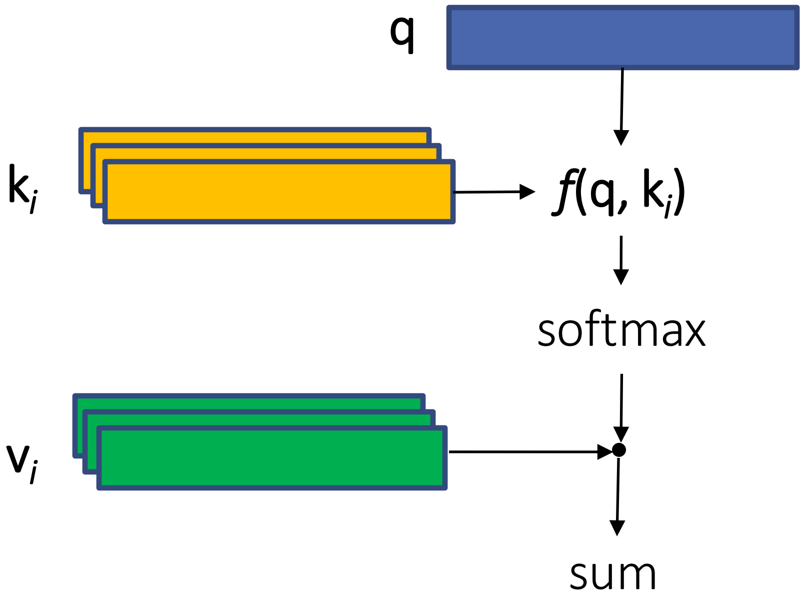
Given a query and pairs of keys and values,

- Compute similarities between queries and keys
- Normalizes similarities via softmax to obtain attention scores
- Multiplies values by the scores

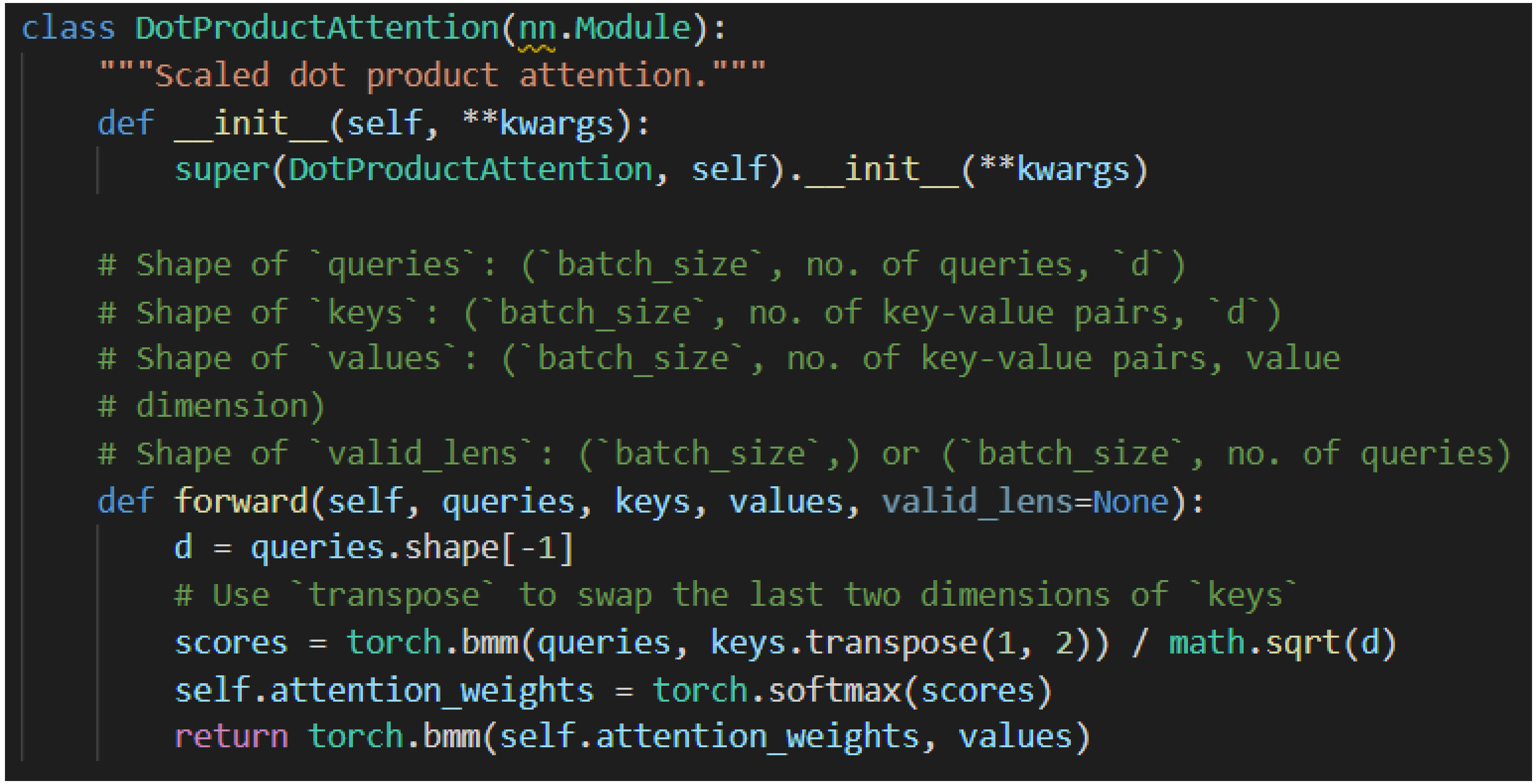
Similarity function

- Additive attention, e.g. $w^T \tanh(w_q q + w_k k)$
- Dot-product attention

  $a(\mathbf{q}, \mathbf{k}) = \mathbf{q}^\top \mathbf{k} / \sqrt{d}$



Dot-product attention



```
class DotProductAttention(nn.Module):
    """Scaled dot product attention."""
    def __init__(self, **kwargs):
        super(DotProductAttention, self).__init__(**kwargs)

    # Shape of `queries`: (`batch_size`, no. of queries, `d`)
    # Shape of `keys`: (`batch_size`, no. of key-value pairs, `d`)
    # Shape of `values`: (`batch_size`, no. of key-value pairs, value
    # dimension)
    # Shape of `valid_lens`: (`batch_size`,) or (`batch_size`, no. of queries)
    def forward(self, queries, keys, values, valid_lens=None):
        d = queries.shape[-1]
        # Use `transpose` to swap the last two dimensions of `keys`
        scores = torch.bmm(queries, keys.transpose(1, 2)) / math.sqrt(d)
        self.attention_weights = torch.softmax(scores)
        return torch.bmm(self.attention_weights, values)
```

Convolution vs Self-attention

Self Attention

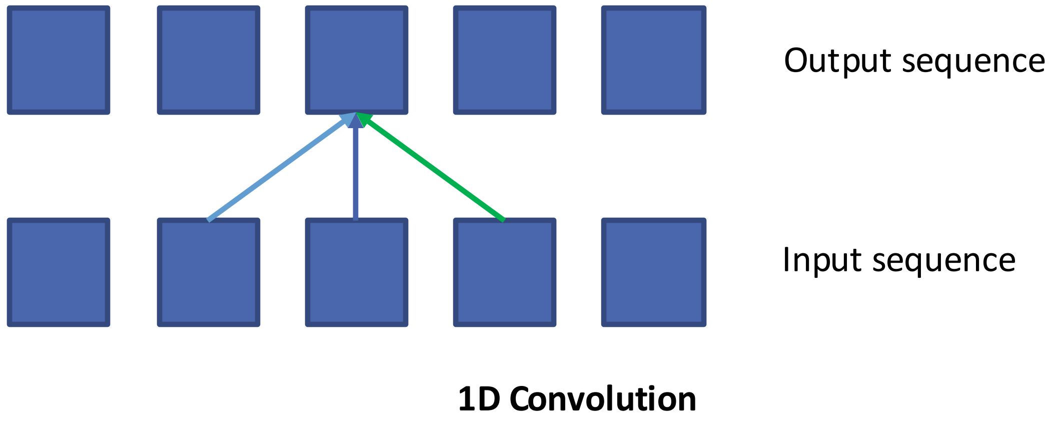
“Refine” each element of the sequence by treating it as **query**, and the whole sequence as keys and values.

Actually: queries, keys and values are three different linear projections of each element of the input sequence.

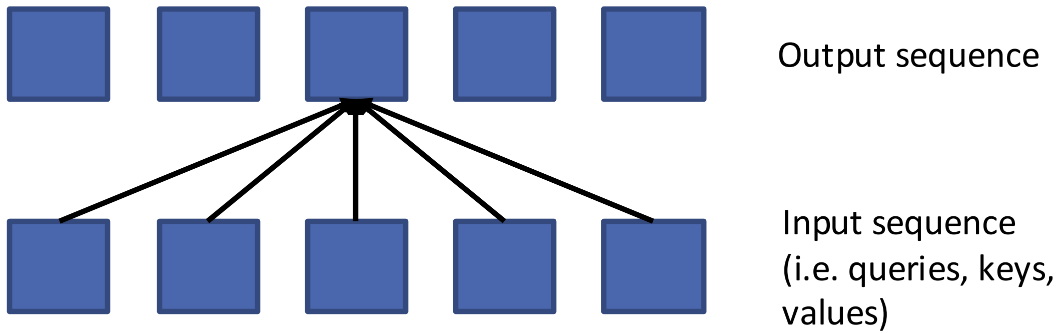
Receptive field is infinite!

Constant path length between two different positions

Trivial to parallelize during training!

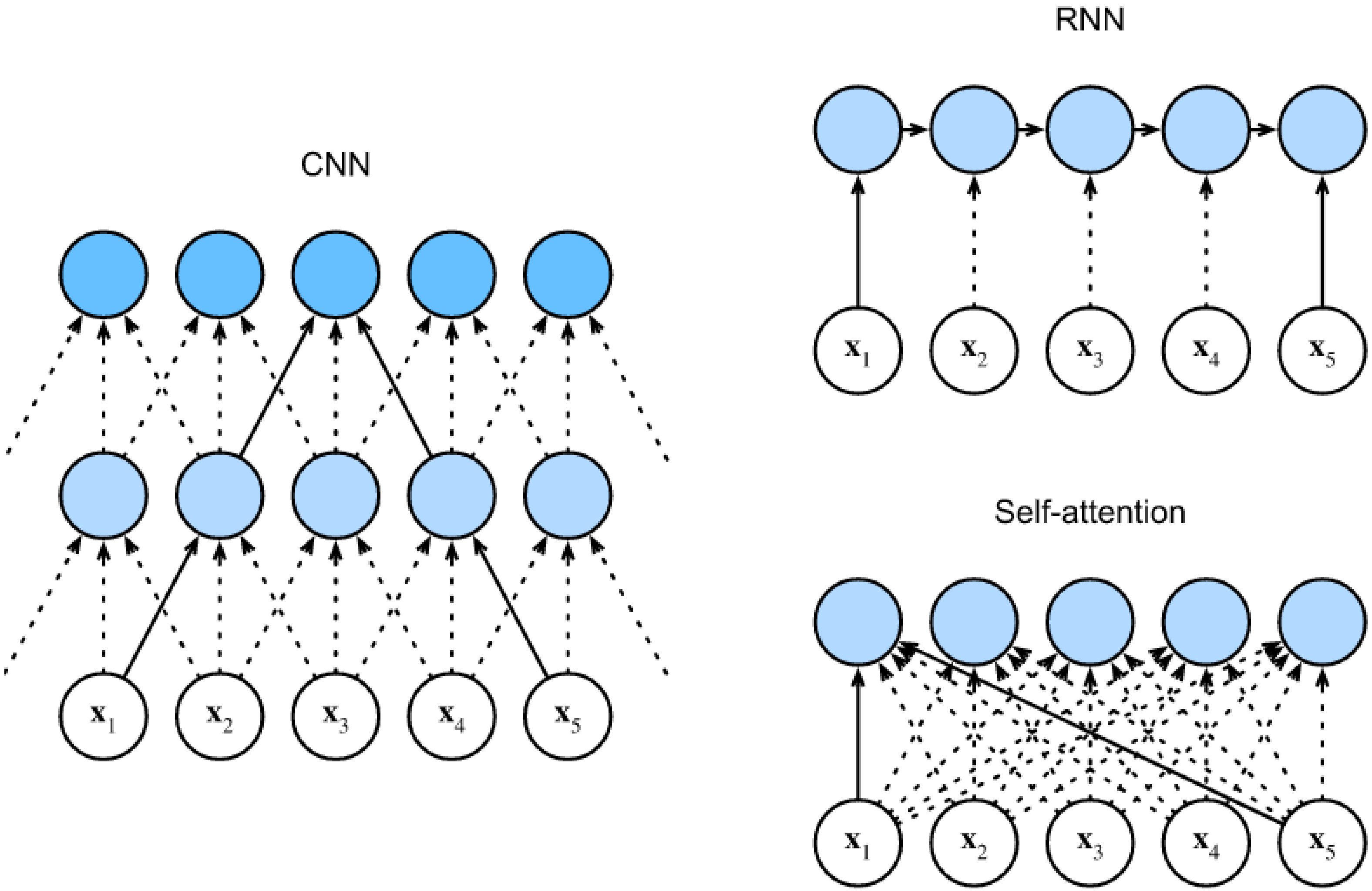


1D Convolution



Self-Attention

CNNs vs RNNs vs Self-attention


CNN

RNN

Self-attention

Multi-head Self-attention

- Linearly project the input sequence *h* times with different weights, instead of doing this only once.
- To From a sequence with length *T*, we obtain:
  - *Q*: matrix of queries, (h, T, d_k)
  - *K*: matrix of keys, (h, T , d_k)
  - *V*: matrix of values, (h, T, d_v)

- Apply scaled dot-product attention over each “head” (i.e. over each element of axis 1)
- Concatenate the result and project back to a lower dimensionality

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_\mathrm{h})W^O$$

$$\text{where } \mathrm{head}_i = \mathrm{Attention}(Q[i], K[i], V[i])$$

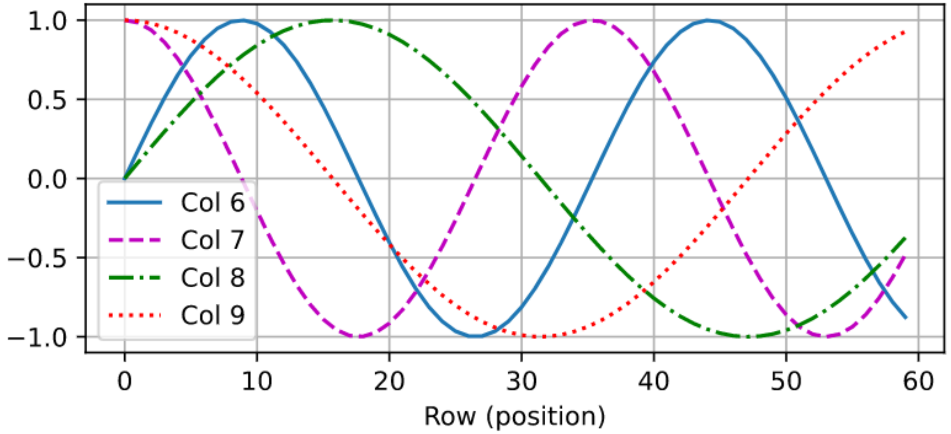
- Can be done in parallel, with batched matrix multiplication.

Positional encoding

- Self-attention is permutation invariant
  - Given a query, if we change the order of keys and values, result does not change.
  - Ok for encoding sets. Not for sequences or images....☹
- To use **order information**, we can inject absolute or relative positional information by adding positional encoding to the input representations.
- Positional encodings can be learned (simple nn.Parameter) or fixed. In the original Transformer, they were defined as sinusoids. With this, attention can capture both absolute and relative positional information (i.e. distances between items!). Now, many more alternatives (e.g. relative, Rotary, ALiBi)

Suppose that the input representation $\mathbf{X} \in \mathbb{R}^{n \times d}$ contains the d-dimensional embeddings for n tokens of a sequence. The positional encoding outputs $\mathbf{X} + \mathbf{P}$ using a positional embedding matrix $\mathbf{P} \in \mathbb{R}^{n \times d}$ of the same shape, whose element on the i^th row and the $(2j)^\mathrm{th}$ or the $(2j+1)^\mathrm{th}$ column is

$$\begin{aligned} p_{i,2j} &= \sin\left(\frac{i}{10000^{2j/d}}\right),\\ p_{i,2j+1} &= \cos\left(\frac{i}{10000^{2j/d}}\right). \end{aligned}$$



A Self-attentive language model for translation

Encoder

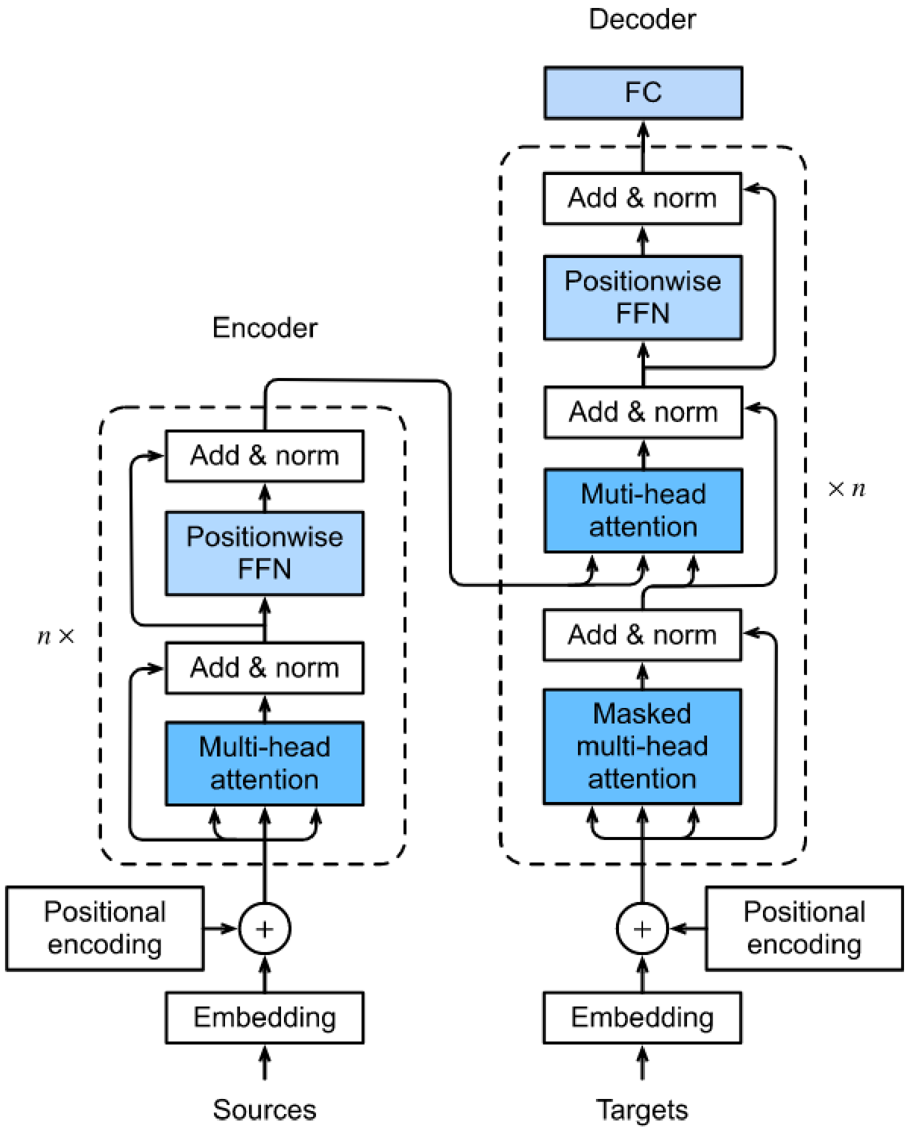
Uses self-attention on its input

Multiple attention layers stacked together (with add+norm) and feed-forward layers (linear layers applied timewise).

Decoder

Self-attention on words

Cross-attention on encoder outputs: use decoder sequences as queries, encoder outputs as key/values.



Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. Attention is all you need. In NIPS 2017.

SCALING LAWS

Training Transformers

“Architecture alone does not make a model”

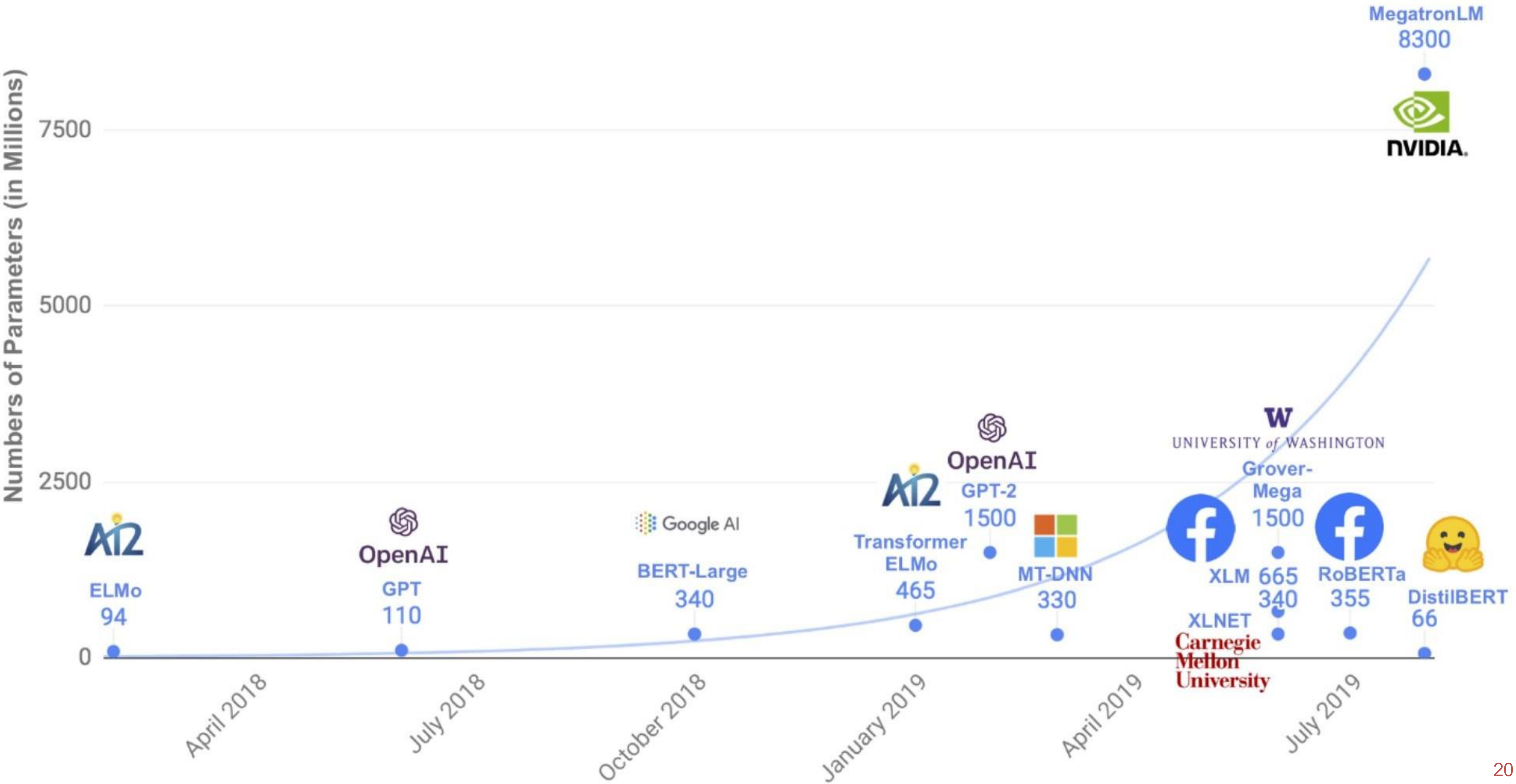
Architecture + Training = Model

- A model expresses different properties depending on how it is trained
- Like nature vs. nurture, both impact what the model does
- Training is what influences parameters

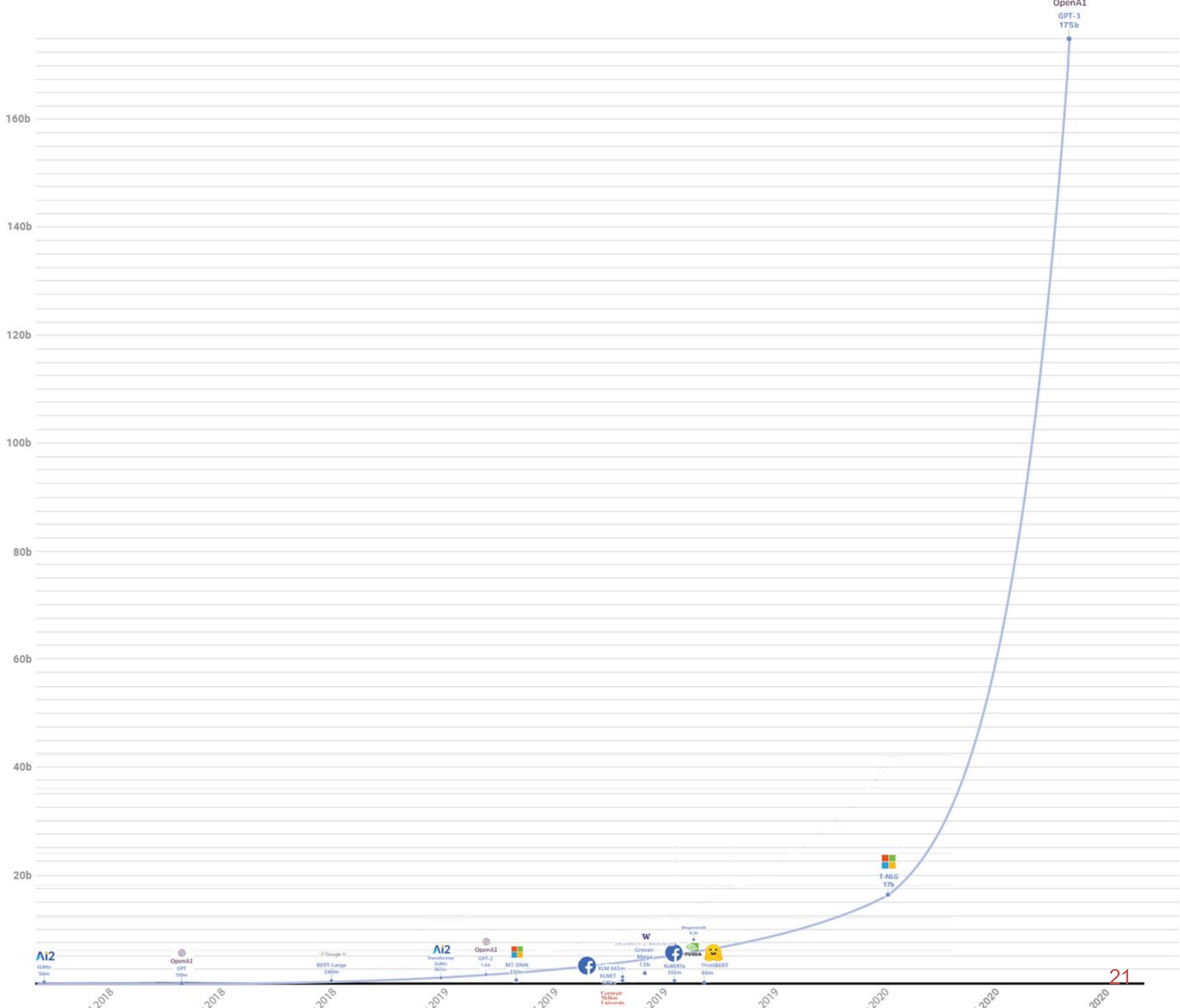
The trend

- Early works: **Task-specific architectures** using shared word vector representations
- BERT: Pretraining and Fine tuning
- GPT-3: **One/Few shot learning**

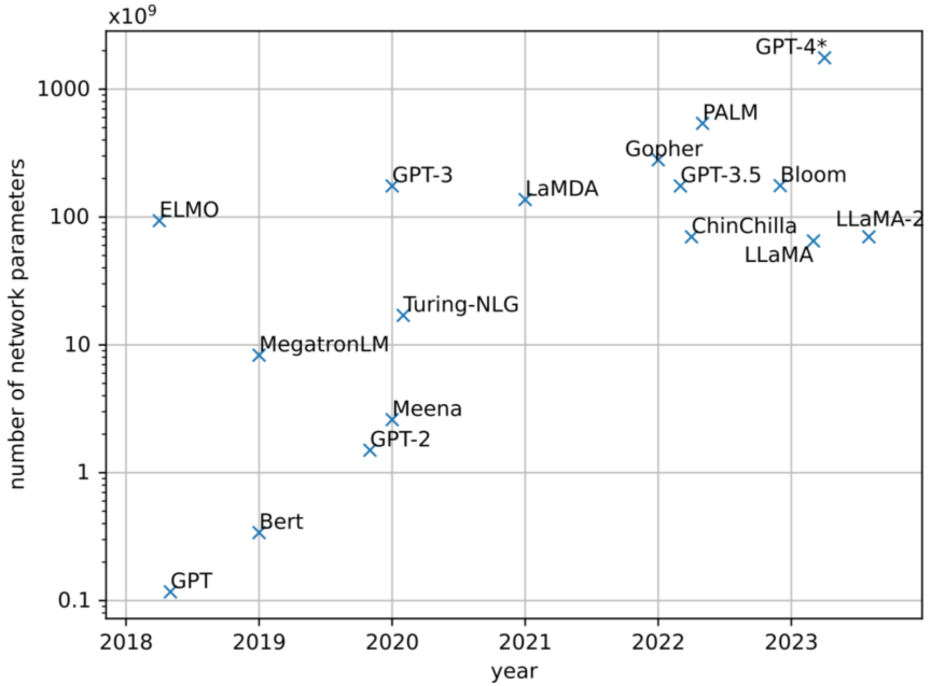
Trend: the bigger, the better



175 Billion Parameters!

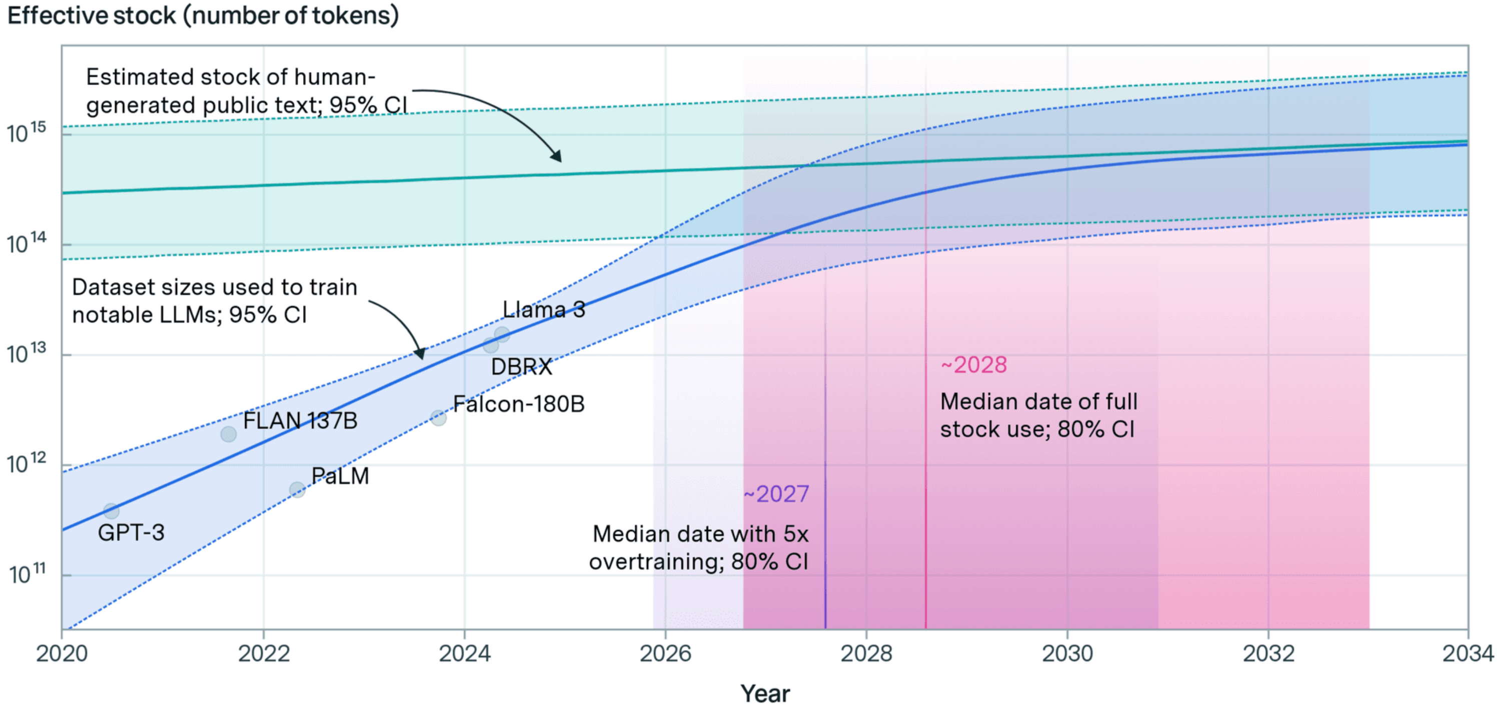


Today…



Projections of the stock of public text and data usage

EPOCH AI



Scaling Laws

- Scientific problem: How does model performance change with increasing scale?
- Engineering importance: Training frontier models costs millions of dollars
- Safety importance: Models are black boxes with often unexpected capabilities

Scaling Laws for Neural Language Models, Kaplan et al. 2020

Main RQ: What is the relationship between compute, data, the number of parameters, and performance?

Kaplan et al., 2020 - Main Idea

- Train models of vastly different scale (data, compute, params) and find the loss is predictably related to the scale
- Offer a predictive framework
- They make certain recommendations based on the scaling laws
  - Ratio between parameters and data
  - Don’t train to convergence
  - Critical batch size

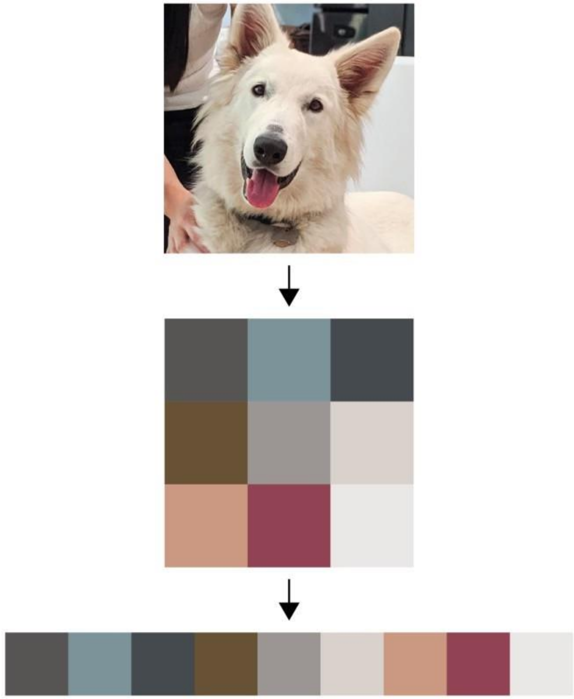
Kaplan et al., 2020 - Main Results

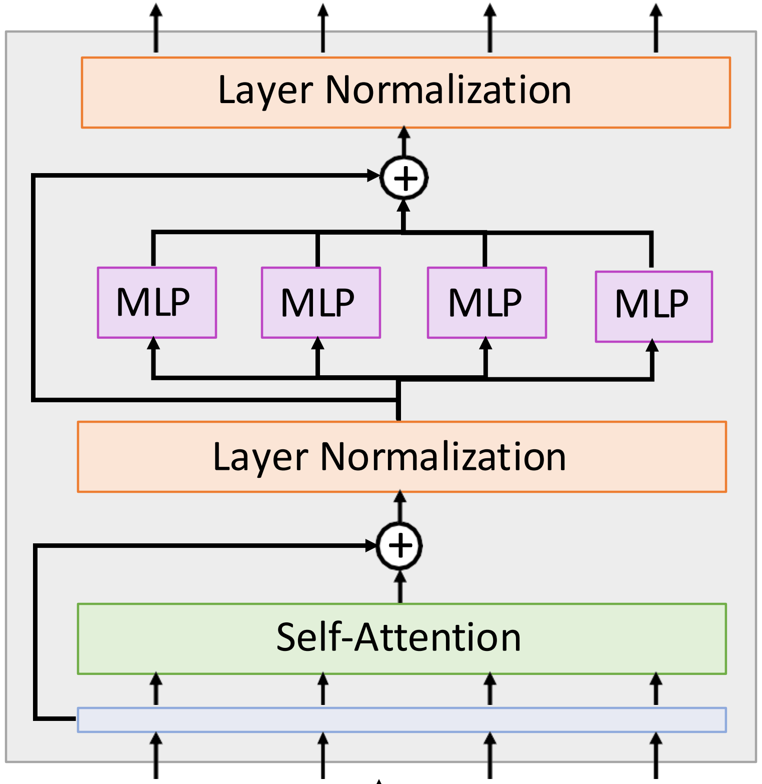
1. Performance depends strongly on scale, weakly on model shape
2. Performance scales with each of compute (C), data (D), and # of parameters (N) when not bottlenecked by the other two
3. Performance improves predictably if we scale N and D in tandem, but suffers diminishing returns if N or D is held fixed
4. Training curves are predictable, so we can (roughly) predict the final loss by extrapolating the early part of the training curve, regardless(ish) of model size
5. We can predict how well the model will perform on OOD data by looking at the training validation accuracy
6. Large models are more sample efficient than small ones
7. Convergence is inefficient!
8. The ideal batch size (B_{crit}) is roughly a power of the loss only

TRANSFORMERS FOR VISION

Idea: Standard Transformer on Pixels

Treat an image as a set of pixel values



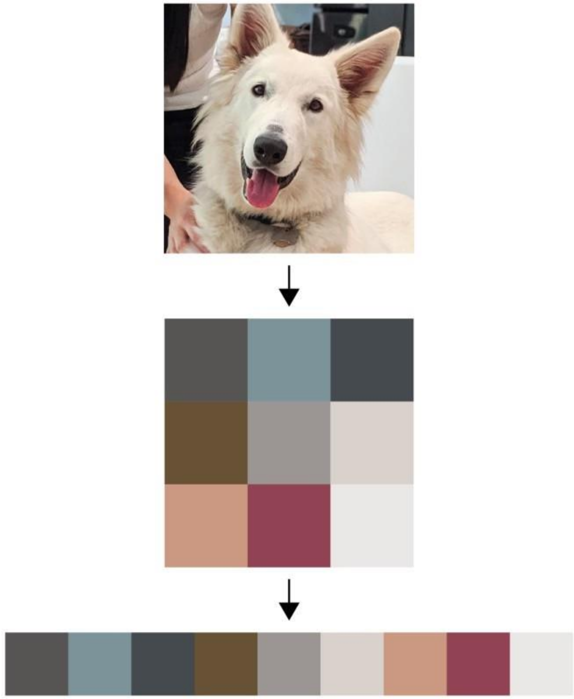


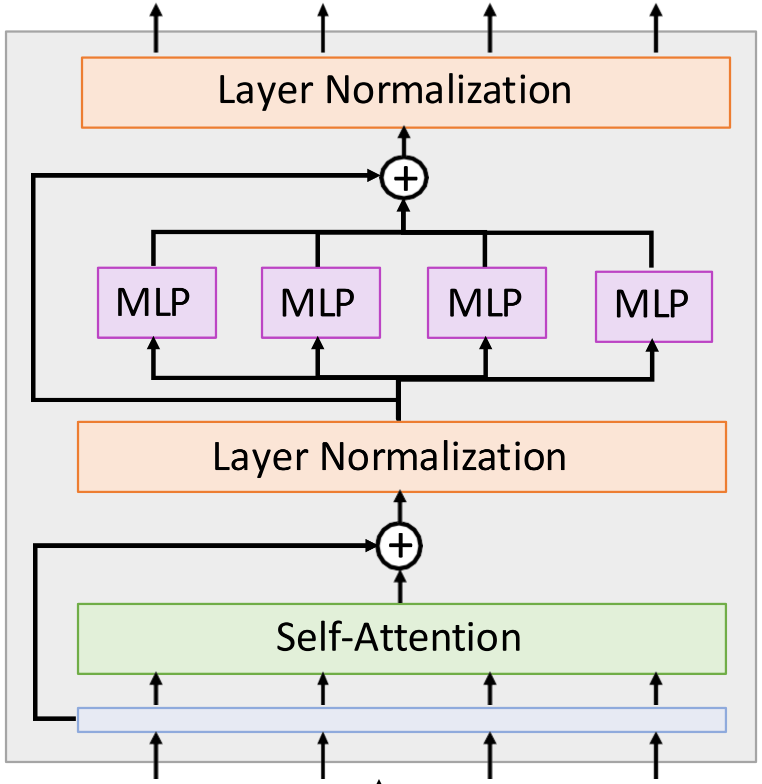
Feed as input to standard Transformer

Chen et al, “Generative Pretraining from Pixels”, ICML 2020

Idea: Standard Transformer on Pixels

Treat an image as a set of pixel values





Feed as input to standard Transformer

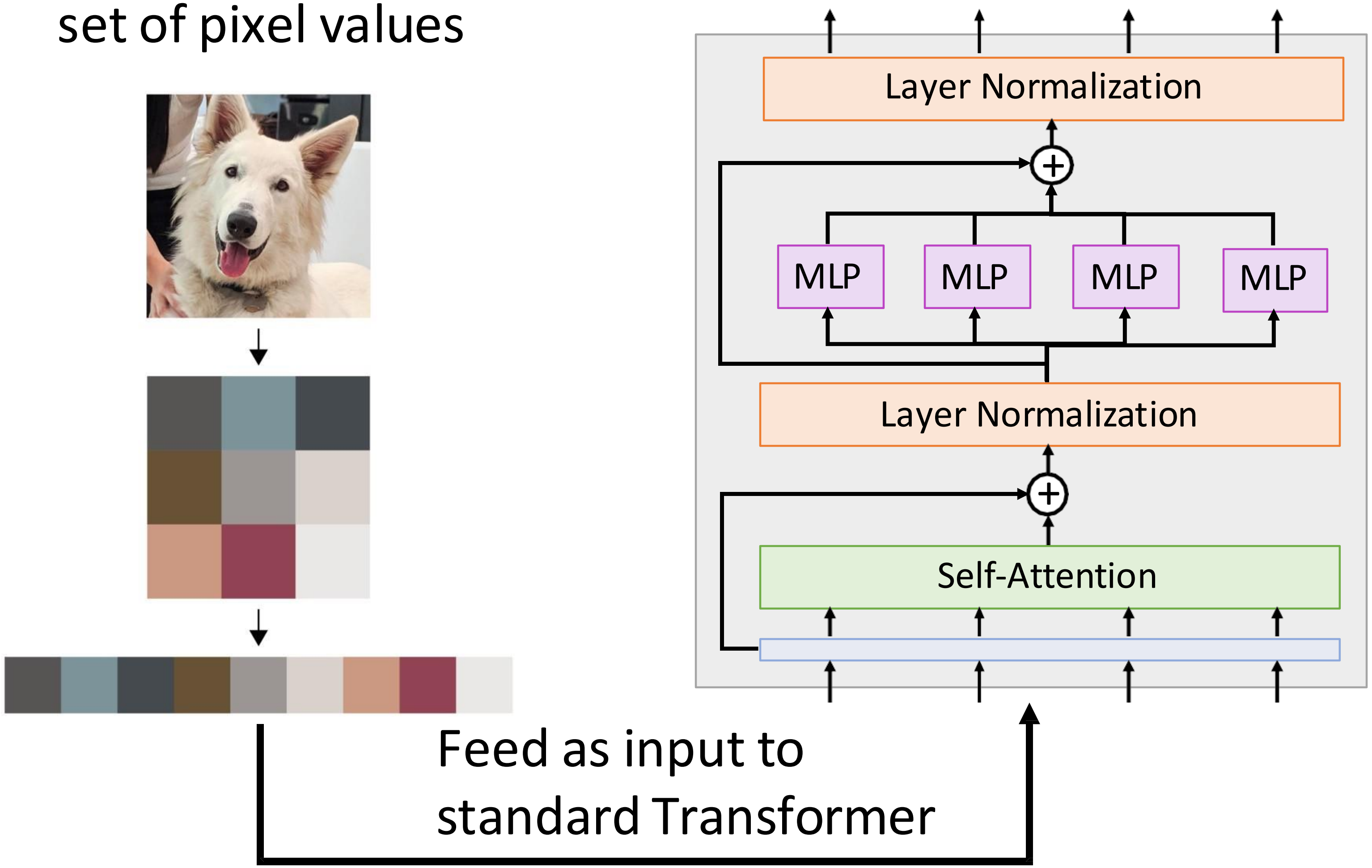
Problem: Memory use!

R x R image needs R^4 elements per attention matrix

Chen et al, “Generative Pretraining from Pixels”, ICML 2020

Idea: Standard Transformer on Pixels

Treat an image as a set of pixel values



Feed as input to standard Transformer

Problem: Memory use!

R x R image needs R^4 elements per attention matrix

R=128, 48 layers, 16 heads per layer takes 768GB of memory for attention matrices for a single example…

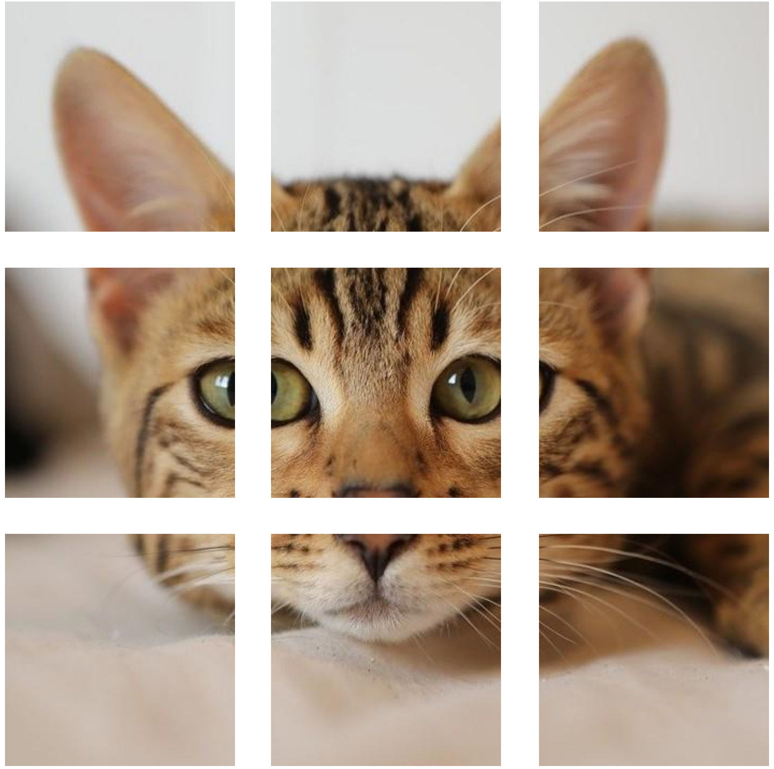
Chen et al, “Generative Pretraining from Pixels”, ICML 2020

Idea: Standard Transformer on Patches

Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Cat image is free for commercial use under a Pixabay license

Idea: Standard Transformer on Patches

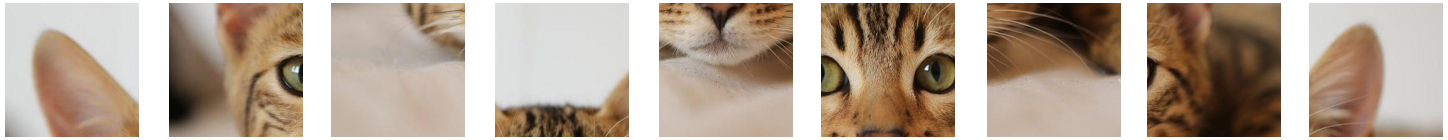


Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Cat image is free for commercial use under a Pixabay license

Idea: Standard Transformer on Patches

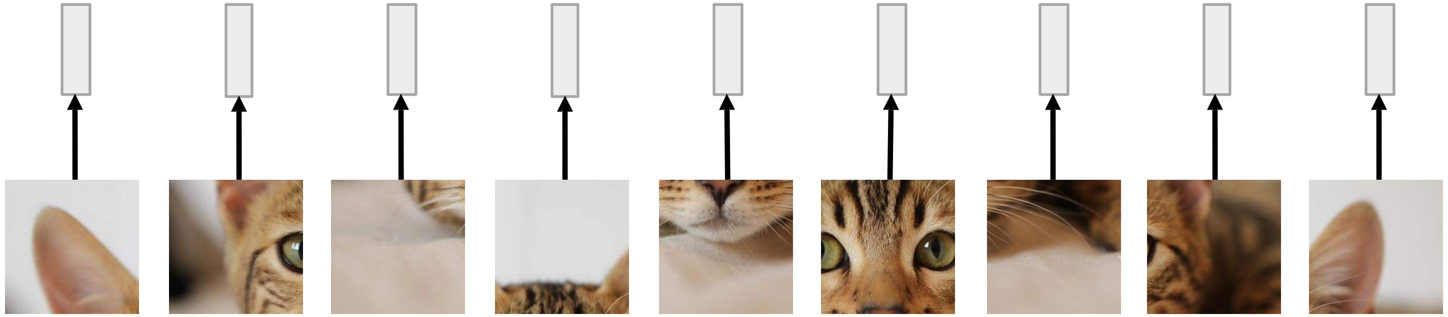
N input patches, each of shape 3x16x16



Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Cat image is free for commercial use under a Pixabay license

Idea: Standard Transformer on Patches



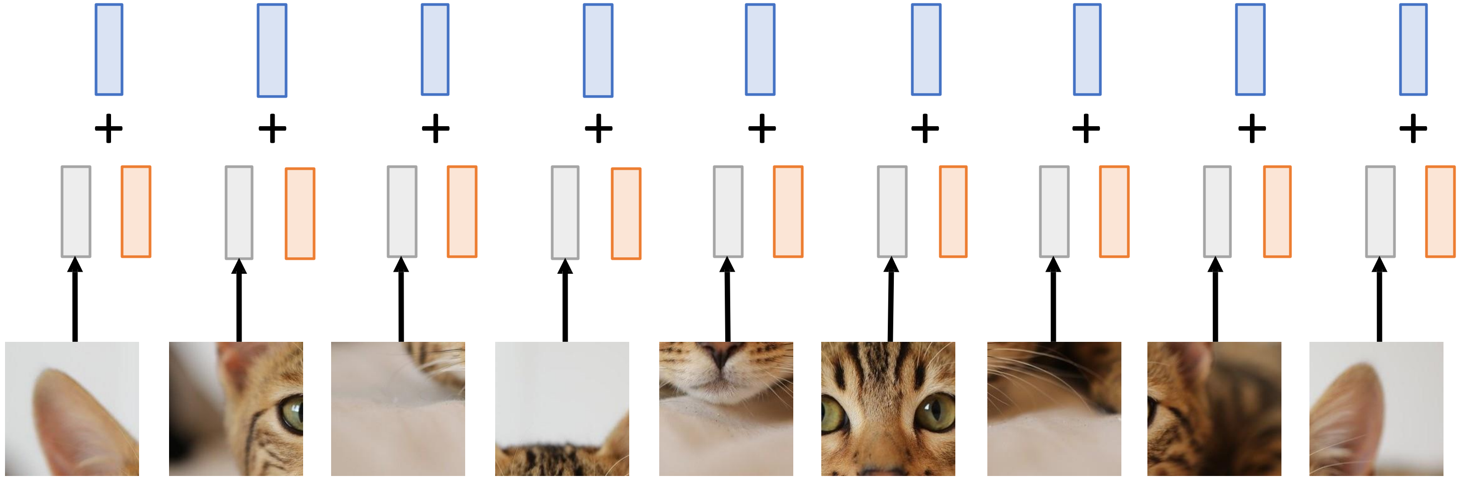
Linear projection to D-dimensional vector

N input patches, each of shape 3x16x16

Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Cat image is free for commercial use under a Pixabay license

Idea: Standard Transformer on Patches



Add positional embedding: learned D-dim vector per position

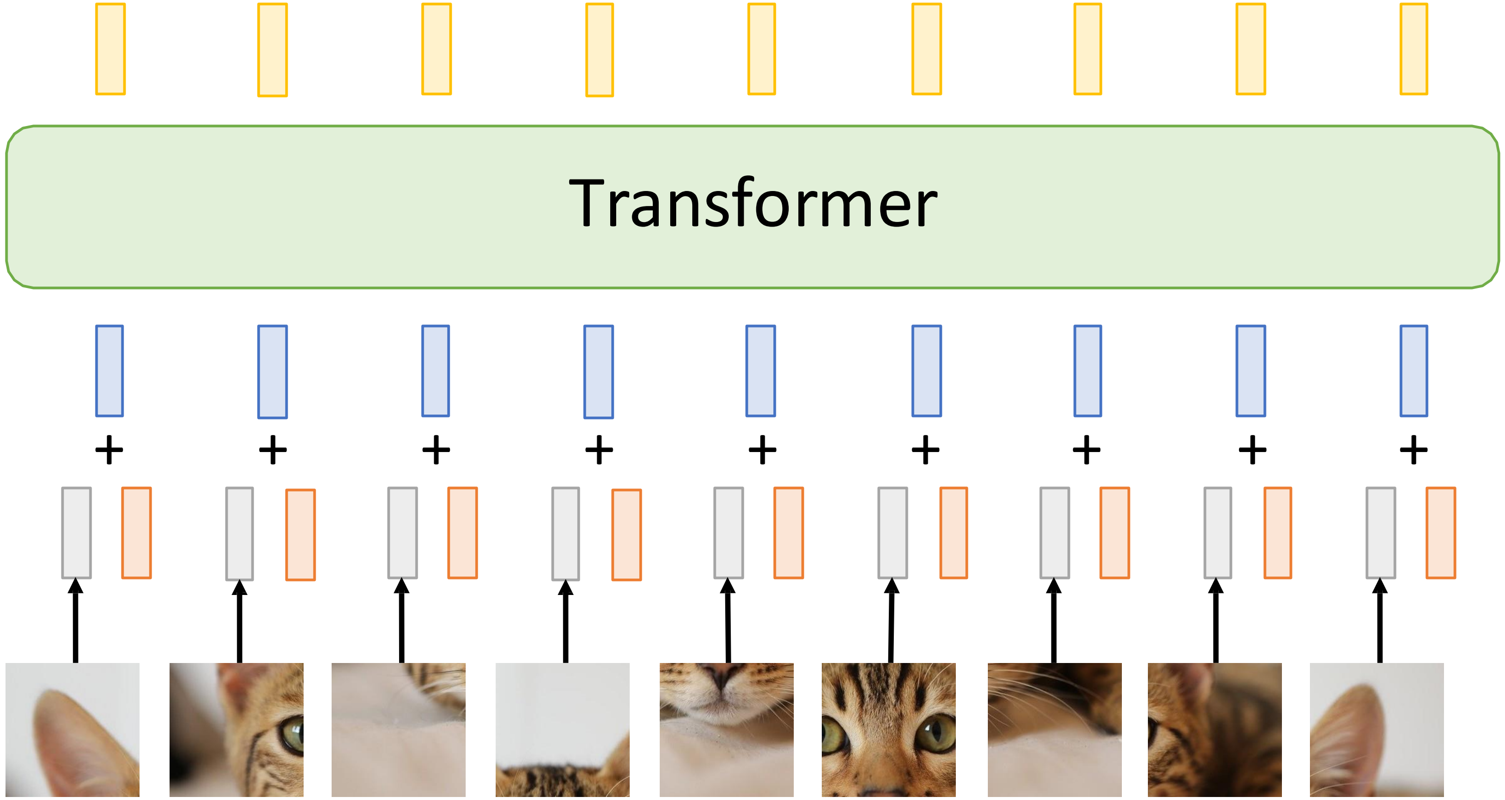
Linear projection to D-dimensional vector

N input patches, each of shape 3x16x16

Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Cat image is free for commercial use under a Pixabay license

Idea: Standard Transformer on Patches



Output vectors

Exact same as NLP Transformer!

Add positional embedding: learned D-dim vector per position

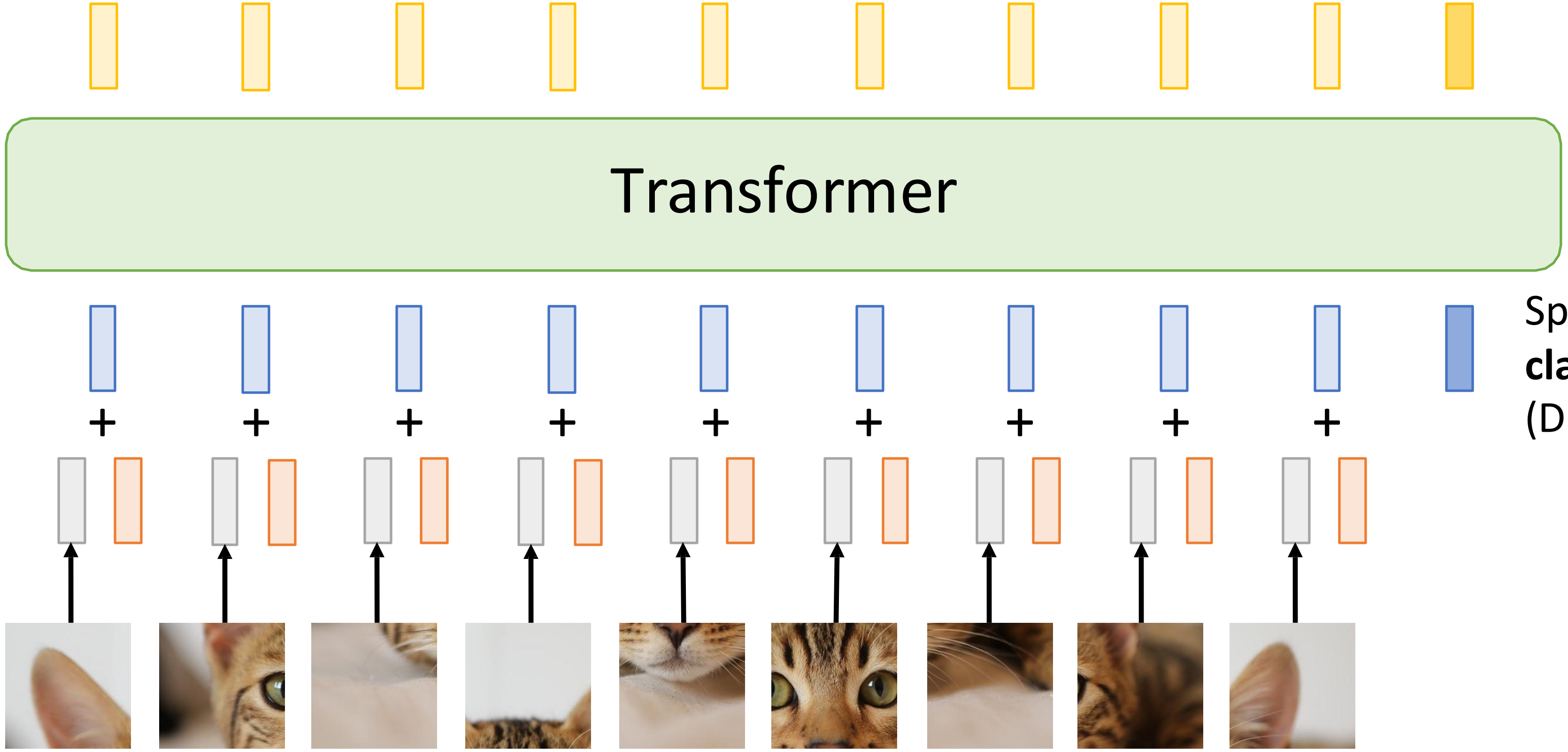
Linear projection to D-dimensional vector

N input patches, each of shape 3x16x16

Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Cat image is free for commercial use under a Pixabay license

Idea: Standard Transformer on Patches



Output vectors

Exact same as NLP Transformer!

Add positional embedding: learned D-dim vector per position

Linear projection to D-dimensional vector

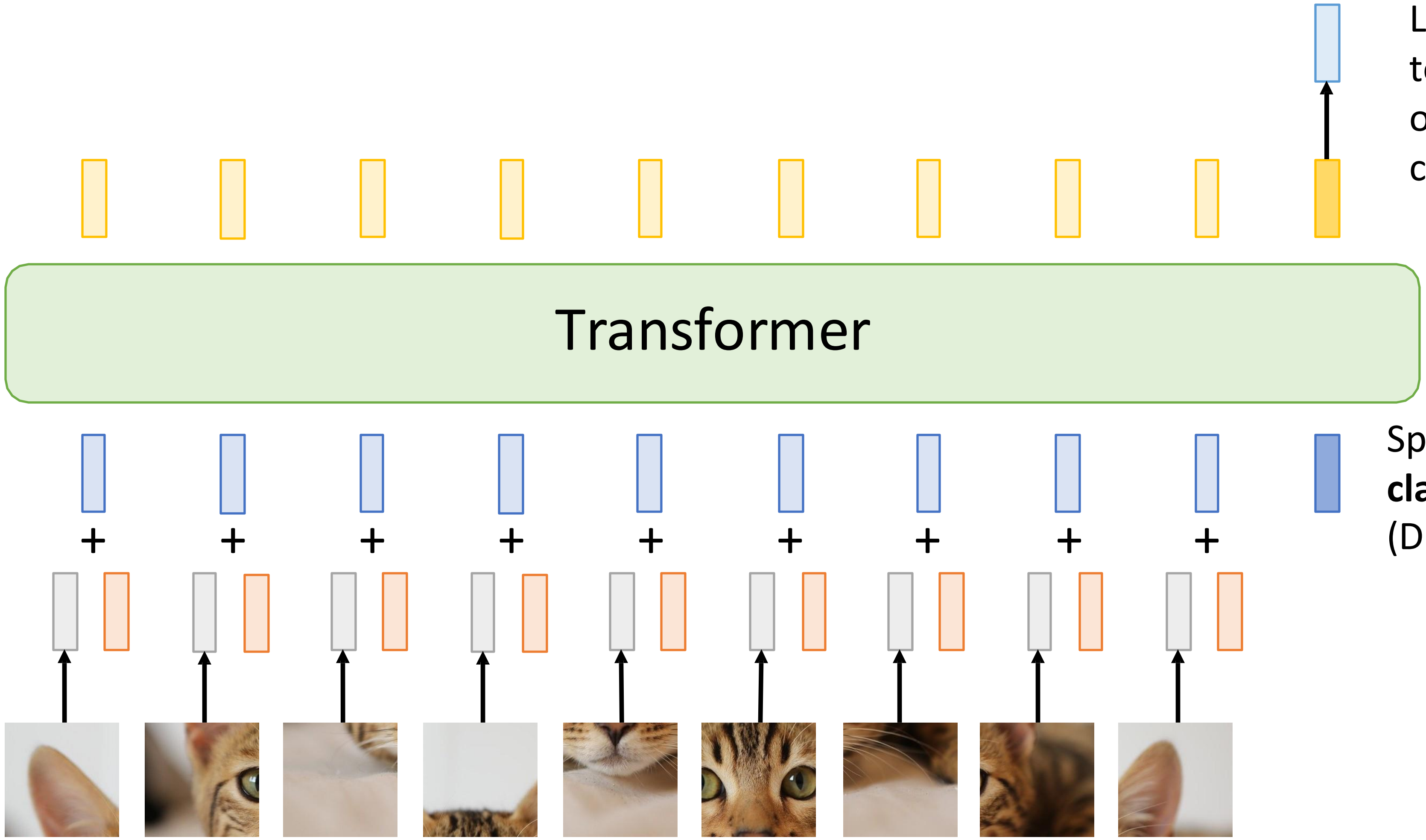
N input patches, each of shape 3x16x16

Special extra input: **classification token** (D dims, learned)

Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Cat image is free for commercial use under a Pixabay license

Idea: Standard Transformer on Patches



Output vectors

Exact same as NLP Transformer!

Add positional embedding: learned D-dim vector per position

Linear projection to D-dimensional vector

N input patches, each of shape 3x16x16

Linear projection to C-dim vector of predicted class scores

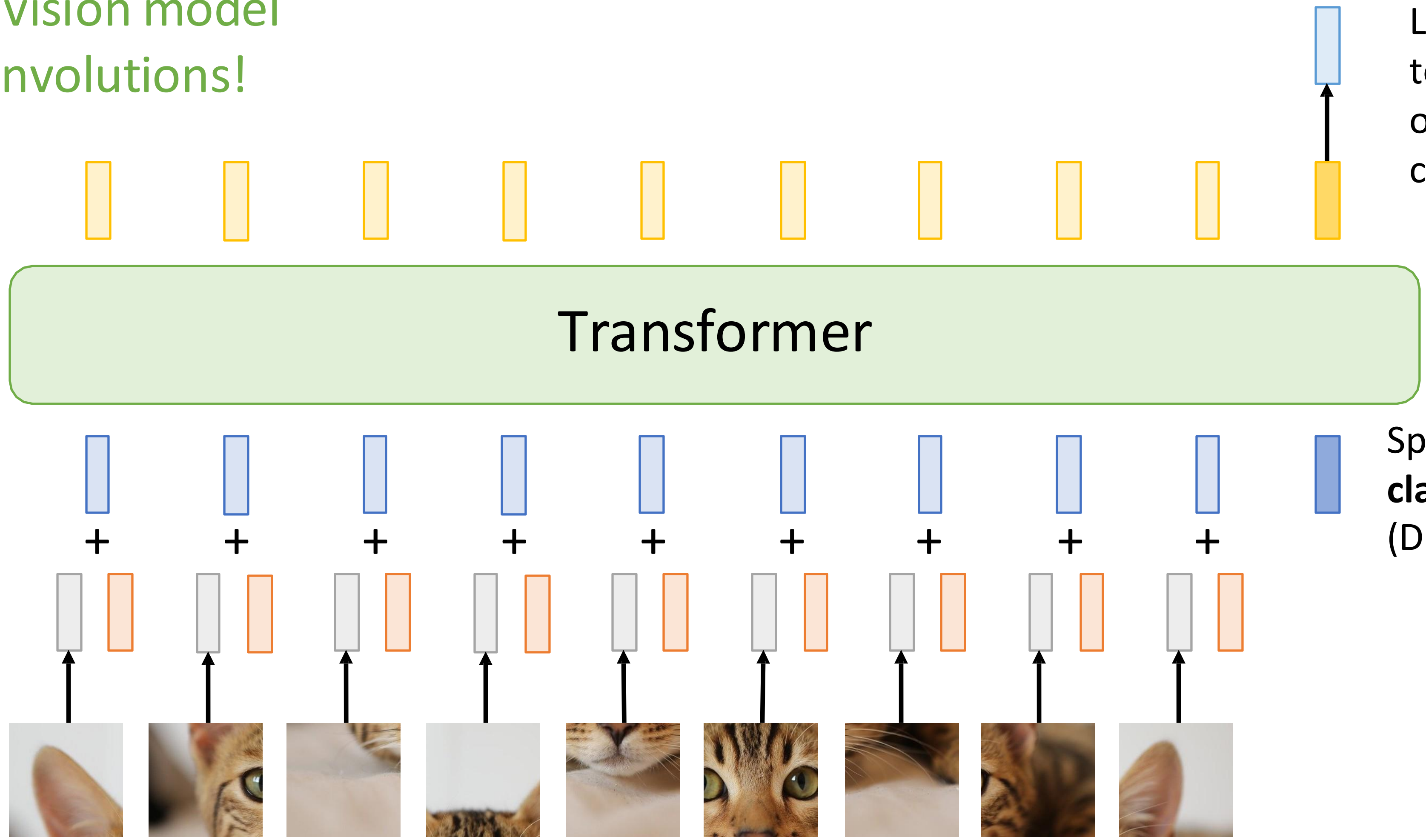
Special extra input: **classification token** (D dims, learned)

Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Cat image is free for commercial use under a Pixabay license

Vision Transformer (ViT)

Computer vision model with no convolutions!


Output vectors
Exact same as NLP Transformer!
Transformer
Add positional embedding: learned D-dim vector per position
Linear projection to D-dimensional vector
N input patches, each of shape 3x16x16
Linear projection to C-dim vector of predicted class scores
Special extra input: **classification token** (D dims, learned)
+

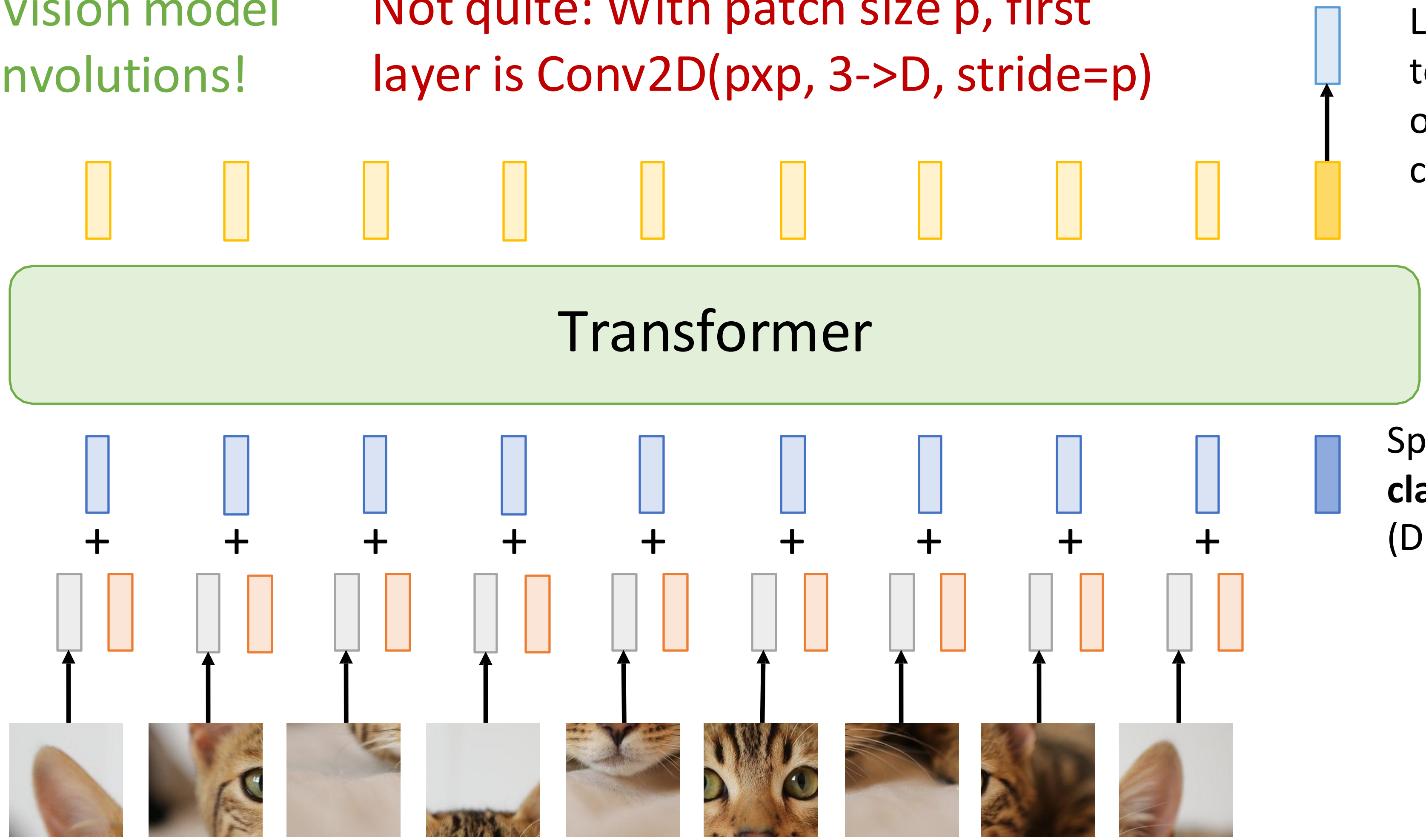
Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Cat image is free for commercial use under a Pixabay license

Vision Transformer (ViT)

Computer vision model with no convolutions!

Not quite: With patch size p, first layer is Conv2D(pxp, 3->D, stride=p)



Linear projection to C-dim vector of predicted class scores

Output vectors

Exact same as NLP Transformer!

Add positional embedding: learned D-dim vector per position

Special extra input: **classification token** (D dims, learned)

Linear projection to D-dimensional vector

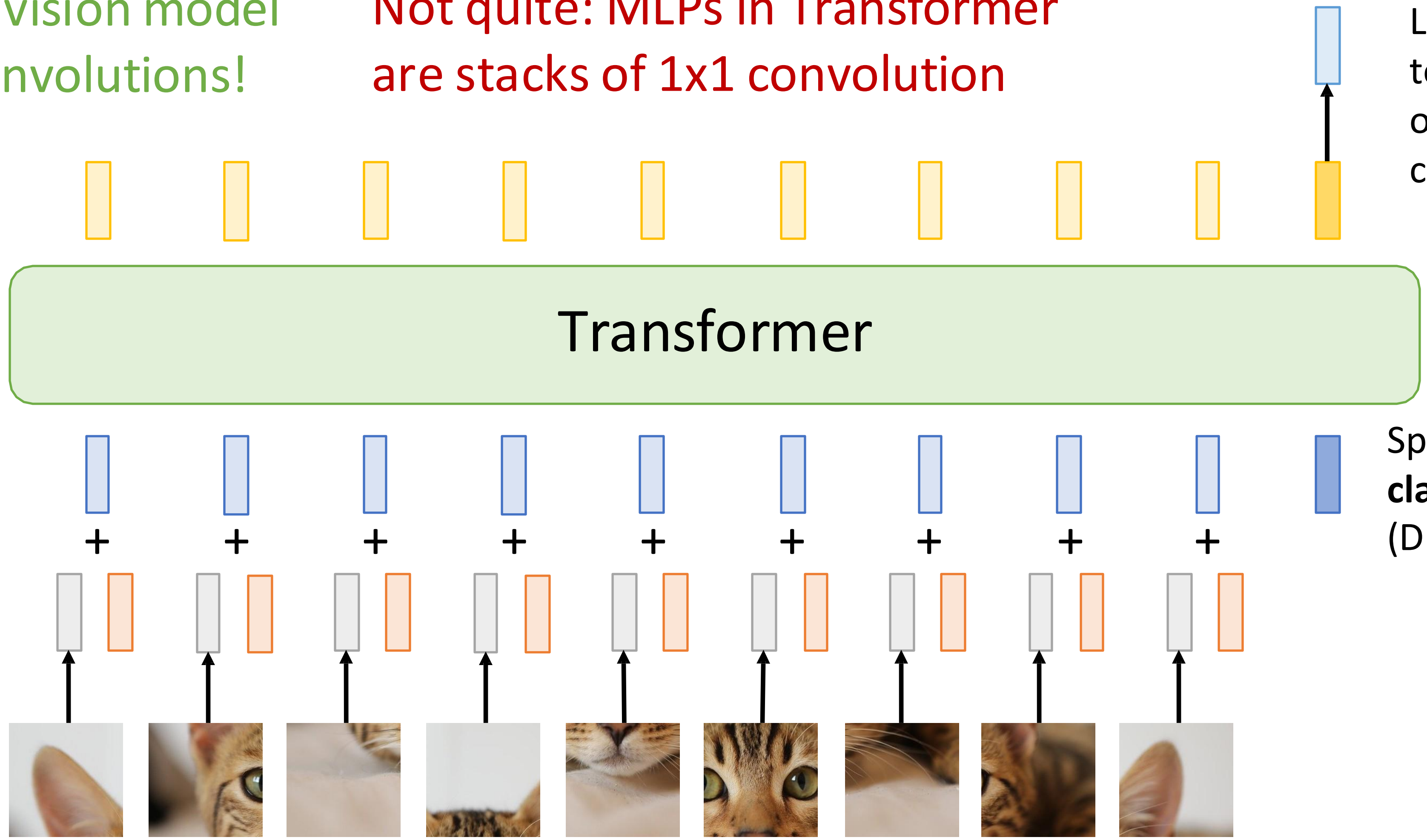
N input patches, each of shape 3x16x16

Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

Vision Transformer (ViT)

Computer vision model with no convolutions!

Not quite: MLPs in Transformer are stacks of 1x1 convolution



Linear projection to C-dim vector of predicted class scores

Output vectors

Exact same as NLP Transformer!

Add positional embedding: learned D-dim vector per position

Special extra input: **classification token** (D dims, learned)

Linear projection to D-dimensional vector

N input patches, each of shape 3x16x16

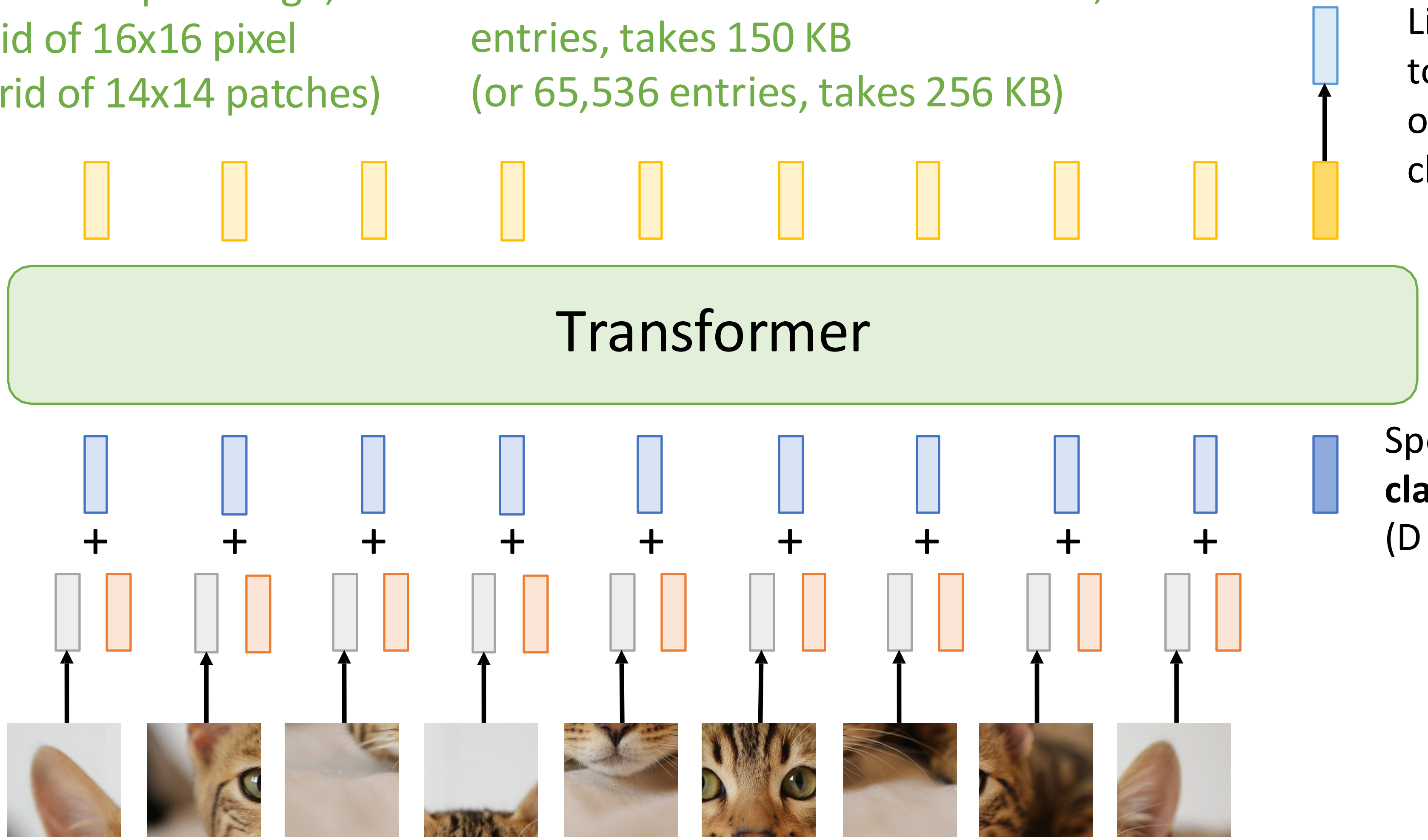
Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

Cat image is free for commercial use under a Pixabay license

Vision Transformer (ViT)

In practice: take 224x224 input image, divide into 14x14 grid of 16x16 pixel patches (or 16x16 grid of 14x14 patches)

Each attention matrix has 14^4 = 38,416 entries, takes 150 KB (or 65,536 entries, takes 256 KB)



Output vectors

Linear projection to C-dim vector of predicted class scores

Exact same as NLP Transformer!

Add positional embedding: learned D-dim vector per position

Special extra input: **classification token** (D dims, learned)

Linear projection to D-dimensional vector

N input patches, each of shape 3x16x16

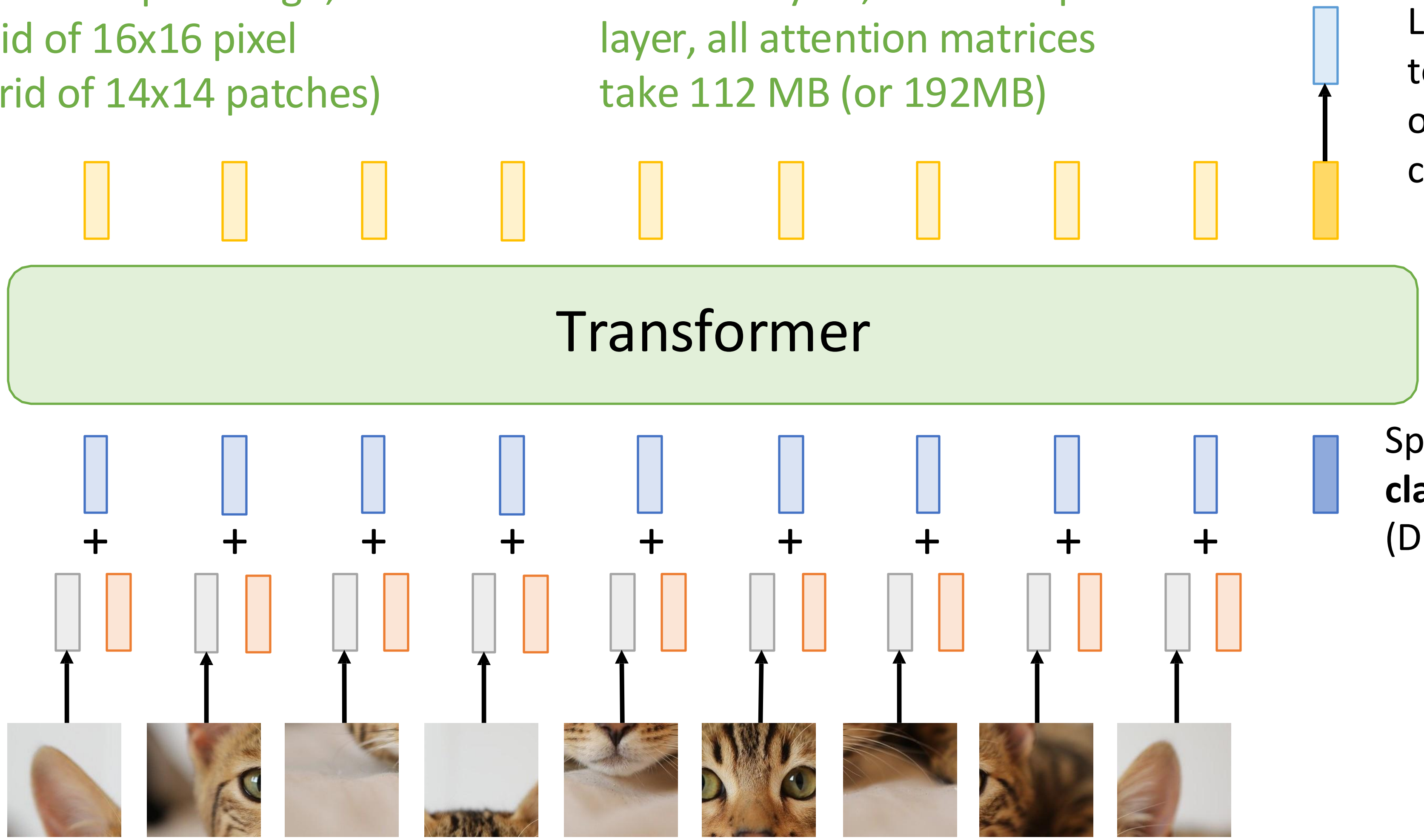
Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Cat image is free for commercial use under a Pixabay license

Vision Transformer (ViT)

In practice: take 224x224 input image, divide into 14x14 grid of 16x16 pixel patches (or 16x16 grid of 14x14 patches)

With 48 layers, 16 heads per layer, all attention matrices take 112 MB (or 192MB)



Output vectors

Linear projection to C-dim vector of predicted class scores

Exact same as NLP Transformer!

Add positional embedding: learned D-dim vector per position

Special extra input: **classification token** (D dims, learned)

Linear projection to D-dimensional vector

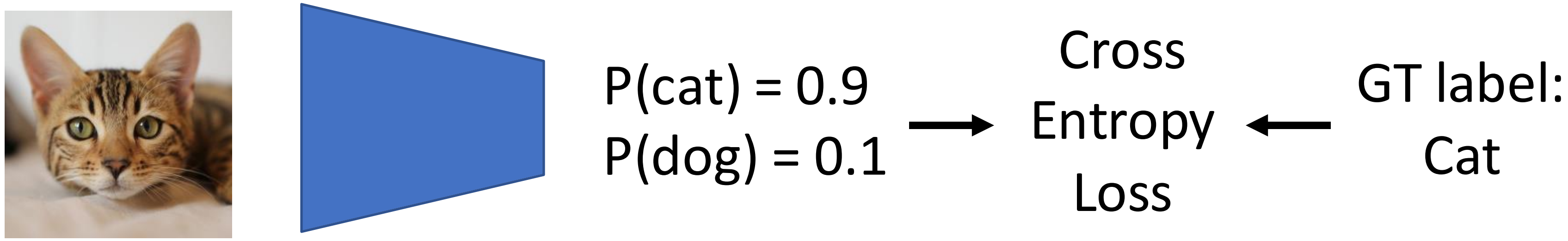
N input patches, each of shape 3x16x16

Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Cat image is free for commercial use under a Pixabay license

Improving ViT: Distillation

Step 1: Train a **teacher model** on images and ground-truth labels



P(cat) = 0.9
P(dog) = 0.1

Cross Entropy Loss

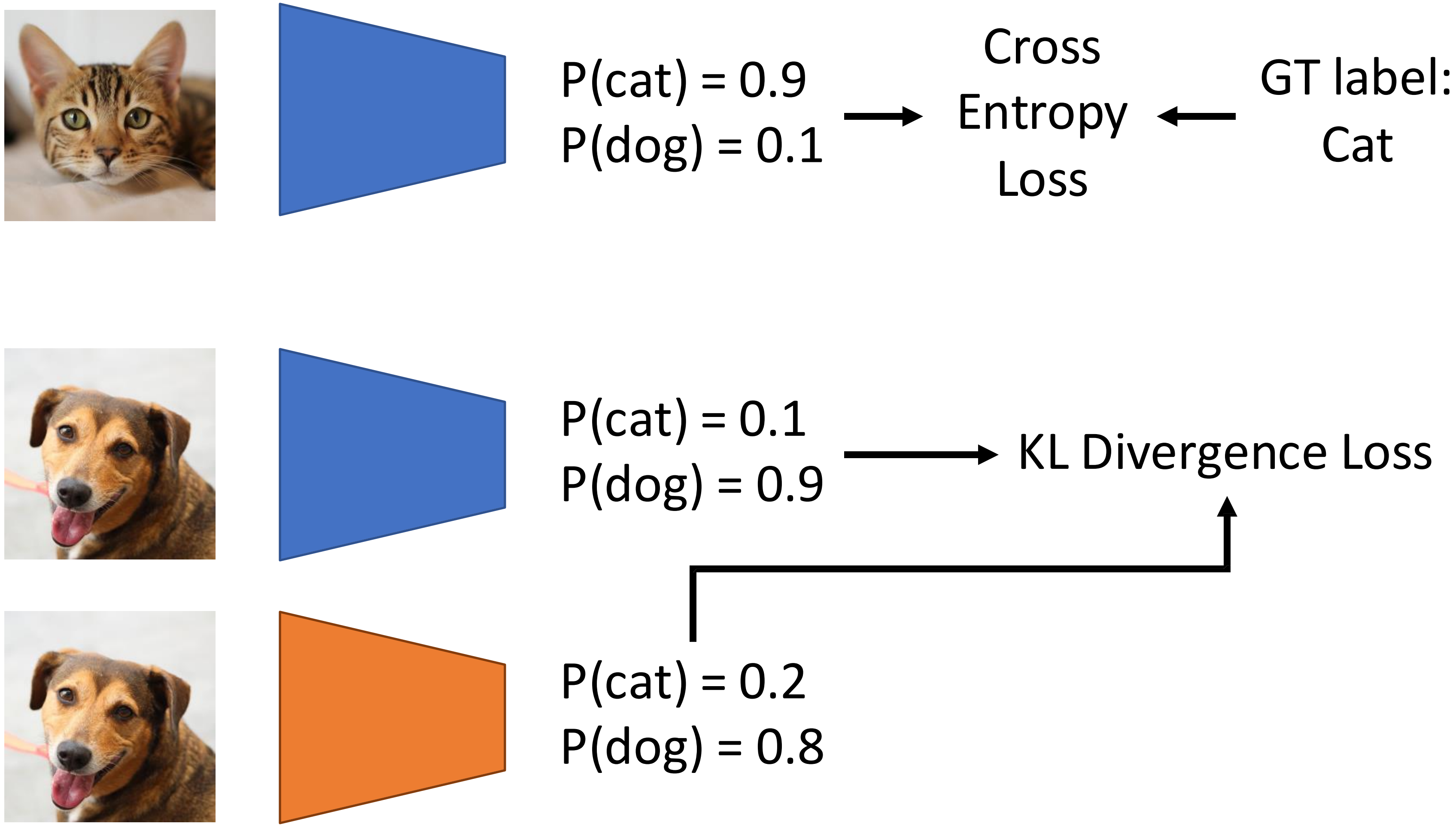
GT label: Cat

Hinton et al, "Distilling the knowledge in a neural network", NeurIPS Deep Learning and Representation Learning Workshop, 2015

Improving ViT: Distillation

Step 1: Train a **teacher model** on images and ground-truth labels

Step 2: Train a **student model** to match predictions from the **teacher**



P(cat) = 0.9
P(dog) = 0.1

Cross Entropy Loss

GT label: Cat

P(cat) = 0.1
P(dog) = 0.9

KL Divergence Loss

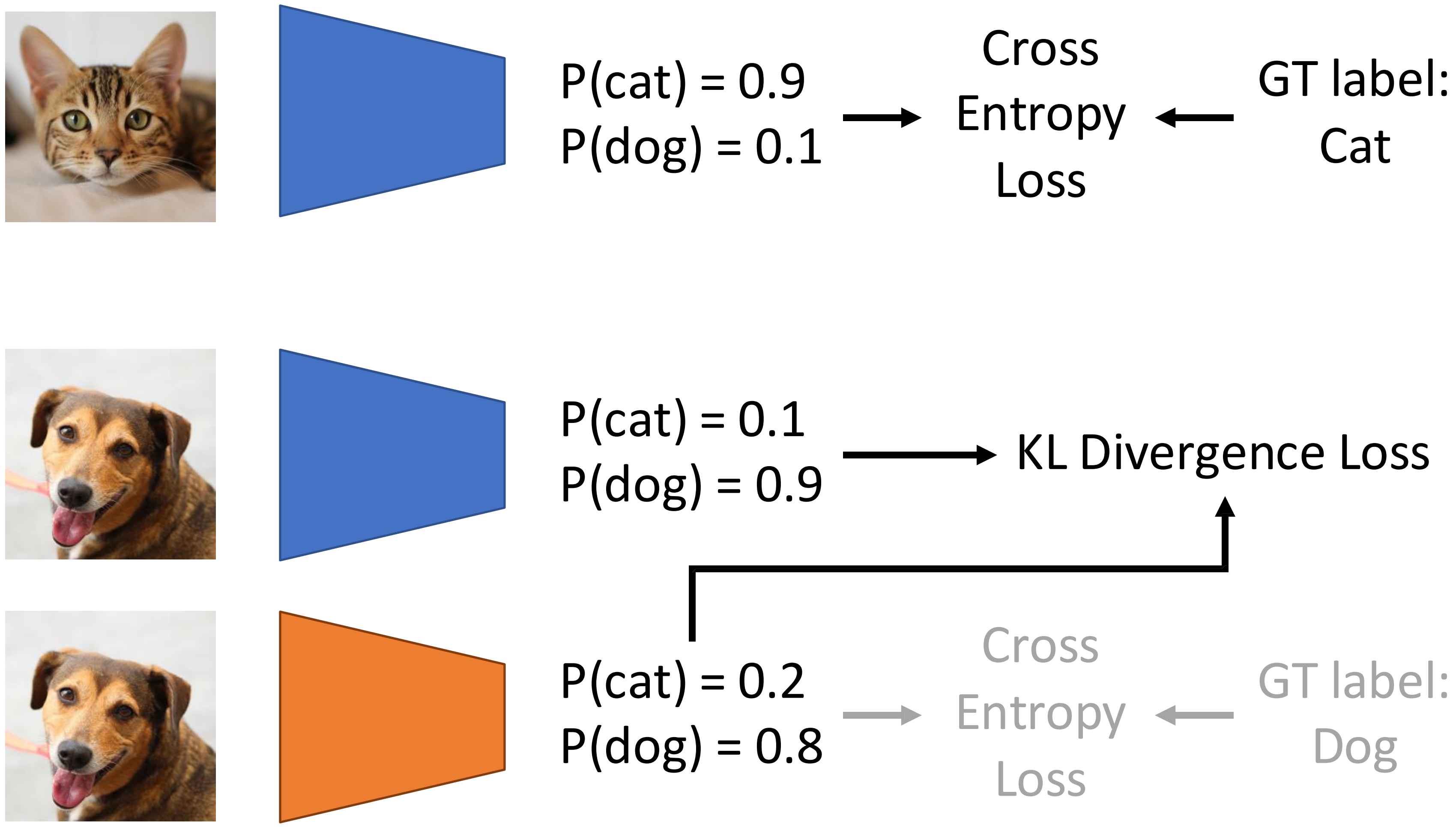
P(cat) = 0.2
P(dog) = 0.8

Hinton et al, "Distilling the knowledge in a neural network", NeurIPS Deep Learning and Representation Learning Workshop, 2015

Improving ViT: Distillation

Step 1: Train a **teacher model** on images and ground-truth labels

Step 2: Train a **student model** to match predictions from the **teacher** (sometimes also to match GT labels)



P(cat) = 0.9
P(dog) = 0.1

Cross Entropy Loss

GT label: Cat

P(cat) = 0.1
P(dog) = 0.9

KL Divergence Loss

P(cat) = 0.2
P(dog) = 0.8

Cross Entropy Loss

GT label: Dog

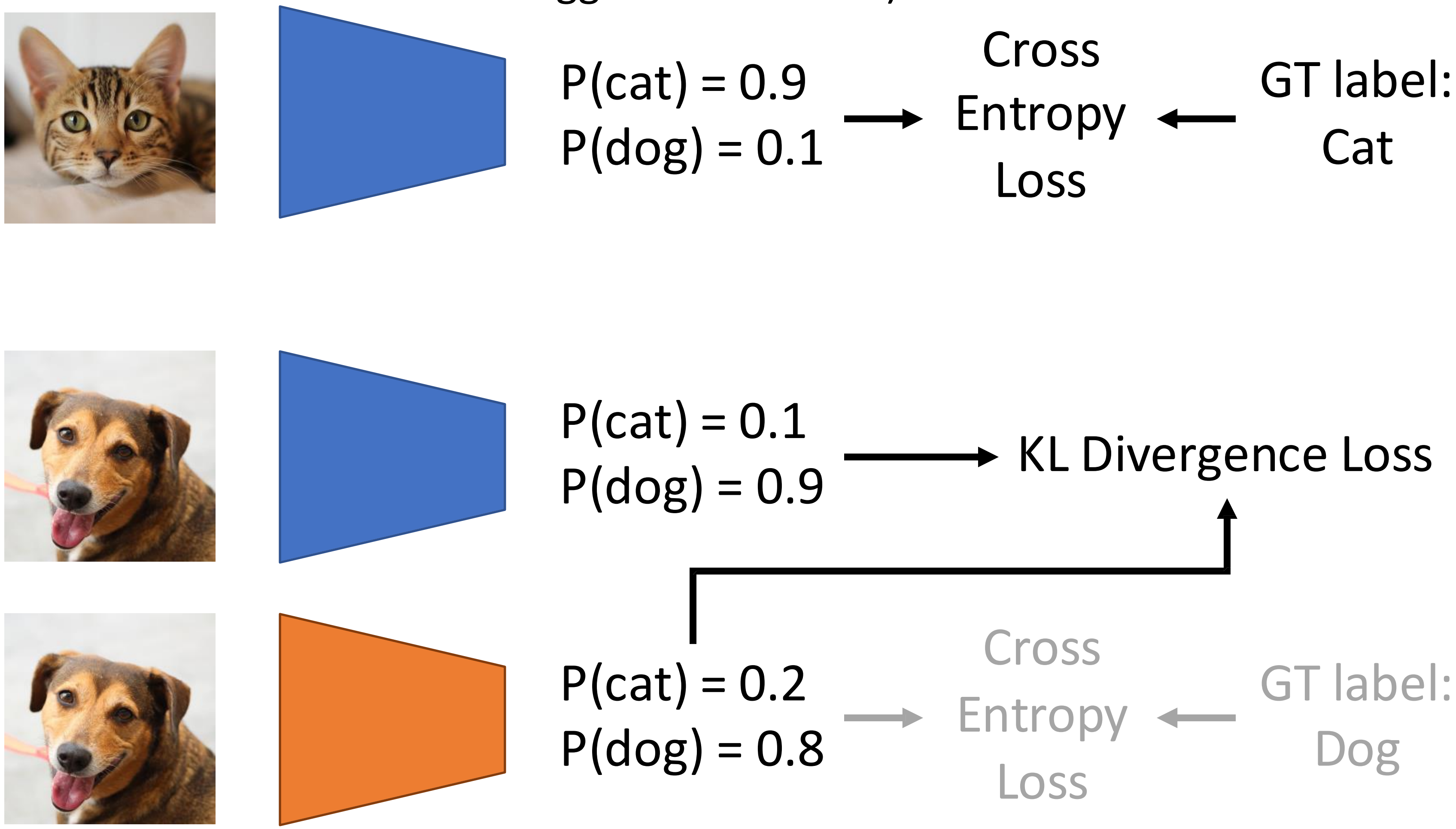
Hinton et al, “Distilling the knowledge in a neural network”, NeurIPS Deep Learning and Representation Learning Workshop, 2015

Improving ViT: Distillation

Often works better than training student from scratch (especially if teacher is bigger than student)

Step 1: Train a **teacher model** on images and ground-truth labels

Step 2: Train a **student model** to match predictions from the **teacher** (sometimes also to match GT labels)



P(cat) = 0.9
P(dog) = 0.1

Cross Entropy Loss

GT label: Cat

P(cat) = 0.1
P(dog) = 0.9

KL Divergence Loss

P(cat) = 0.2
P(dog) = 0.8

Cross Entropy Loss

GT label: Dog

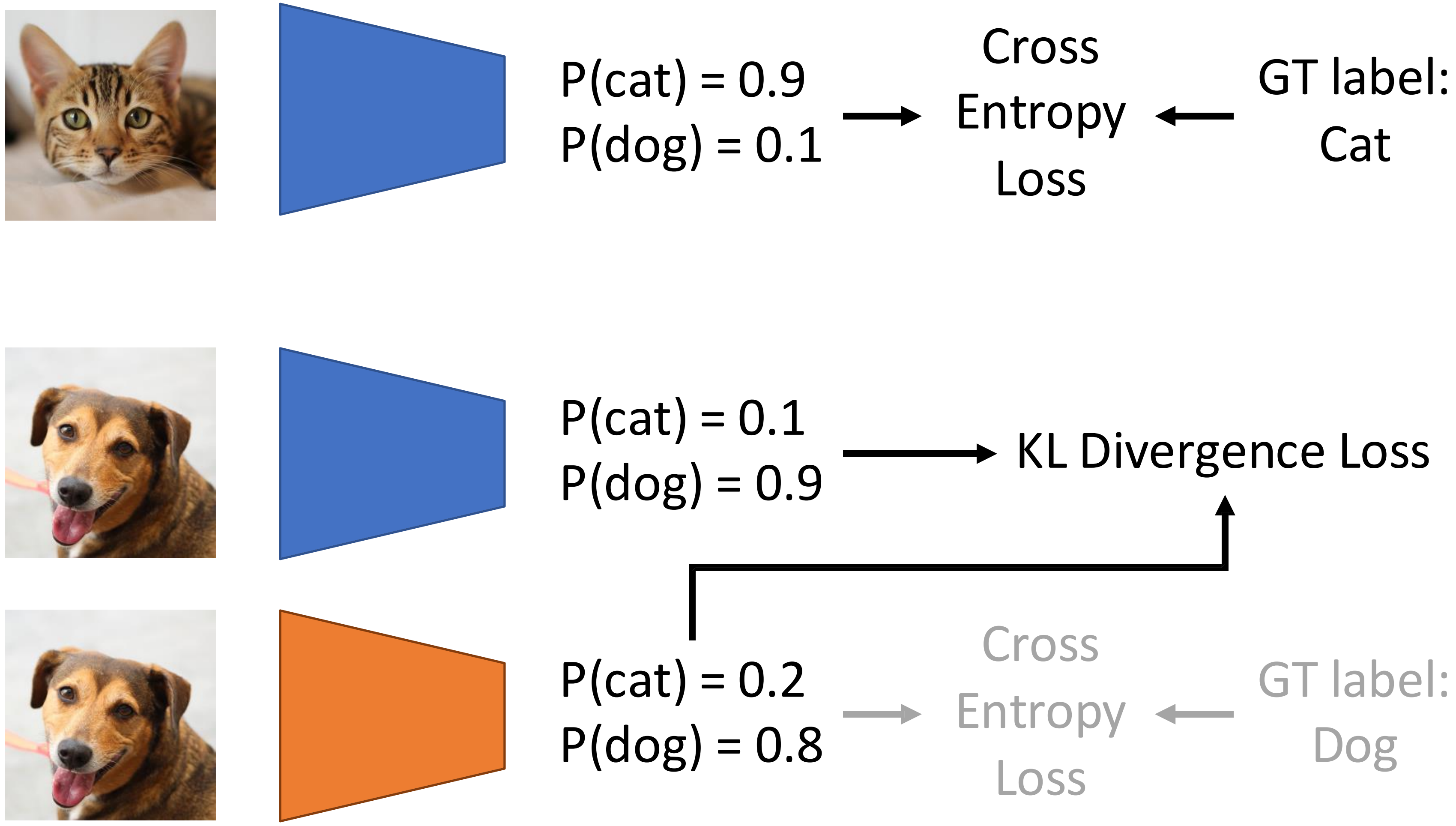
Hinton et al, “Distilling the knowledge in a neural network”, NeurIPS Deep Learning and Representation Learning Workshop, 2015

Improving ViT: Distillation

Can also train student on **unlabeled** data! (Semi-supervised learning)

Step 1: Train a **teacher model** on images and ground-truth labels

Step 2: Train a **student model** to match predictions from the **teacher** (sometimes also to match GT labels)



P(cat) = 0.9
P(dog) = 0.1

Cross Entropy Loss

GT label: Cat

P(cat) = 0.1
P(dog) = 0.9

KL Divergence Loss

P(cat) = 0.2
P(dog) = 0.8

Cross Entropy Loss

GT label: Dog

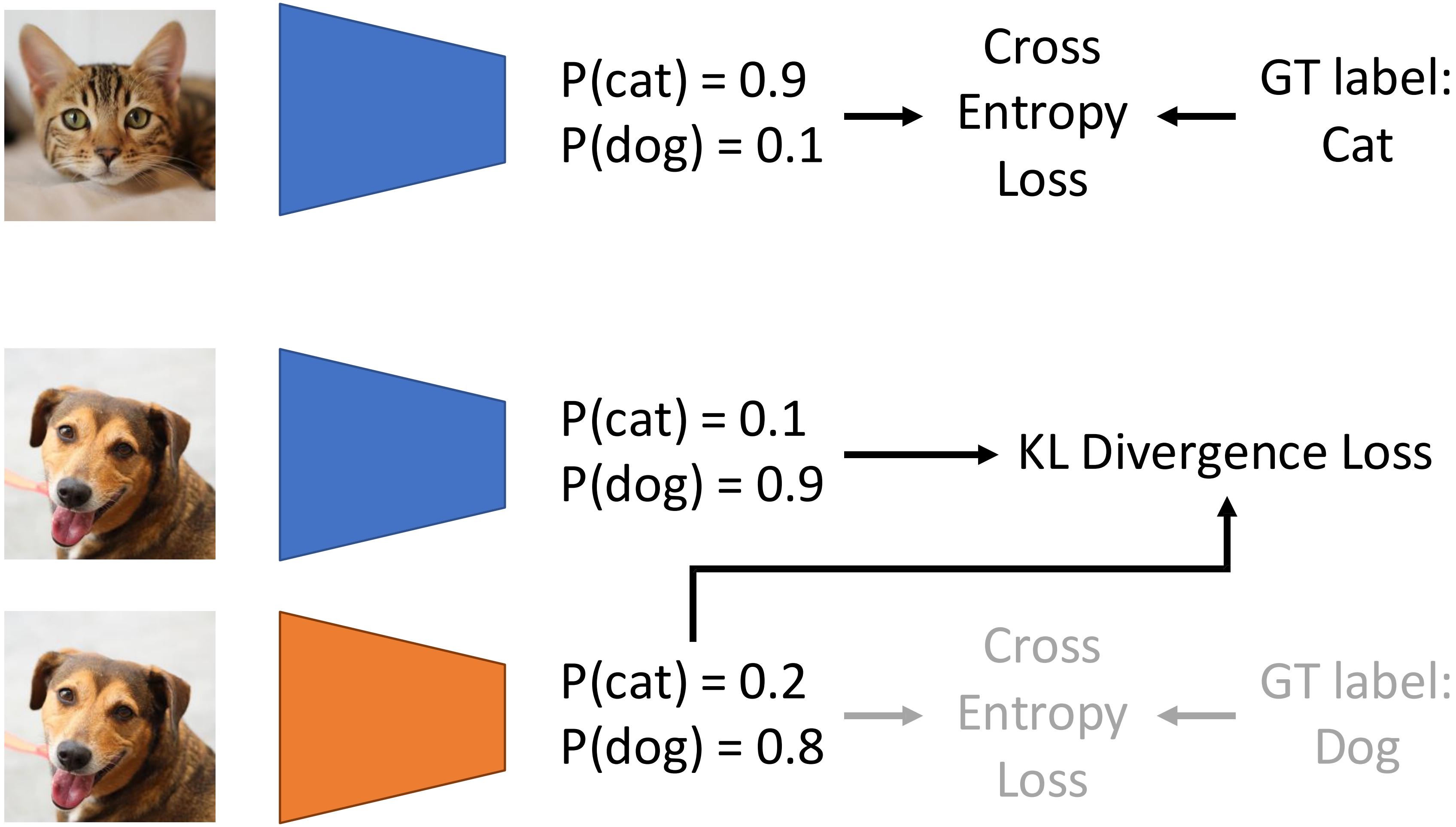
Hinton et al, “Distilling the knowledge in a neural network”, NeurIPS Deep Learning and Representation Learning Workshop, 2015

Improving ViT: Distillation

Step 1: Train a **teacher CNN** on ImageNet

Step 2: Train a **student ViT** to match ImageNet predictions from the **teacher CNN**

(and match GT labels)



P(cat) = 0.9
P(dog) = 0.1

Cross Entropy Loss

GT label: Cat

P(cat) = 0.1
P(dog) = 0.9

KL Divergence Loss

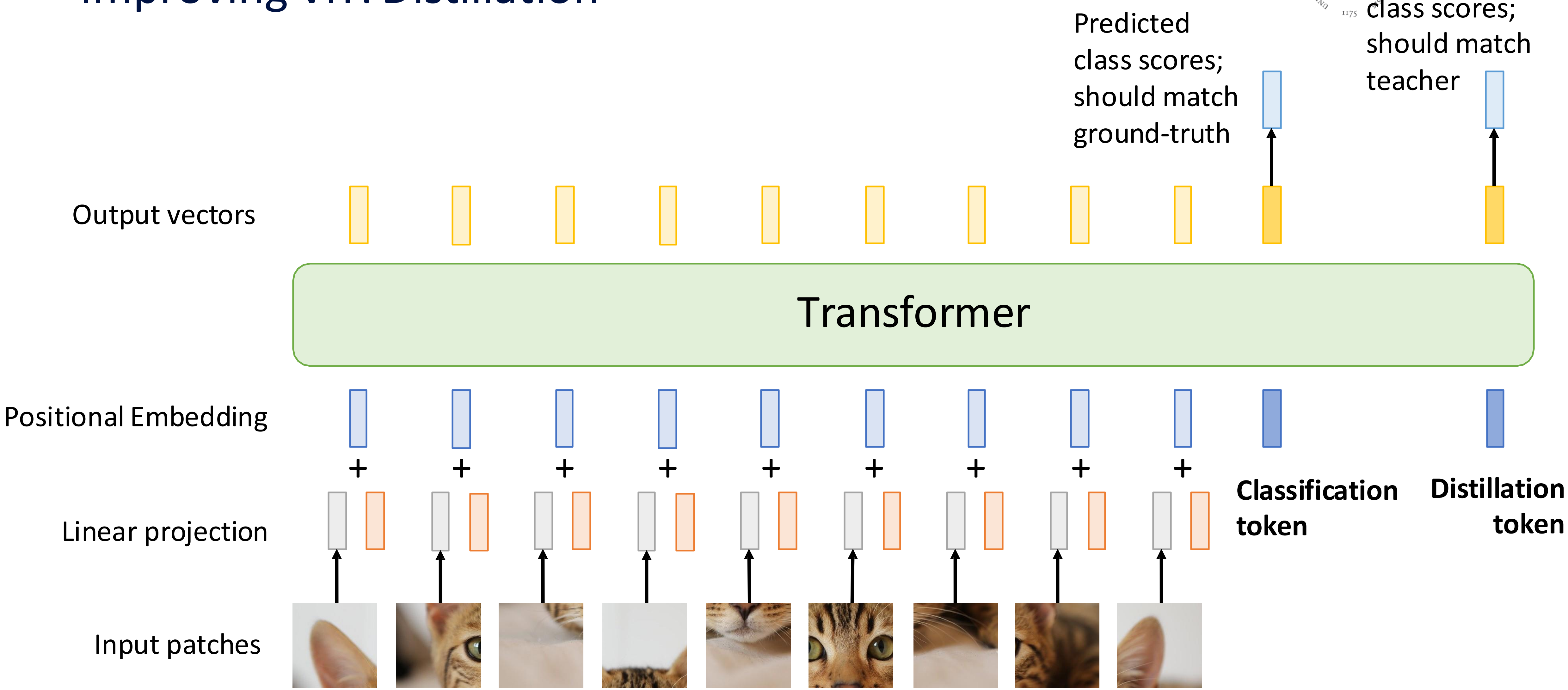
P(cat) = 0.2
P(dog) = 0.8

Cross Entropy Loss

GT label: Dog

Touvrom et al, “Training data-efficient image transformers & distillation through attention”, ICML 2021

Improving ViT: Distillation



Touvrom et al, “Training data-efficient image transformers & distillation through attention”, ICML 2021

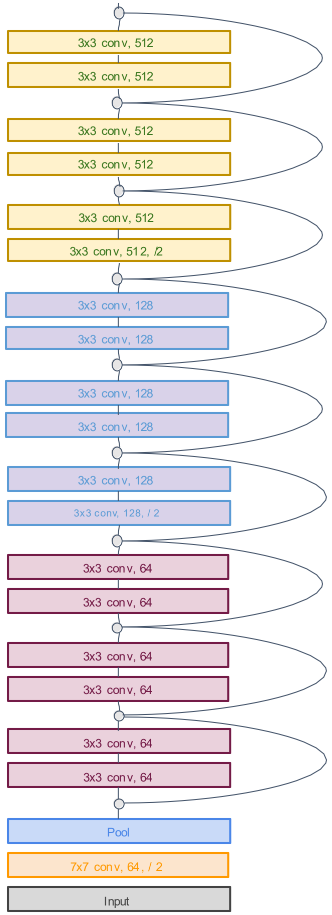
ViT vs CNN

Stage 3:
256 x 14 x 14

Stage 2:
128 x 28 x 28

Stage 1:
64 x 56 x 56

Input:
3 x 224 x 224



In most CNNs (including ResNets), **decrease** resolution and **increase** channels as you go deeper in the network (Hierarchical architecture)

Useful since objects in images can occur at various scales

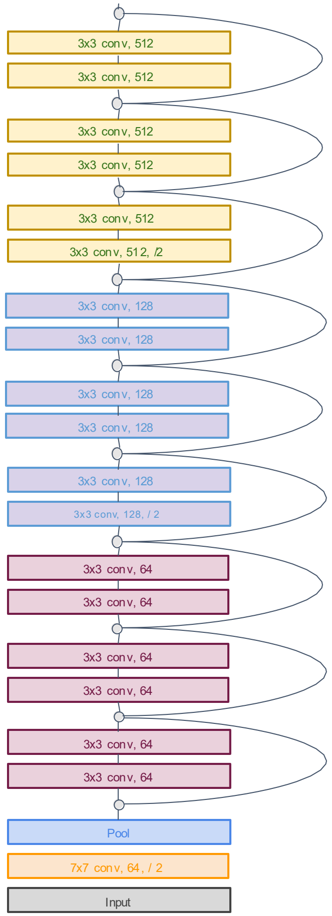
ViT vs CNN

Stage 3:
256 x 14 x 14

Stage 2:
128 x 28 x 28

Stage 1:
64 x 56 x 56

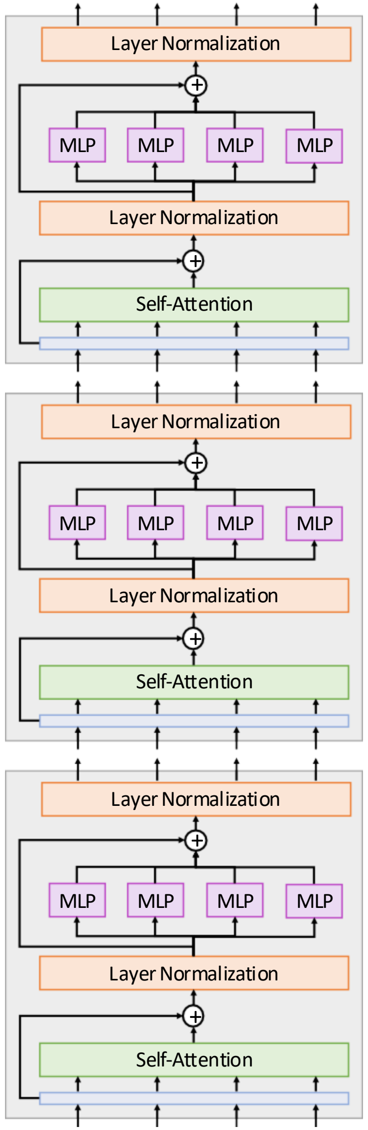
Input:
3 x 224 x 224



In most CNNs (including ResNets), **decrease** resolution and **increase** channels as you go deeper in the network (Hierarchical architecture)

Useful since objects in images can occur at various scales

In a ViT, all blocks have same resolution and number of channels (Isotropic architecture)



3rd block:
768 x 14 x 14

2nd block:
768 x 14 x 14

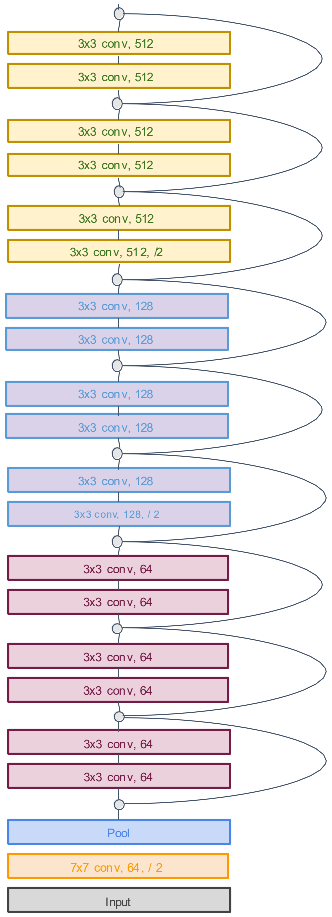
1st block:
768 x 14 x 14

Input:
3 x 224 x 224

ViT vs CNN

In most CNNs (including ResNets), **decrease** resolution and **increase** channels as you go deeper in the network (Hierarchical architecture)

Useful since objects in images can occur at various scales



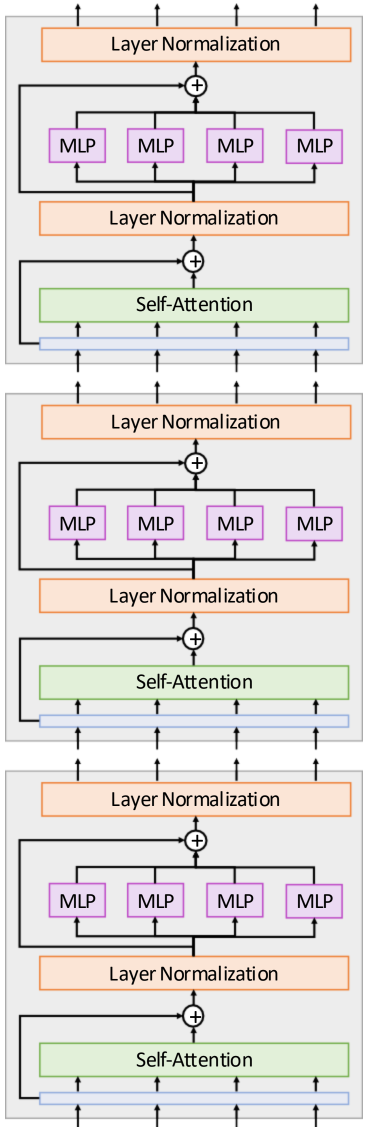
Stage 3:
256 x 14 x 14

Stage 2:
128 x 28 x 28

Stage 1:
64 x 56 x 56

Input:
3 x 224 x 224

In a ViT, all blocks have same resolution and number of channels (Isotropic architecture)



3rd block:
768 x 14 x 14

2nd block:
768 x 14 x 14

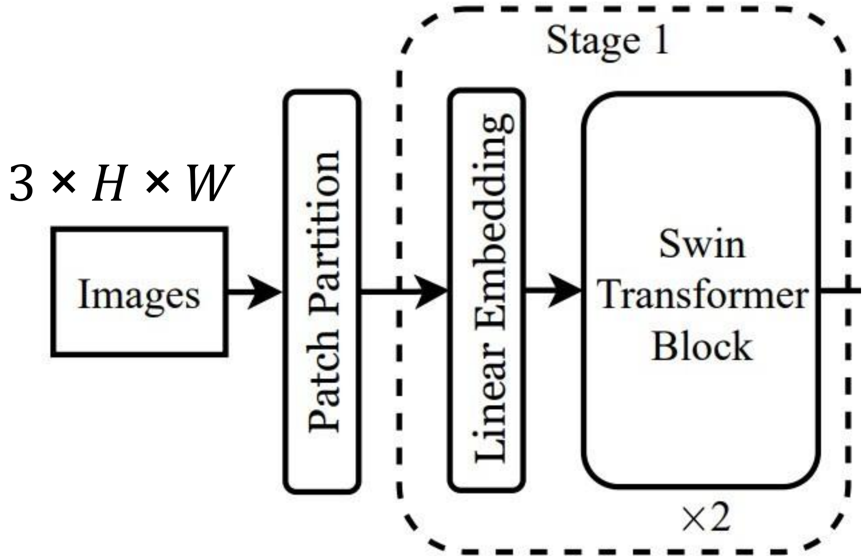
1st block:
768 x 14 x 14

Input:
3 x 224 x 224

Can we build a **hierarchical** ViT model?

Hierarchical ViT: Swin Transformer

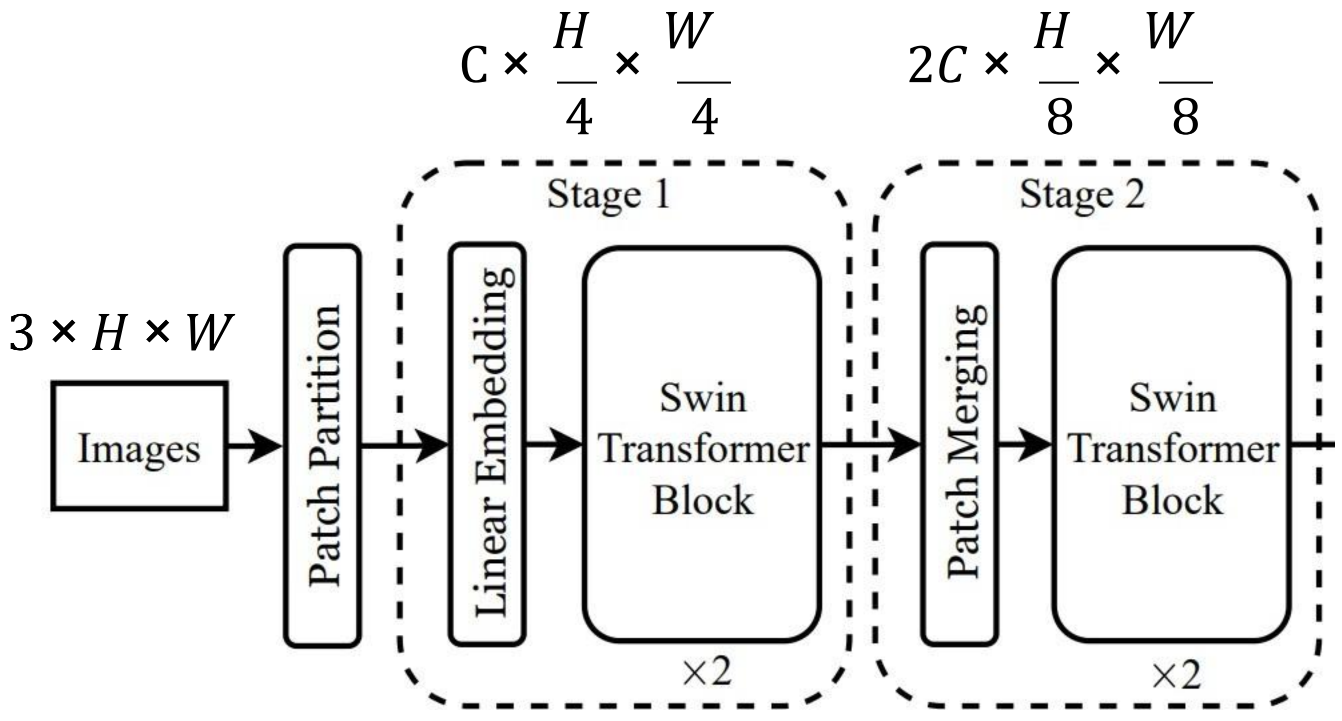
$\mathrm{C} \times \frac{H}{4} \times \frac{W}{4}$



Divide image into 4x4
patches and project
to C dimensions

Liu et al, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”, CVPR 2021

Hierarchical ViT: Swin Transformer

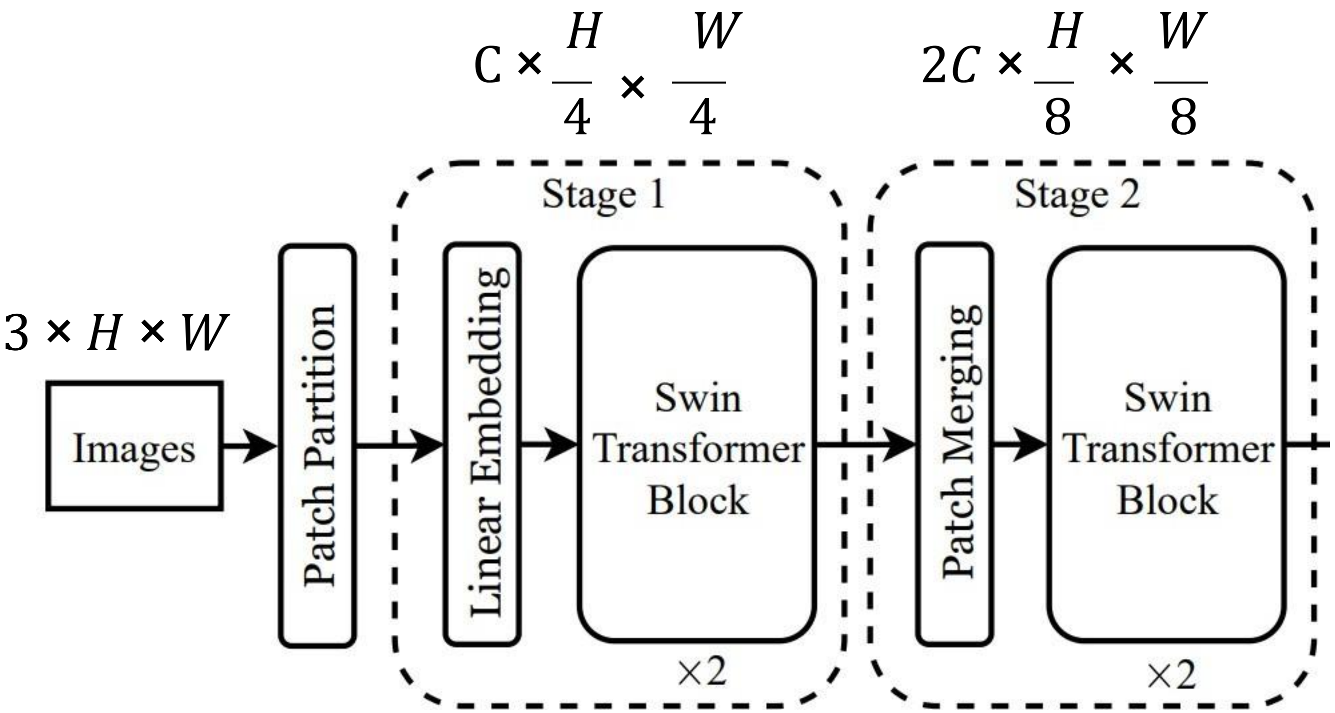


Divide image into 4x4 patches and project to C dimensions

Merge 2x2 neighborhoods; now patches are (effectively) 8x8

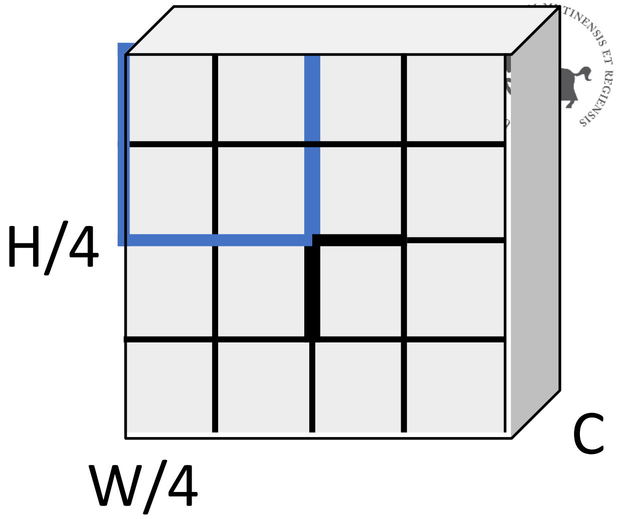
Liu et al, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”, CVPR 2021

Hierarchical ViT: Swin Transformer



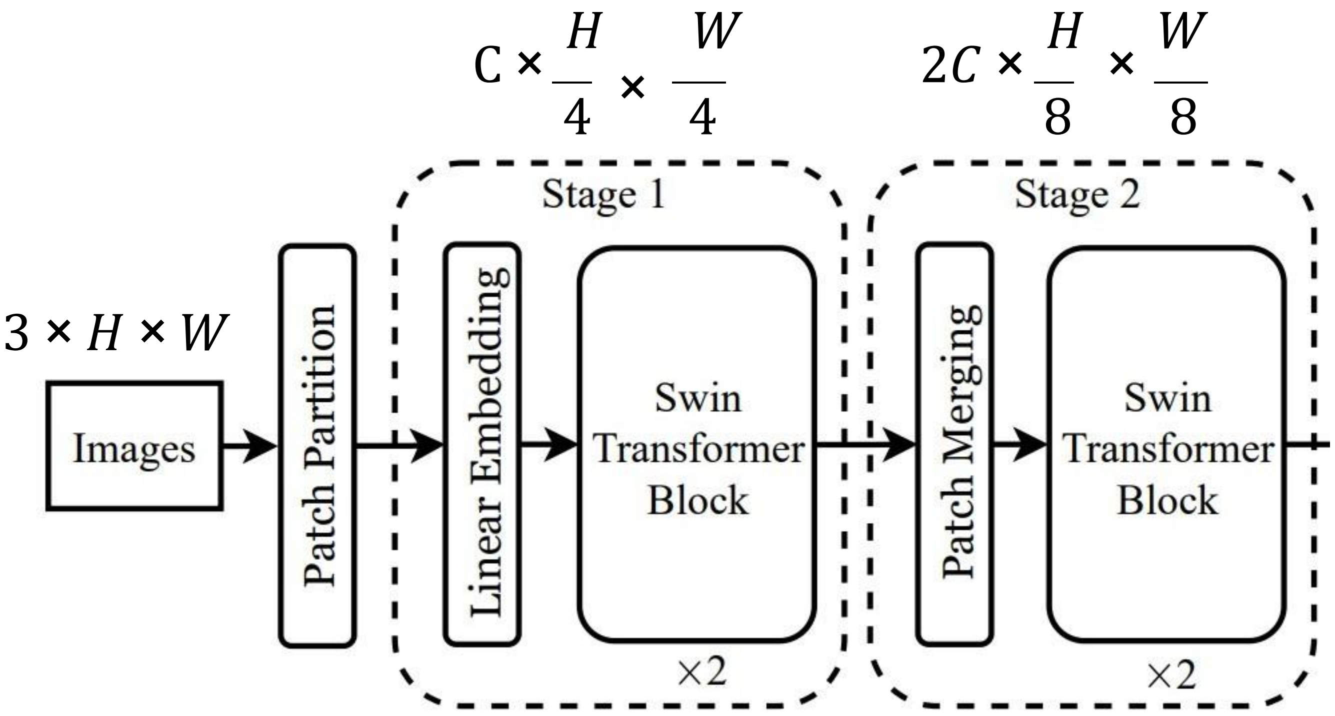
Divide image into 4x4 patches and project to C dimensions

Merge 2x2 neighborhoods; now patches are (effectively) 8x8



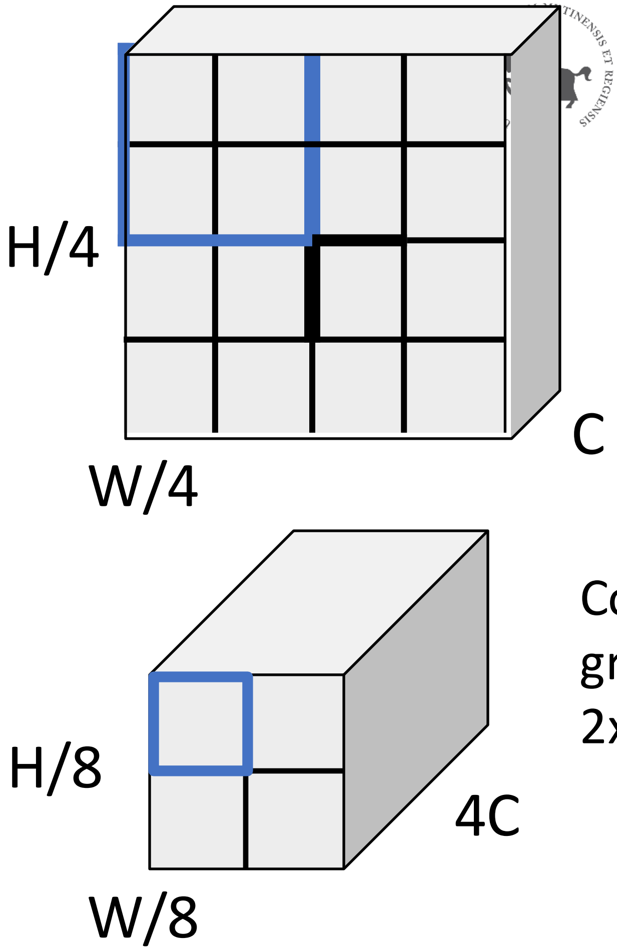
Liu et al, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”, CVPR 2021

Hierarchical ViT: Swin Transformer



Divide image into 4x4 patches and project to C dimensions

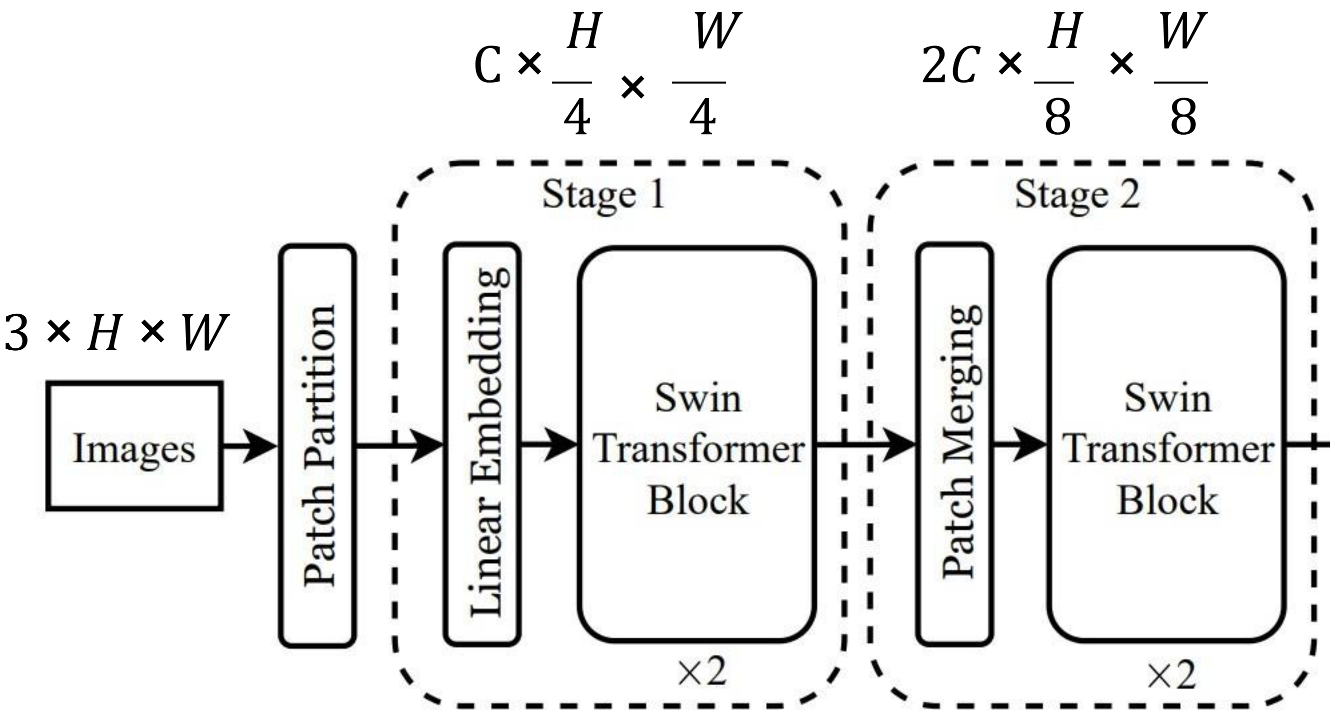
Merge 2x2 neighborhoods; now patches are (effectively) 8x8



Concatenate groups of 2x2 features

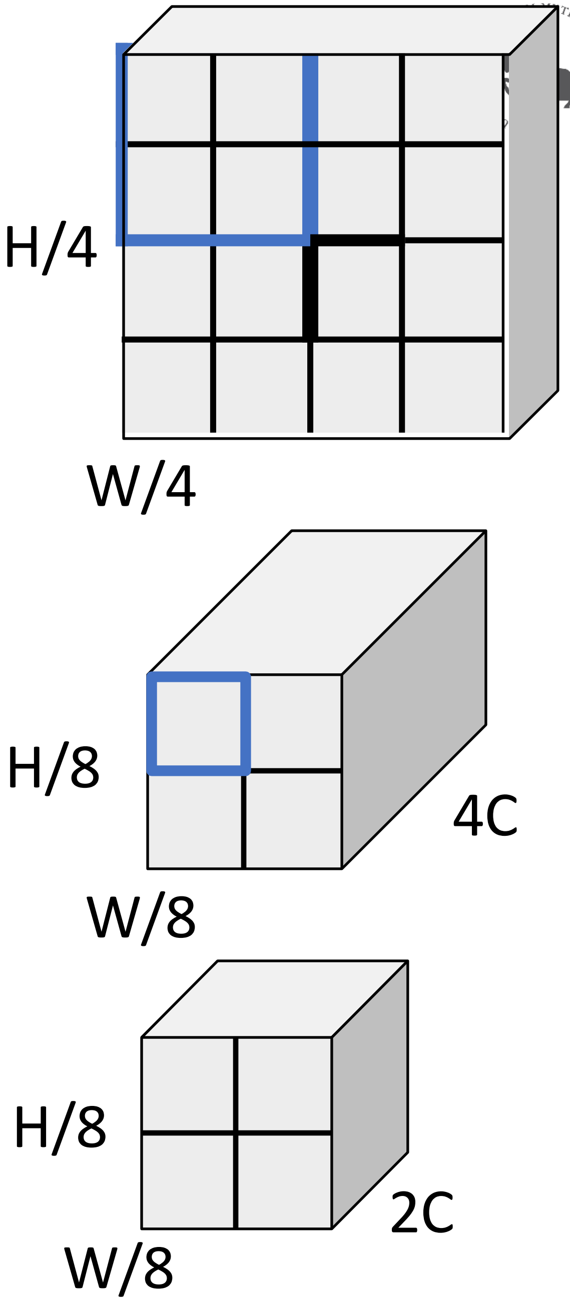
Liu et al, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”, CVPR 2021

Hierarchical ViT: Swin Transformer



Divide image into 4x4 patches and project to C dimensions

Merge 2x2 neighborhoods; now patches are (effectively) 8x8

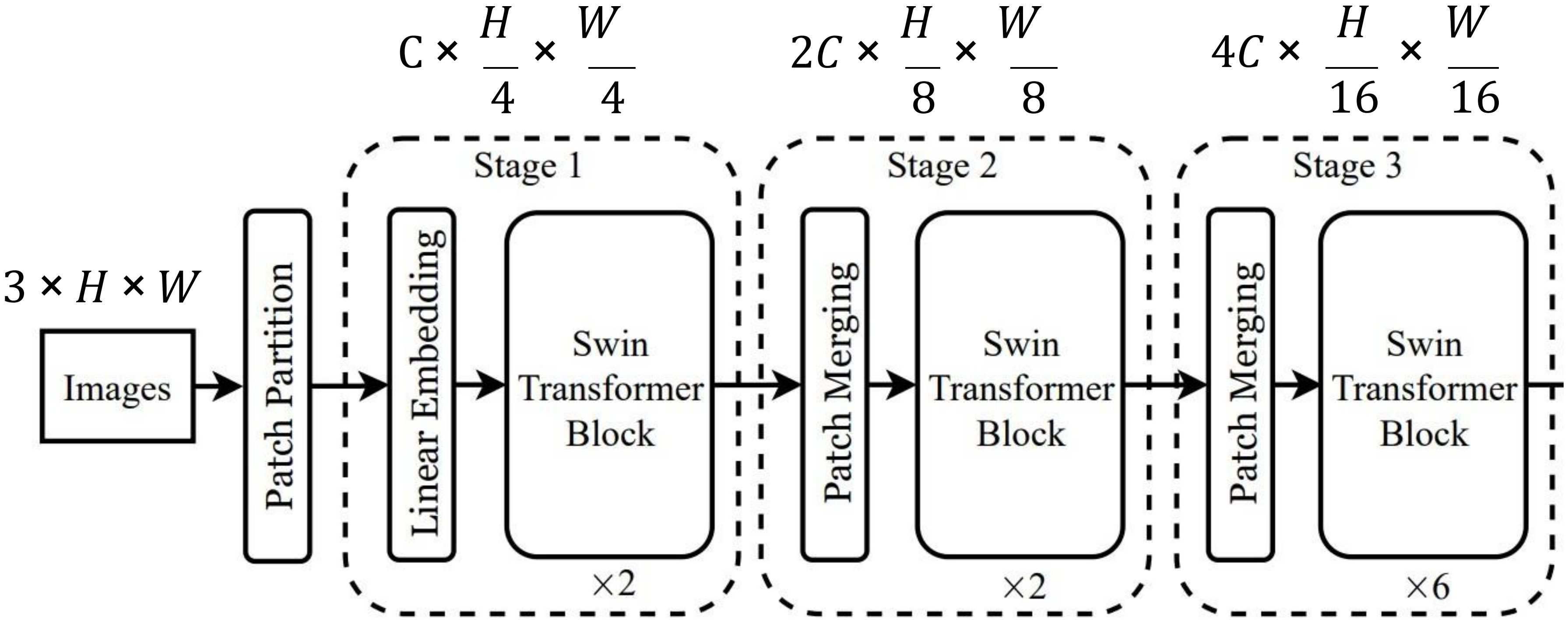


Concatenate groups of 2x2 features

Linear projection from 4C to 2C channels (1x1 conv)

Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Hierarchical ViT: Swin Transformer



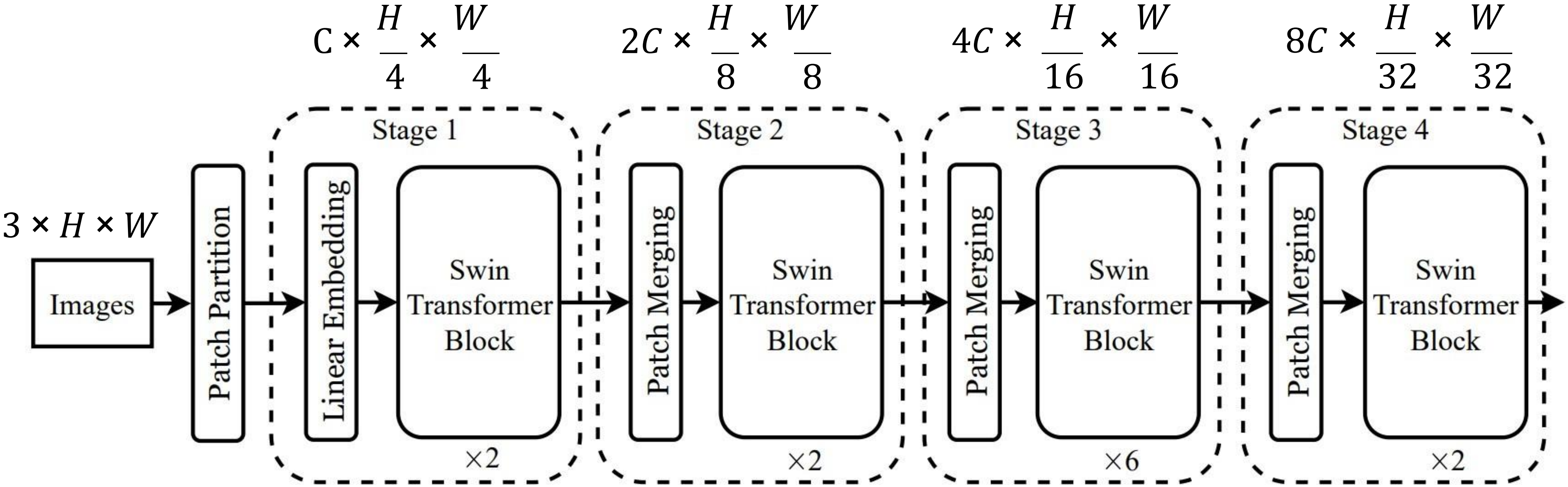
Divide image into 4x4 patches and project to C dimensions

Merge 2x2 neighborhoods; now patches are (effectively) 8x8

Merge 2x2 neighborhoods; now patches are (effectively) 16x16

Liu et al, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”, CVPR 2021

Hierarchical ViT: Swin Transformer



Divide image into 4x4 patches and project to C dimensions

Merge 2x2 neighborhoods; now patches are (effectively) 8x8

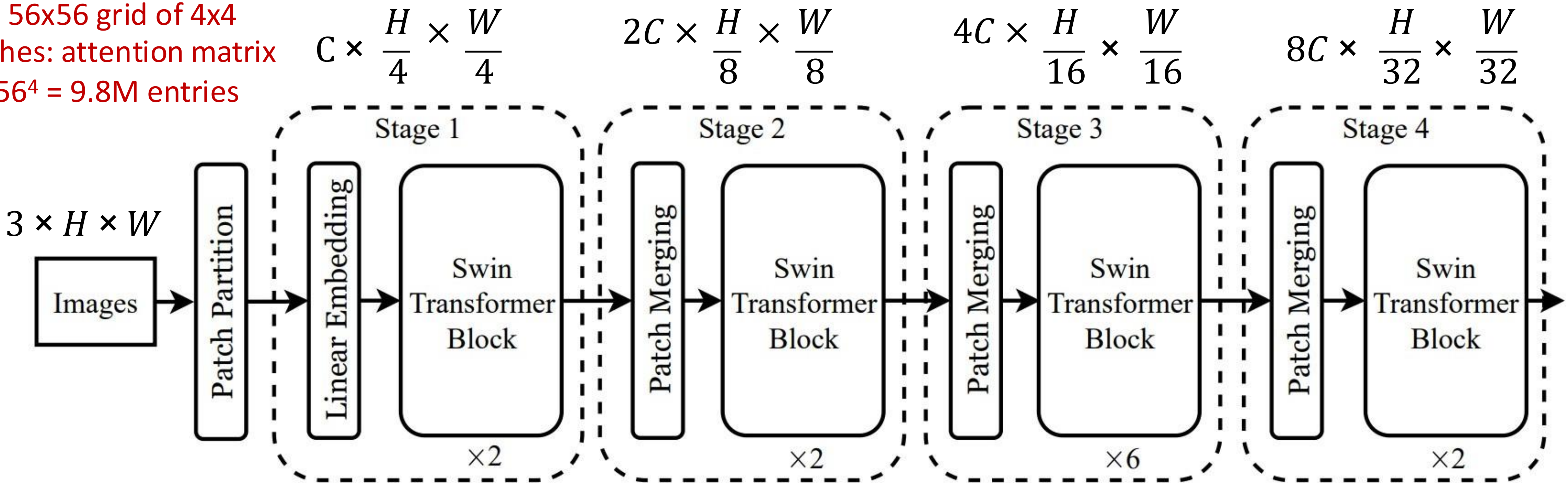
Merge 2x2 neighborhoods; now patches are (effectively) 16x16

Merge 2x2 neighborhoods; now patches are (effectively) 32x32

Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Hierarchical ViT: Swin Transformer

Problem: 224x224 image with 56x56 grid of 4x4 patches: attention matrix has 56^4 = 9.8M entries



Divide image into 4x4 patches and project to C dimensions

Merge 2x2 neighborhoods; now patches are (effectively) 8x8

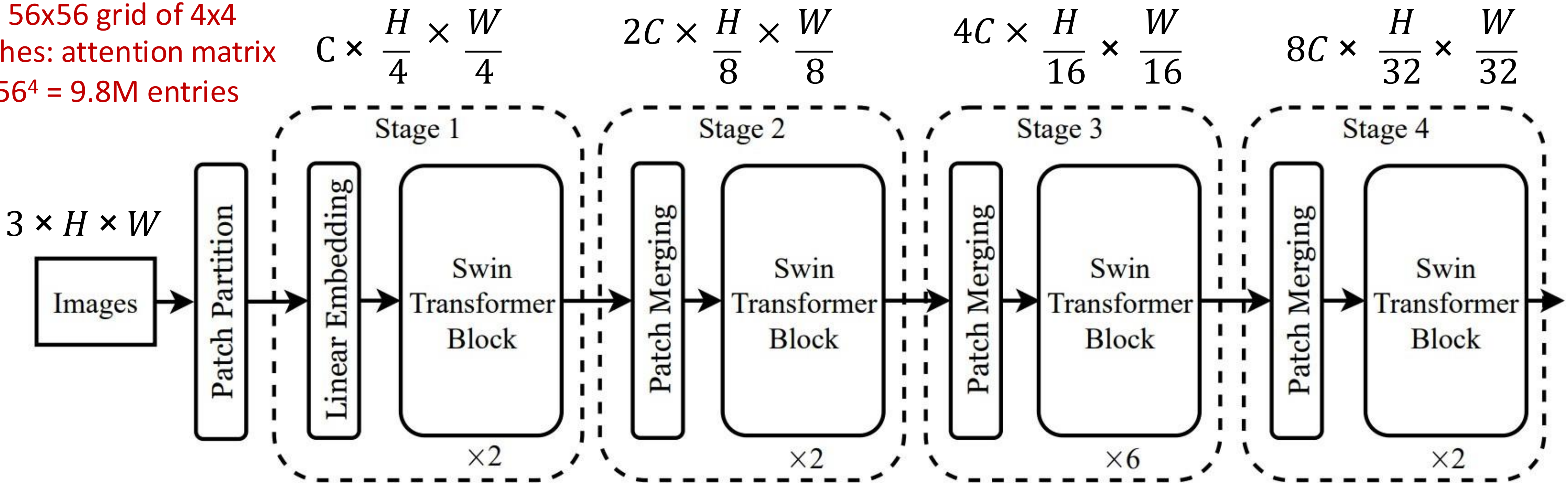
Merge 2x2 neighborhoods; now patches are (effectively) 16x16

Merge 2x2 neighborhoods; now patches are (effectively) 32x32

Liu et al, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”, CVPR 2021

Hierarchical ViT: Swin Transformer

Problem: 224x224 image with 56x56 grid of 4x4 patches: attention matrix has 56^4 = 9.8M entries



Divide image into 4x4 patches and project to C dimensions

Merge 2x2 neighborhoods; now patches are (effectively) 8x8

Merge 2x2 neighborhoods; now patches are (effectively) 16x16

Merge 2x2 neighborhoods; now patches are (effectively) 32x32

Solution: don't use full attention, instead use attention over patches

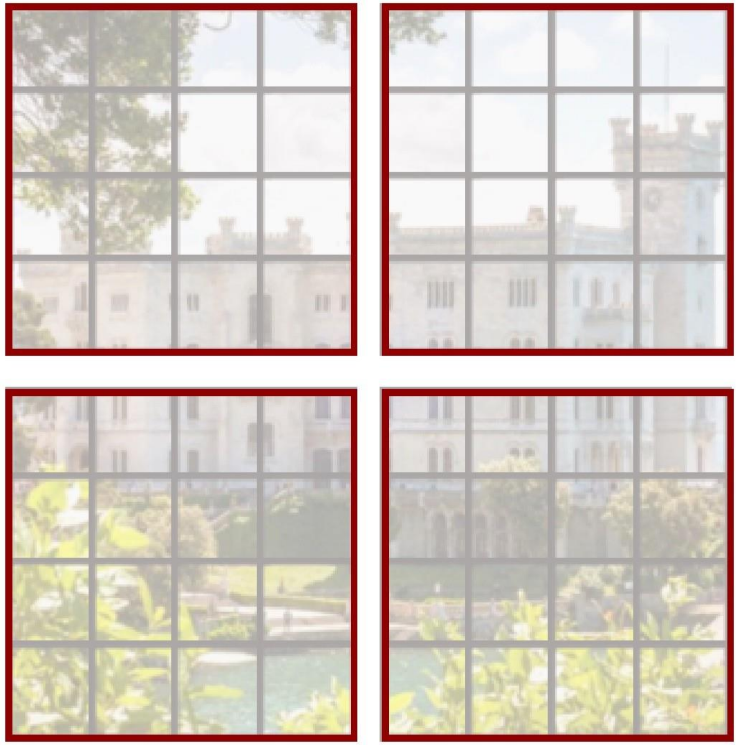
Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Swin Transformer: Window Attention

With H x W grid of **tokens**, each attention matrix is H^2W^2 – **quadratic** in image size

Liu et al, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”, CVPR 2021

Swin Transformer: Window Attention

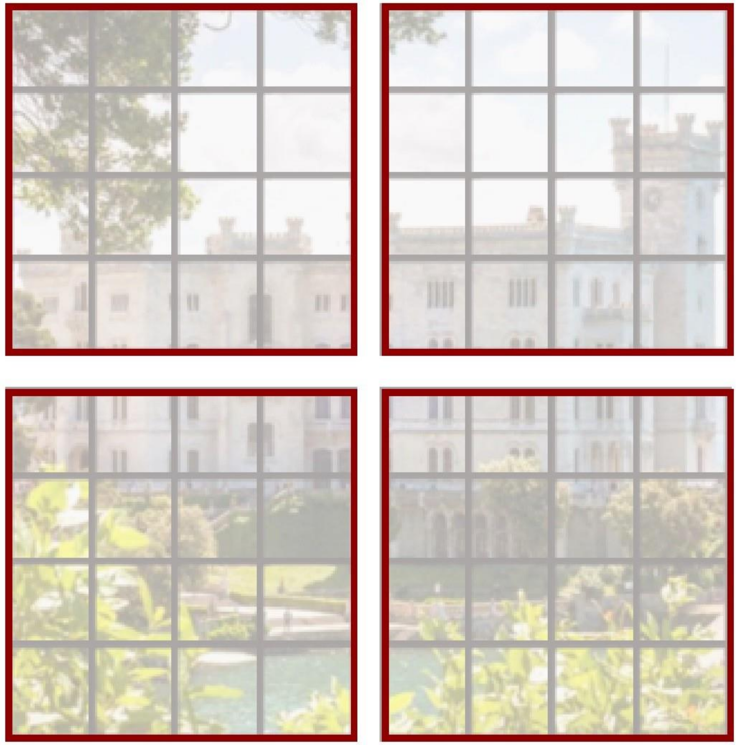


With H x W grid of tokens, each attention matrix is H^2W^2 – **quadratic** in image size

Rather than allowing each token to attend to all other tokens, instead divide into **windows** of M x M tokens (here M=4); only compute attention within each window

Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Swin Transformer: Window Attention



With H x W grid of **tokens**, each attention matrix is H^2W^2 – **quadratic** in image size

Rather than allowing each **token** to attend to all other tokens, instead divide into **windows** of M x M tokens (here M=4); only compute attention within each window

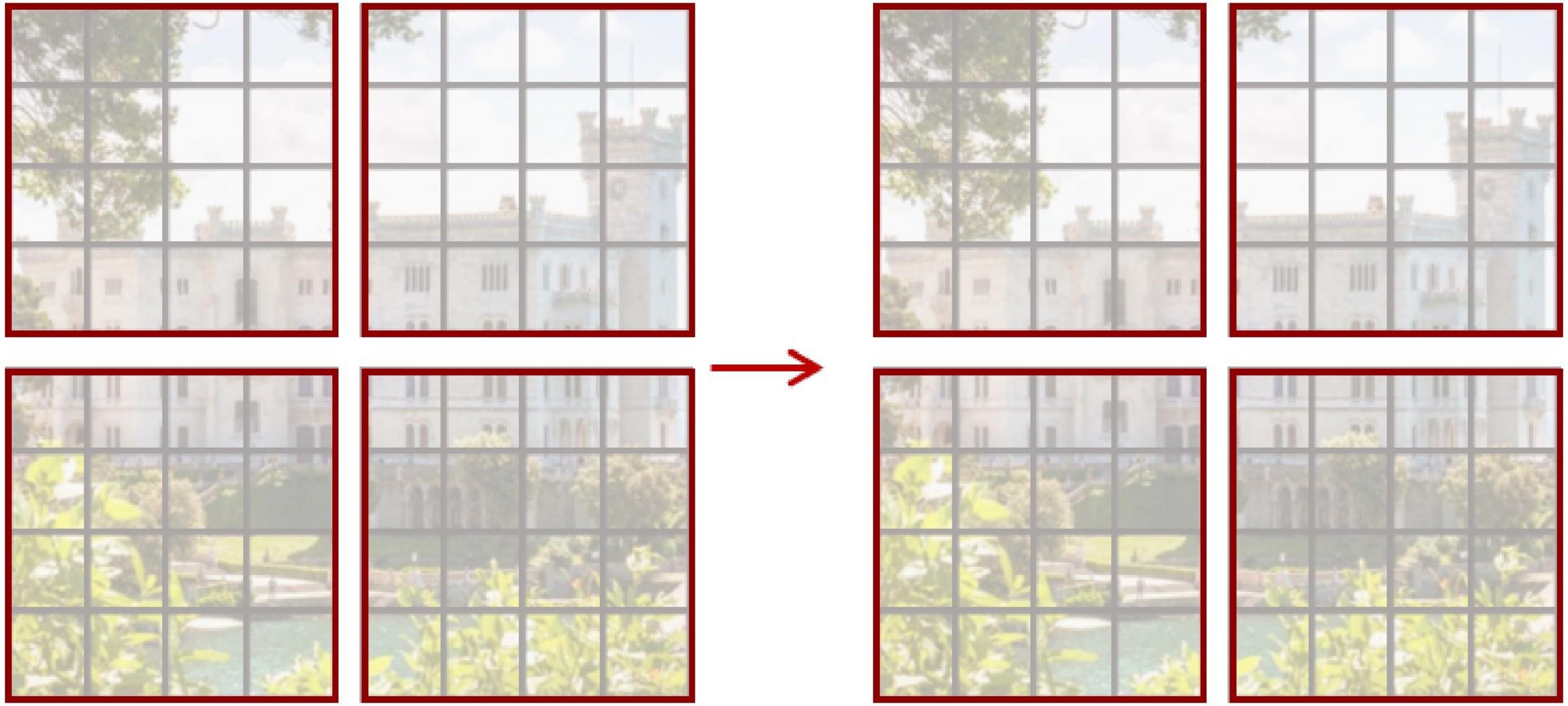
Total size of all attention matrices is now: $M^4(H/M)(W/M) = M^2HW$

Linear in image size for fixed M!
Swin uses M=7 throughout the network

Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Swin Transformer: Window Attention

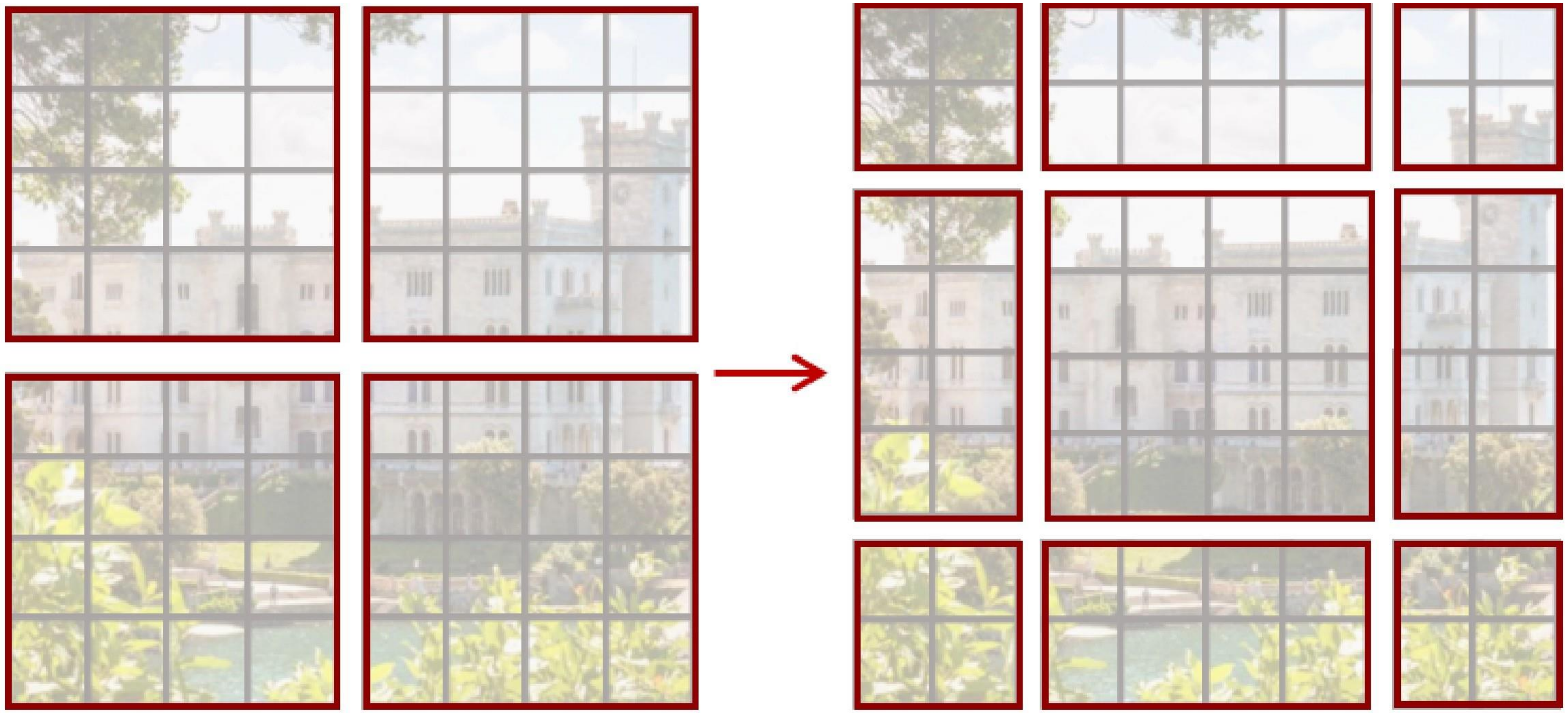
Problem: tokens only interact with other tokens within the same window; no communication across windows



Liu et al, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”, CVPR 2021

Swin Transformer: Shifted Window Attention

Solution: Alternate between normal windows and shifted windows in successive Transformer blocks



Ugly detail: Non-square windows at edges and corners

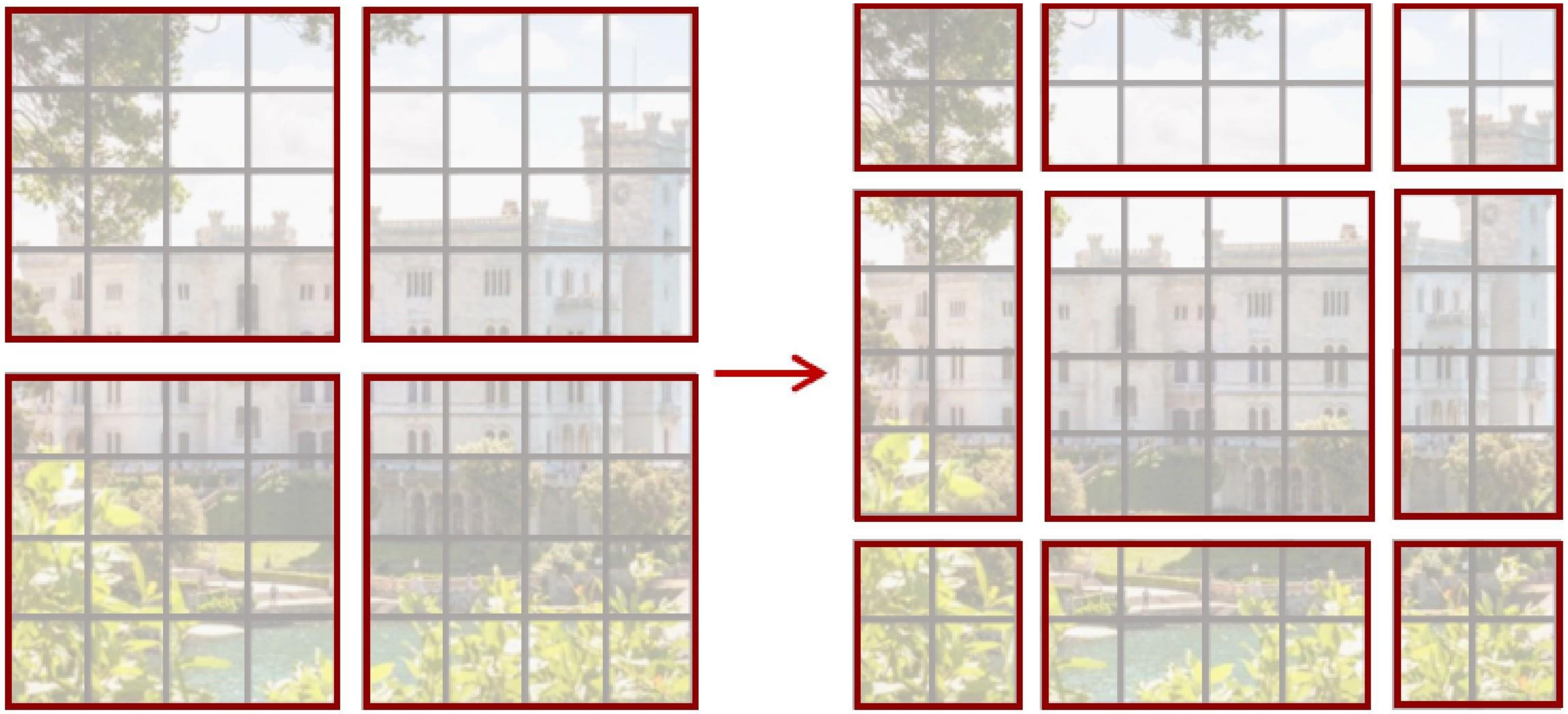
Block L: Normal windows

Block L+1: Shifted Windows

Liu et al, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”, CVPR 2021

Swin Transformer: Shifted Window Attention

Solution: Alternate between normal windows and shifted windows in successive Transformer blocks



Block L: Normal windows

Block L+1: Shifted Windows

Detail: Relative Positional Bias

ViT adds positional embedding to input tokens, encodes *absolute position* of each token in the image

Swin does not use positional embeddings, instead encodes *relative position* between patches when computing attention:

Attention with relative bias:

$$A = Softmax\left(\frac{QK^T}{\sqrt{D}} + B\right)V$$

$Q, K, V: M^2 \times D$ (Query, Key, Value)

$B: M^2 \times M^2$ (learned biases)

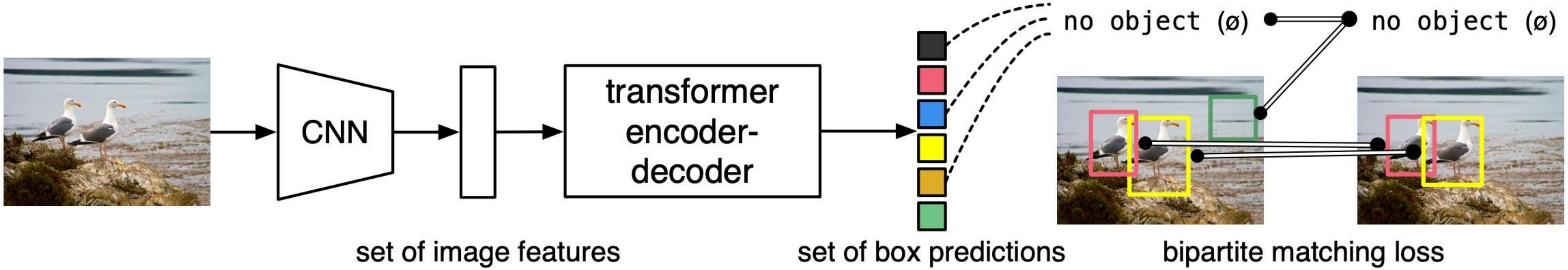
Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Object Detection with Transformers: DETR

Simple object detection pipeline: directly output a set of boxes from a Transformer

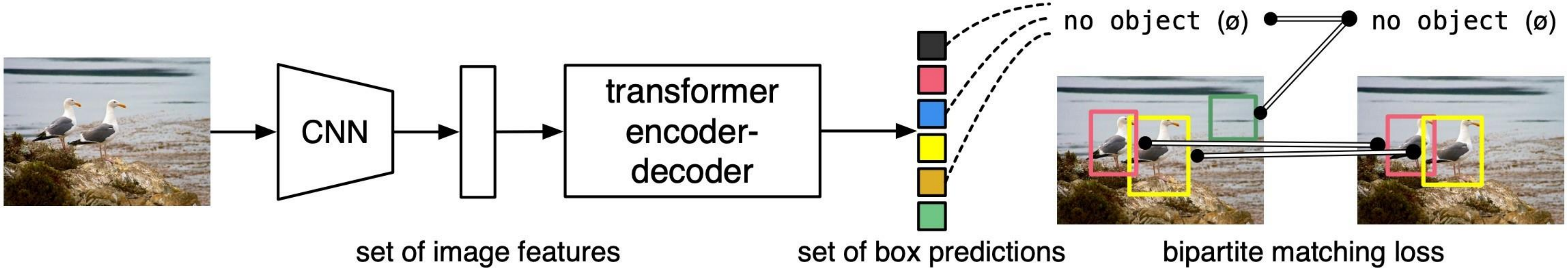
No anchors, no regression of box transforms

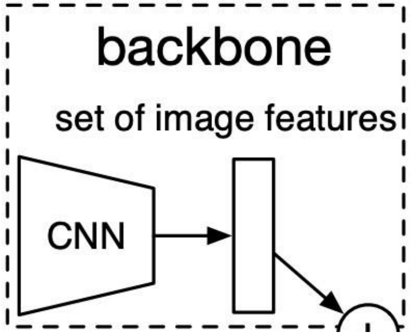
Match predicted boxes to GT boxes with bipartite matching; train to regress box coordinates



Carion et al, “End-to-End Object Detection with Transformers”, ECCV 2020

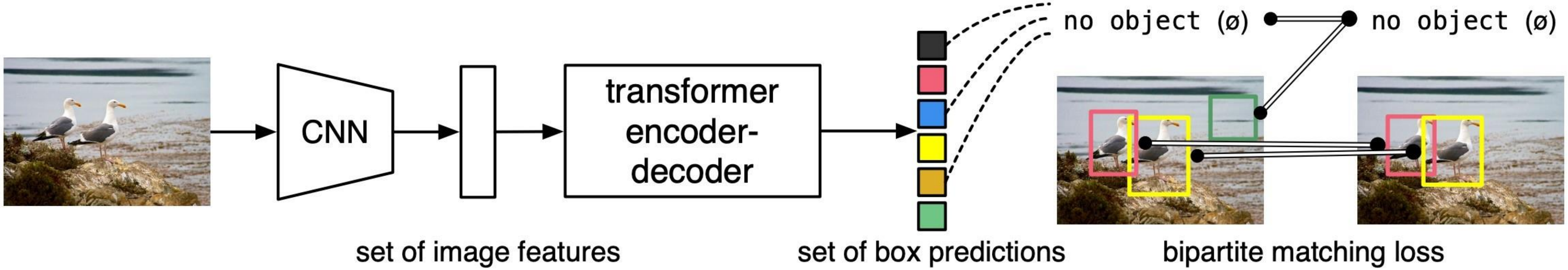
Object Detection with Transformers: DETR

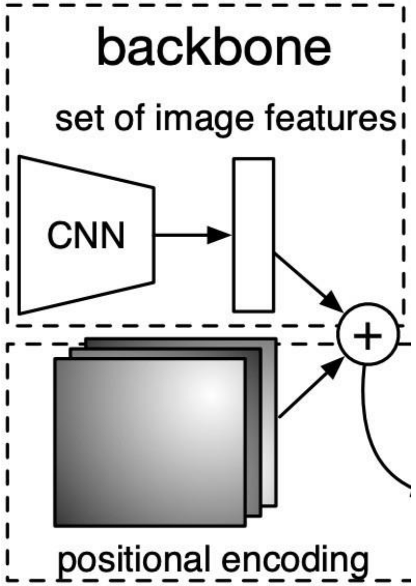




Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

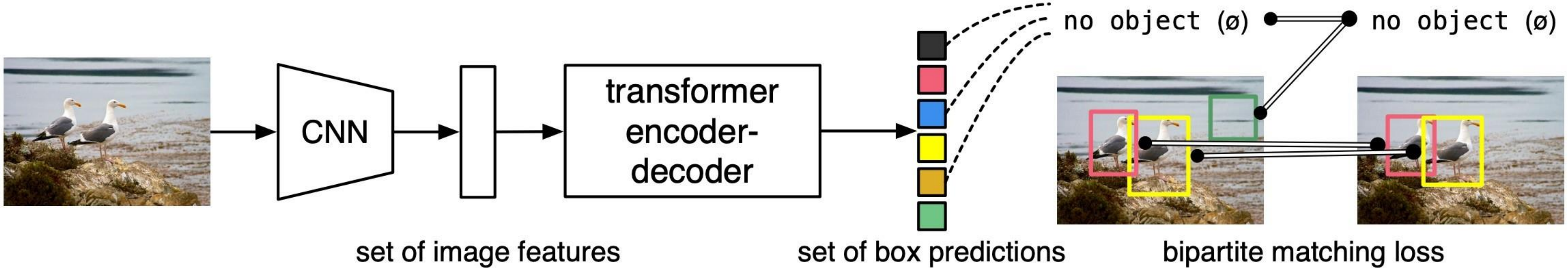
Object Detection with Transformers: DETR

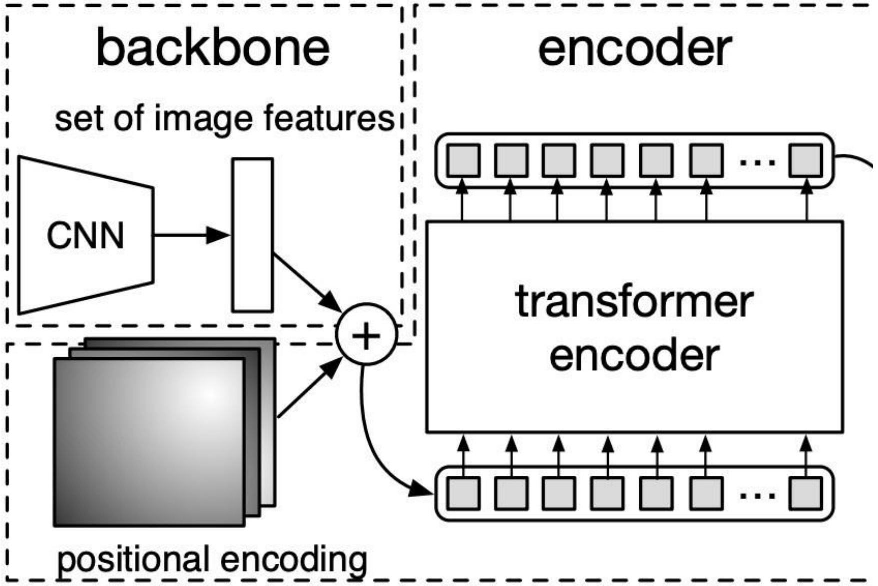




Carion et al, “End-to-End Object Detection with Transformers”, ECCV 2020

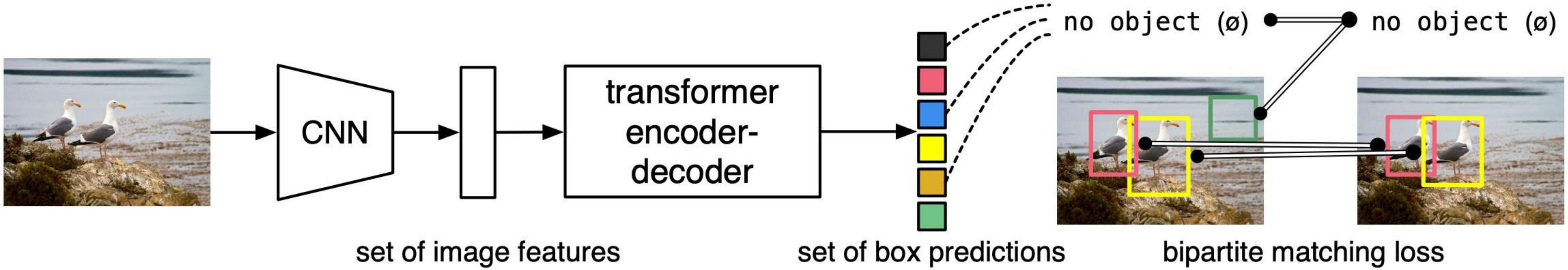
Object Detection with Transformers: DETR

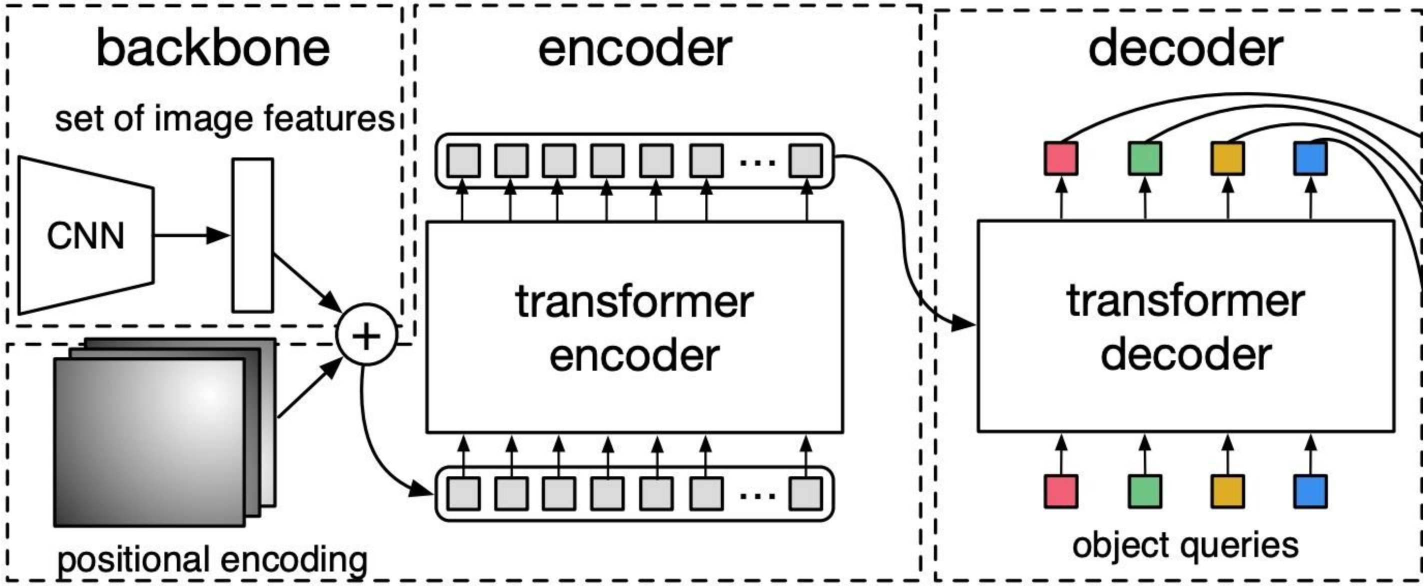




Carion et al, “End-to-End Object Detection with Transformers”, ECCV 2020

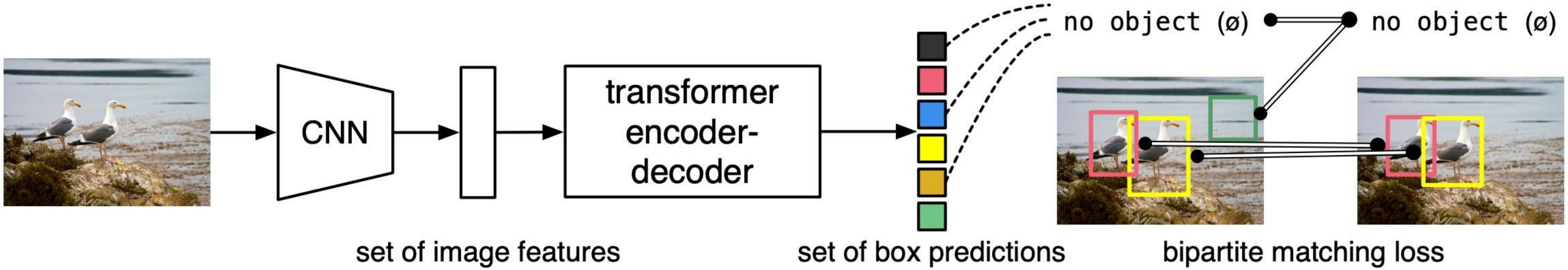
Object Detection with Transformers: DETR

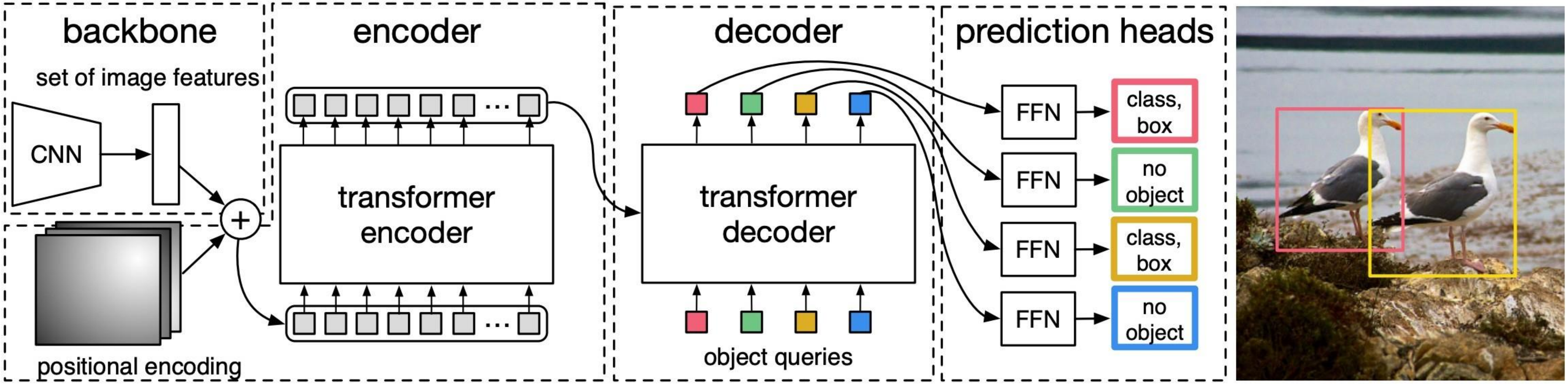




Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

Object Detection with Transformers: DETR

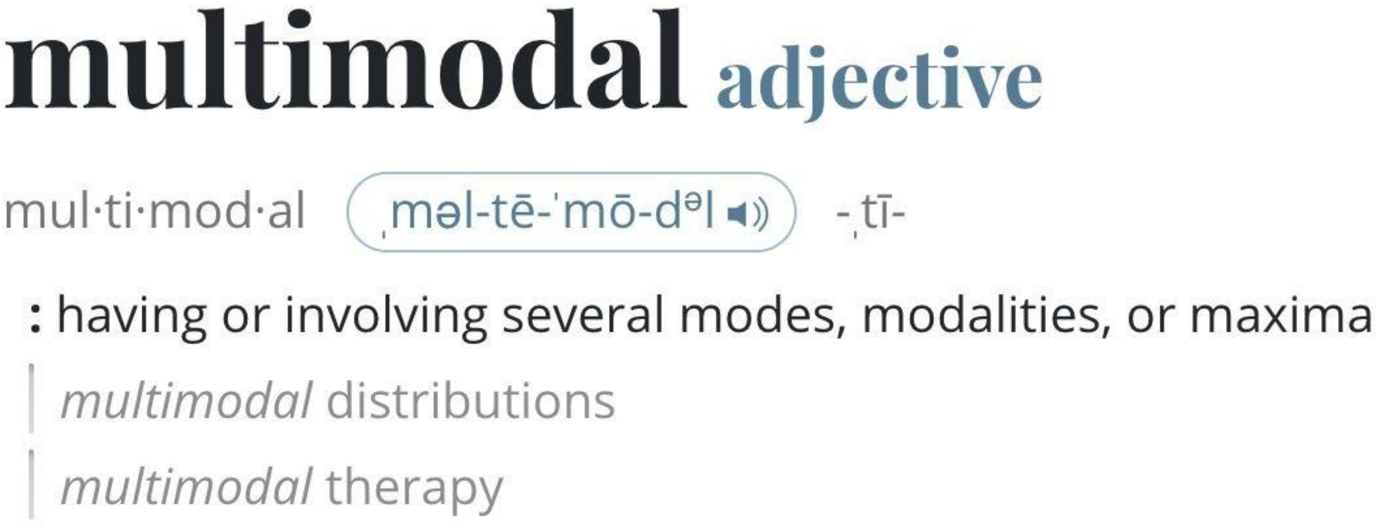




Carion et al, “End-to-End Object Detection with Transformers”, ECCV 2020

MULTIMODAL MODELS

What is multimodality?



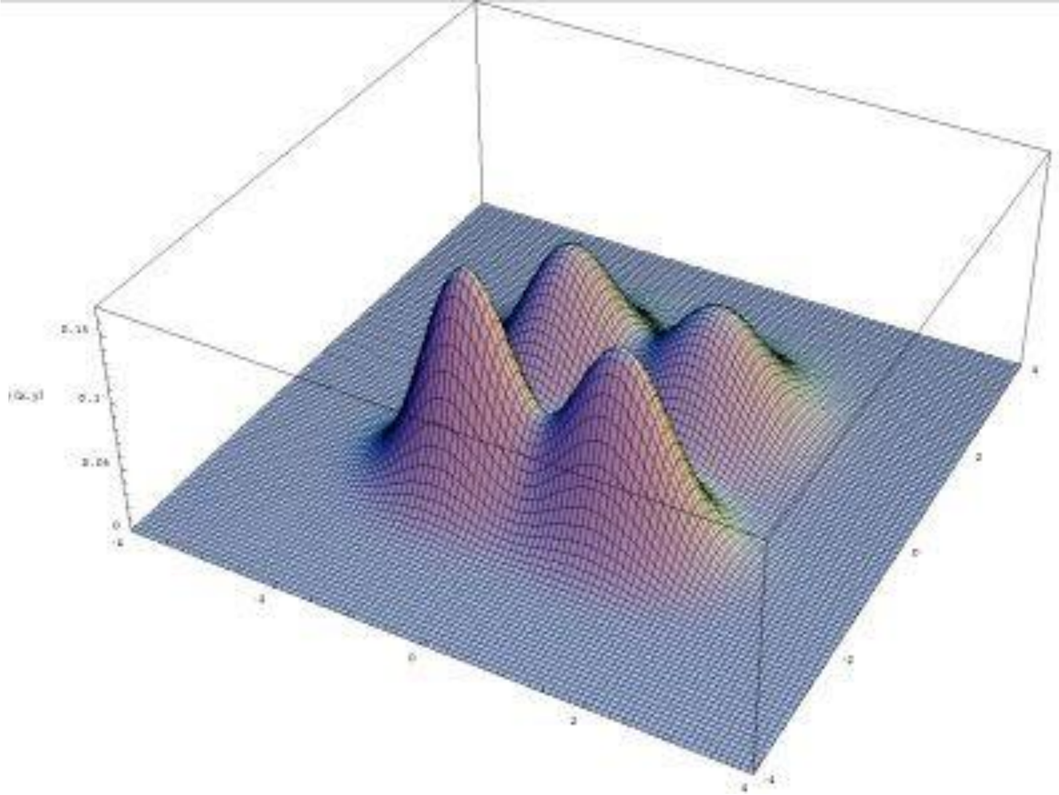
multimodal adjective

mul·ti·mod·al ˌməl-tē-ˈmō-dᵊl -ˌtī-

: having or involving several modes, modalities, or maxima

multimodal distributions

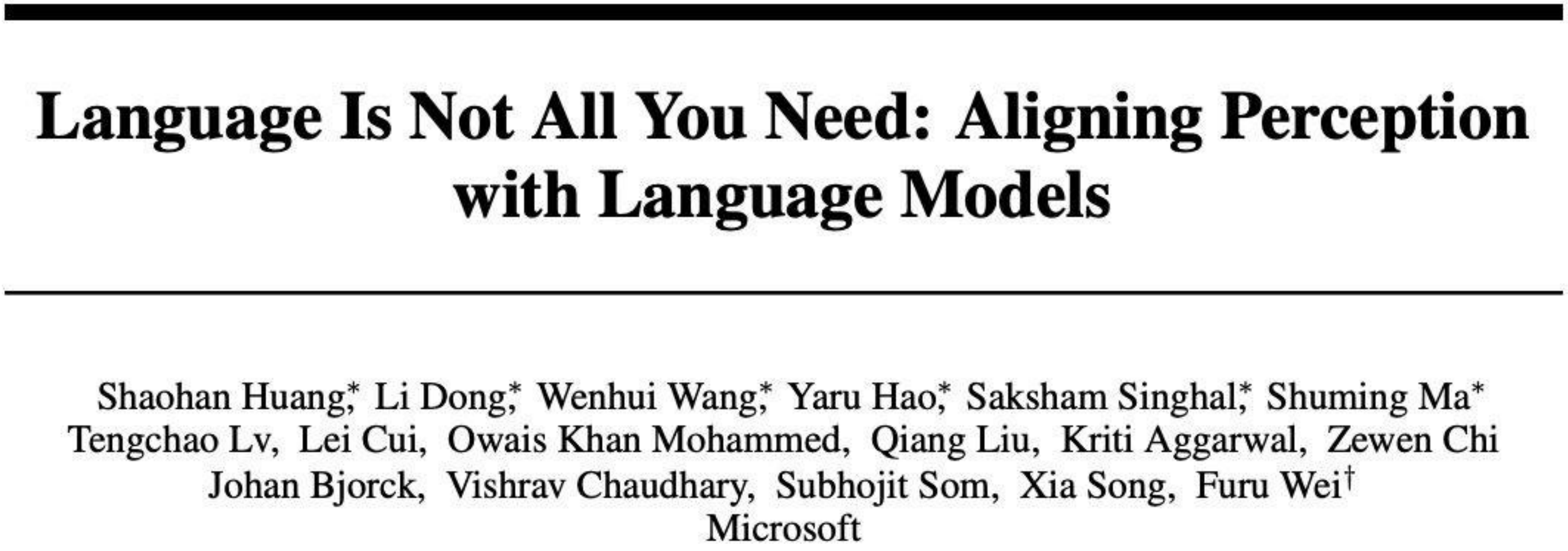
multimodal therapy



In our case, focusing on NLP: text + one or more other *modality* (images, speech, audio, olfaction, others). We’ll mostly focus on images as the other modality.

Multimodal is hot right now

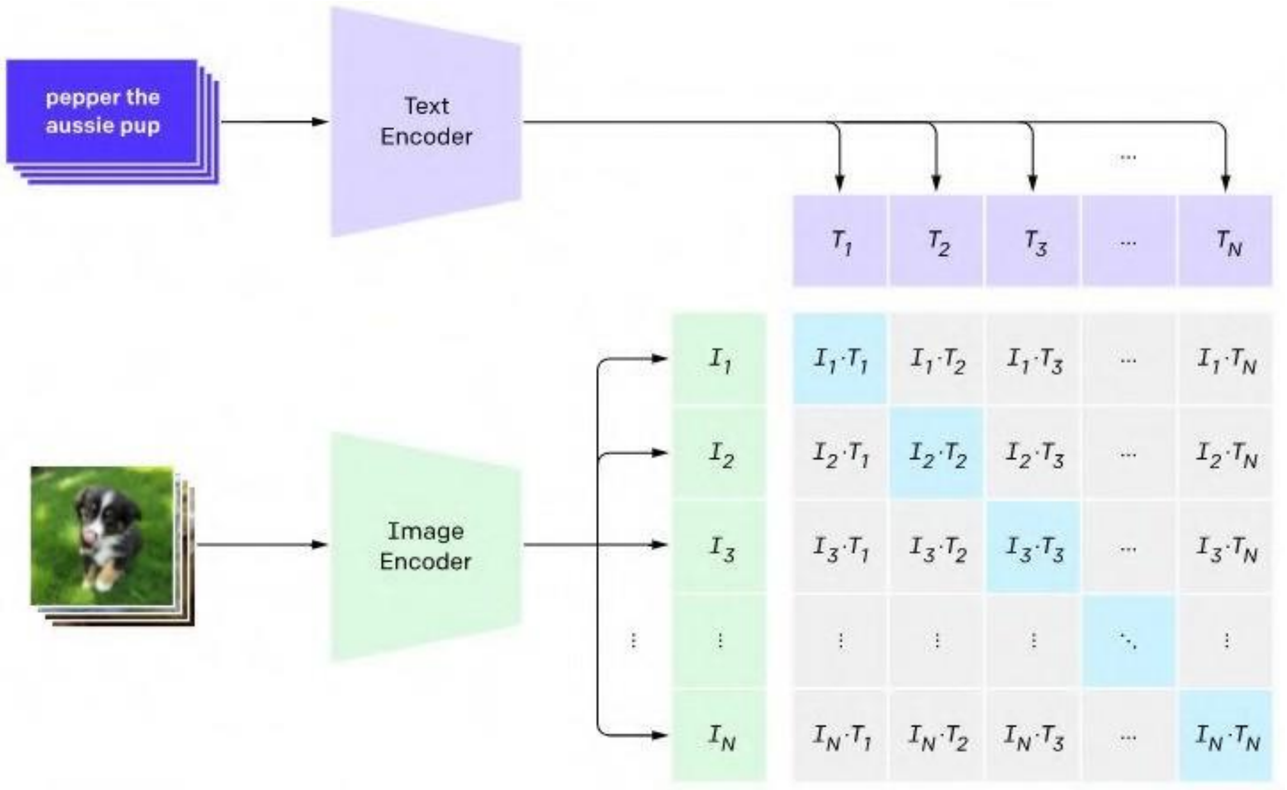
.. and/but has been “the next big thing” for almost a decade!



Language Is Not All You Need: Aligning Perception with Language Models

Shaohan Huang*, Li Dong*, Wenhui Wang*, Yaru Hao*, Saksham Singhal*, Shuming Ma*
Tengchao Lv, Lei Cui, Owais Khan Mohammed, Qiang Liu, Kriti Aggarwal, Zewen Chi
Johan Bjorck, Vishrav Chaudhary, Subhojit Som, Xia Song, Furu Wei†
Microsoft

CLIP: Models and Training Complexity



	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	$I_3 \cdot T_1$	$I_3 \cdot T_2$	$I_3 \cdot T_3$	...	$I_3 \cdot T_N$
⋮	⋮	⋮	⋮	⋱	⋮
I_N	$I_N \cdot T_1$	$I_N \cdot T_2$	$I_N \cdot T_3$	...	$I_N \cdot T_N$

- Text encoder:
  - 12-layer Transformer with causal mask

- Image encoder:
  - ResNet families: RN50, RN101, RN50x4, RN50x16, RN50x64
  - ViT families: ViT-B/32, ViT-B/16, ViT-L/14

Radford et al. "Learning Transferable Visual Models From Natural Language Supervision", ICML 2021

Vision-language models: Contrastive learning

- Contrastive training to bridge the image and text embedding spaces

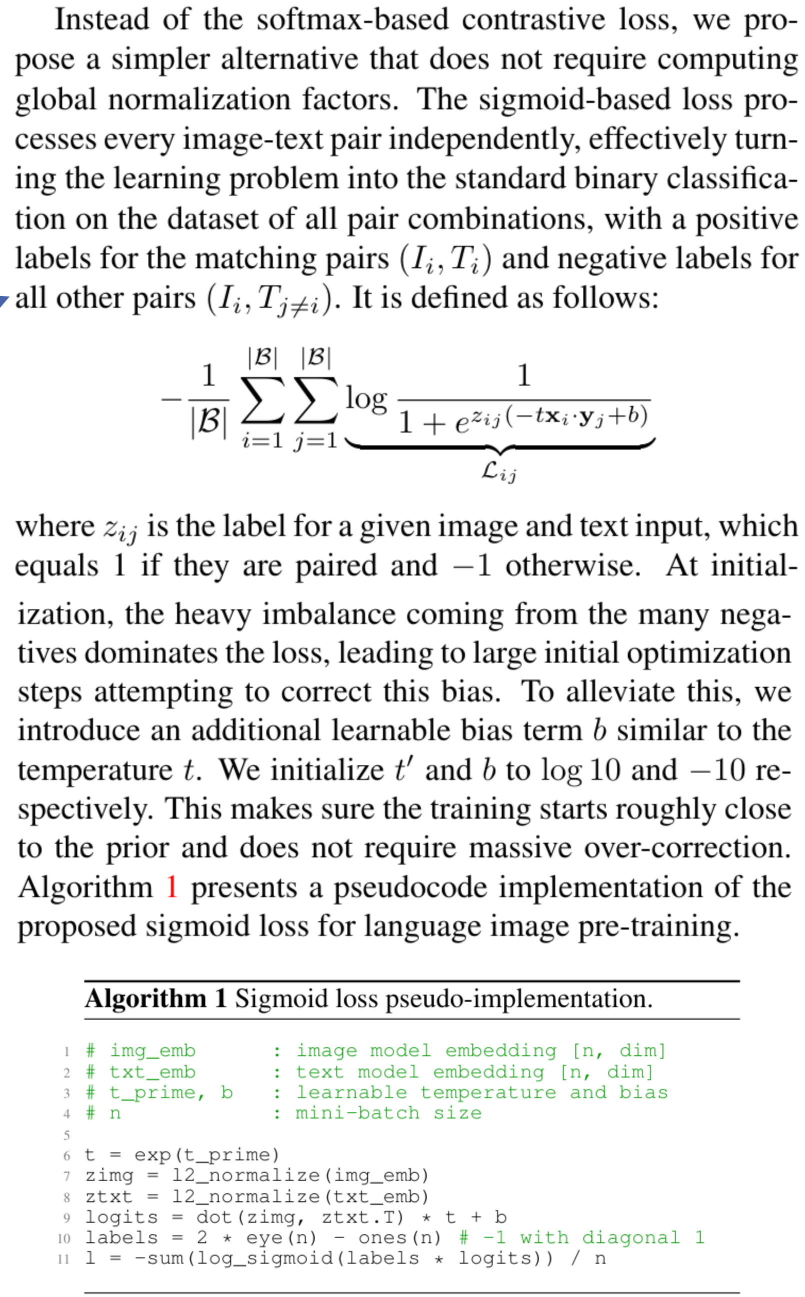
- Making embedding of (image, text) pairs similar and that of non-pairs dissimilar

- This embedding space is super helpful for performing searches across modalities
    - Can return the best caption given an image
    - Has impressive capabilities for zero-shot adaptation to unseen tasks, without the need for fine-tuning

Radford et al. "Learning Transferable Visual Models From Natural Language Supervision", ICML 2021

CLIP Variants

- Objective function or pretraining
  - Combining CLIP with label supervision (BASIC, UniCL, LiT, MOFI)
  - Contrastive + self-supervised image representation learning
    - Contrastive + Self-supervlised methods like SimCLR (SLIP, DeCLIP, nCLIP)
    - Contrastve + Masked Image Modeling (EVA, EVA-02, MVP)
  - Fine-grained matching loss (FILIP)
  - Region-level pretraining (RegionCLIP, GLIP)
  - Sigmoid loss for language-image pre-training (SigCLIP)



Instead of the softmax-based contrastive loss, we propose a simpler alternative that does not require computing global normalization factors. The sigmoid-based loss processes every image-text pair independently, effectively turning the learning problem into the standard binary classification on the dataset of all pair combinations, with a positive labels for the matching pairs (I_i, T_i) and negative labels for all other pairs $(I_i, T_{j\neq i})$. It is defined as follows:

$$-\frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}\sum_{j=1}^{|\mathcal{B}|}\underbrace{\log\frac{1}{1+e^{z_{ij}(-t\mathbf{x}_i\cdot\mathbf{y}_j+b)}}}_{\mathcal{L}_{ij}}$$

where z_{ij} is the label for a given image and text input, which equals 1 if they are paired and -1 otherwise. At initialization, the heavy imbalance coming from the many negatives dominates the loss, leading to large initial optimization steps attempting to correct this bias. To alleviate this, we introduce an additional learnable bias term b similar to the temperature t. We initialize t' and b to $\log 10$ and -10 respectively. This makes sure the training starts roughly close to the prior and does not require massive over-correction. Algorithm 1 presents a pseudocode implementation of the proposed sigmoid loss for language image pre-training.

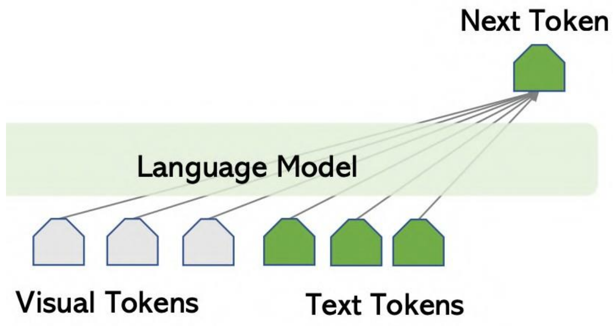
Algorithm 1 Sigmoid loss pseudo-implementation.

```
# img_emb      : image model embedding [n, dim]
# txt_emb      : text model embedding [n, dim]
# t_prime, b   : learnable temperature and bias
# n            : mini-batch size

t = exp(t_prime)
zimg = l2_normalize(img_emb)
ztxt = l2_normalize(txt_emb)
logits = dot(zimg, ztxt.T) * t + b
labels = 2 * eye(n) - ones(n) # -1 with diagonal 1
l = -sum(log_sigmoid(labels * logits)) / n
```

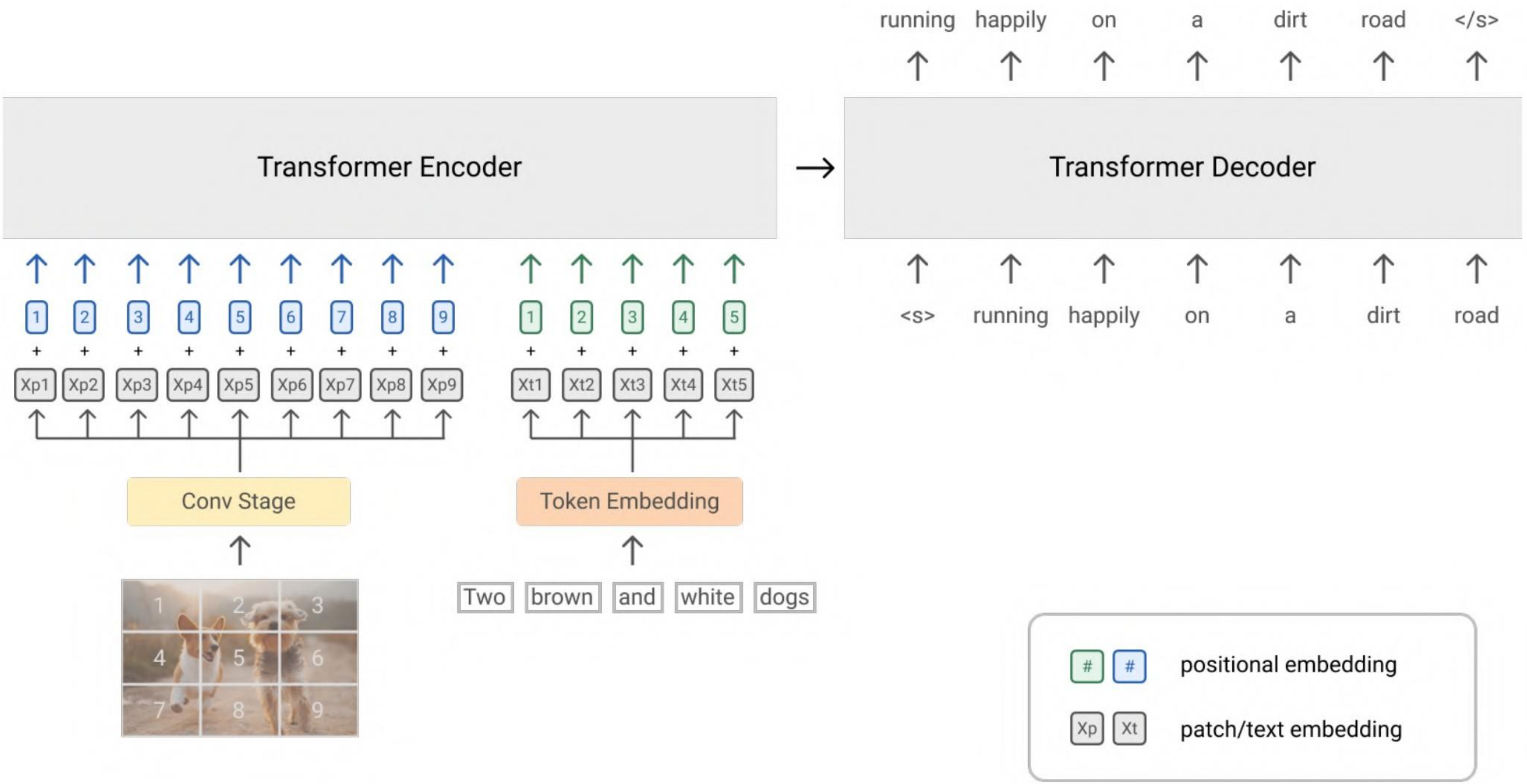
Vision-Language Models: Toward generative models

- Architecture
  - Dual encoders → CLIP & its mentioned variants
  - Encoder-decoder
  - Fusion decoder

SimVLM



Slowly moving from contrastive/discriminative to generative.



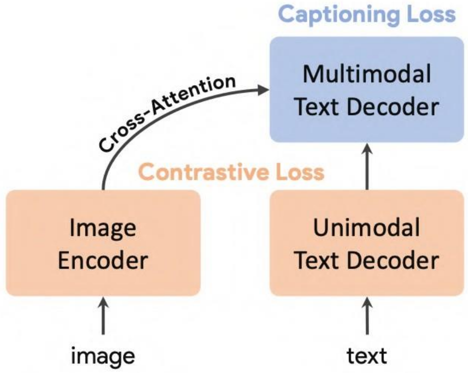
Wang et al., “SimVLM: Simple Visual Language Model Pretraining with Weak Supervision”, ICLR 2022

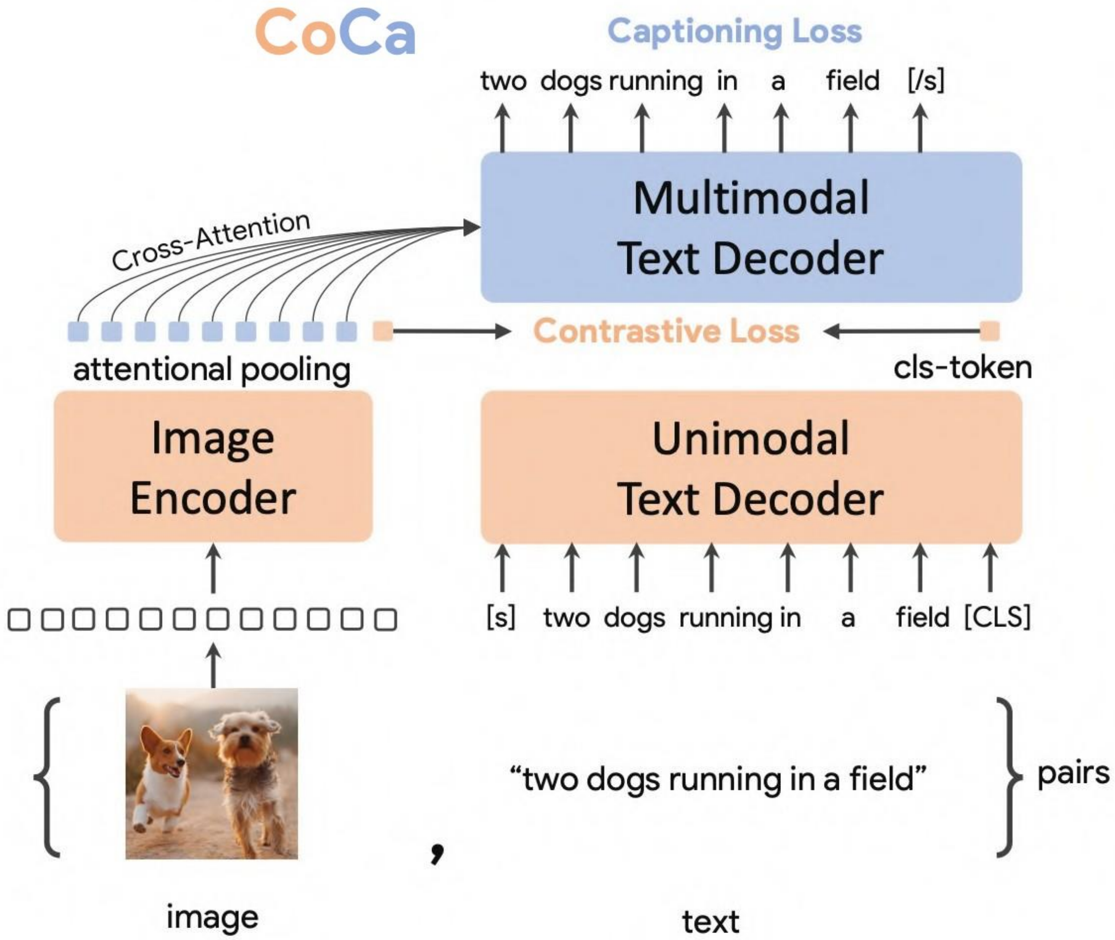
CoCa: Contrastive Captioner

- Use mixed image-text and image-label (JFT-3B) data for pre-training

- A generative branch for enhanced performance and enabling new capabilities (image captioning and VQA)

- CoCa aims to learn a better image encoder from scratch





Yu et al., "Coca: Contrastive captioners are image-text foundation models", 2022

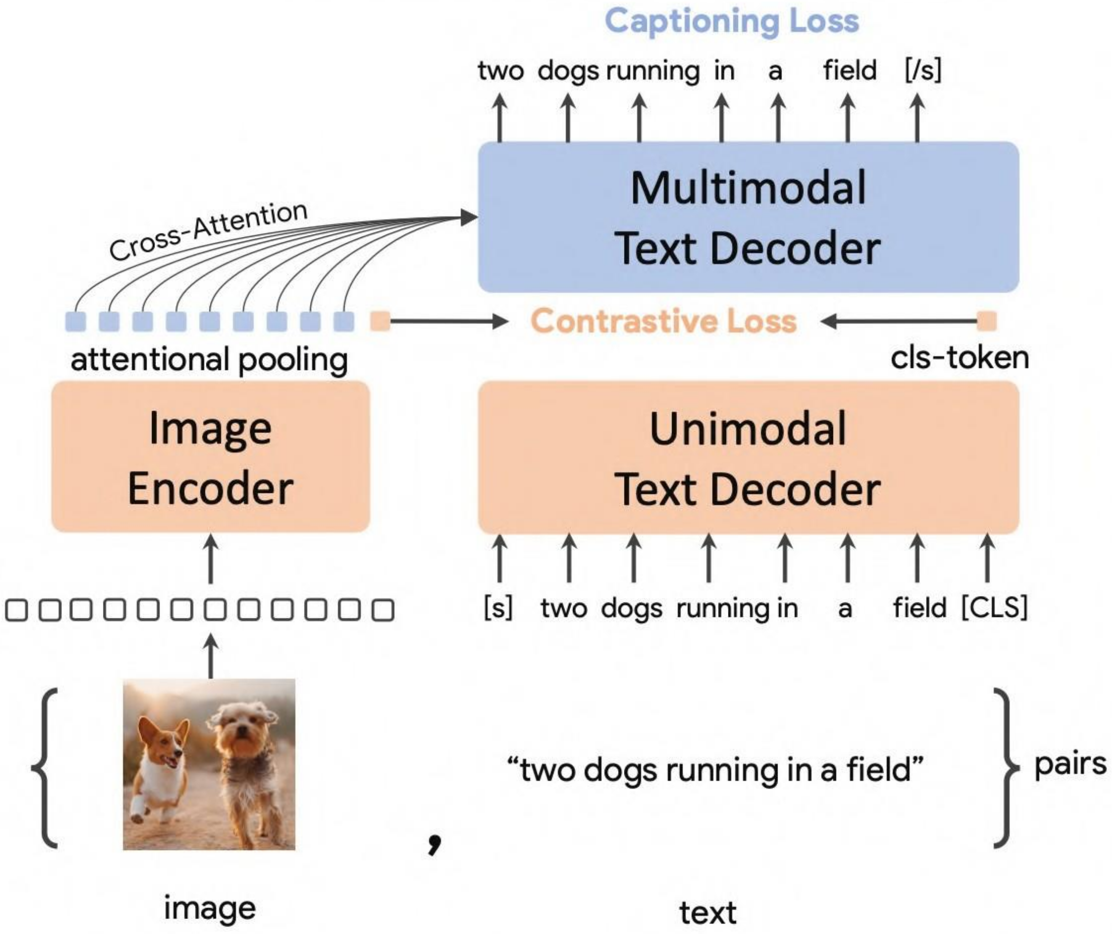
CoCa: Contrastive Captioner

- Use mixed image-text and image-label (JFT-3B) data for pre-training

- A generative branch for enhanced performance and enabling new capabilities (image captioning and VQA)

- CoCa aims to learn a better image encoder from scratch

$$\mathcal{L}_{\text{Cap}} = -\sum_{t=1}^{T} \log P_\theta(y_t | y_{<t}, x).$$



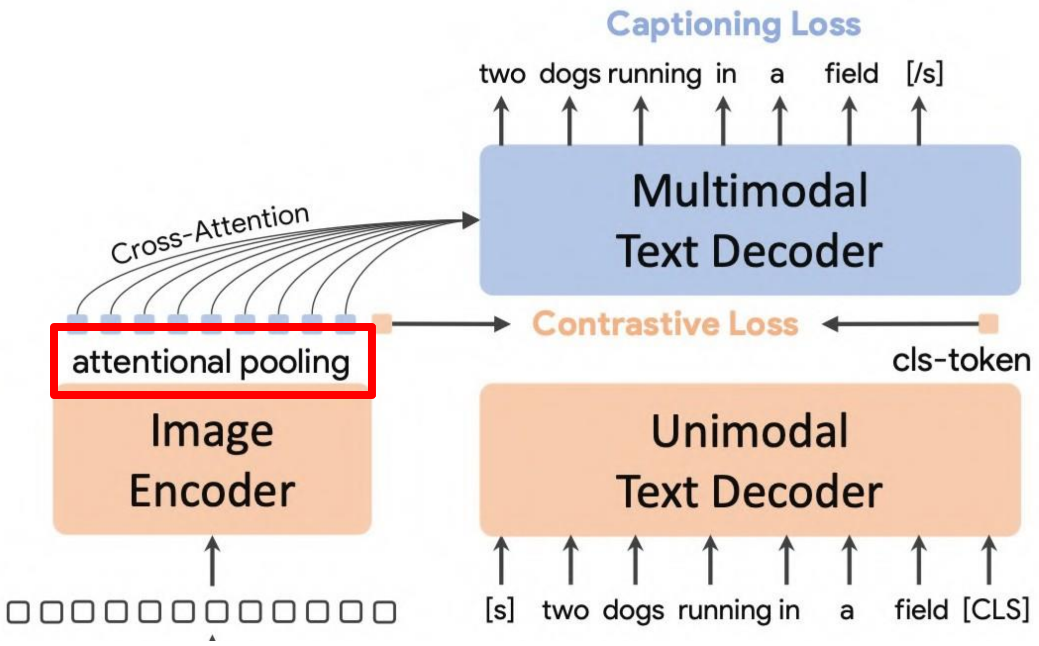
Yu et al., "Coca: Contrastive captioners are image-text foundation models", 2022

CoCa Architecture

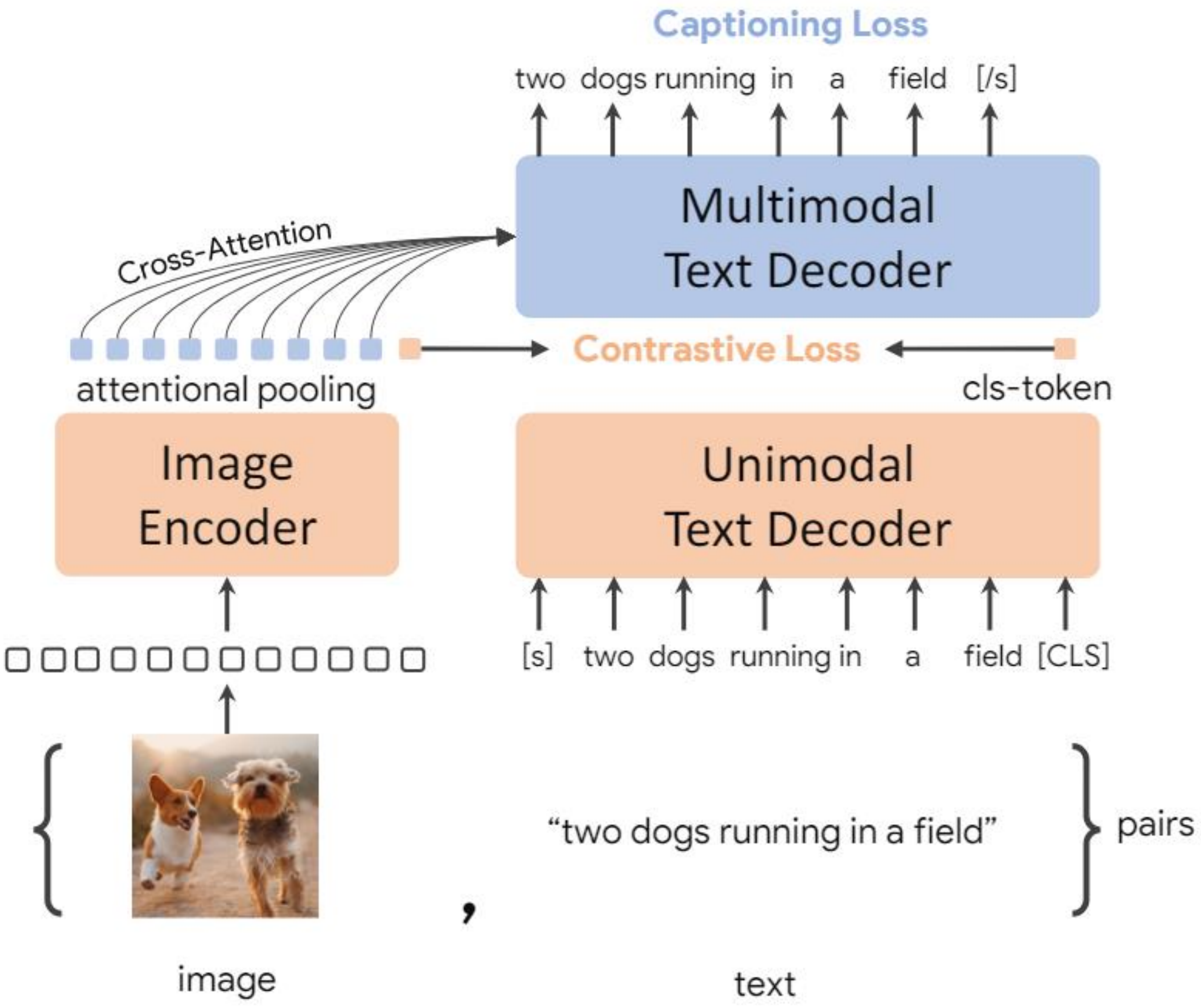
- Unified single-encoder, dual-encoder, and encoder-decoder paradigms
    - one image-text foundation model with the capabilities of all three approaches

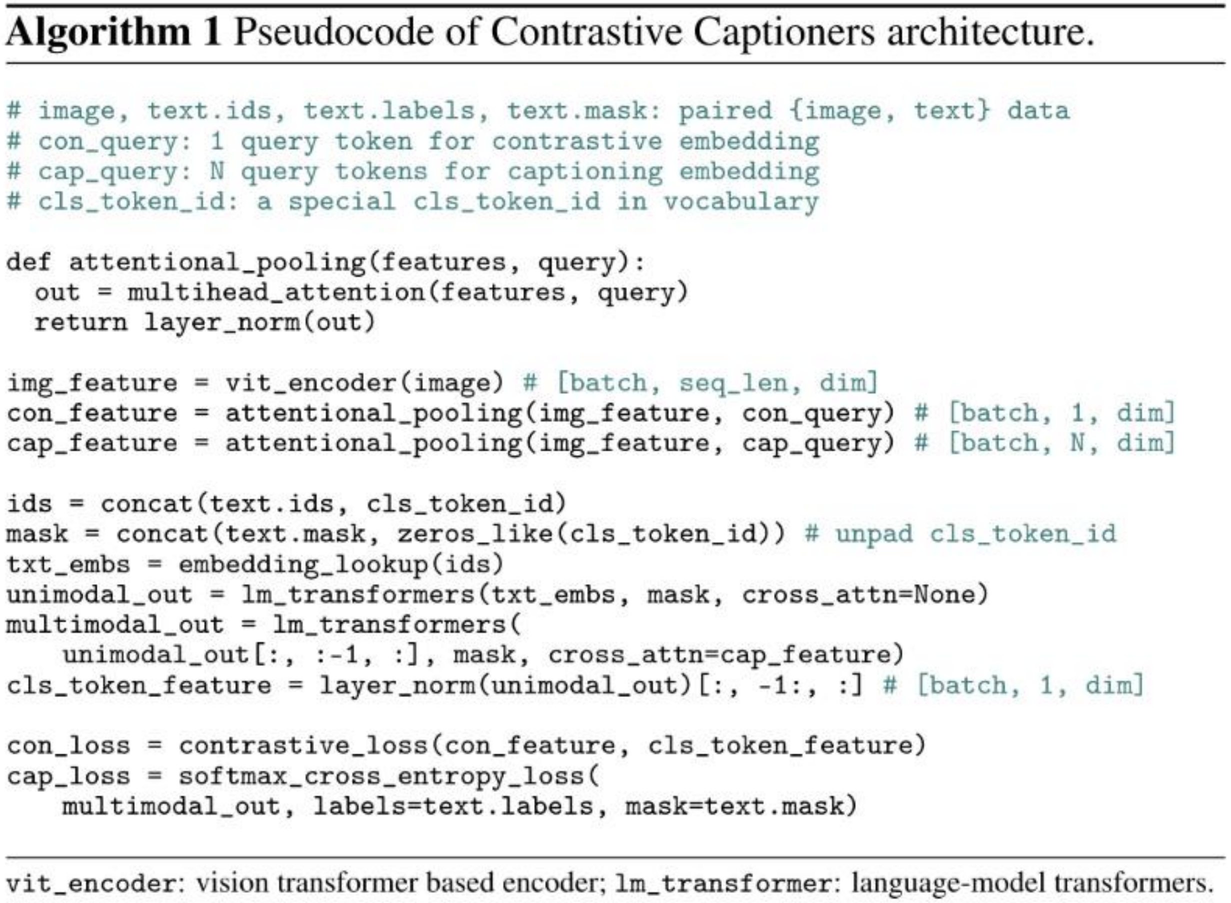
- Cross-attention is omitted in unimodal decoder layers to encode text-only representations

- Multimodal decoder cross-attending to image encoder outputs to learn multimodal representations.



Yu et al., “Coca: Contrastive captioners are image-text foundation models”, 2022





Algorithm 1 Pseudocode of Contrastive Captioners architecture.

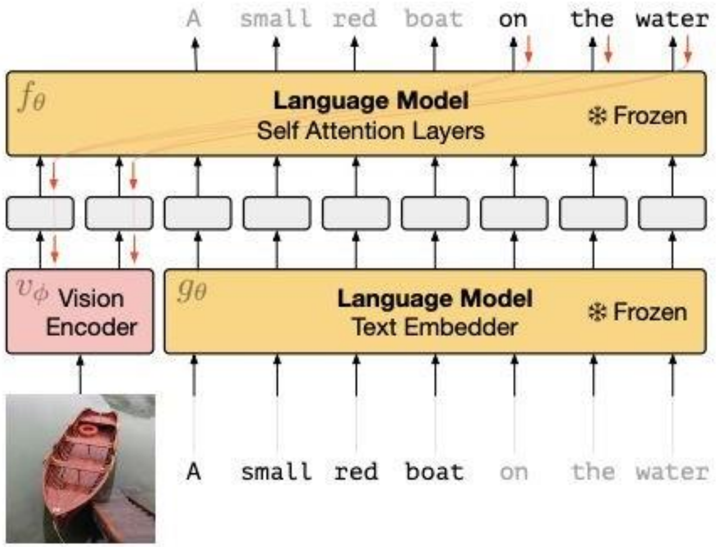
```
# image, text.ids, text.labels, text.mask: paired {image, text} data
# con_query: 1 query token for contrastive embedding
# cap_query: N query tokens for captioning embedding
# cls_token_id: a special cls_token_id in vocabulary

def attentional_pooling(features, query):
  out = multihead_attention(features, query)
  return layer_norm(out)

img_feature = vit_encoder(image) # [batch, seq_len, dim]
con_feature = attentional_pooling(img_feature, con_query) # [batch, 1, dim]
cap_feature = attentional_pooling(img_feature, cap_query) # [batch, N, dim]

ids = concat(text.ids, cls_token_id)
mask = concat(text.mask, zeros_like(cls_token_id)) # unpad cls_token_id
txt_embs = embedding_lookup(ids)
unimodal_out = lm_transformers(txt_embs, mask, cross_attn=None)
multimodal_out = lm_transformers(
    unimodal_out[:, :-1, :], mask, cross_attn=cap_feature)
cls_token_feature = layer_norm(unimodal_out)[:, -1:, :] # [batch, 1, dim]

con_loss = contrastive_loss(con_feature, cls_token_feature)
cap_loss = softmax_cross_entropy_loss(
    multimodal_out, labels=text.labels, mask=text.mask)
```

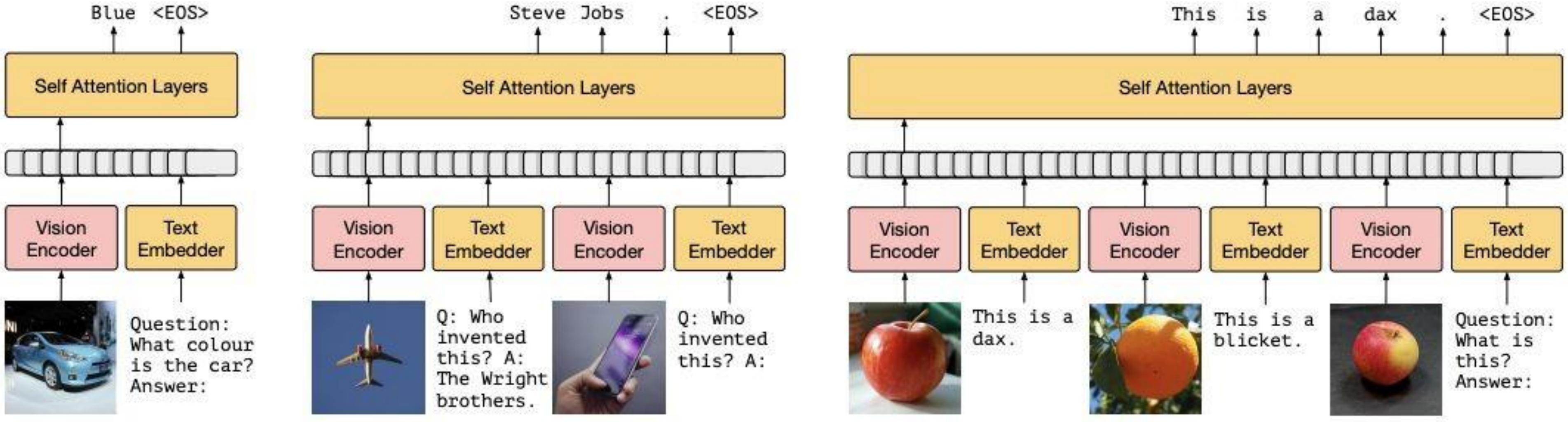
`vit_encoder`: vision transformer based encoder; `lm_transformer`: language-model transformers.

Frozen (Tsimpoukelli, Menick, Cabi, et al., 2021)

Kind of like MMBT but with a better LLM (T5) and a better vision encoder (NF-ResNet).

Multi-Modal Few-Shot Learners!





(a) **0-shot VQA**

(b) **1-shot outside-knowledge VQA**

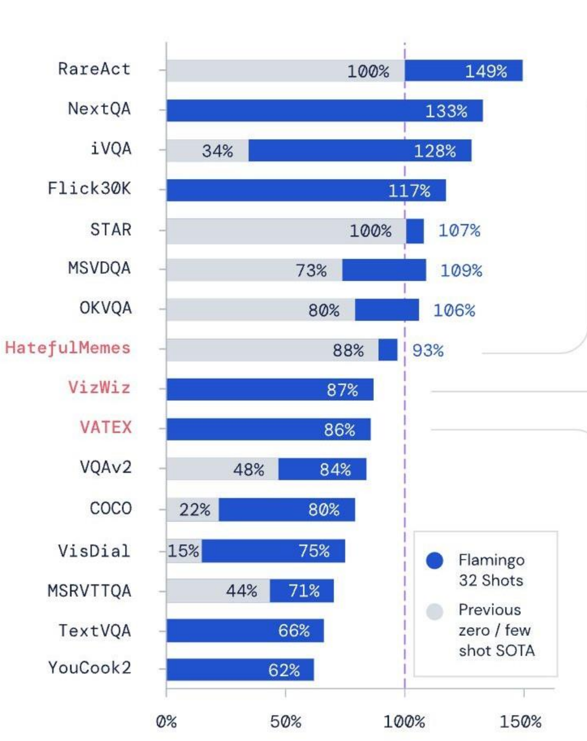
(c) **Few-shot image classification**

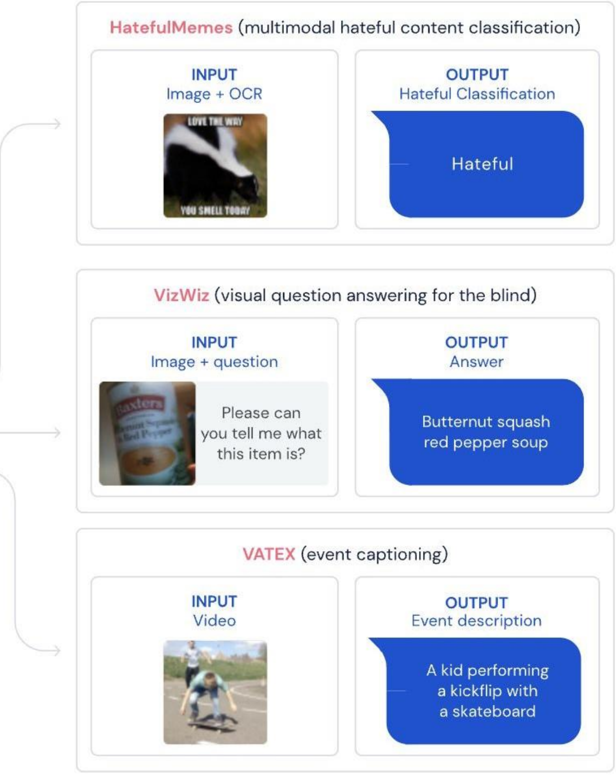
Flamingo (Alayrac et al., 2022)

80b param model based on Chinchilla.

Multi-image.

Performance relative to SOTA





HatefulMemes (multimodal hateful content classification)

INPUT
Image + OCR

OUTPUT
Hateful Classification

Hateful

VizWiz (visual question answering for the blind)

INPUT
Image + question

Please can you tell me what this item is?

OUTPUT
Answer

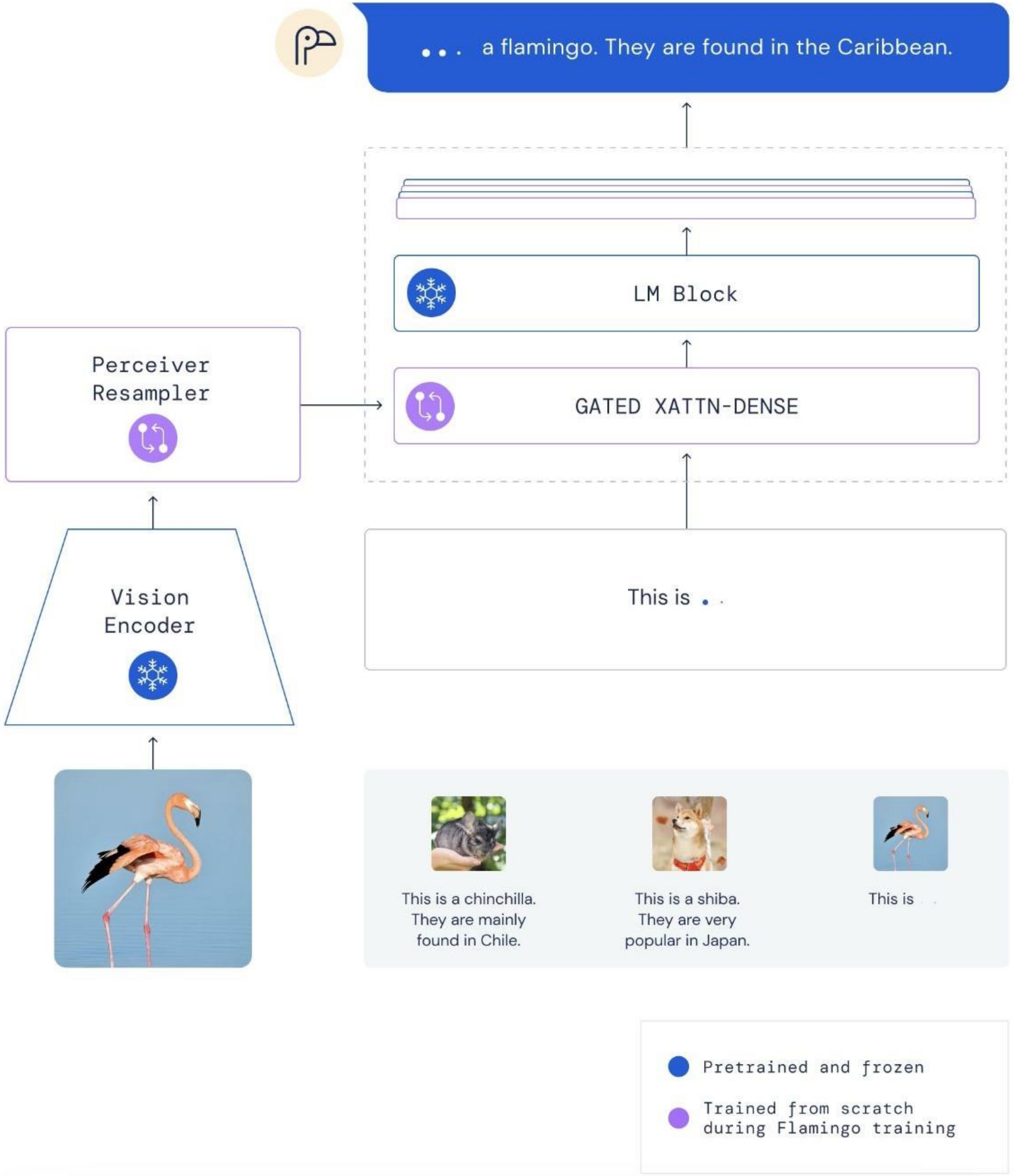
Butternut squash red pepper soup

VATEX (event captioning)

INPUT
Video

OUTPUT
Event description

A kid performing a kickflip with a skateboard



. . . a flamingo. They are found in the Caribbean.

LM Block

GATED XATTN-DENSE

Perceiver Resampler

Vision Encoder

This is . .

This is a chinchilla. They are mainly found in Chile.

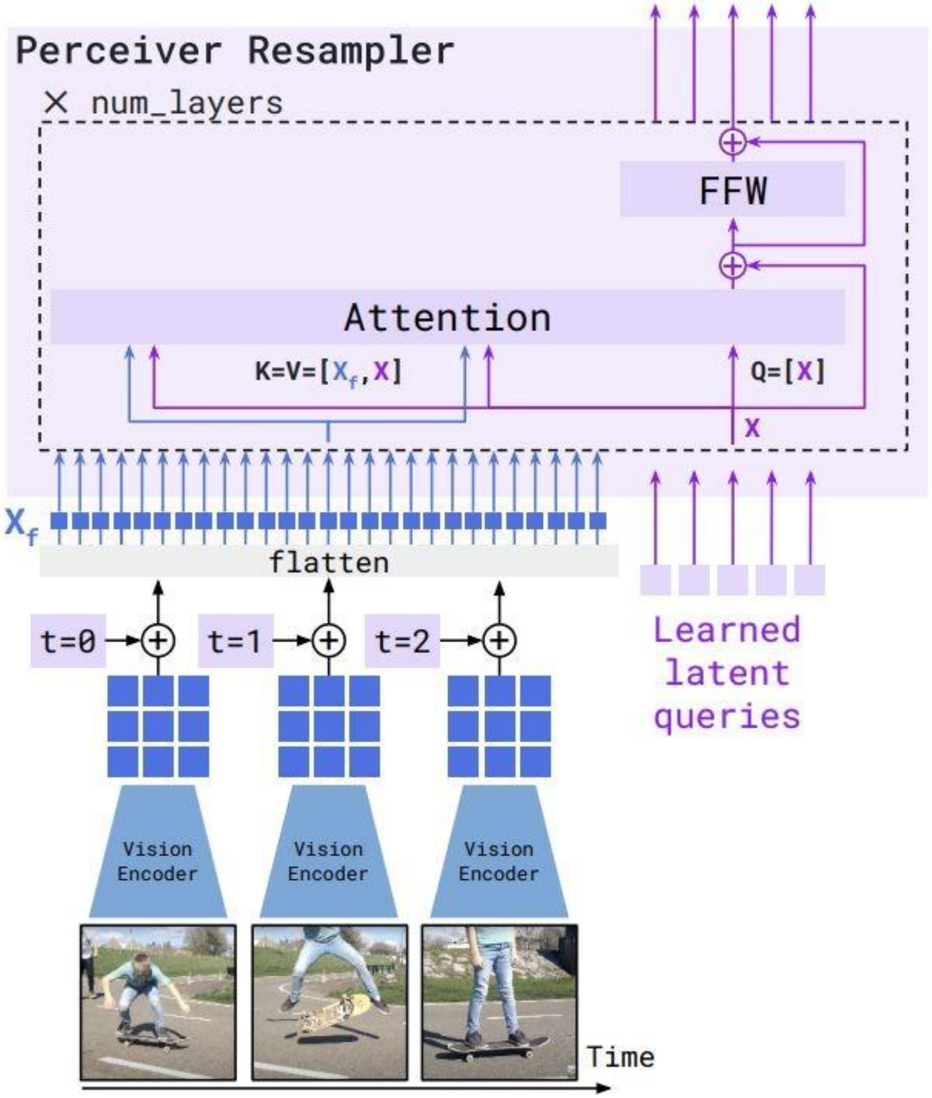
This is a shiba. They are very popular in Japan.

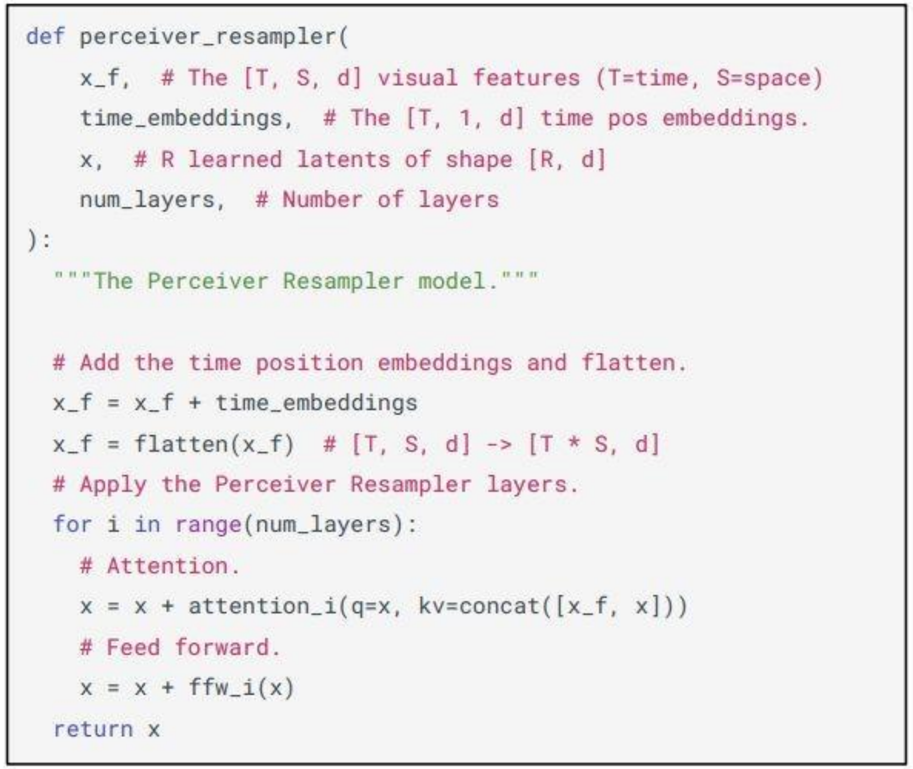
This is

Pretrained and frozen

Trained from scratch during Flamingo training

Perceiver Resampler

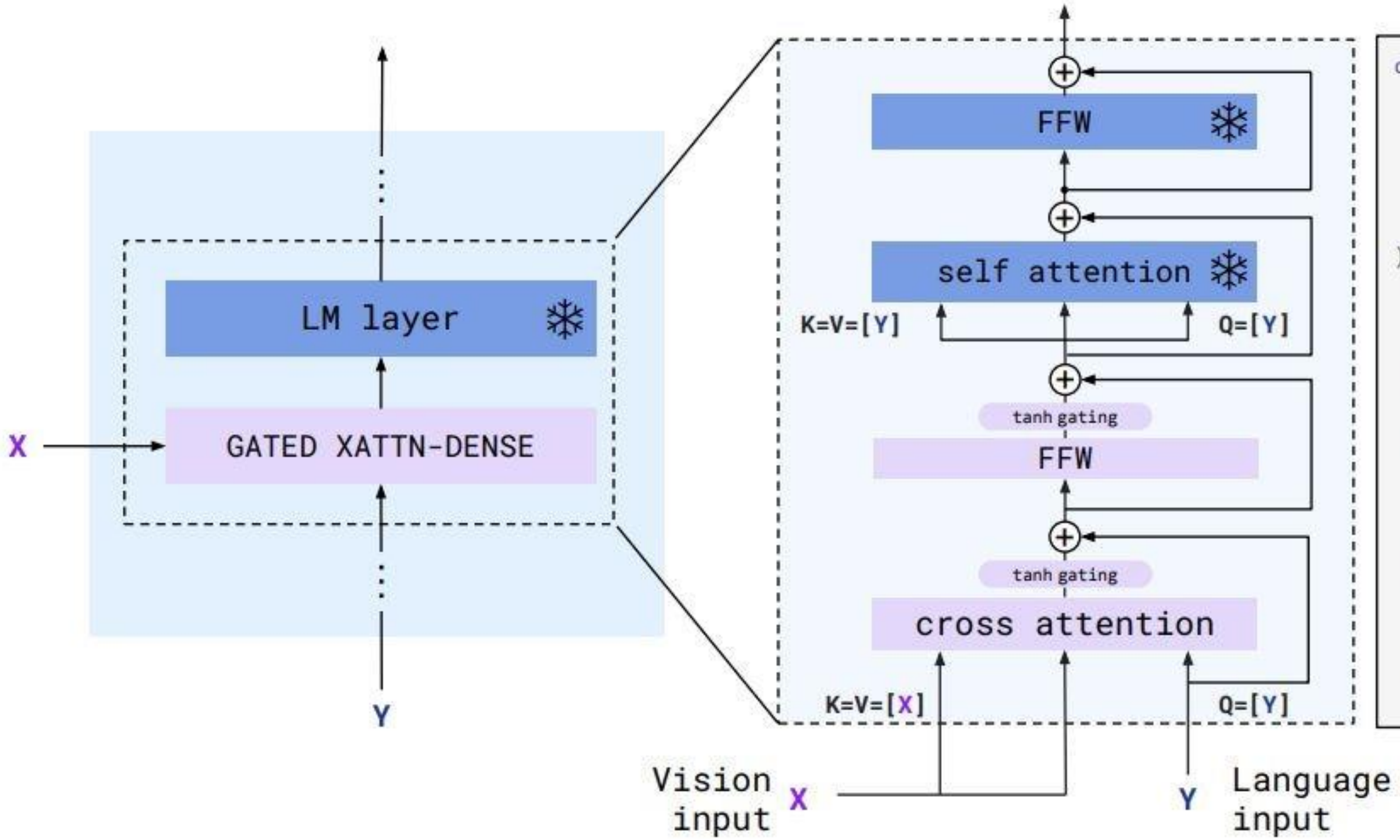


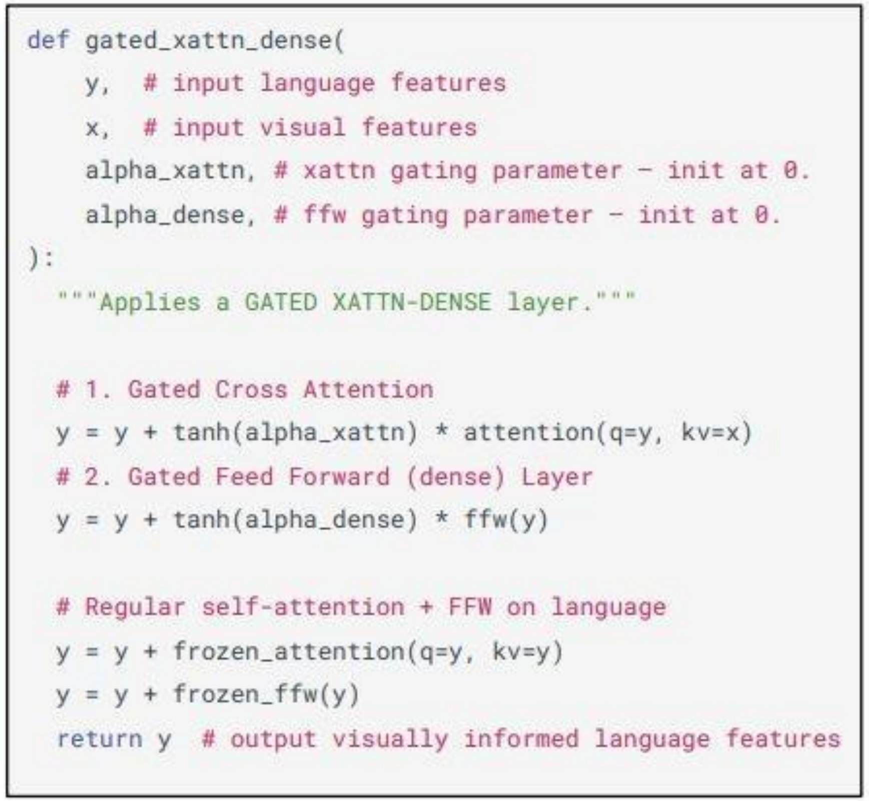


```
def perceiver_resampler(
    x_f,  # The [T, S, d] visual features (T=time, S=space)
    time_embeddings,  # The [T, 1, d] time pos embeddings.
    x,  # R learned latents of shape [R, d]
    num_layers,  # Number of layers
):
  """The Perceiver Resampler model."""

  # Add the time position embeddings and flatten.
  x_f = x_f + time_embeddings
  x_f = flatten(x_f)  # [T, S, d] -> [T * S, d]
  # Apply the Perceiver Resampler layers.
  for i in range(num_layers):
    # Attention.
    x = x + attention_i(q=x, kv=concat([x_f, x]))
    # Feed forward.
    x = x + ffw_i(x)
  return x
```

Gated XATTN

Inject visual info directly into a frozen LM via cross-attention (remember FiLM?).

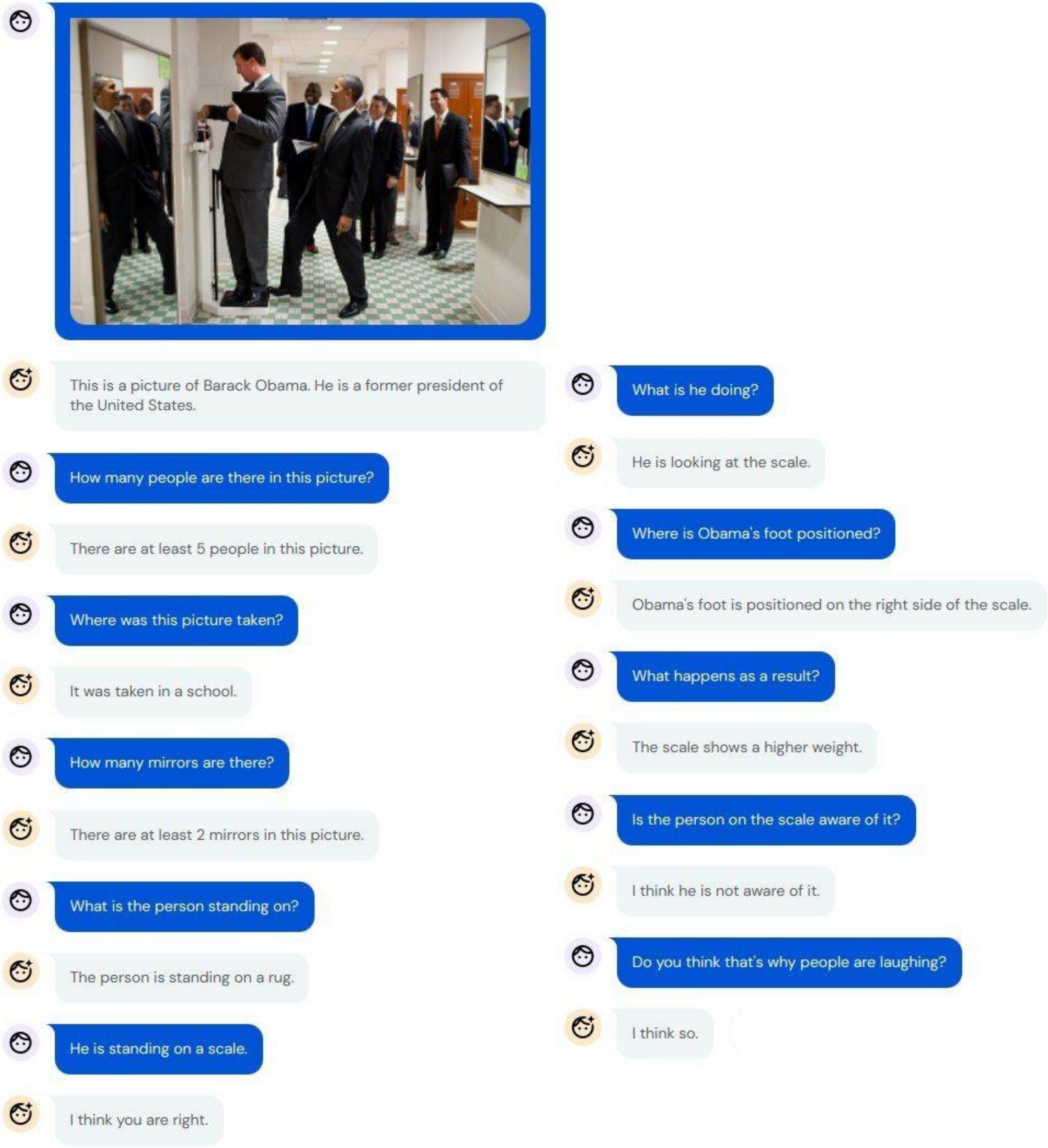




```
def gated_xattn_dense(
    y,  # input language features
    x,  # input visual features
    alpha_xattn, # xattn gating parameter – init at 0.
    alpha_dense, # ffw gating parameter – init at 0.
):
  """Applies a GATED XATTN-DENSE layer."""

  # 1. Gated Cross Attention
  y = y + tanh(alpha_xattn) * attention(q=y, kv=x)
  # 2. Gated Feed Forward (dense) Layer
  y = y + tanh(alpha_dense) * ffw(y)

  # Regular self-attention + FFW on language
  y = y + frozen_attention(q=y, kv=y)
  y = y + frozen_ffw(y)
  return y  # output visually informed language features
```

Why is this funny?

Original image from Karpathy as a "visual Turing test" →



This is a picture of Barack Obama. He is a former president of the United States.

How many people are there in this picture?

There are at least 5 people in this picture.

Where was this picture taken?

It was taken in a school.

How many mirrors are there?

There are at least 2 mirrors in this picture.

What is the person standing on?

The person is standing on a rug.

He is standing on a scale.

I think you are right.

What is he doing?

He is looking at the scale.

Where is Obama's foot positioned?

Obama's foot is positioned on the right side of the scale.

What happens as a result?

The scale shows a higher weight.

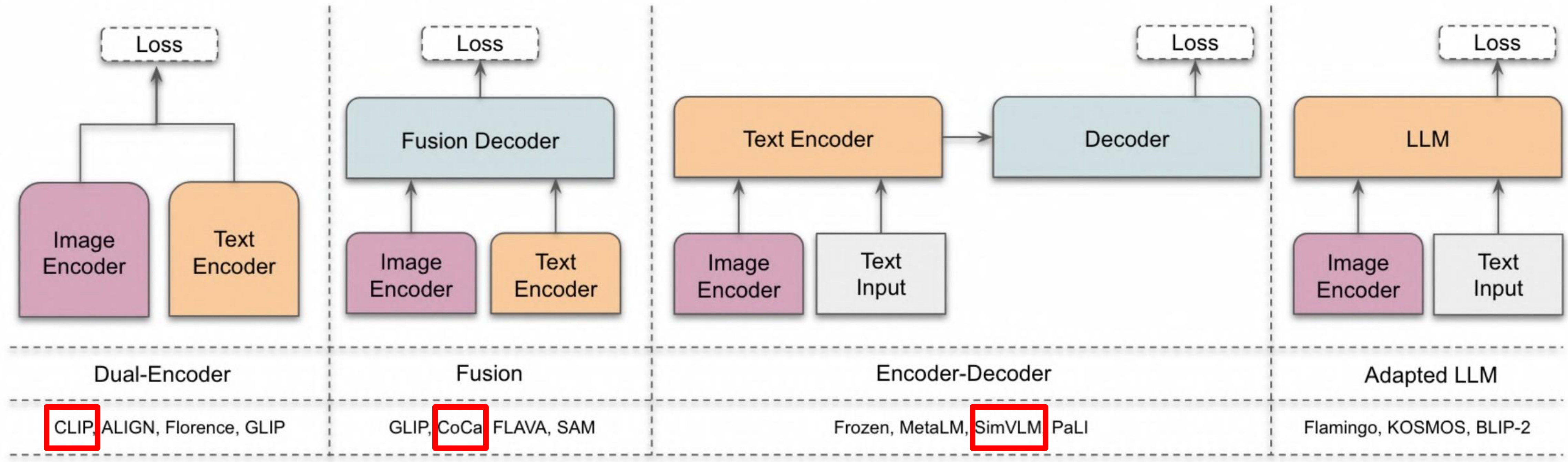
Is the person on the scale aware of it?

I think he is not aware of it.

Do you think that's why people are laughing?

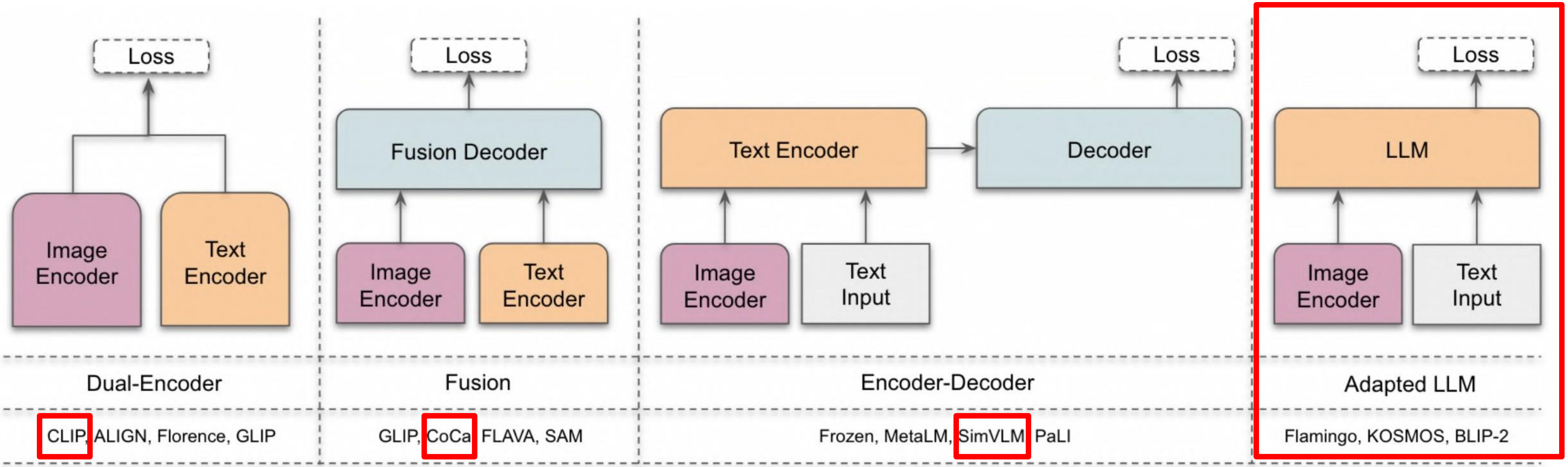
I think so.

Architecture of Multimodal Models

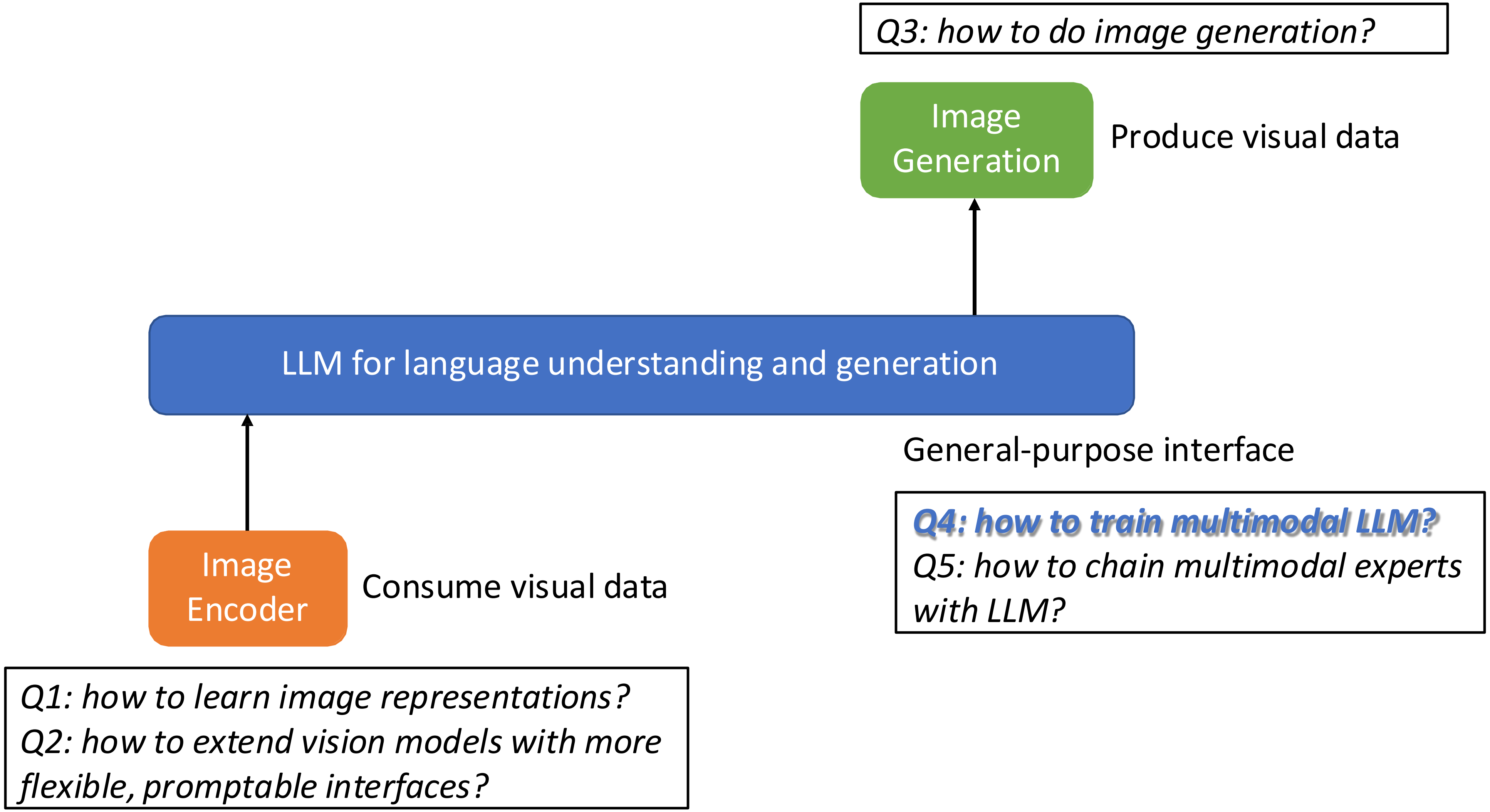


Li et al., “Multimodal Foundation Models: From Specialists to General-Purpose Assistants”, 2023

Architecture of Multimodal Models



Li et al., "Multimodal Foundation Models: From Specialists to General-Purpose Assistants", 2023



Q3: how to do image generation?

Image Generation
Produce visual data
LLM for language understanding and generation
General-purpose interface
Image Encoder
Consume visual data

Q4: how to train multimodal LLM?
Q5: how to chain multimodal experts with LLM?

Q1: how to learn image representations?
Q2: how to extend vision models with more flexible, promptable interfaces?

Visual Instruction Tuning with GPT-4

https://llava-vl.github.io/

Haotian Liu*, Chunyuan Li*, Qingyang Wu, Yong Jae Lee (* Equal contribution)

Self-Instruct with Strong Teacher LLMs

	LLaMA	Alpaca	Vicuna
Teacher		GPT-3.5	ShareGPT (Human & GPT)
Instruction-following Data	None	52K	700K (70 conversions)

But No Teacher is available on multiGPT4?

GPT-4-LLM	LLaVA
GPT-4 (text-only)	GPT-4 (text-only)

- 158K multimodal instruction following data (First & High Quality)

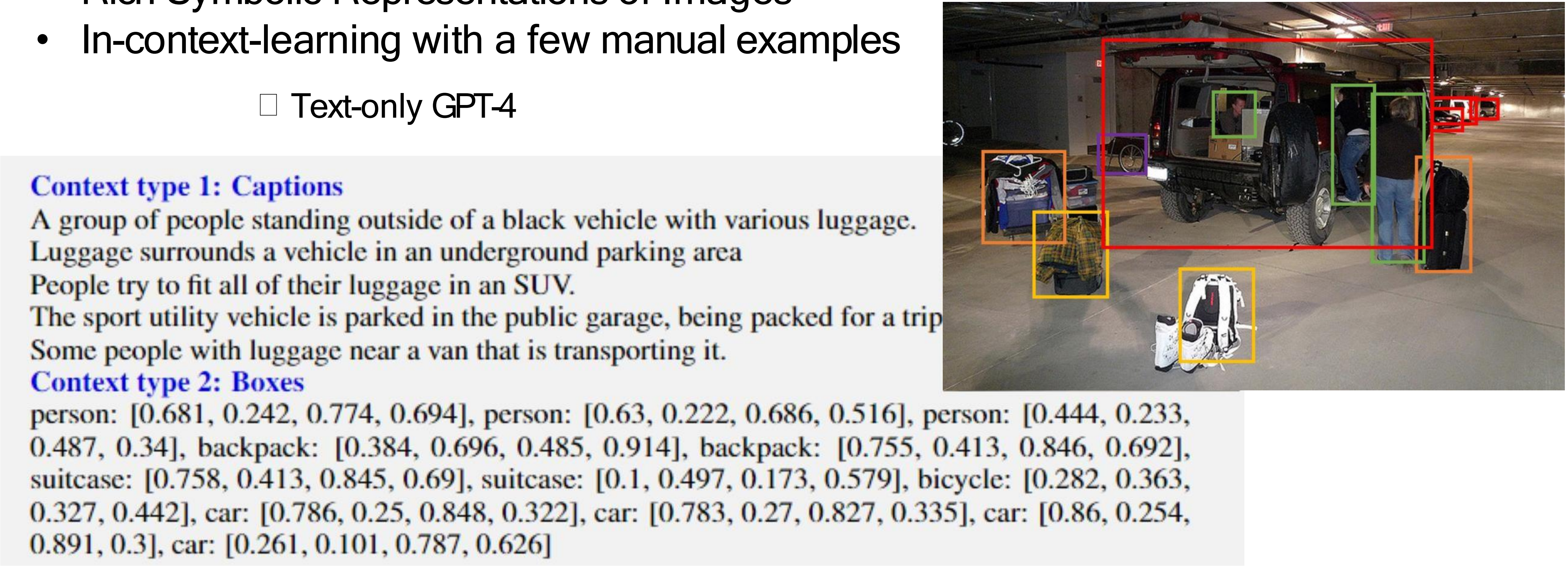
→ Multimodal Chatbot

Large Language and Vision Assistant

GPT-assisted Visual Instruction Data Generation

- Rich Symbolic Representations of Images
- In-context-learning with a few manual examples

□ Text-only GPT-4



Context type 1: Captions

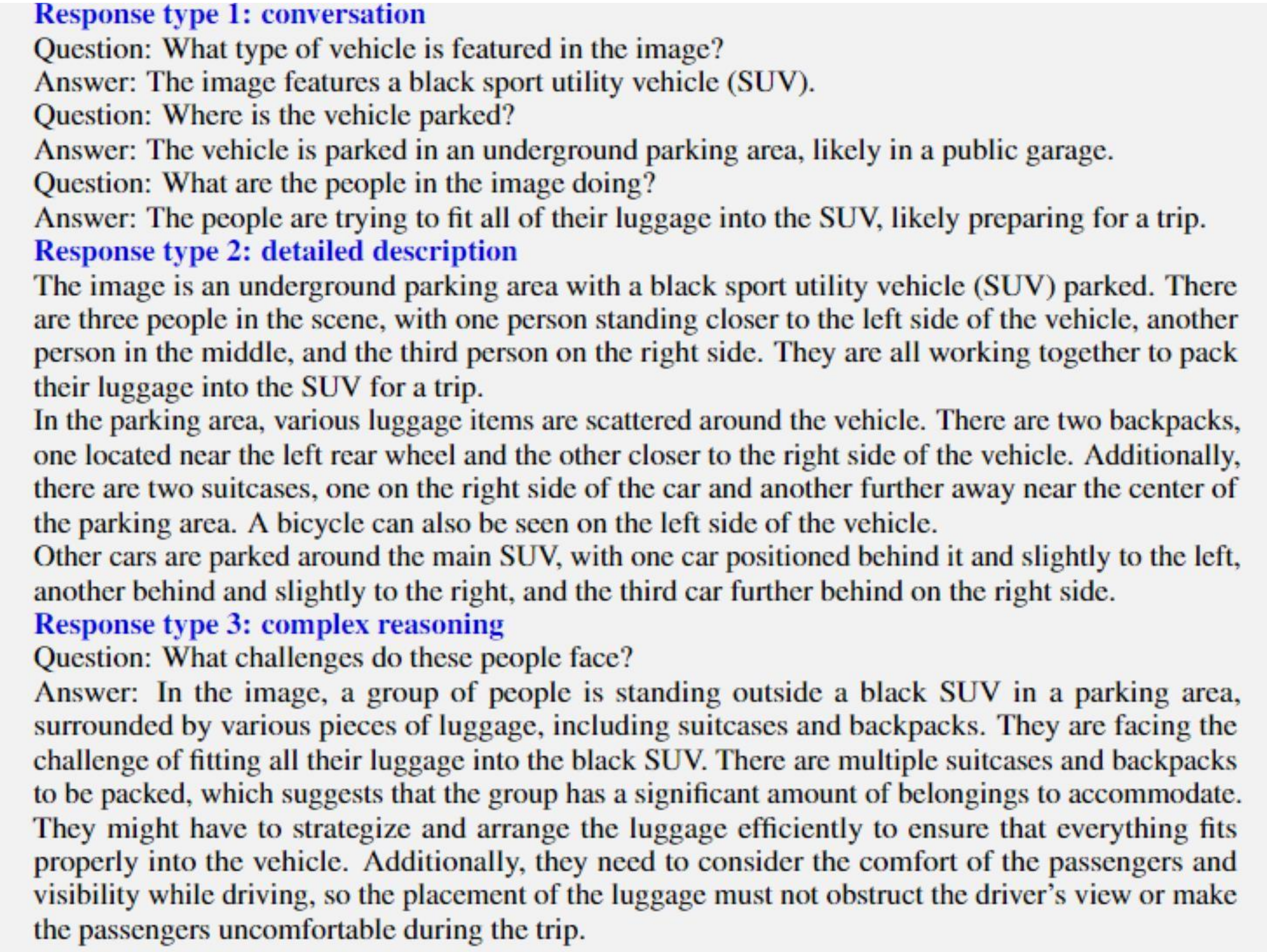
A group of people standing outside of a black vehicle with various luggage.
Luggage surrounds a vehicle in an underground parking area
People try to fit all of their luggage in an SUV.
The sport utility vehicle is parked in the public garage, being packed for a trip
Some people with luggage near a van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

Context type 2: Boxes

person: [0.681, 0.242, 0.774, 0.694], person: [0.63, 0.222, 0.686, 0.516], person: [0.444, 0.233, 0.487, 0.34], backpack: [0.384, 0.696, 0.485, 0.914], backpack: [0.755, 0.413, 0.846, 0.692], suitcase: [0.758, 0.413, 0.845, 0.69], suitcase: [0.1, 0.497, 0.173, 0.579], bicycle: [0.282, 0.363, 0.327, 0.442], car: [0.786, 0.25, 0.848, 0.322], car: [0.783, 0.27, 0.827, 0.335], car: [0.86, 0.254, 0.891, 0.3], car: [0.261, 0.101, 0.787, 0.626]

GPT-assisted Visual Instruction Data Generation

Three type of instruction-following responses



Response type 1: conversation

Question: What type of vehicle is featured in the image?
Answer: The image features a black sport utility vehicle (SUV).
Question: Where is the vehicle parked?
Answer: The vehicle is parked in an underground parking area, likely in a public garage.
Question: What are the people in the image doing?
Answer: The people are trying to fit all of their luggage into the SUV, likely preparing for a trip.

Response type 2: detailed description

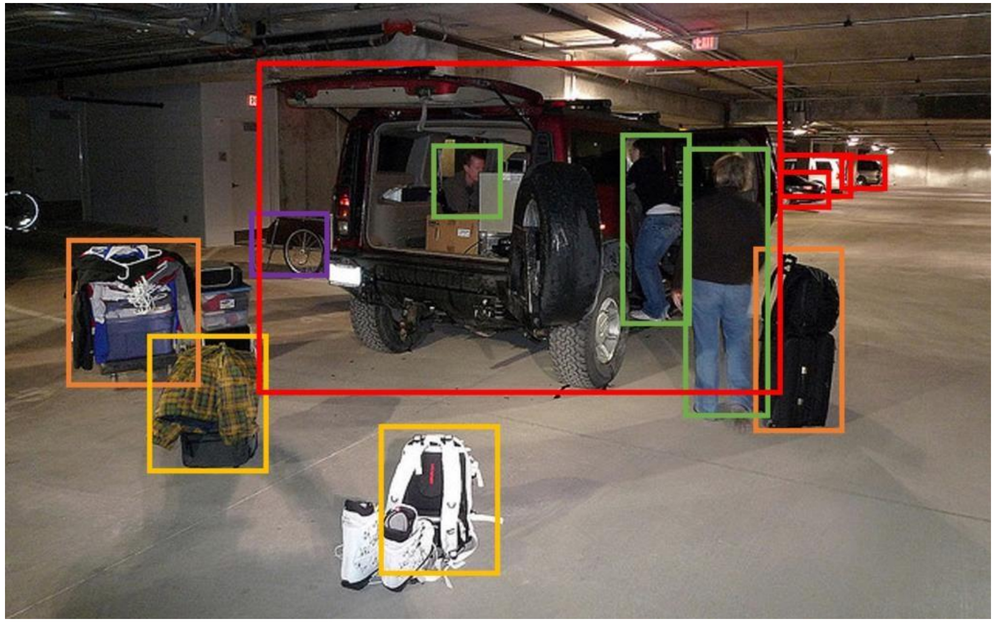
The image is an underground parking area with a black sport utility vehicle (SUV) parked. There are three people in the scene, with one person standing closer to the left side of the vehicle, another person in the middle, and the third person on the right side. They are all working together to pack their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

In the parking area, various luggage items are scattered around the vehicle. There are two backpacks, one located near the left rear wheel and the other closer to the right side of the vehicle. Additionally, there are two suitcases, one on the right side of the car and another further away near the center of the parking area. A bicycle can also be seen on the left side of the vehicle.

Other cars are parked around the main SUV, with one car positioned behind it and slightly to the left, another behind and slightly to the right, and the third car further behind on the right side.

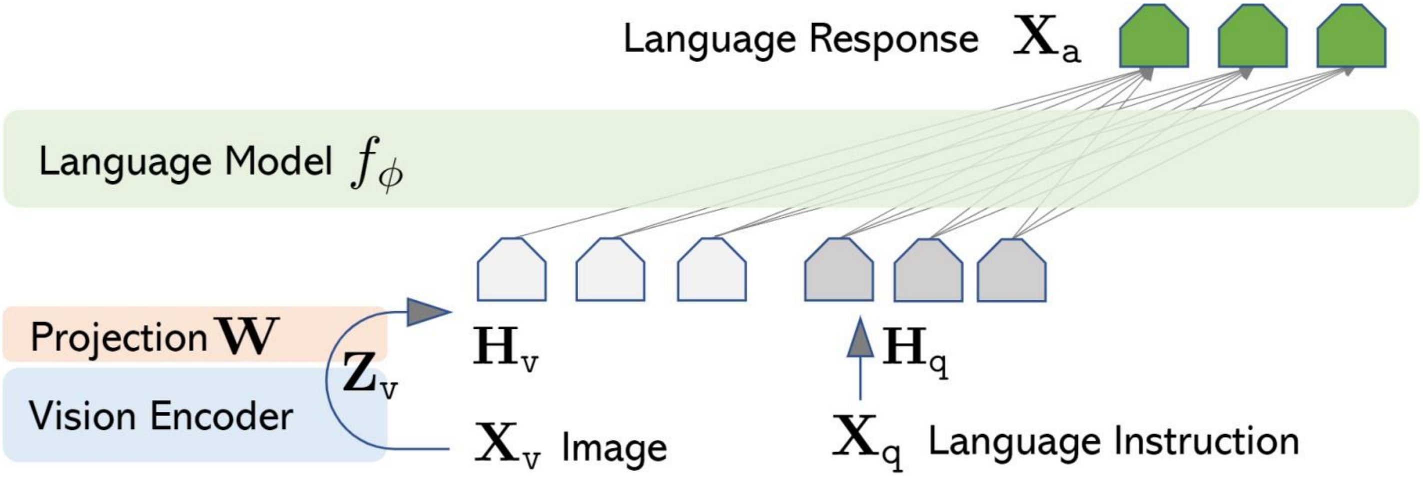
Response type 3: complex reasoning

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 to accommodate. They might have to strategize and arrange the luggage efficiently to ensure that everything fits properly into the vehicle. Additionally, they need to consider the comfort of the passengers and visibility while driving, so the placement of the luggage must not obstruct the driver's view or make the passengers uncomfortable during the trip.



LLaVA: Large Language-and-Vision Assistant

❑ Architecture

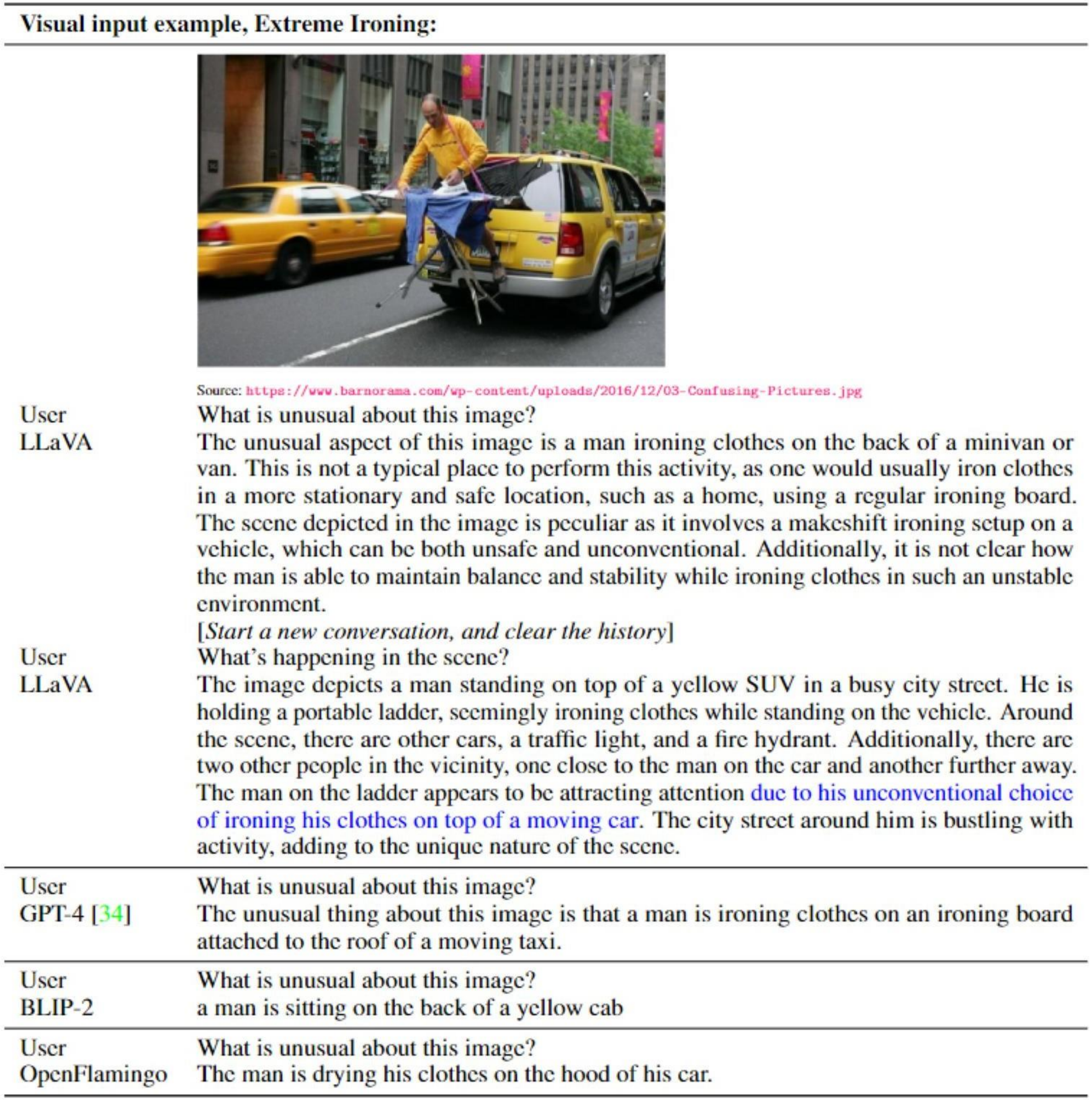


❑ Two-stage Training

- **Stage 1: Pre-training for Feature Alignment**.

  Only the projection matrix is updated, based on a subset of CC3M.
- **Stage 2: Fine-tuning End-to-End.** Both the projection matrix and LLM are updated
  - **Visual Chat**: Our generated multimodal instruction data for daily user-oriented applications.
  - **Science QA**: Multimodal reasoning dataset for the science domain.

Example 1: Extreme Ironing

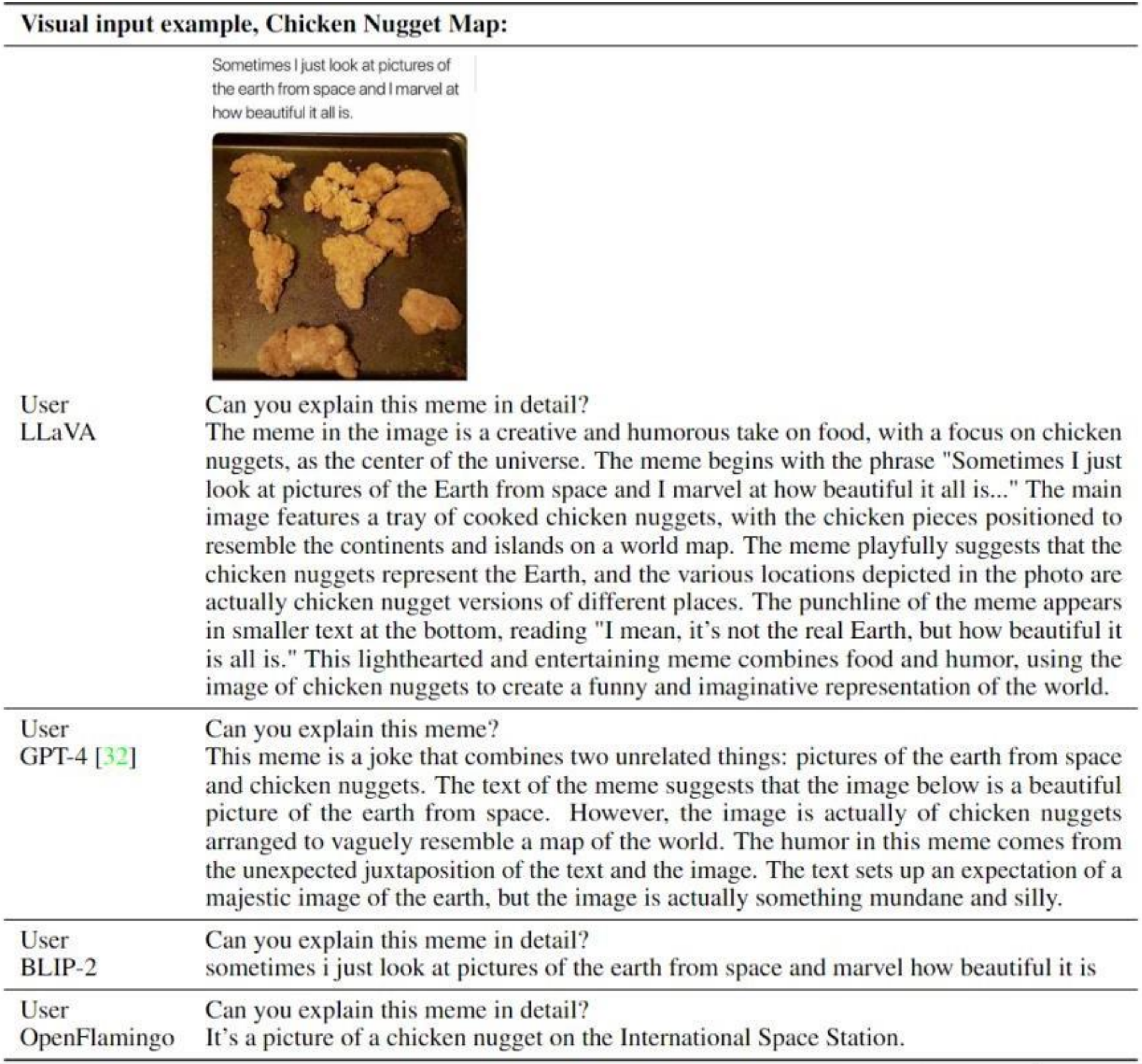


Visual input example, Extreme Ironing:

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.
	[Start a new conversation, and clear the history]
User	What's happening in the scene?
LLaVA	The image depicts a man standing on top of a yellow SUV in a busy city street. He is holding a portable ladder, seemingly ironing clothes while standing on the vehicle. Around the scene, there are other cars, a traffic light, and a fire hydrant. Additionally, there are two other people in the vicinity, one close to the man on the car and another further away. The man on the ladder appears to be attracting attention due to his unconventional choice of ironing his clothes on top of a moving car. The city street around him is bustling with activity, adding to the unique nature of the scene.
User	What is unusual about this image?
GPT-4 [34]	The unusual thing about this image is that a man is ironing clothes on an ironing board attached to the roof of a moving taxi.
User	What is unusual about this image?
BLIP-2	a man is sitting on the back of a yellow cab
User	What is unusual about this image?
OpenFlamingo	The man is drying his clothes on the hood of his car.

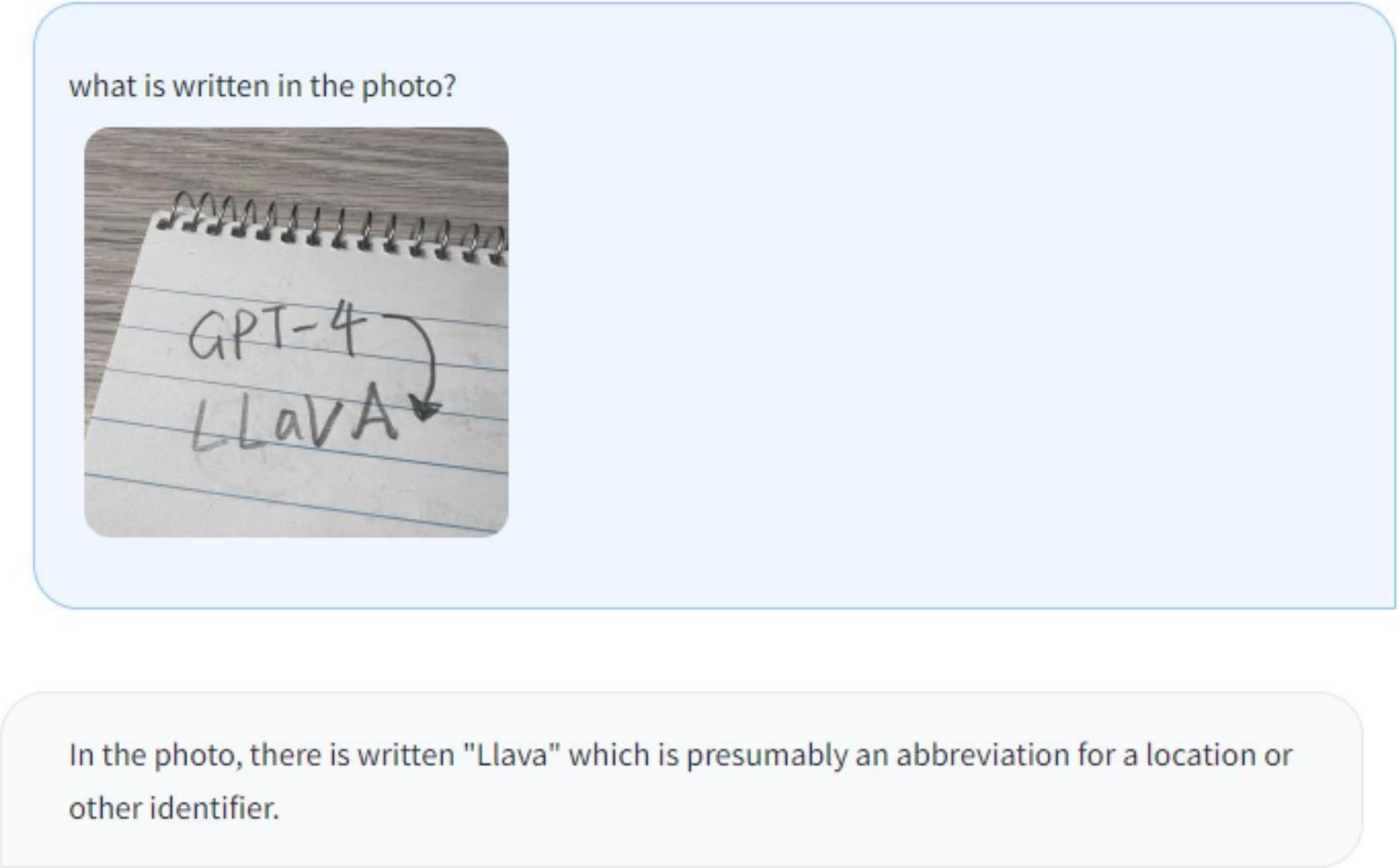
Example 2: Chicken Nugget Map



Visual input example, Chicken Nugget Map:

	Sometimes I just look at pictures of the earth from space and I marvel at how beautiful it all is.
User LLaVA	Can you explain this meme in detail? The meme in the image is a creative and humorous take on food, with a focus on chicken nuggets, as the center of the universe. The meme begins with the phrase "Sometimes I just look at pictures of the Earth from space and I marvel at how beautiful it all is..." The main image features a tray of cooked chicken nuggets, with the chicken pieces positioned to resemble the continents and islands on a world map. The meme playfully suggests that the chicken nuggets represent the Earth, and the various locations depicted in the photo are actually chicken nugget versions of different places. The punchline of the meme appears in smaller text at the bottom, reading "I mean, it's not the real Earth, but how beautiful it is all is." This lighthearted and entertaining meme combines food and humor, using the image of chicken nuggets to create a funny and imaginative representation of the world.
User GPT-4 [32]	Can you explain this meme? This meme is a joke that combines two unrelated things: pictures of the earth from space and chicken nuggets. The text of the meme suggests that the image below is a beautiful picture of the earth from space. However, the image is actually of chicken nuggets arranged to vaguely resemble a map of the world. The humor in this meme comes from the unexpected juxtaposition of the text and the image. The text sets up an expectation of a majestic image of the earth, but the image is actually something mundane and silly.
User BLIP-2	Can you explain this meme in detail? sometimes i just look at pictures of the earth from space and marvel how beautiful it is
User OpenFlamingo	Can you explain this meme in detail? It's a picture of a chicken nugget on the International Space Station.

Example 3: LLaVA



what is written in the photo?

In the photo, there is written "Llava" which is presumably an abbreviation for a location or other identifier.

THANK YOU!

Questions?