Topics:

- Vision Transformers
- Intro to VLMs

CS 8803-VLM

ZSOLT KIRA

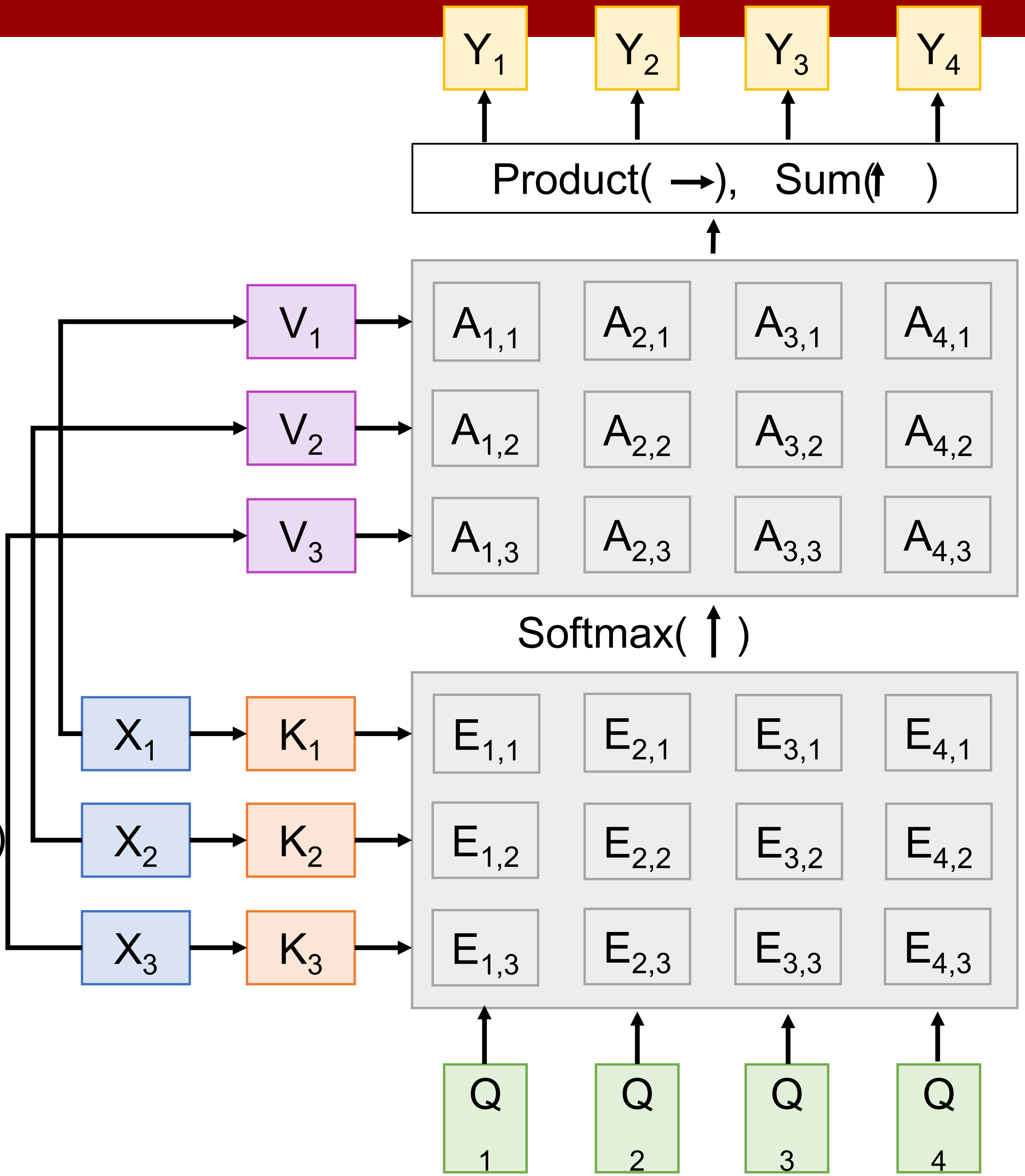
Many slides by Justin Johnson

- This week:
  - VLM background (lectures)
  - No reviews due this week
- Papers will be out today (sorry!)
  - Signup will be due this week
  - Please sign up for first ones!
- Attendance sheet being passed around

Attention Layer

Inputs:
Query vectors: $\mathbf{Q}$ (Shape: N_Q x D_Q)
Input vectors: $\mathbf{X}$ (Shape: N_X x D_X)
Key matrix: $\mathbf{W_K}$ (Shape: D_X x D_Q)
Value matrix: $\mathbf{W_V}$ (Shape: D_X x D_V)

Computation:
Key vectors: $\mathbf{K} = \mathbf{XW_K}$ (Shape: N_X x D_Q)
Value vectors: $\mathbf{V} = \mathbf{XW_V}$ (Shape: N_X x D_V)
Similarities: $E = \mathbf{QK}^T$ (Shape: N_Q x N_X) $E_{i,j} = \mathbf{Q_i} \cdot \mathbf{K_j} / \mathrm{sqrt}(D_Q)$
Attention weights: $A = \mathrm{softmax}(E, \mathrm{dim}=1)$ (Shape: N_Q x N_X)
Output vectors: $Y = A\mathbf{V}$ (Shape: N_Q x D_V) $Y_i = \sum_j A_{i,j}\mathbf{V_j}$



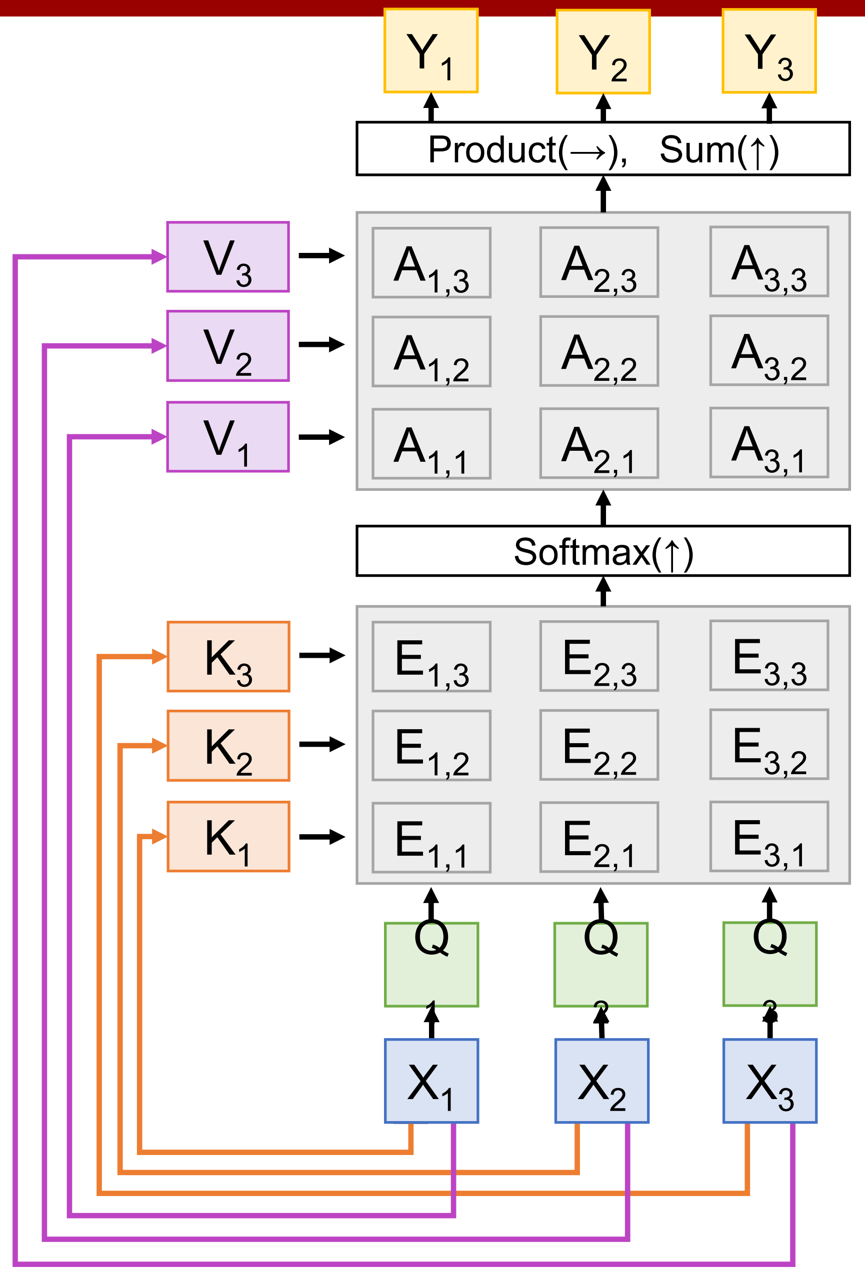
Slide credit: Justin Johnson

Self-Attention Layer

One **query** per **input vector**

Inputs:
Input vectors: $\mathbf{X}$ (Shape: N_X x D_X)
Key matrix: $\mathbf{W_K}$ (Shape: D_X x D_Q)
Value matrix: $\mathbf{W_V}$ (Shape: D_X x D_V)
Query matrix: $\mathbf{W_Q}$ (Shape: D_X x D_Q)

Computation:
Query vectors: $\mathbf{Q} = \mathbf{XW_Q}$
Key vectors: $\mathbf{K} = \mathbf{XW_K}$ (Shape: N_X x D_Q)
Value vectors: $\mathbf{V} = \mathbf{XW_V}$ (Shape: N_X x D_V)
Similarities: $E = \mathbf{QK}^T$ (Shape: N_X x N_X) $E_{i,j} = \mathbf{Q_i} \cdot \mathbf{K_j} / \mathrm{sqrt}(D_Q)$
Attention weights: $A = \mathrm{softmax}(E, \mathrm{dim}=1)$ (Shape: N_X x N_X)
Output vectors: $Y = A\mathbf{V}$ (Shape: N_X x D_V) $Y_i = \sum_j A_{i,j}\mathbf{V_j}$



Slide credit: Justin Johnson

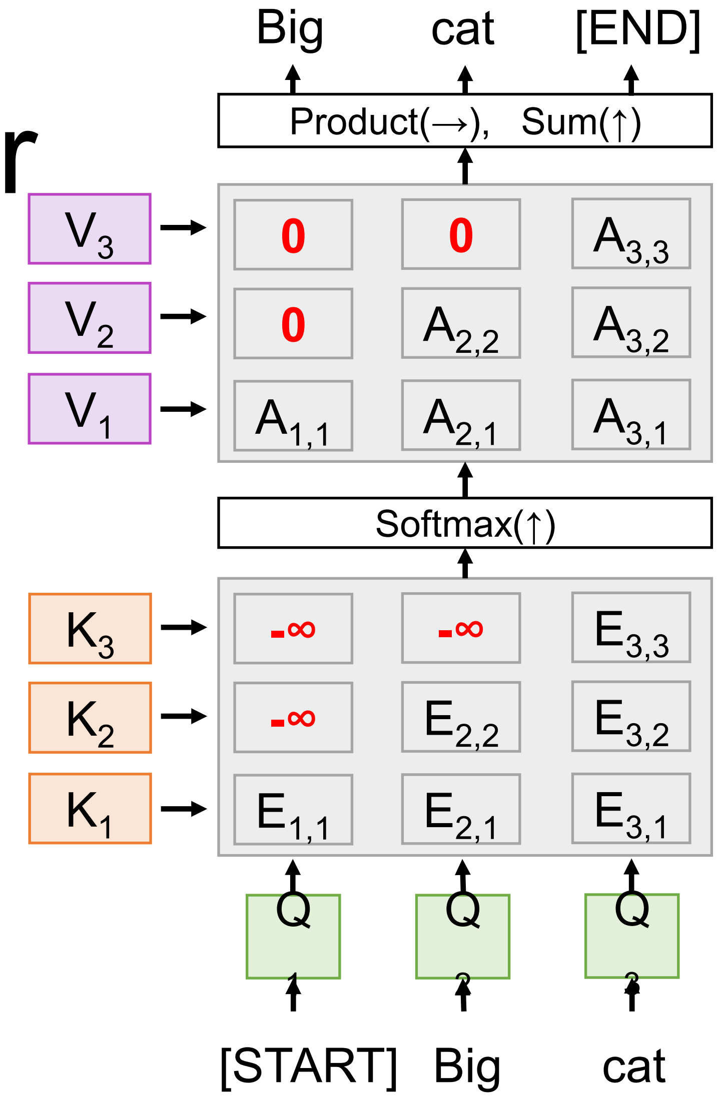
Masked Self-Attention Layer

Inputs:
Input vectors: **X** (Shape: N_X x D_X)
Key matrix: **W_K** (Shape: D_X x D_Q)
Value matrix: **W_V** (Shape: D_X x D_V)
Query matrix: **W_Q** (Shape: D_X x D_Q)

Computation:
Query vectors: **Q** = **XW_Q**
Key vectors: **K** = **XW_K** (Shape: N_X x D_Q)
Value vectors: **V** = **XW_V** (Shape: N_X x D_V)
Similarities: E = **QK^T** (Shape: N_X x N_X) $E_{i,j} = Q_i \cdot K_j / \mathrm{sqrt}(D_Q)$
Attention weights: A = softmax(E, dim=1) (Shape: N_X x N_X)
Output vectors: Y = AV (Shape: N_X x D_V) $Y_i = \sum_j A_{i,j} V_j$

Don't let vectors "look ahead" in the sequence

Used for language modeling (predict next word)



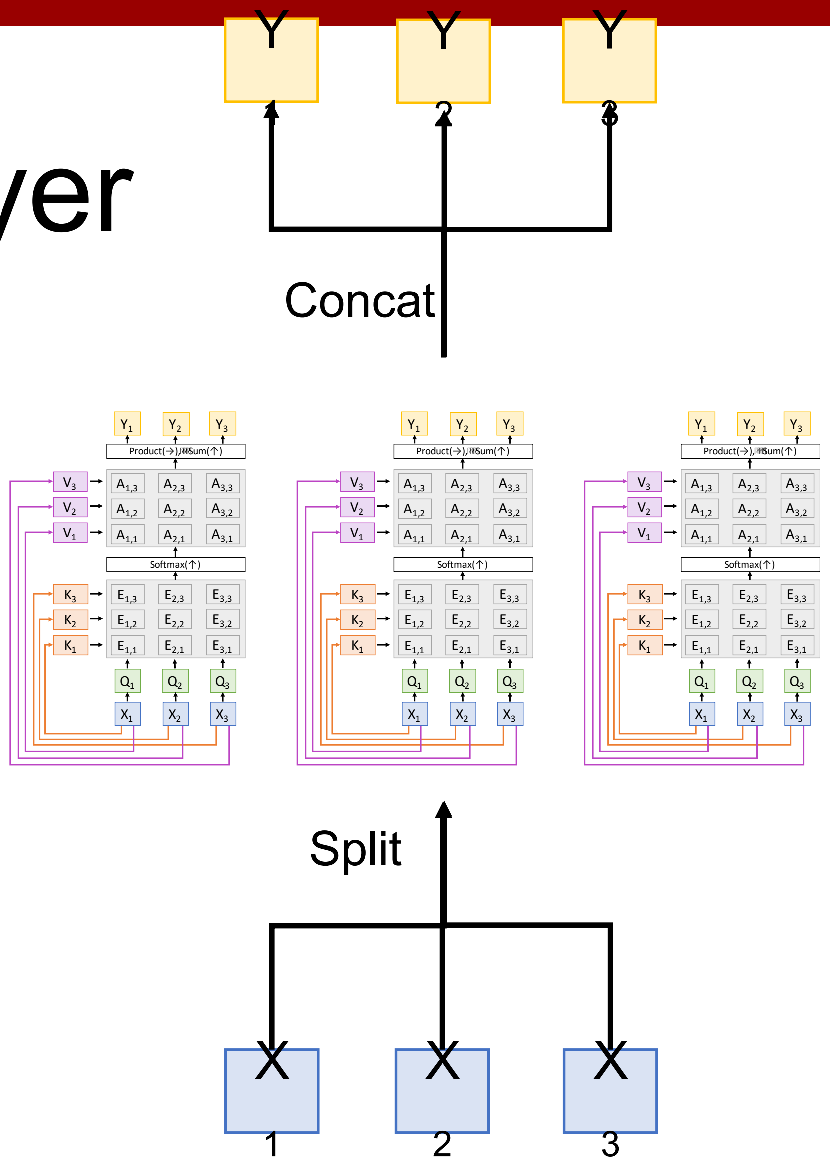
Slide credit: Justin Johnson

Multihead Self-Attention Layer

Inputs:
Input vectors: $\mathbf{X}$ (Shape: $N_X \times D_X$)
Key matrix: $\mathbf{W_K}$ (Shape: $D_X \times D_Q$)
Value matrix: $\mathbf{W_V}$ (Shape: $D_X \times D_V$)
Query matrix: $\mathbf{W_Q}$ (Shape: $D_X \times D_Q$)

Computation:
Query vectors: $\mathbf{Q} = \mathbf{XW_Q}$
Key vectors: $\mathbf{K} = \mathbf{XW_K}$ (Shape: $N_X \times D_Q$)
Value vectors: $\mathbf{V} = \mathbf{XW_V}$ (Shape: $N_X \times D_V$)
Similarities: $E = \mathbf{QK}^T$ (Shape: $N_X \times N_X$) $E_{i,j} = \mathbf{Q_i} \cdot \mathbf{K_j} / \mathrm{sqrt}(D_Q)$
Attention weights: $A = \mathrm{softmax}(E, \mathrm{dim}=1)$ (Shape: $N_X \times N_X$)
Output vectors: $Y = A\mathbf{V}$ (Shape: $N_X \times D_V$) $Y_i = \sum_j A_{i,j} \mathbf{V_j}$

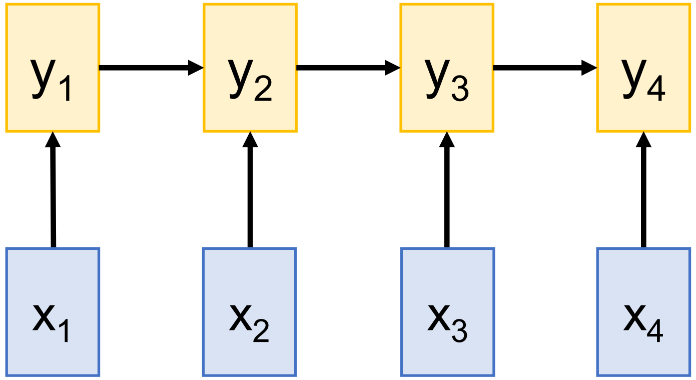
Use H independent "Attention Heads" in parallel



Slide credit: Justin Johnson

Three Ways of Processing Sequences

Recurrent Neural Network

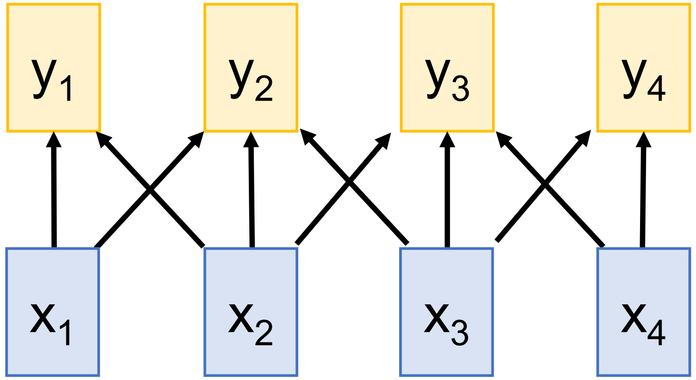


Works on **Ordered Sequences**

(+) Good at long sequences: After one RNN layer, h_T "sees" the whole sequence

(-) Not parallelizable: need to compute hidden states sequentially

1D Convolution

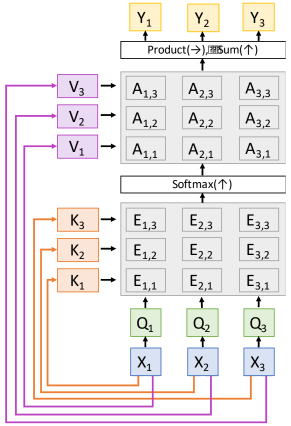


Works on **Multidimensional Grids**

(-) Bad at long sequences: Need to stack many conv layers for outputs to "see" the whole sequence

(+) Highly parallel: Each output can be computed in parallel

Self-Attention



Works on **Sets of Vectors**

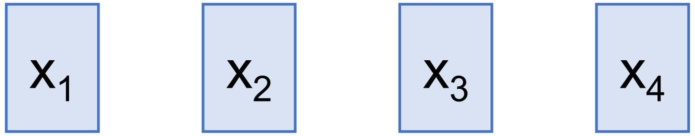
(+) Good at long sequences: after one self-attention layer, each output "sees" all inputs!

(+) Highly parallel: Each output can be computed in parallel

(-) Very memory intensive

Slide credit: Justin Johnson

The Transformer

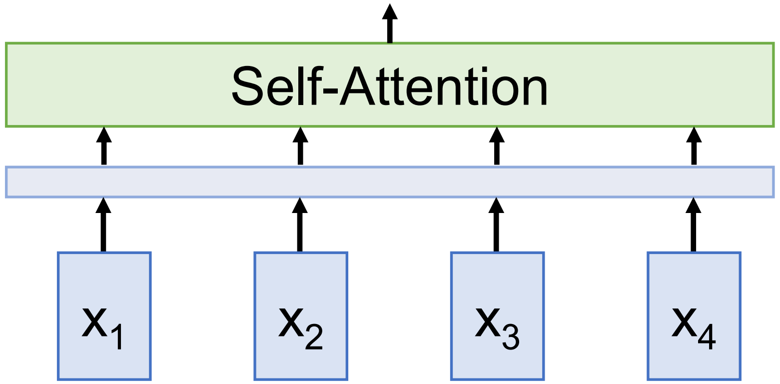


Vaswani et al, “Attention is all you need”, NeurIPS 2017

Slide credit: Justin Johnson

The Transformer

All vectors interact
with each other



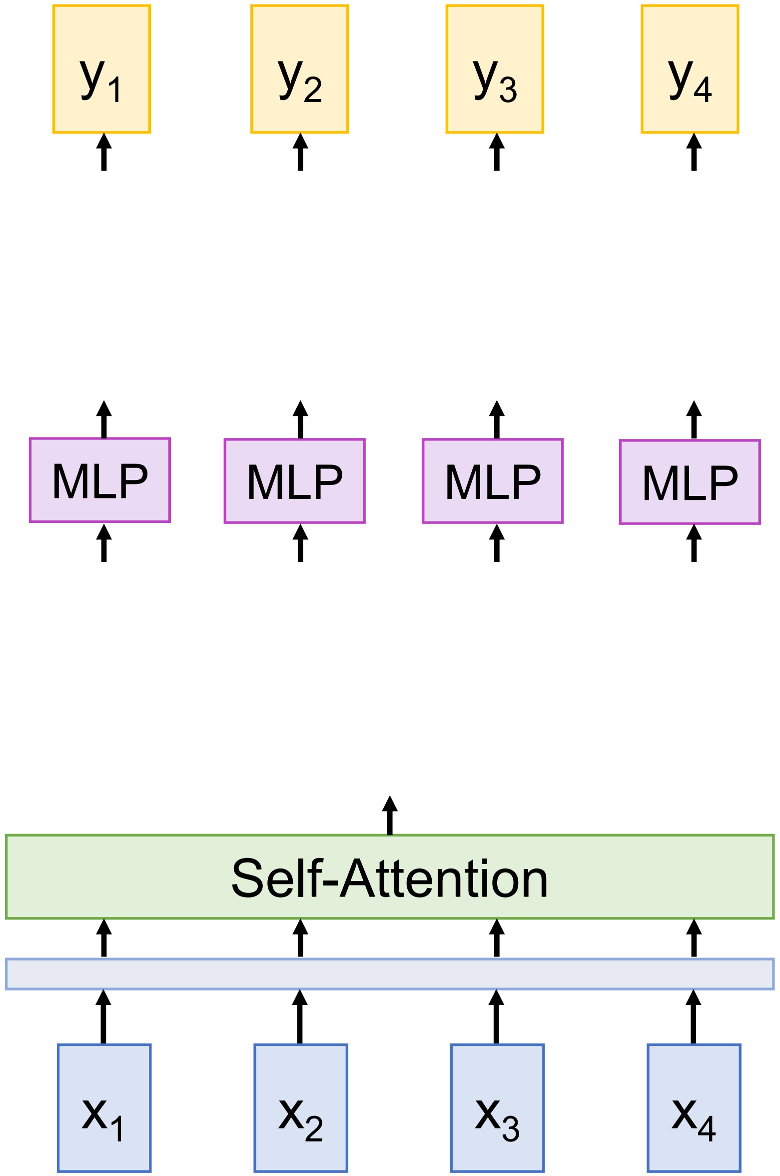
Vaswani et al, “Attention is all you need”, NeurIPS 2017

Slide credit: Justin Johnson

The Transformer

MLP independently on each vector **(weight shared!)**

All vectors interact with each other



Vaswani et al, "Attention is all you need", NeurIPS 2017

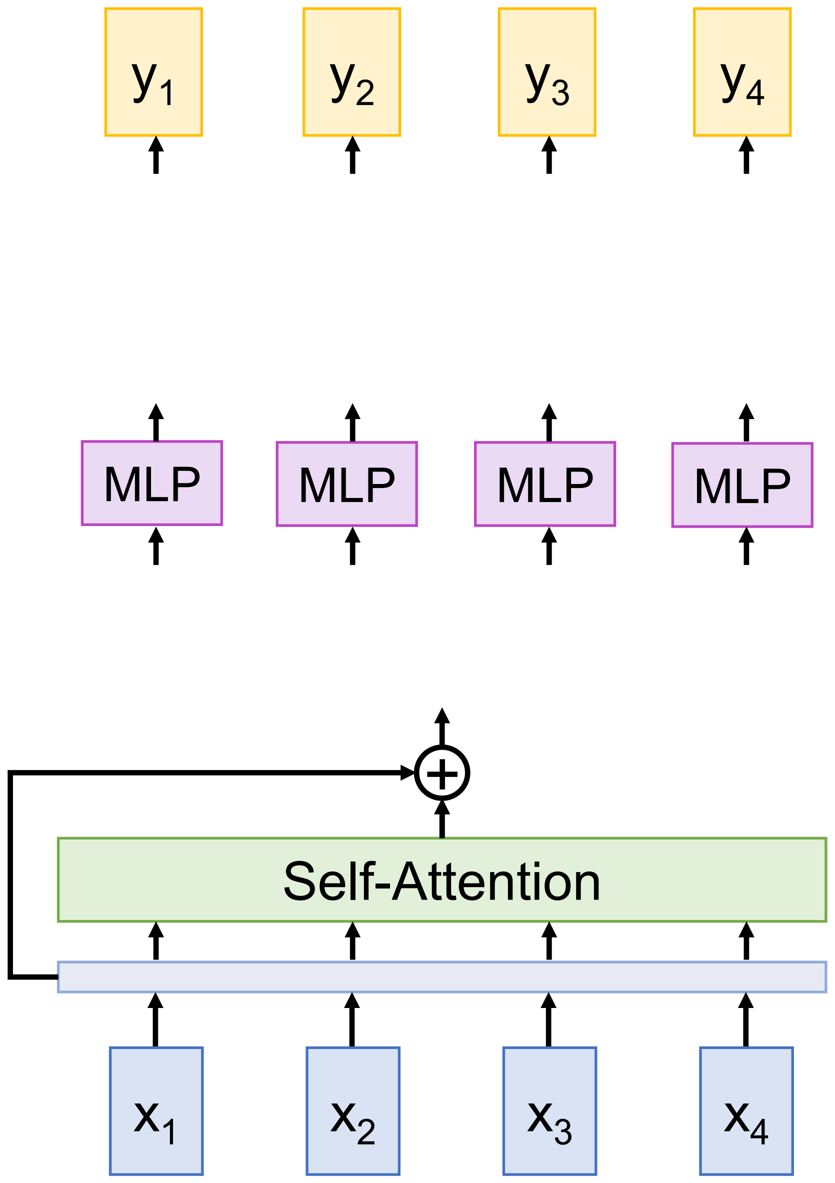
Slide credit: Justin Johnson

The Transformer

MLP independently on each vector

Residual connection

All vectors interact with each other



Vaswani et al, “Attention is all you need”, NeurIPS 2017

Slide credit: Justin Johnson

The Transformer

Recall **Layer Normalization**:

Given $h_1, \ldots, h_N$ (Shape: D)

scale: γ (Shape: D)

shift: β (Shape: D)

$\mu_i = (1/D)\sum_j h_{i,j}$ (scalar)

$\sigma_i = (\sum_j (h_{i,j} - \mu_i)^2)^{1/2}$ (scalar)

$z_i = (h_i - \mu_i) / \sigma_i$

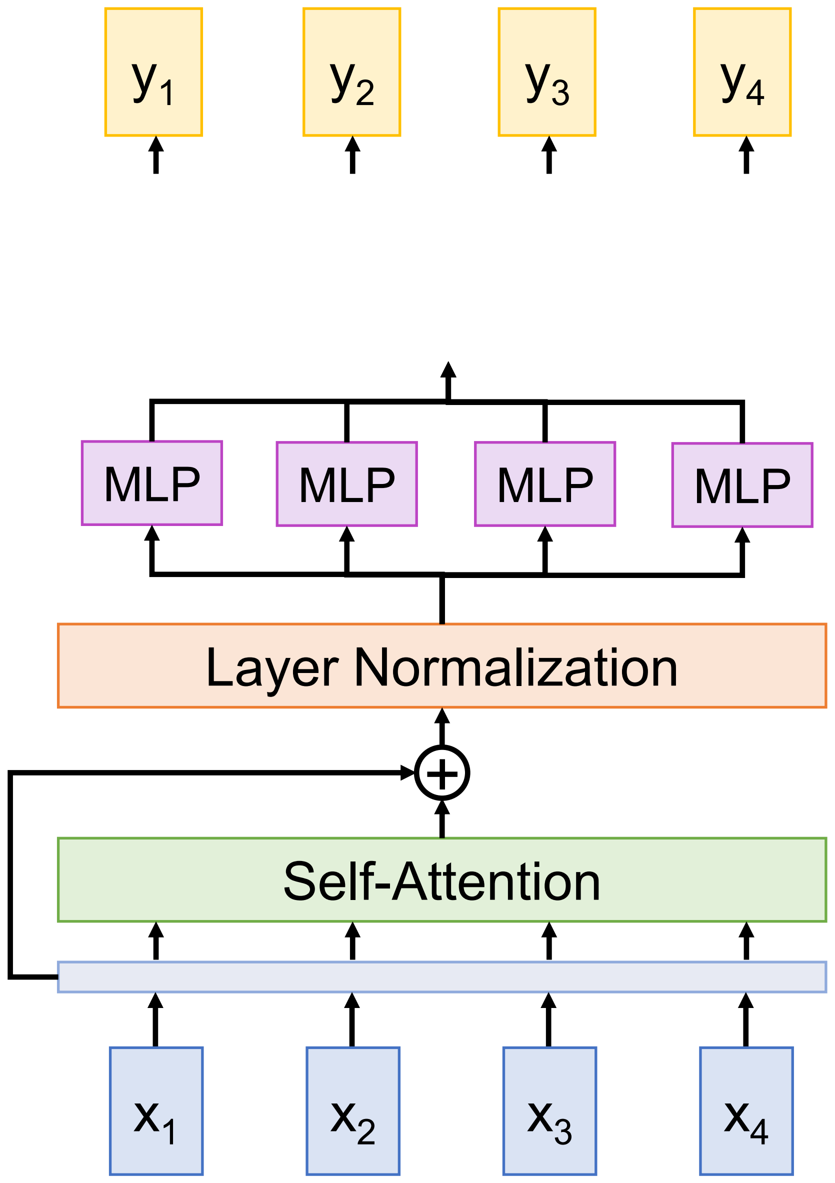
$y_i = \gamma * z_i + \beta$

Ba et al, 2016

MLP independently on each vector

Residual connection

All vectors interact with each other



Vaswani et al, "Attention is all you need", NeurIPS 2017

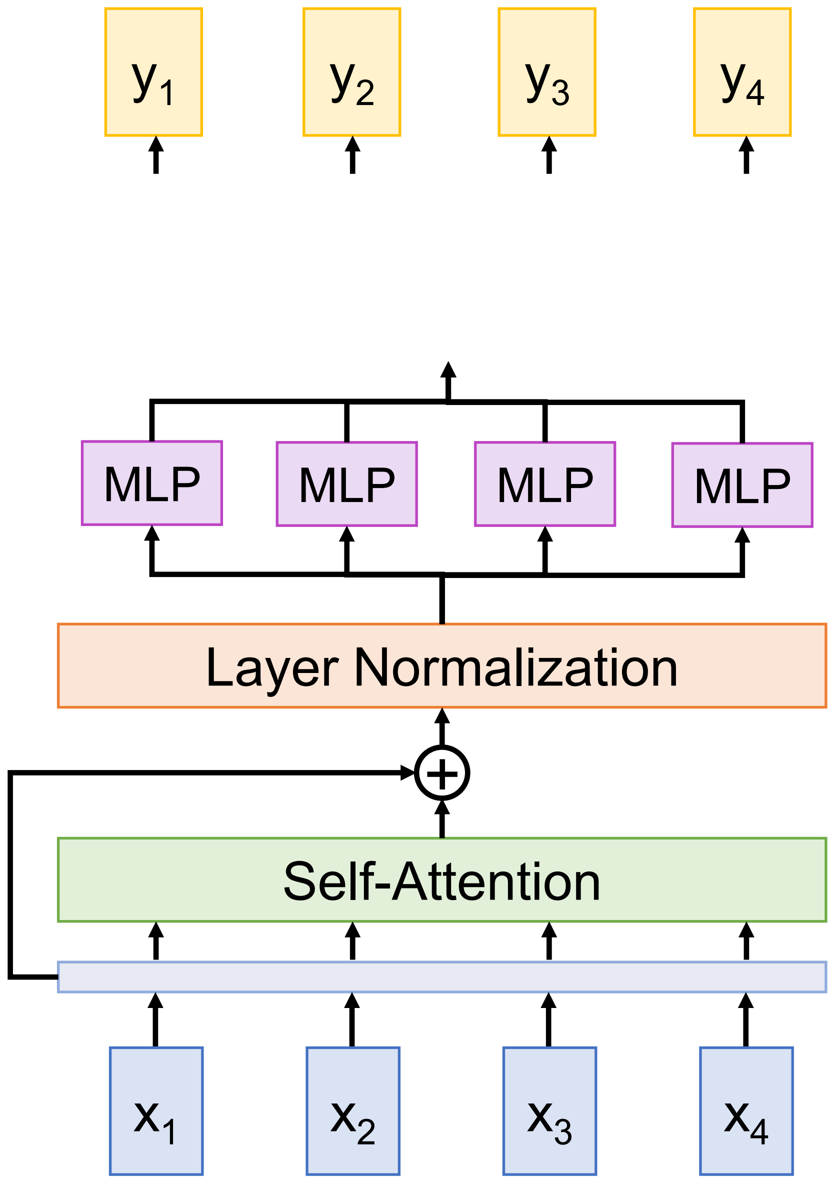
Slide credit: Justin Johnson

The Transformer

MLP independently on each vector

Residual connection

All vectors interact with each other



Vaswani et al, "Attention is all you need", NeurIPS 2017

Slide credit: Justin Johnson

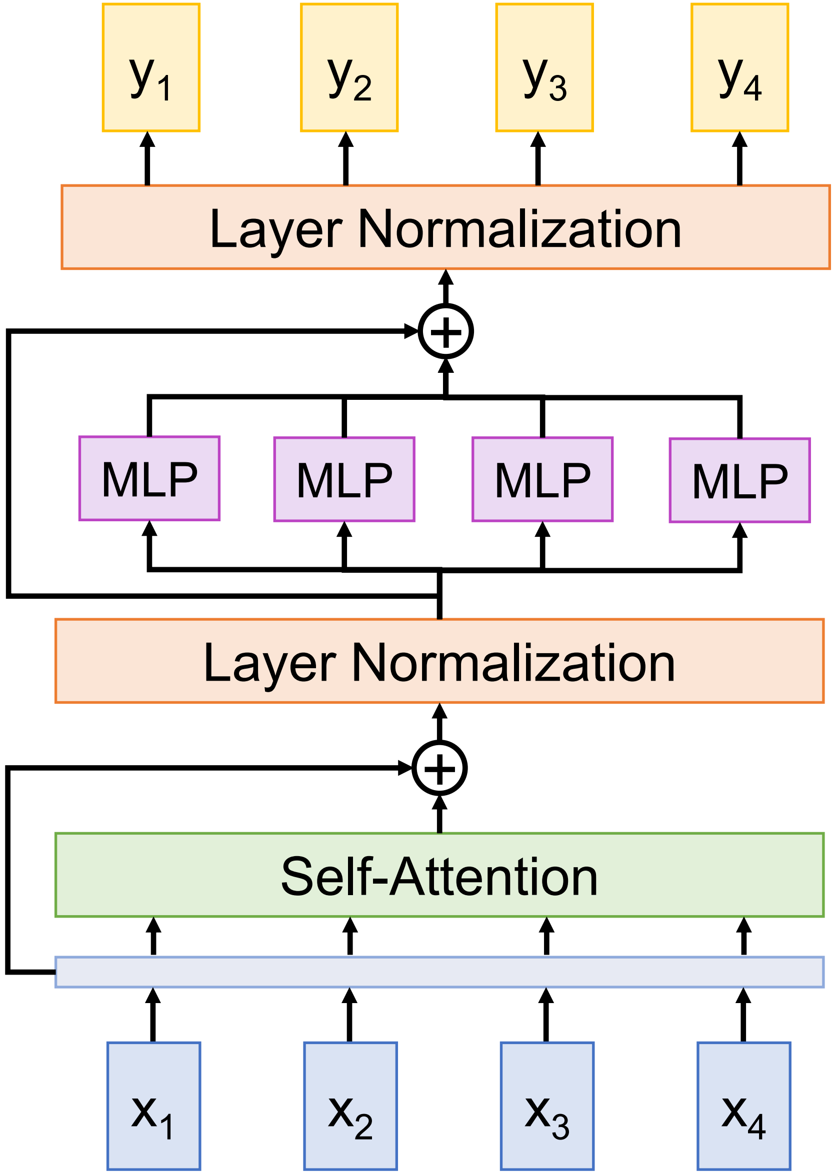
The Transformer

Residual connection

MLP independently on each vector

Residual connection

All vectors interact with each other



Vaswani et al, “Attention is all you need”, NeurIPS 2017

Slide credit: Justin Johnson

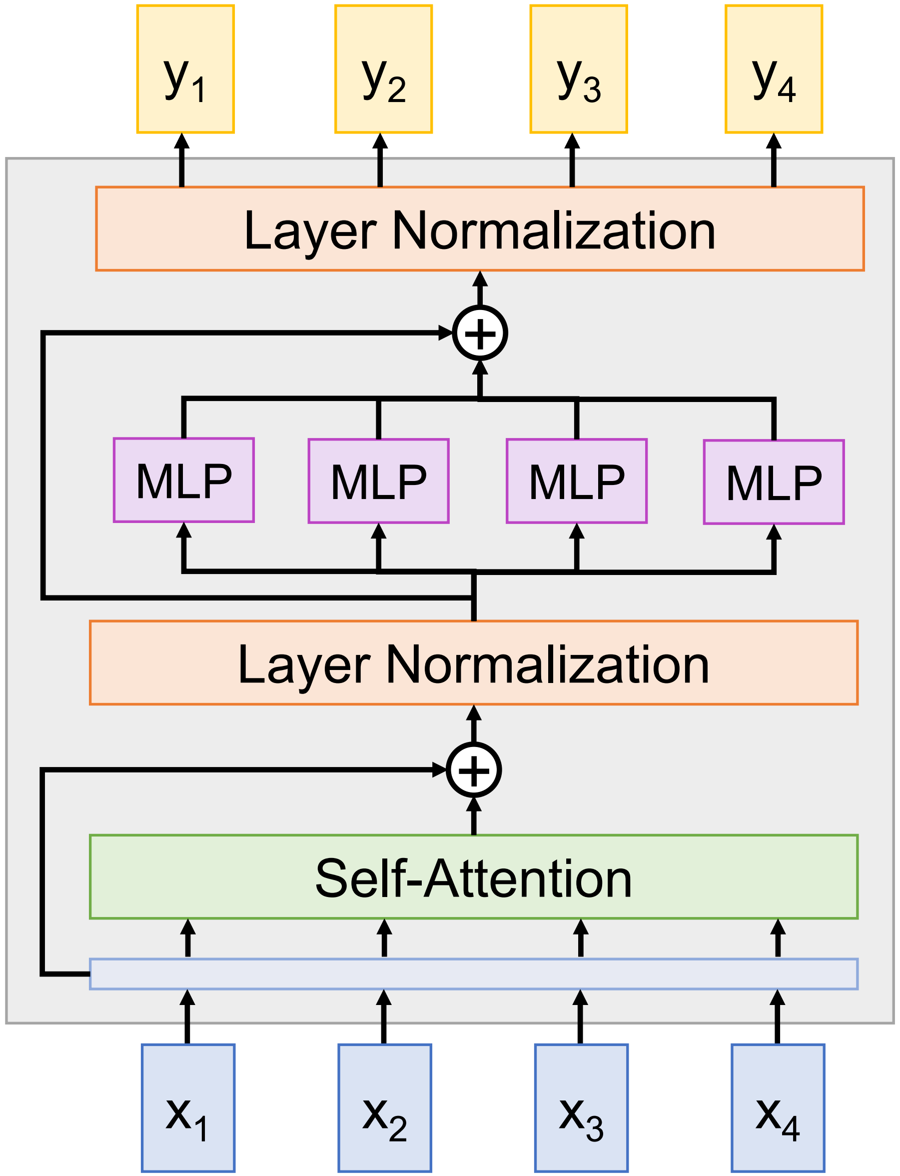
The Transformer

<u>**Transformer Block**</u>:
Input: Set of vectors x
Output: Set of vectors y

Self-attention is the only interaction between vectors!

Layer norm and MLP work independently per vector

Highly scalable, highly parallelizable



Vaswani et al, "Attention is all you need", NeurIPS 2017

Slide credit: Justin Johnson

The Transformer

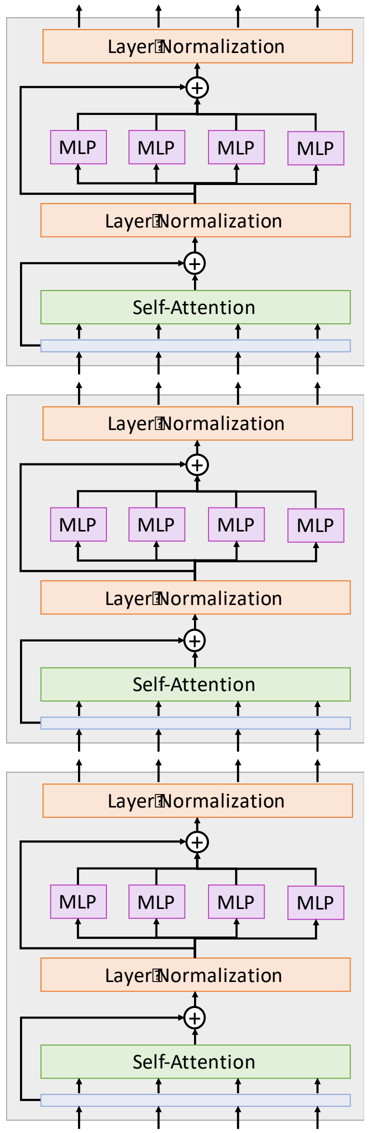
<u>Transformer Block</u>:
Input: Set of vectors x
Output: Set of vectors y

Self-attention is the only interaction between vectors!

Layer norm and MLP work independently per vector

Highly scalable, highly parallelizable

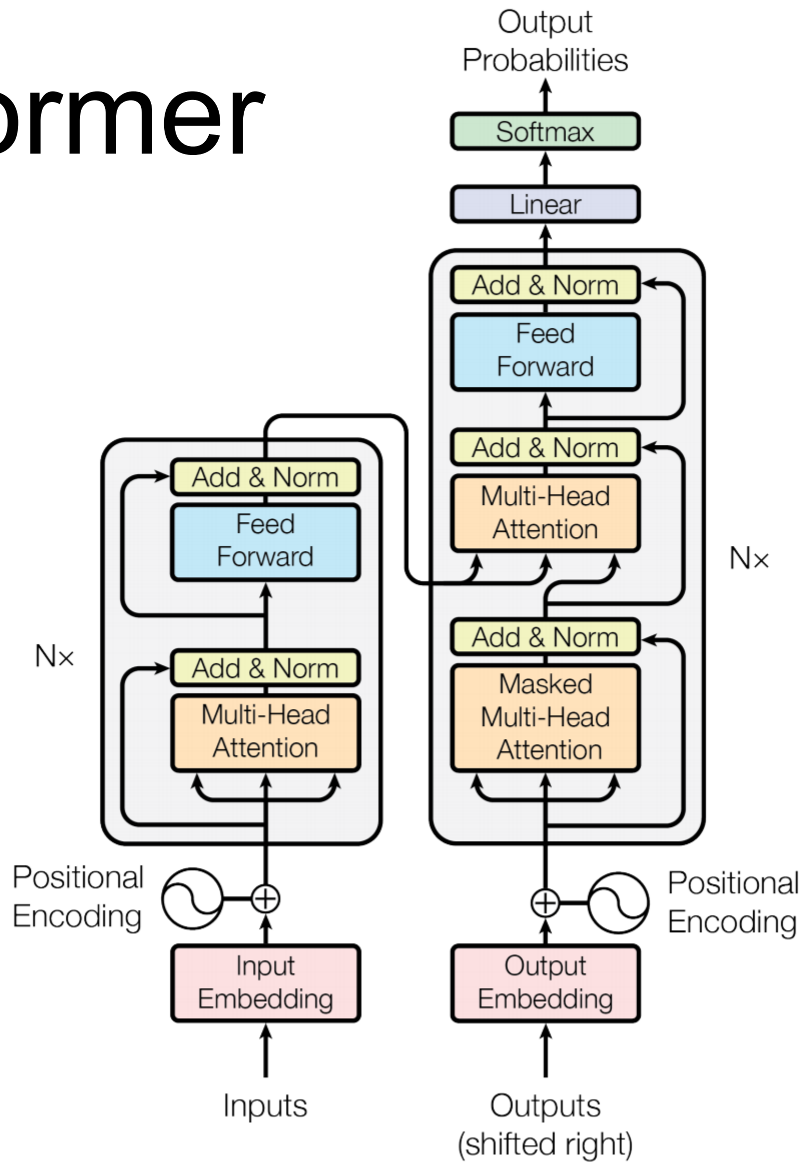
A **Transformer** is a sequence of transformer blocks



Vaswani et al, “Attention is all you need”, NeurIPS 2017

Slide credit: Justin Johnson

The Transformer



Encoder-Decoder

Details:

- Tokenization is messy! Trained chunking mechanism
- Position encoding
  - sin/cos: Normalized, nearby tokens have similar values, etc.
  - Added to input embedding
- When to use decoder-only versus encoder-decoder model is open problem
  - GPT is decoder only!

Vaswani et al, “Attention is all you need”, NeurIPS 2017

- Language models estimate the probability of sequences of words:

$$p(\boldsymbol{s}) = p(w_1, w_2, \ldots, w_n)$$

- ***Another task: Masked language modeling*** is a related ***pre-training task*** – an auxiliary task, different from the final task we're really interested in, but which can help us achieve better performance by finding good initial parameters for the model.

- By pre-training on masked language modeling before training on our final task, it is usually possible to obtain higher performance than by simply training on the final task.

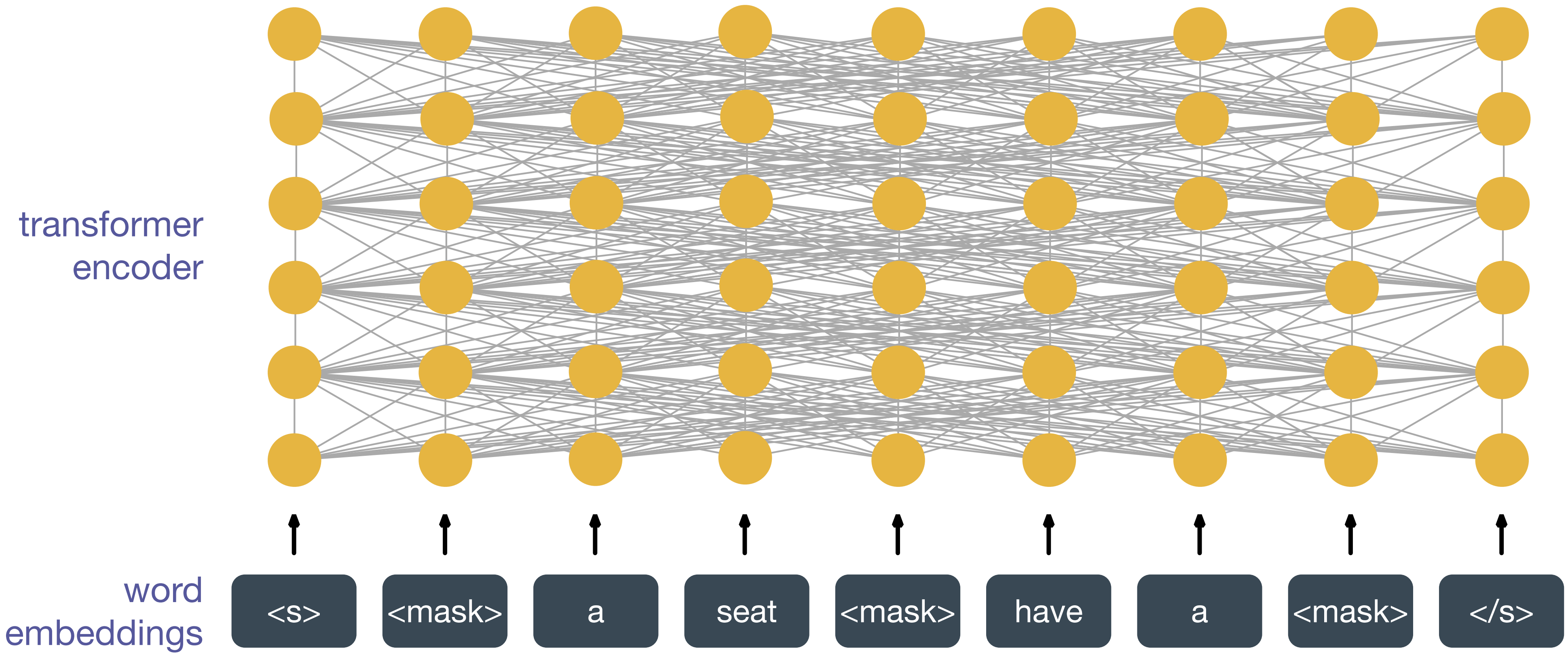
take | a | seat | , | have | a | drink

Masked Language Models

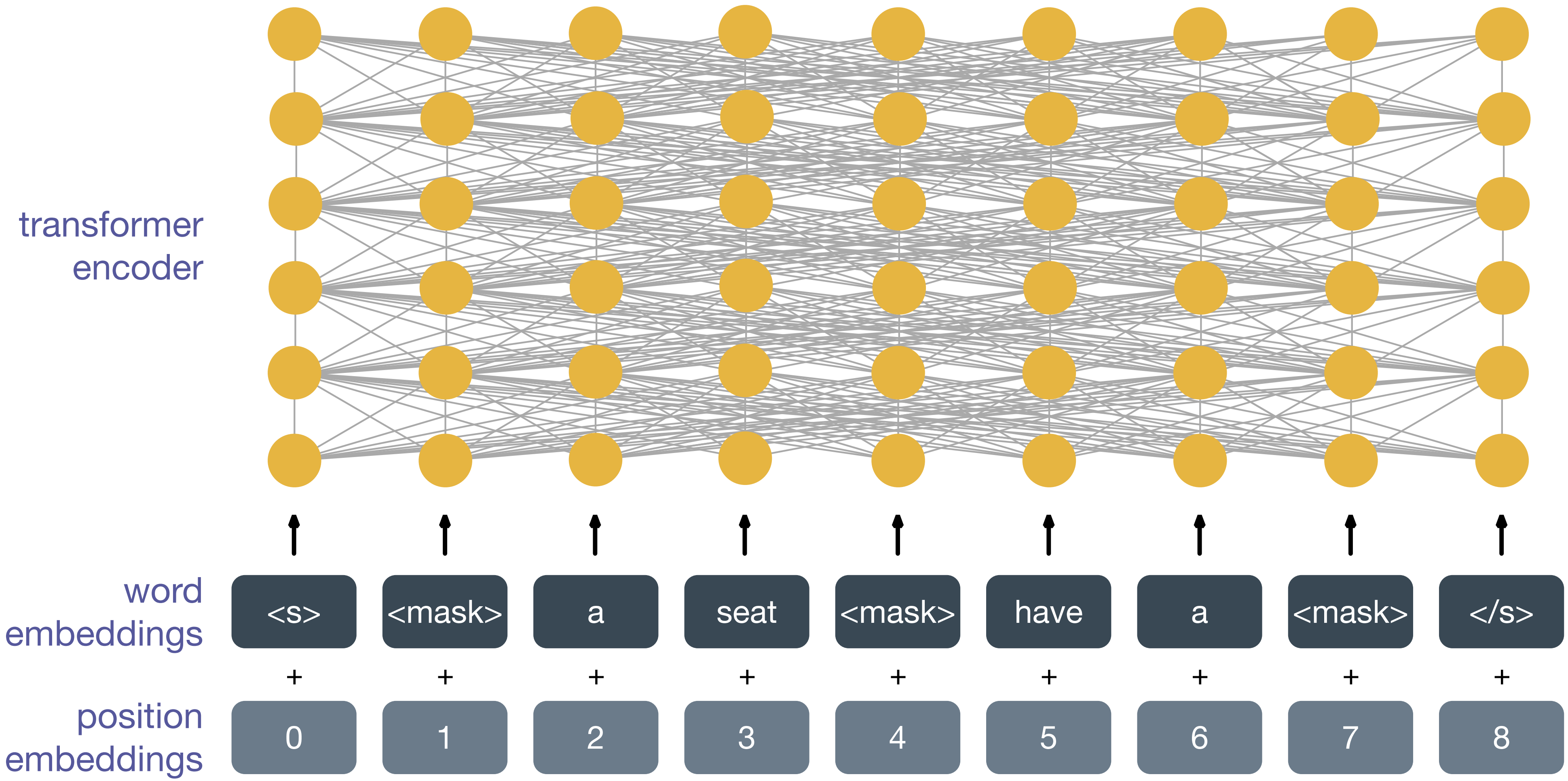
FACEBOOK AI

<s> <mask> a seat <mask> have a <mask> </s>

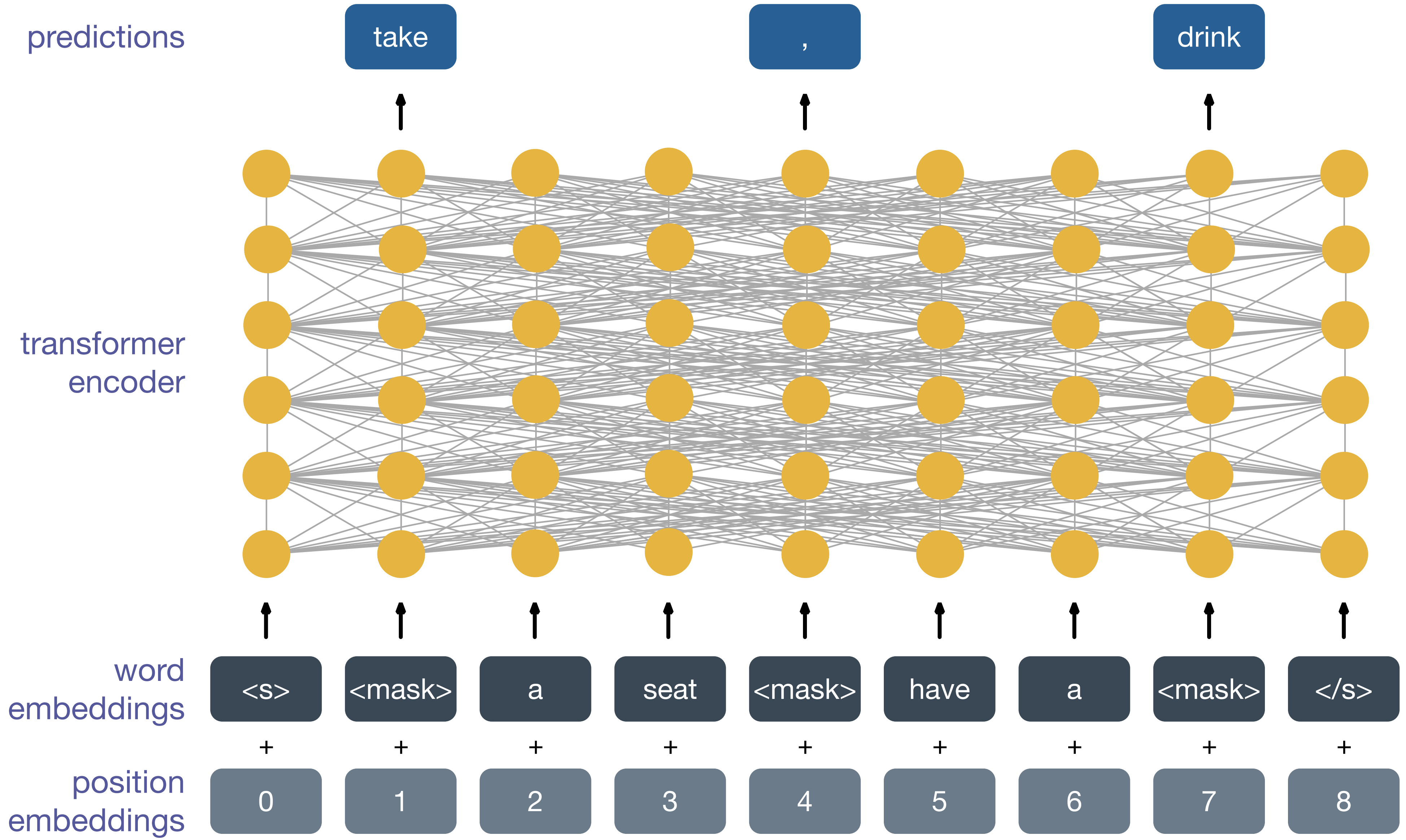
Masked Language Models



Masked Language Models



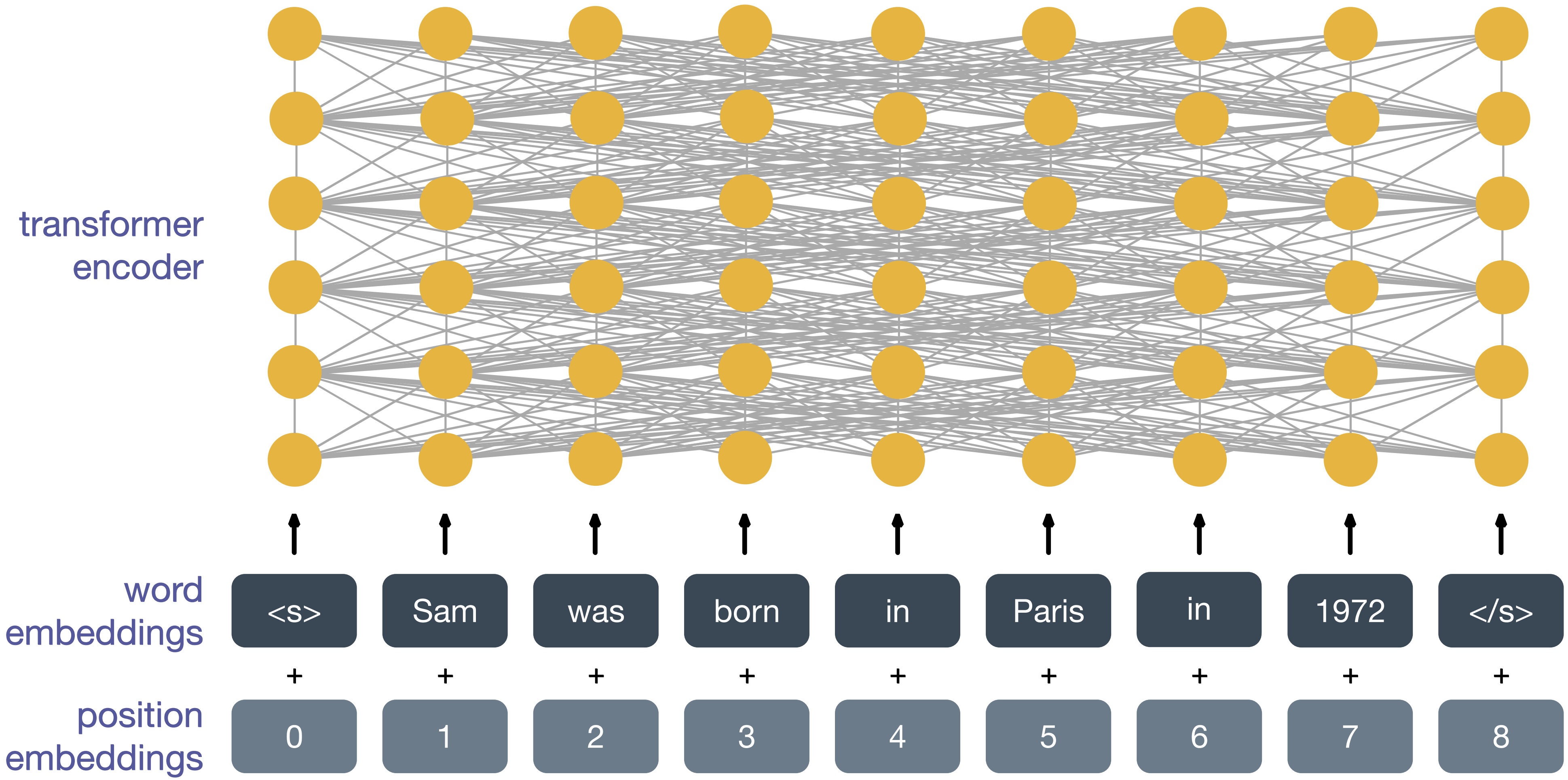
Masked Language Models



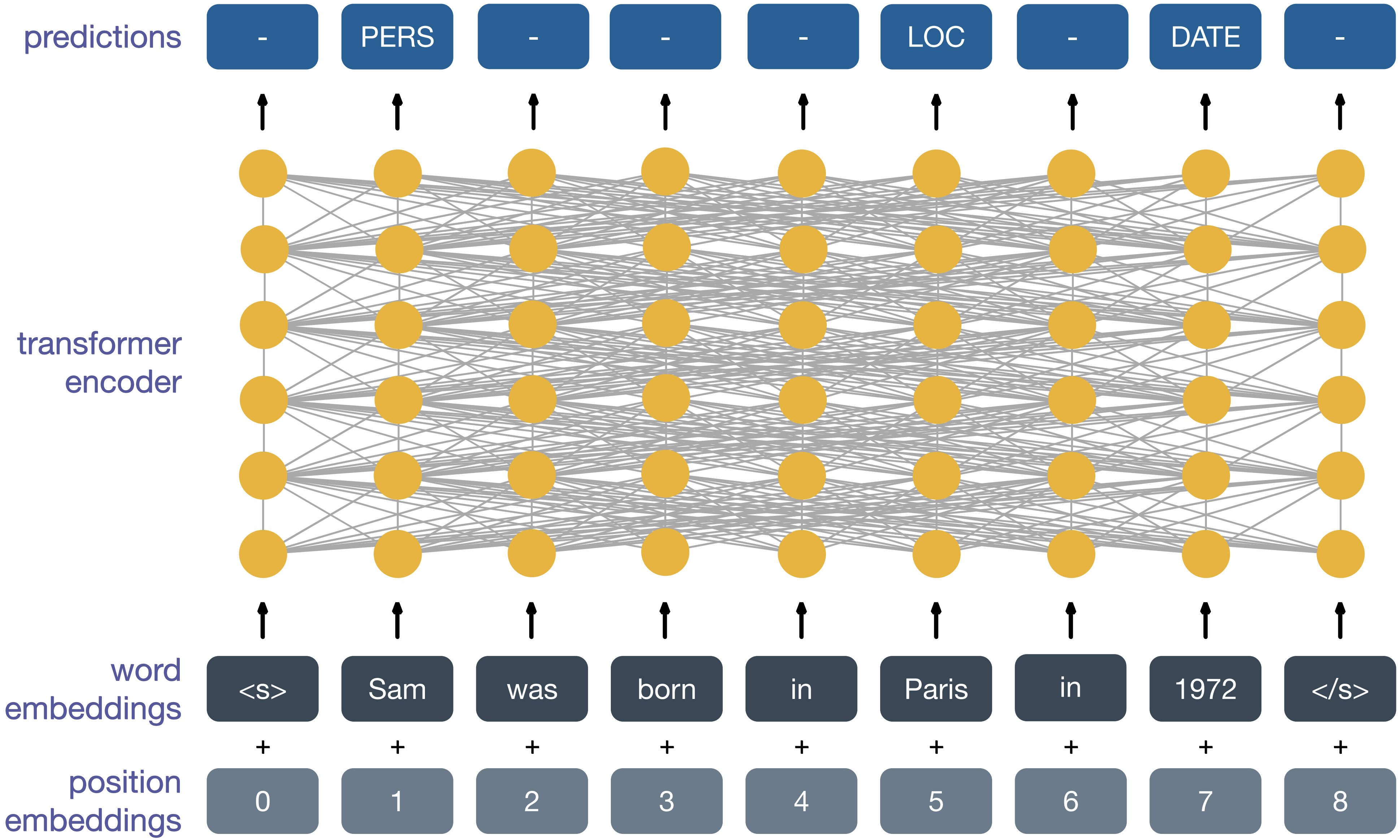
Masked Language Models

FACEBOOK AI

Georgia Tech

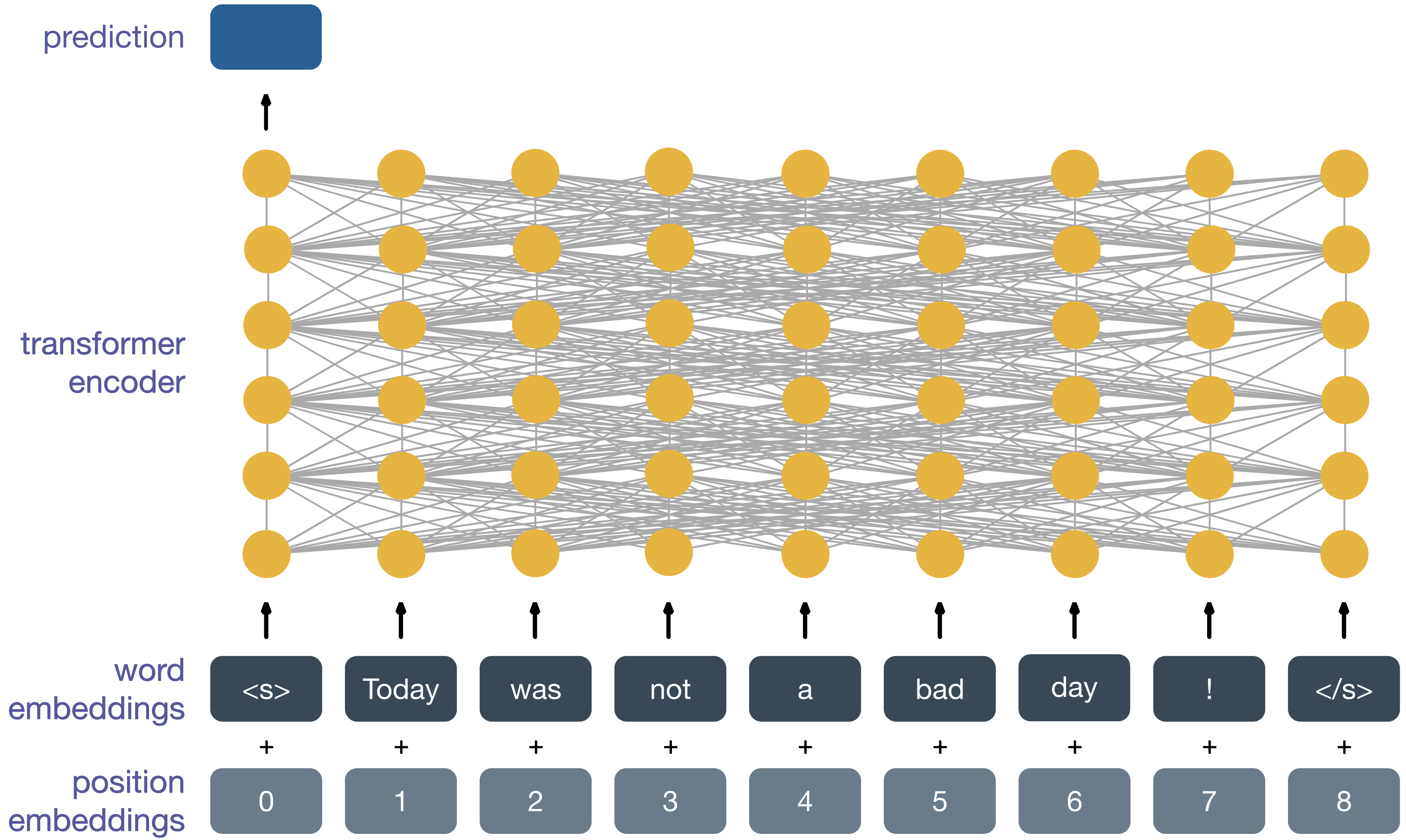


Token-level Tasks

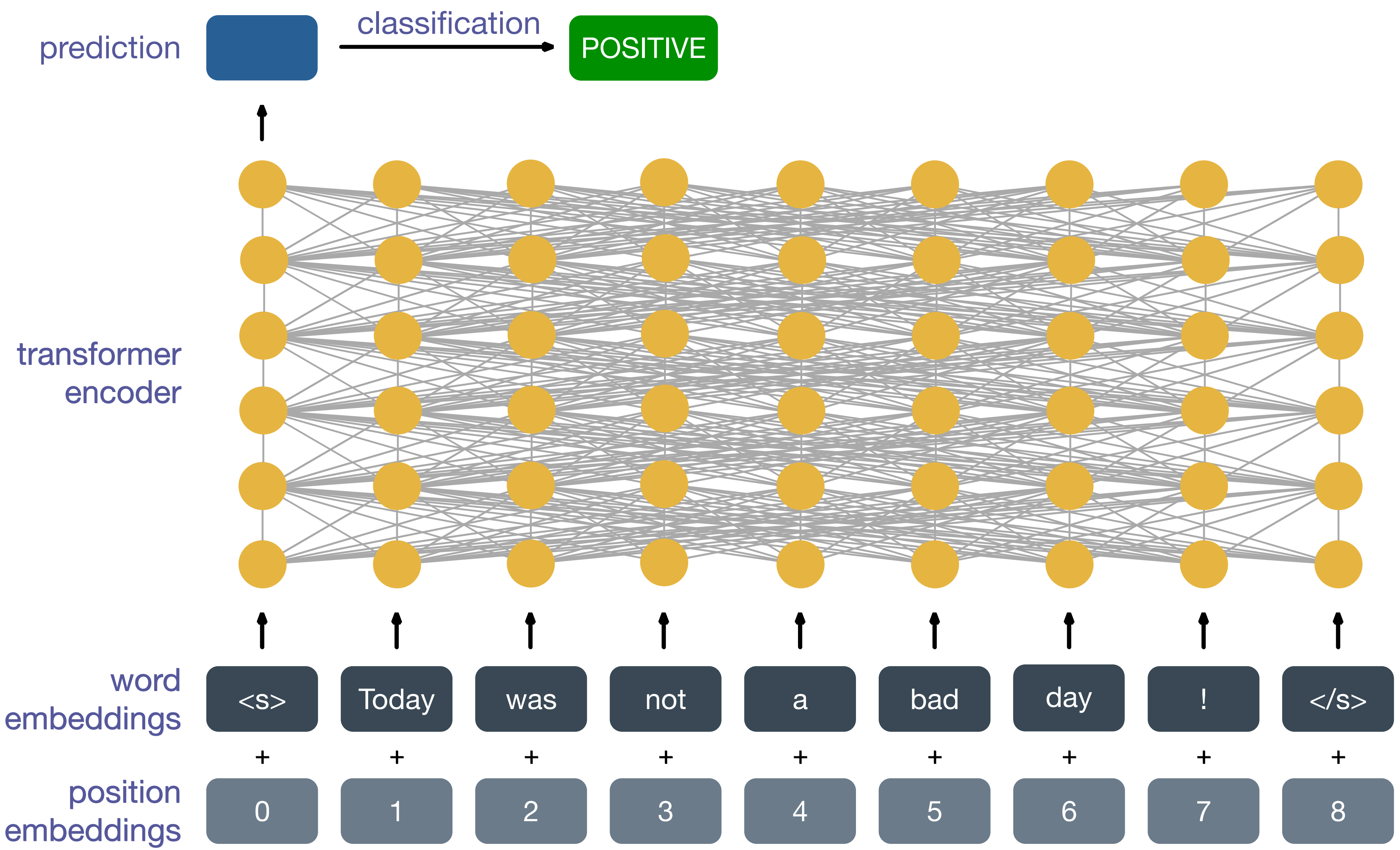


Token-level Tasks

FACEBOOK AI Georgia Tech



Sentence-level Tasks

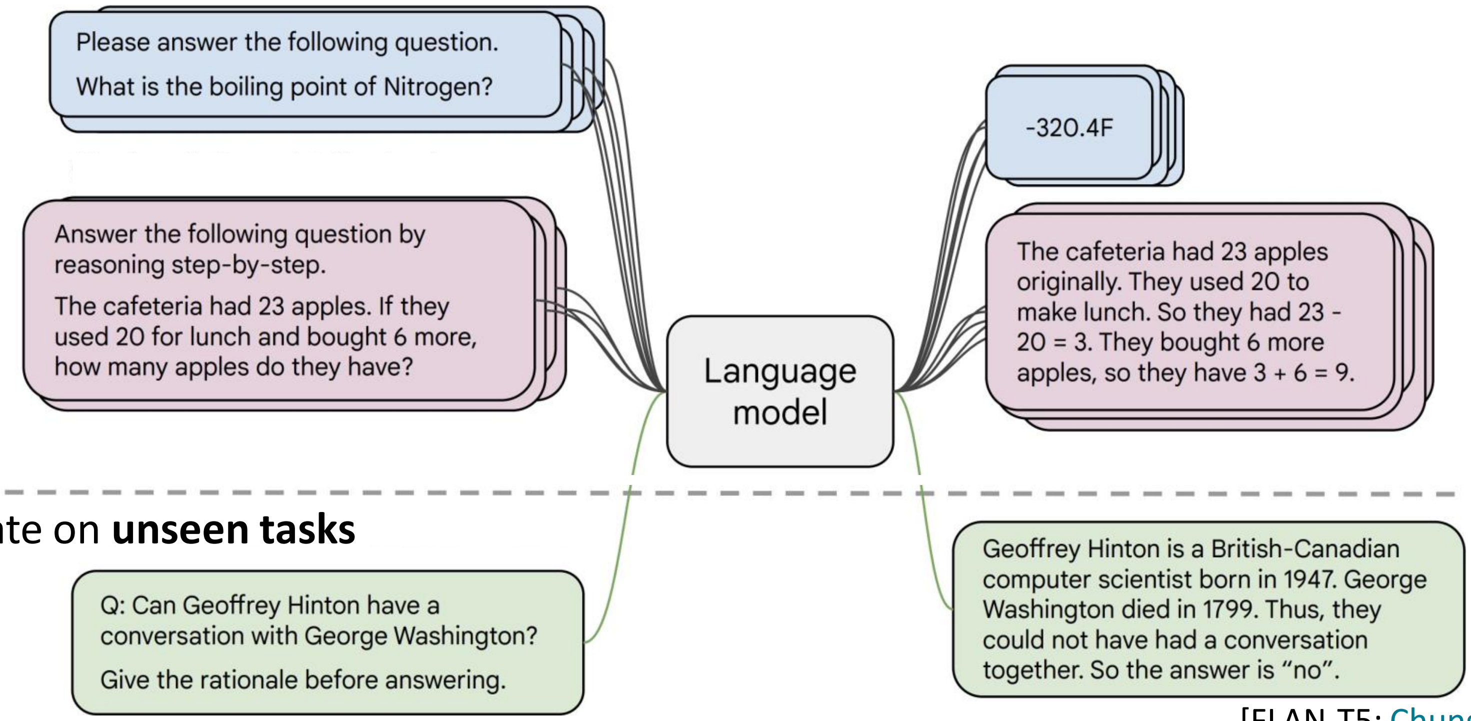


Sentence-level Tasks

Training Stages

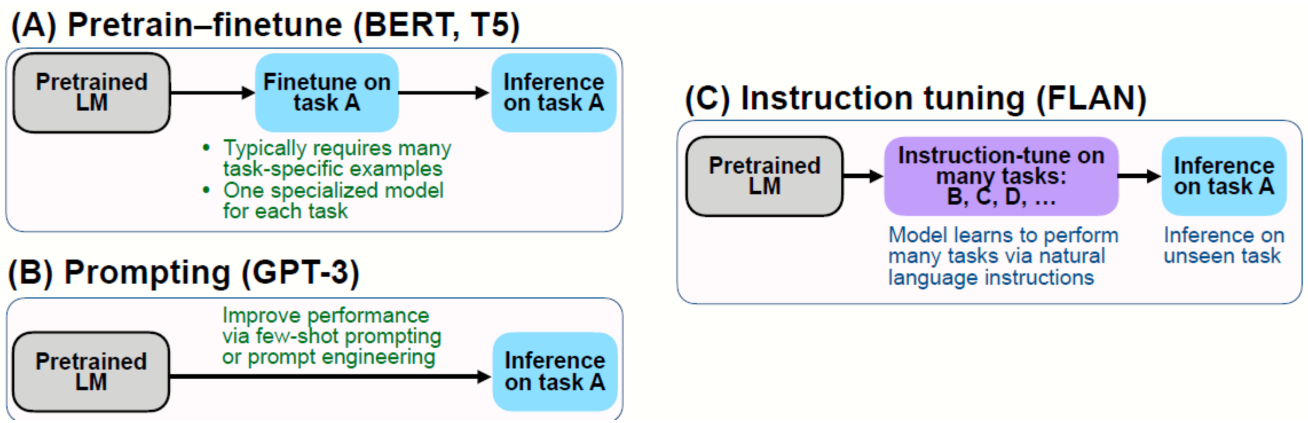
- **Collect examples** of (instruction, output) pairs across many tasks and finetune an LM

1. Pre-training
2. Instruction Tuning
3. Alignment Tuning
4. Post-Training



- Evaluate on **unseen tasks**

[FLAN-T5; Chung et al., 2022



Yin et al., A Survey on Multimodal Large Language Models

Pre-training

- Goal: Align modalities & learn multimodal knowledge
  - Data: Large-scale image-text pairs
- Example datasets
  - CC3M, LAION-5B, COYO-700M
  - Trend: GPT-4V for high-quality fine-grained data

Dataset	Samples	Date
Coarse-grained Image-Text		
CC-3M [84]	3.3M	2018
CC-12M [85]	12.4M	2020
SBU Captions [86]	1M	2011
LAION-5B [87]	5.9B	Mar-2022
LAION-2B [87]	2.3B	Mar-2022
LAION-COCO [88]	600M	Sep-2022
COYO-700M [90]	747M	Aug-2022
Fine-grained Image-Text		
ShareGPT4V-PT [83]	1.2M	Nov-2023
LVIS-Instruct4V [91]	111K	Nov-2023
ALLaVA [92]	709K	Feb-2024
Video-Text		
MSR-VTT [93]	200K	2016
Audio-Text		
WavCaps [94]	24K	Mar-2023

Yin et al., A Survey on Multimodal Large Language Models

Instruction Tuning

- Goal: Teach models to follow multimodal instructions
  - Data collection methods:
    1. Adapting existing datasets
    2. Self-instruction: LLM expands instructions
    3. Mixing language-only and multimodal data
  - LLaVA-instruct: Bounding boxes/caption -> GPT4 -> more data
- Data quality is important!

Below is an instruction that describes a task. Write a response that appropriately completes the request

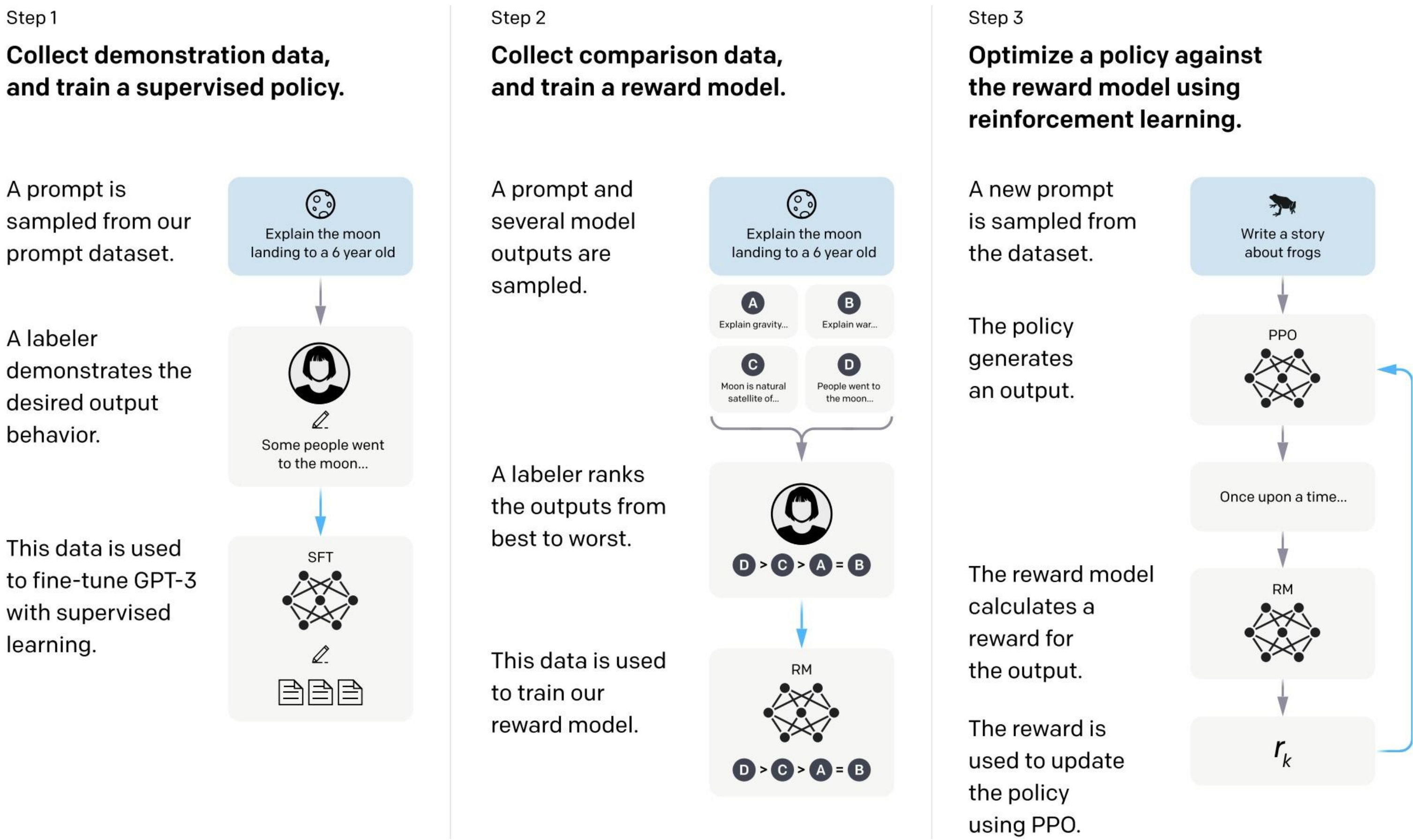
Instruction: <instruction>
Input: {<image>, <text>}
Response: <output>

Dataset	Sample	Modality	Source	Composition
LLaVA-Instruct	158K	I + T → T	MS-COCO	23K caption + 58K M-T QA + 77K reasoning
LVIS-Instruct	220K	I + T → T	LVIS	110K caption + 110K M-T QA
ALLaVA	1.4M	I + T → T	VFlan, LAION	709K caption + 709K S-T QA
Video-ChatGPT	100K	V + T → T	ActivityNet	7K description + 4K M-T QA
VideoChat	11K	V+T → T	WebVid	description + summarization + creation
Clotho-Detail	3.9K	A + T → T	Clotho	caption

Yin et al., A Survey on Multimodal Large Language Models

Alignment Tuning

- Goal: Align outputs with human preferences
  - Techniques:
  - RLHF (Reinforcement Learning with Human Feedback)
  - DPO (Direct Preference Optimization)
  - Key papers: LLaVA-RLHF, RLHF-V, Silkie (uses GPT4-V)



[Ouyang et al., 2022]

Yin et al., A Survey on Multimodal Large Language Models

Model	Layers	Width	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hours)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)
BERT-Base	12	768	12	110M	13 GB	
BERT-Large	24	1024	16	340M	13 GB	
XLNet-Large	24	1024	16	~340M	126 GB	512x TPU-v3 (2.5 days)
RoBERTa	24	1024	16	355M	160 GB	1024x V100 GPU (1 day)
GPT-2	48	1600	?	1.5B	40 GB	
Megatron-LM	72	3072	32	8.3B	174 GB	512x V100 GPU (9 days)
Turing-NLG	78	4256	28	17B	?	256x V100 GPU
GPT-3	96	12,288	96	175B	694GB	?
Gopher	80	16,384	128	280B	10.55 TB	4096x TPUv3 (38 days)

Can Attention/Transformers be used from more than text processing?

ViLBERT: A Visolinguistic Transformer

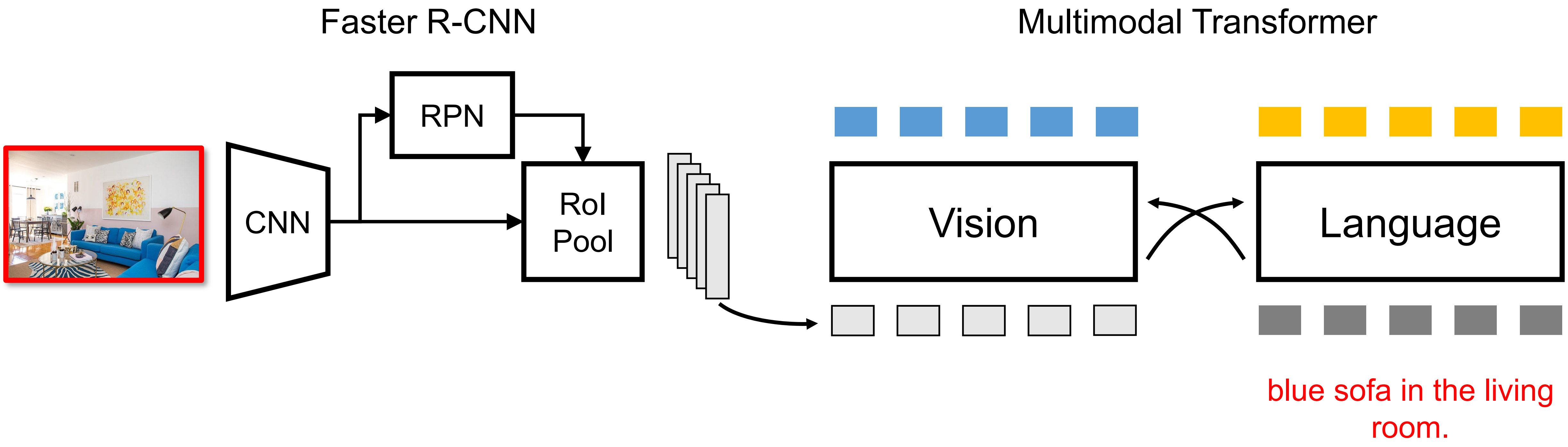
pop artist performs at the festival in a city.

a worker helps to clear the debris.

blue sofa in the living room.

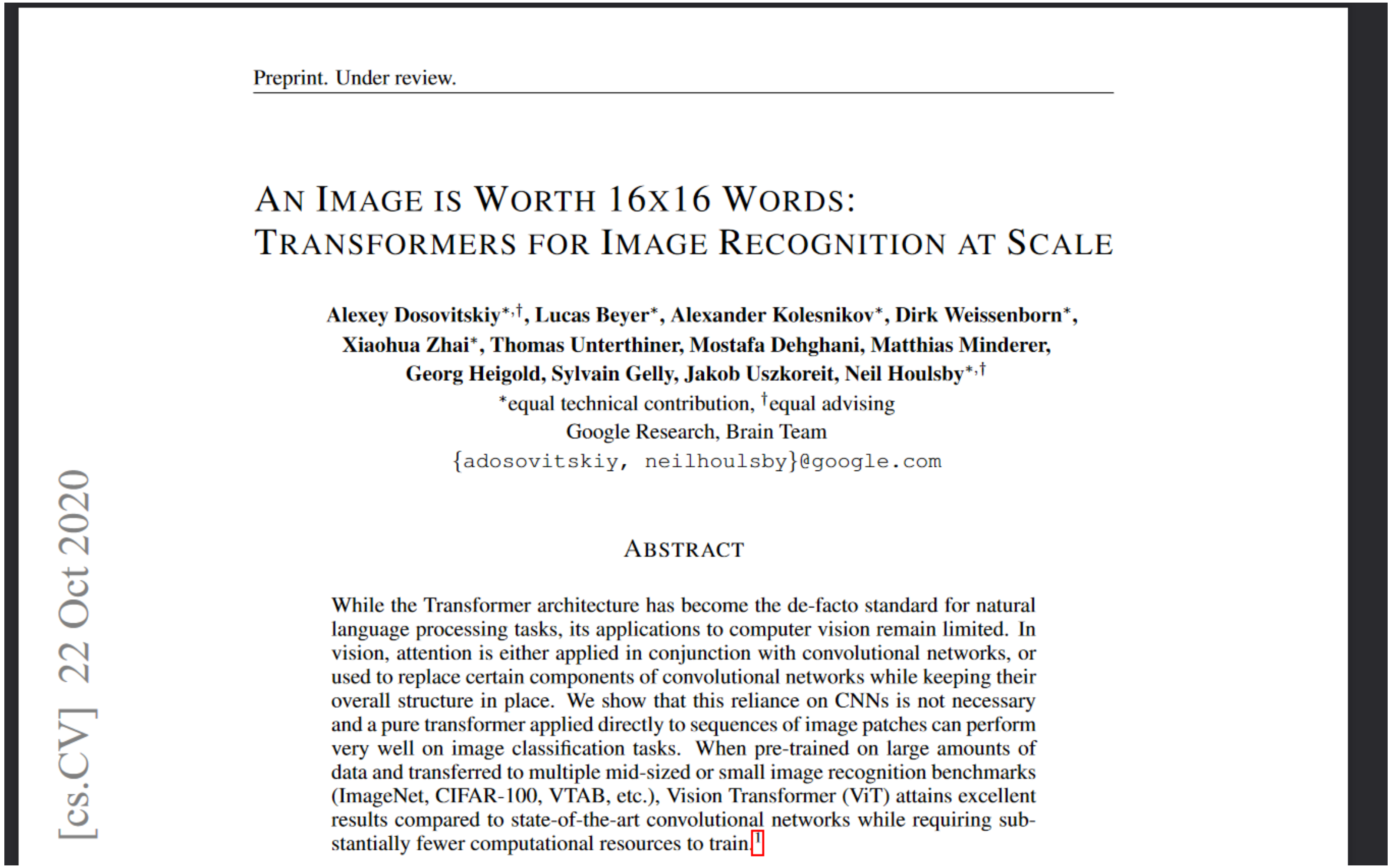
Image and captions from: Sharma, Piyush, et al. "Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning." ACL. 2018.

ViLBERT: A Visolinguistic Transformer



Lu et al "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks." *NeurIPS*. 2019.
Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *NeurIPS*. 2015.

What about for just image inputs? Without Convolution?



AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

Alexey Dosovitskiy*,†, Lucas Beyer*, Alexander Kolesnikov*, Dirk Weissenborn*, Xiaohua Zhai*, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby*,†

*equal technical contribution, †equal advising

Google Research, Brain Team

{adosovitskiy, neilhoulsby}@google.com

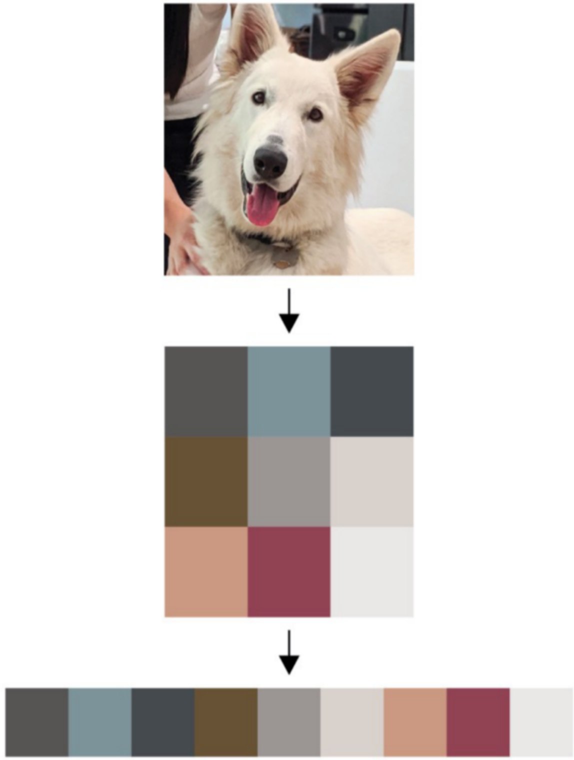
ABSTRACT

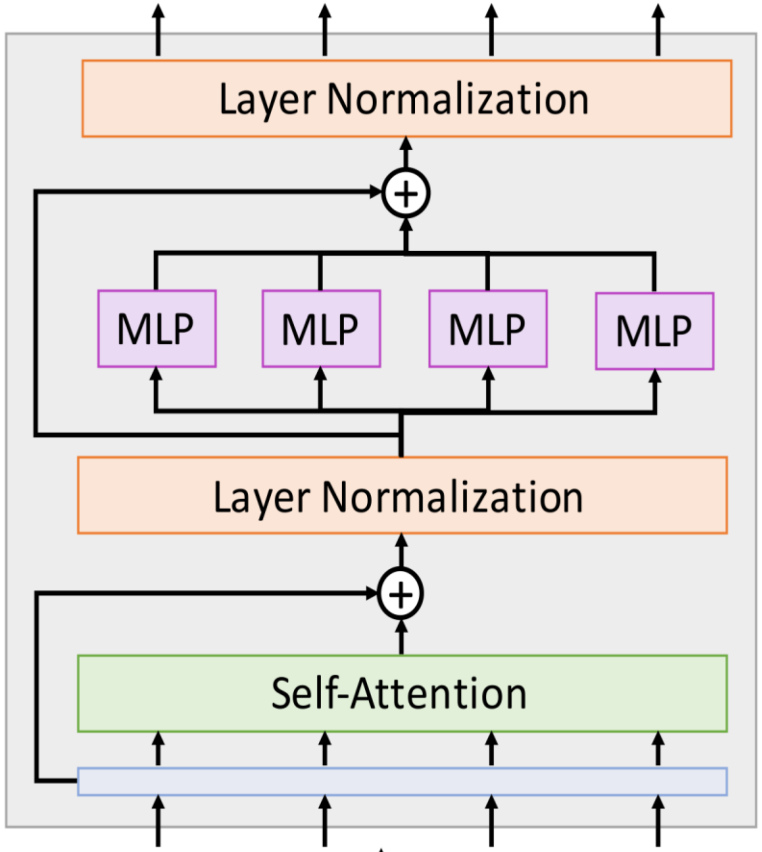
While the Transformer architecture has become the de-facto standard for natural language processing tasks, its applications to computer vision remain limited. In vision, attention is either applied in conjunction with convolutional networks, or used to replace certain components of convolutional networks while keeping their overall structure in place. We show that this reliance on CNNs is not necessary and a pure transformer applied directly to sequences of image patches can perform very well on image classification tasks. When pre-trained on large amounts of data and transferred to multiple mid-sized or small image recognition benchmarks (ImageNet, CIFAR-100, VTAB, etc.), Vision Transformer (ViT) attains excellent results compared to state-of-the-art convolutional networks while requiring substantially fewer computational resources to train.[1]

Slide progression inspired by Soheil Feizi

Idea #3: Standard Transformer on Pixels

Treat an image as a set of pixel values





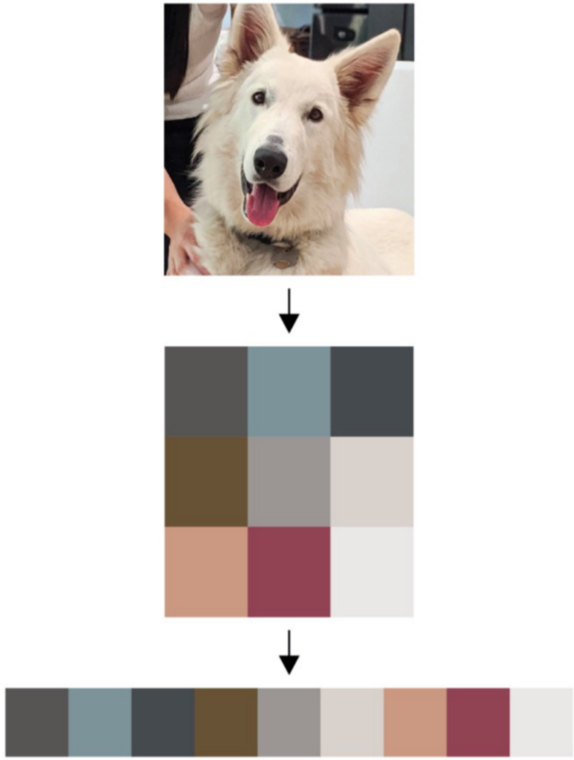
Feed as input to standard Transformer

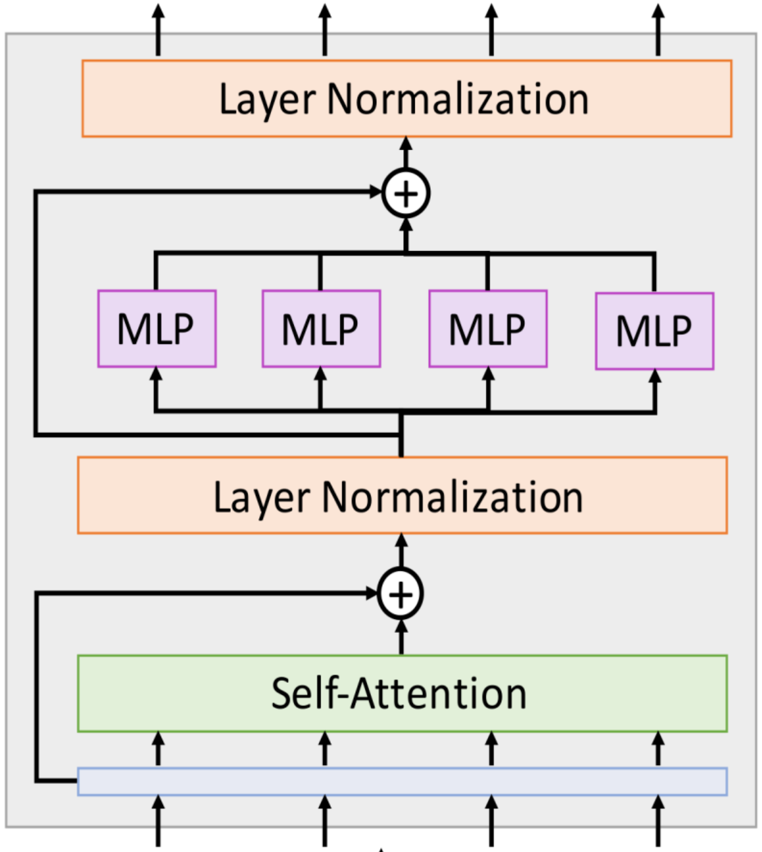
Chen et al, "Generative Pretraining from Pixels", ICML 2020

Slide credit: Justin Johnson

Idea #3: Standard Transformer on Pixels

Treat an image as a set of pixel values





Feed as input to standard Transformer

Problem: Memory use!

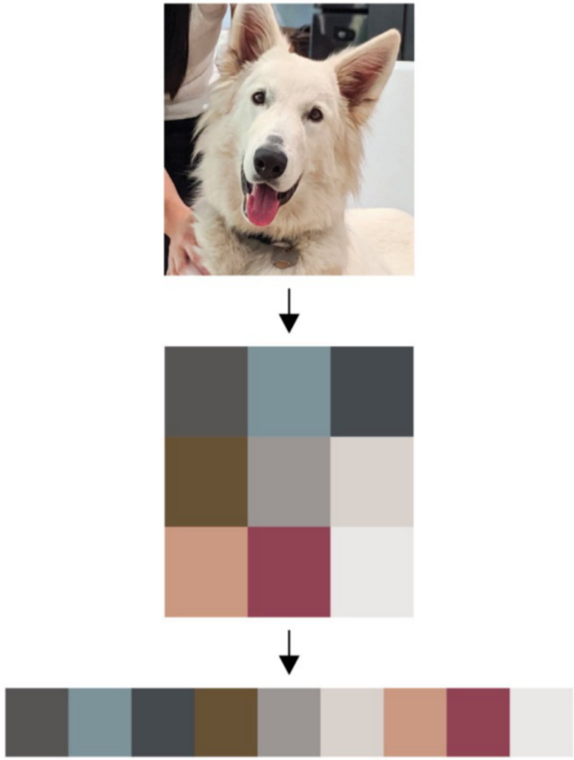
R x R image needs R^4 elements per attention matrix

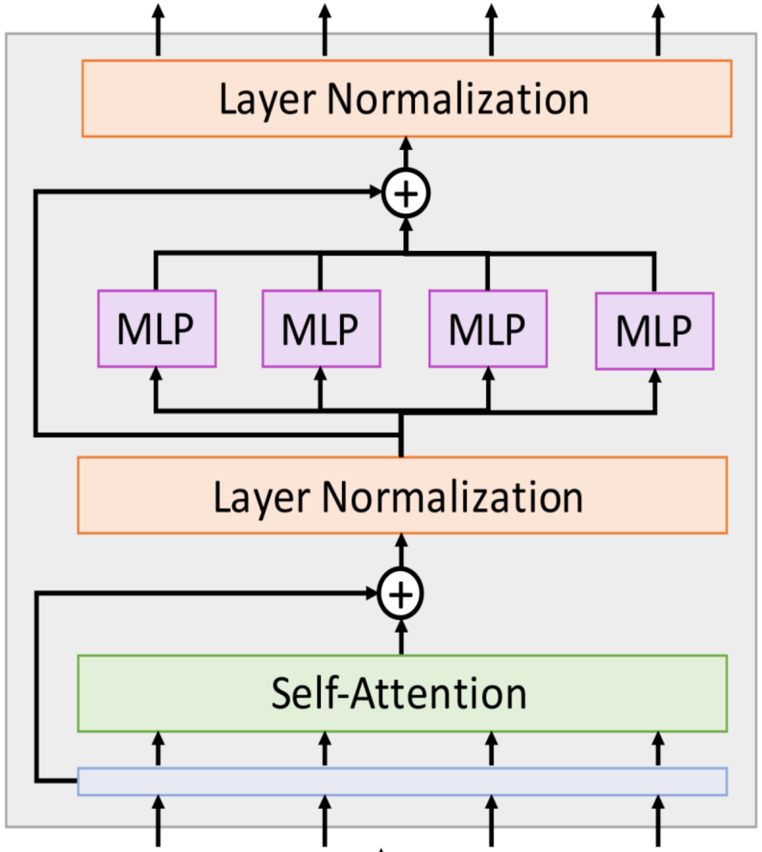
Chen et al, “Generative Pretraining from Pixels”, ICML 2020

Slide credit: Justin Johnson

Idea #3: 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

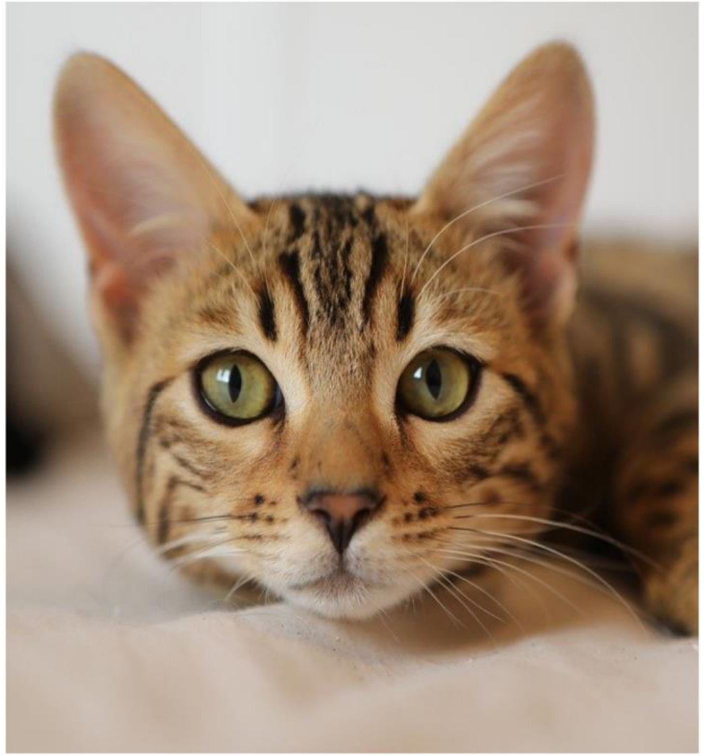
R=128, 48 layers, 16 heads per layer takes 768GB of memory for attention matrices for a single example…

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

Slide credit: Justin Johnson

Slide credit: Justin Johnson

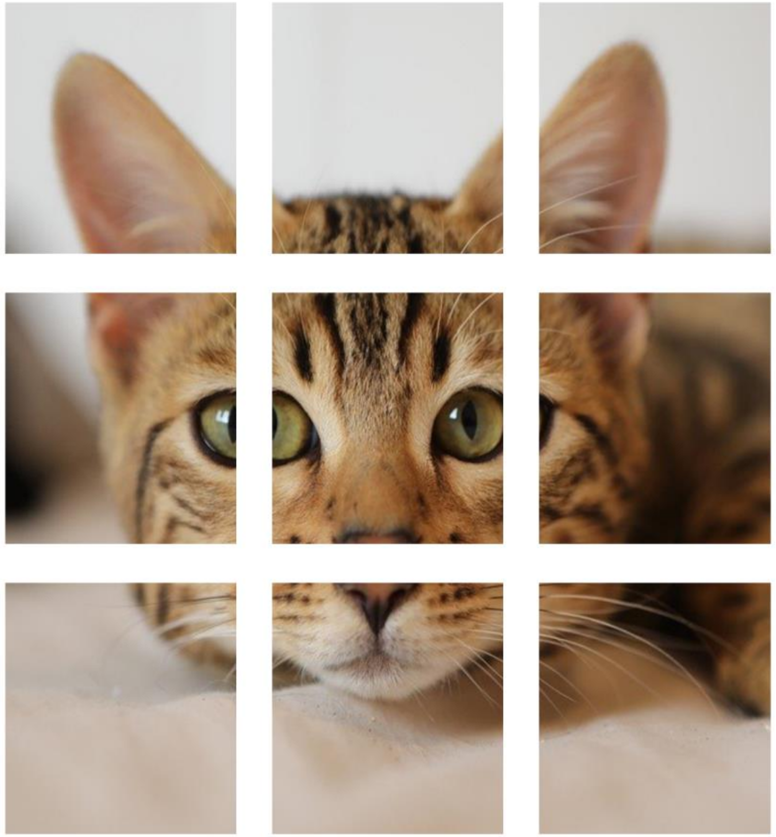
Idea #4: Standard Transformer on Patches



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

Slide credit: Justin Johnson

Idea #4: Standard Transformer on Patches

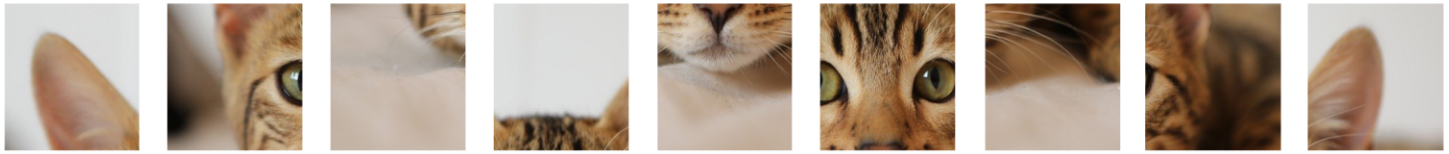


Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

Slide credit: Justin Johnson

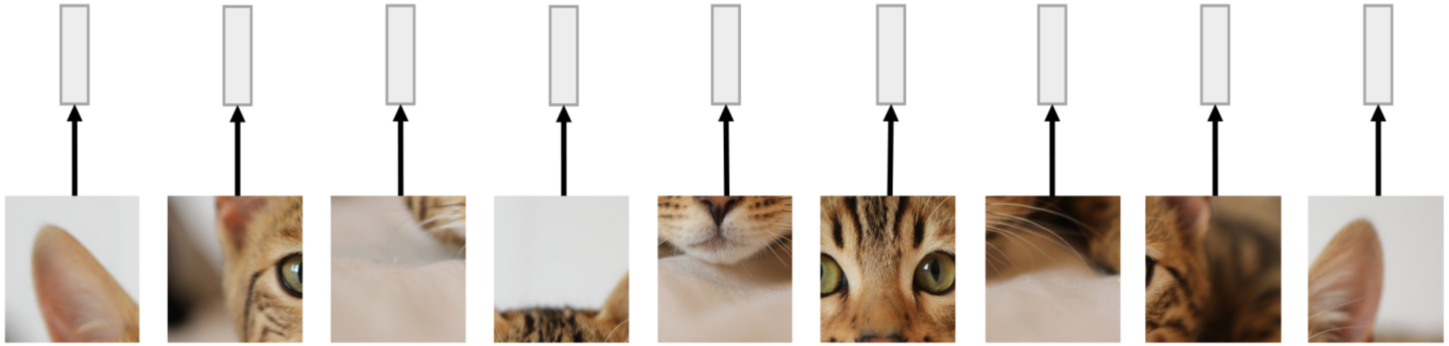
Idea #4: Standard Transformer on Patches

N input patches, each of shape 3x16x16



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

Idea #4: Standard Transformer on Patches



Linear projection to D-dimensional vector

N input patches, each of shape 3x16x16

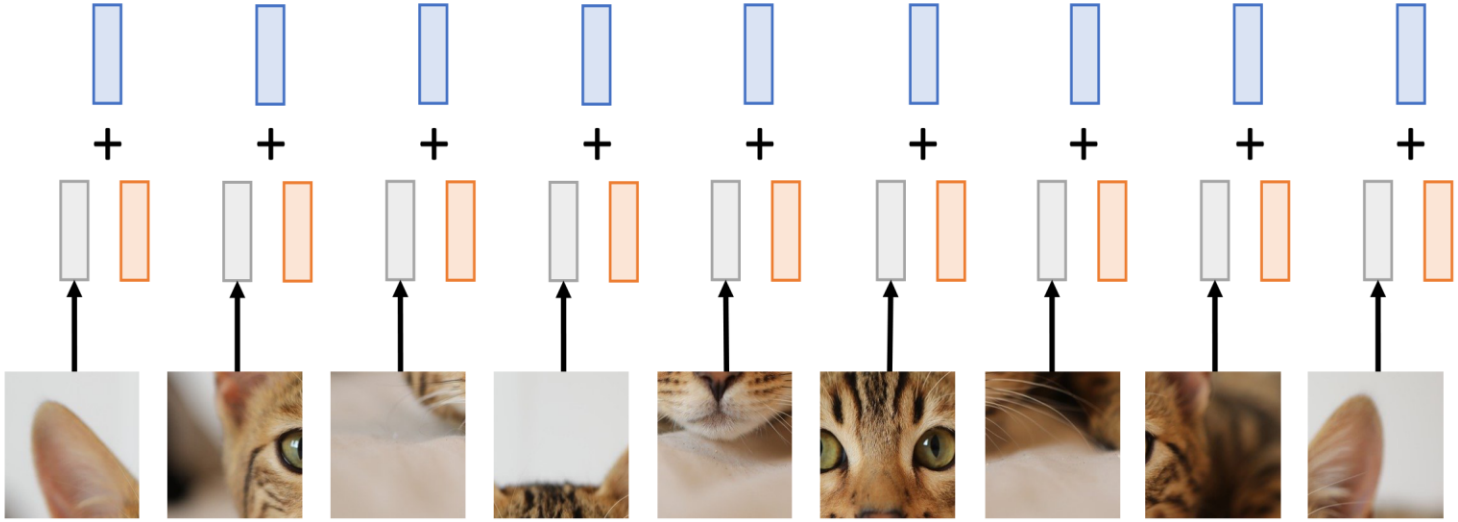
Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Idea #4: 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

Idea #4: Standard Transformer on Patches

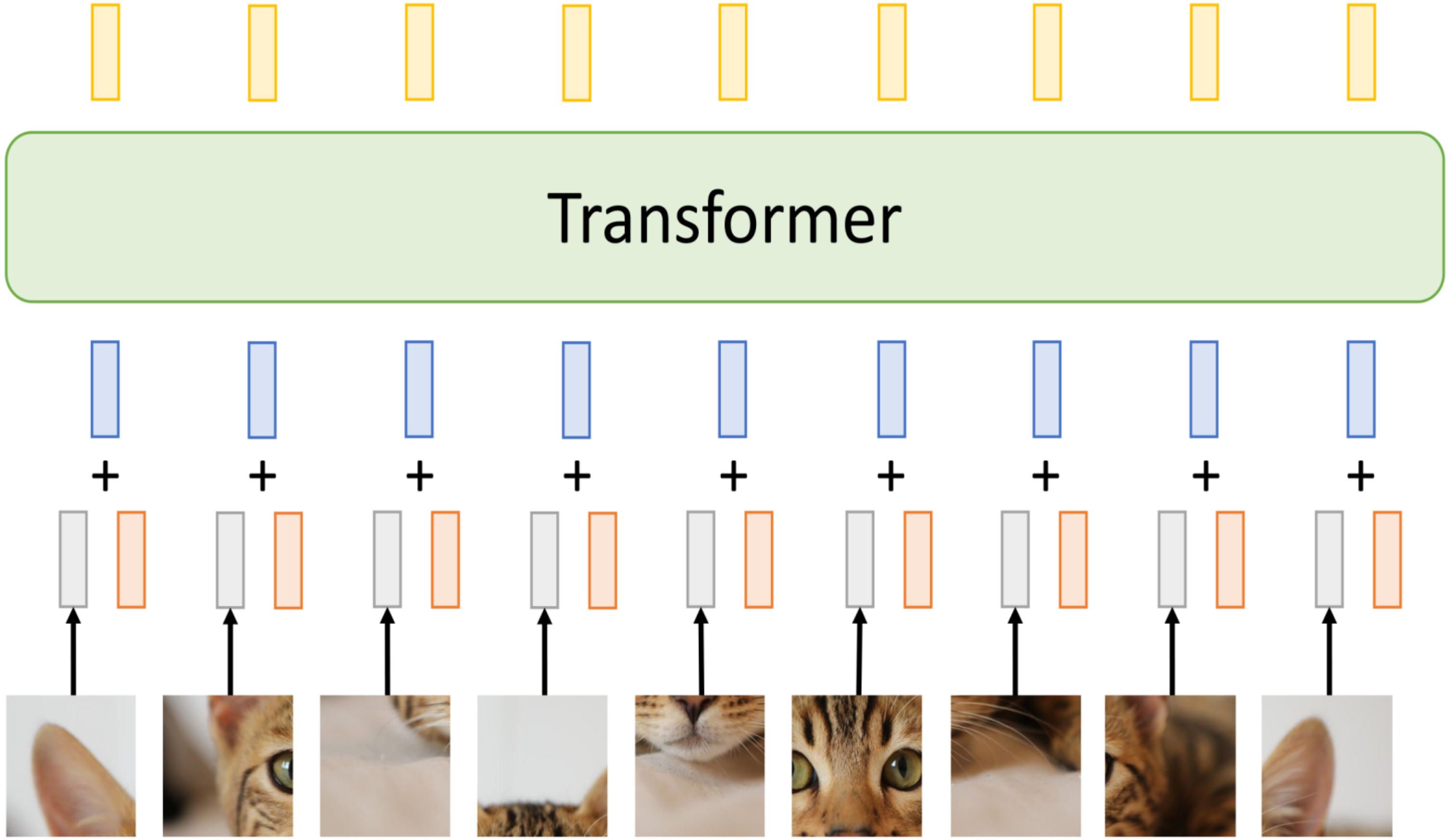
Output vectors

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

Slide credit: Justin Johnson

Idea #4: Standard Transformer on Patches

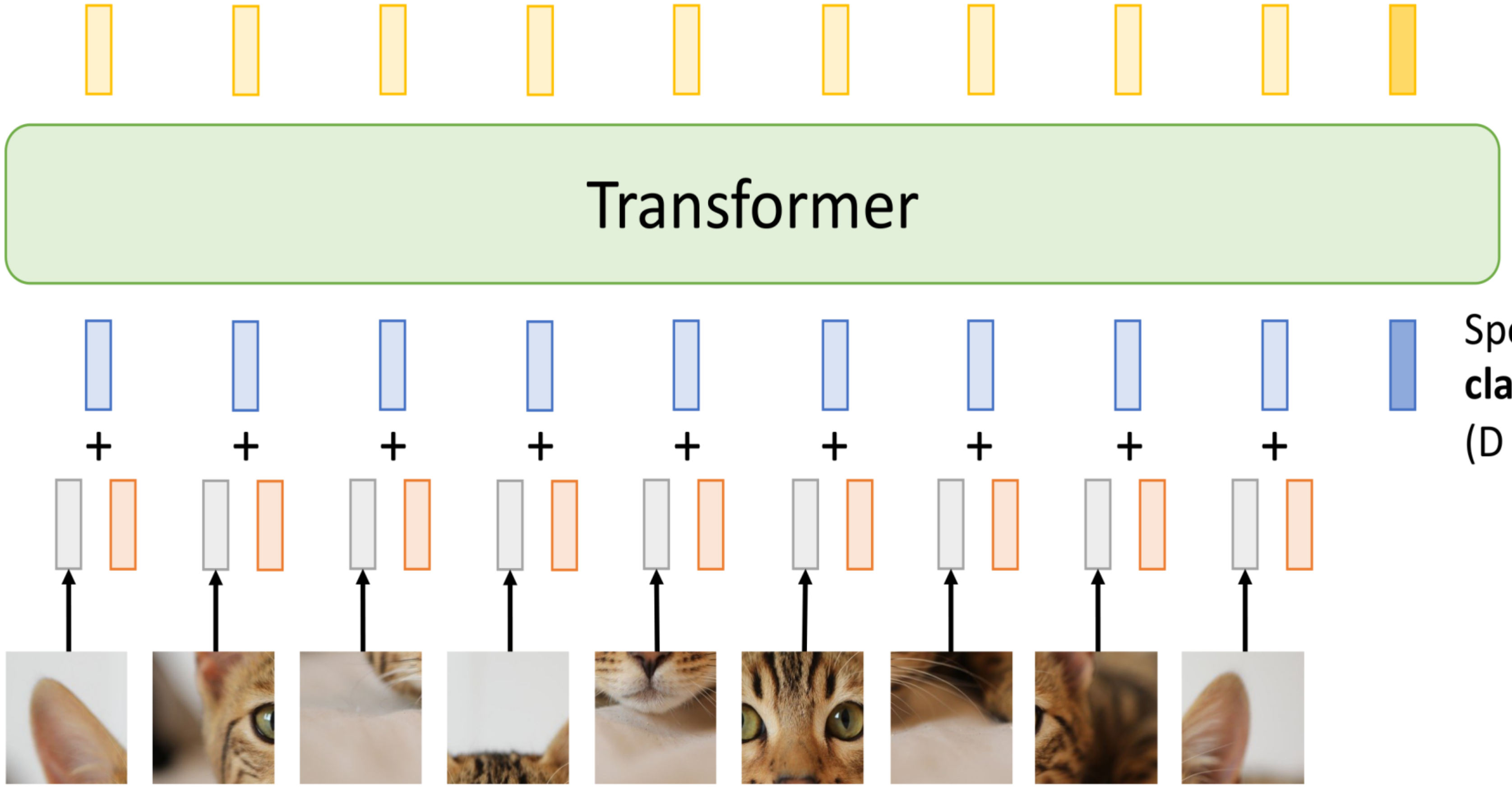
Output vectors

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

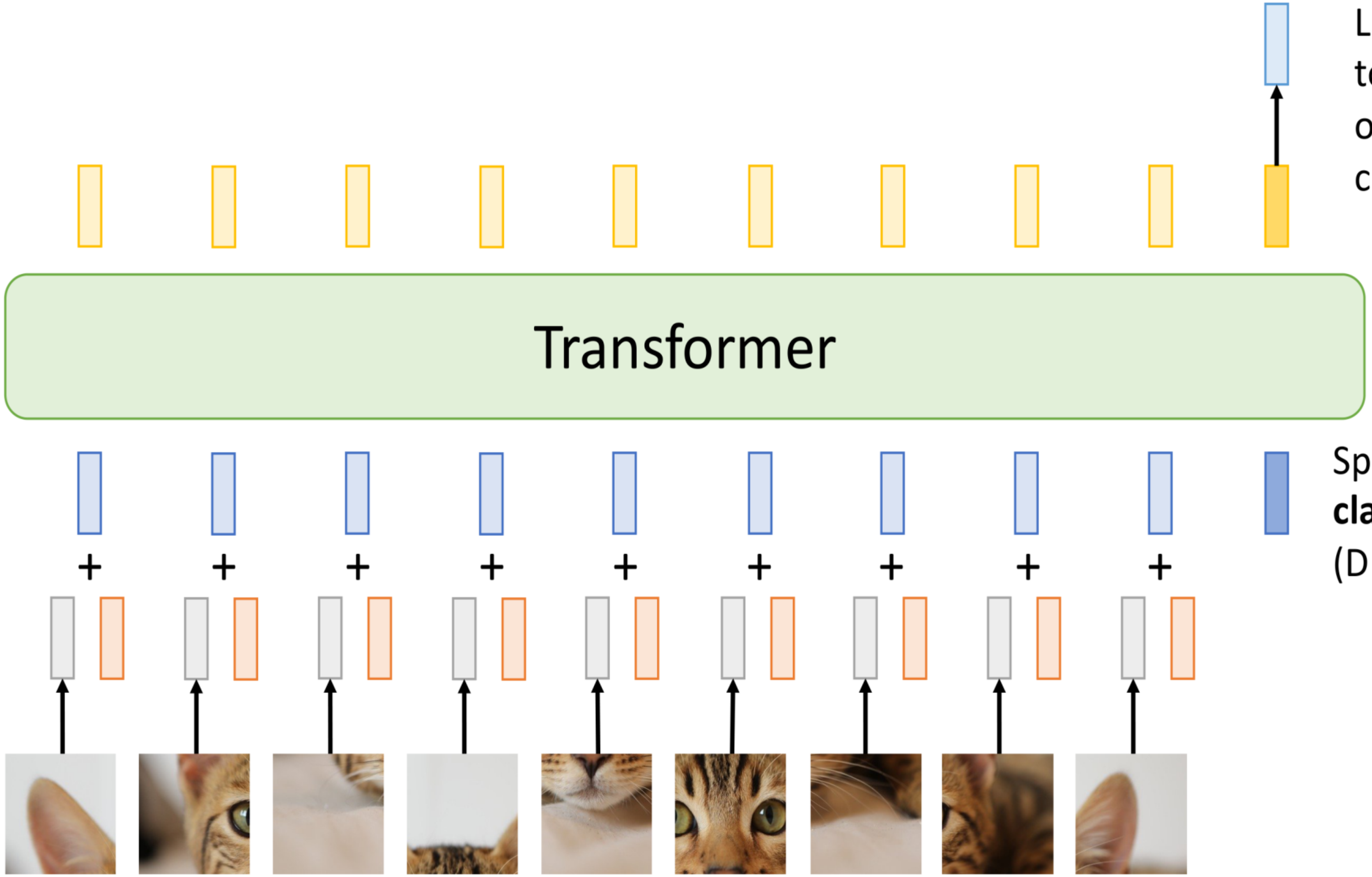


Special extra input: **classification token** (D dims, learned)

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

Slide credit: Justin Johnson

Idea #4: 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

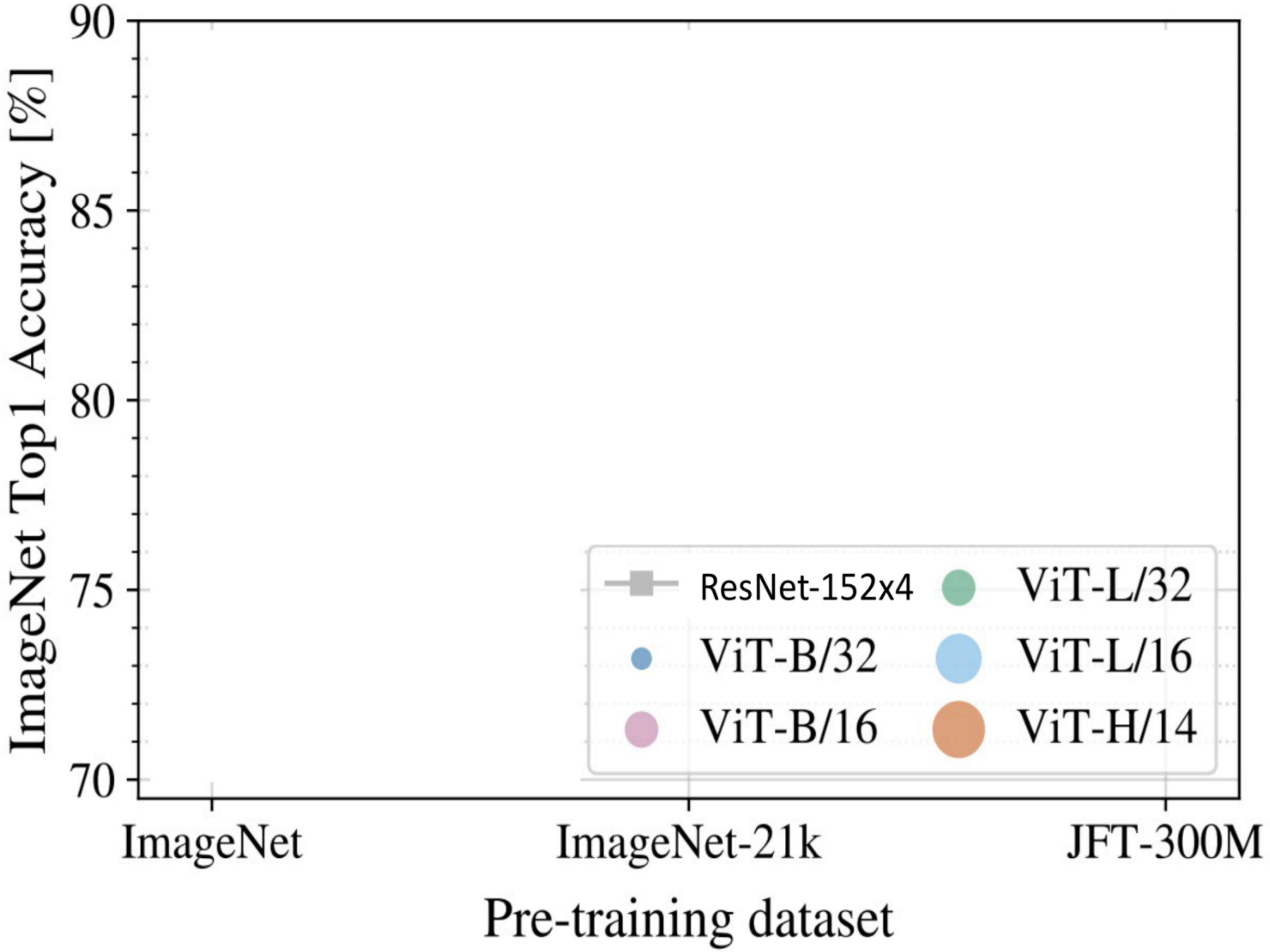
N input patches, each of shape 3x16x16

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

Vision Transformer (ViT) vs ResNets



B = Base
L = Large
H = Huge

/32, /16, /14 is patch size; smaller patch size is a bigger model (more patches)

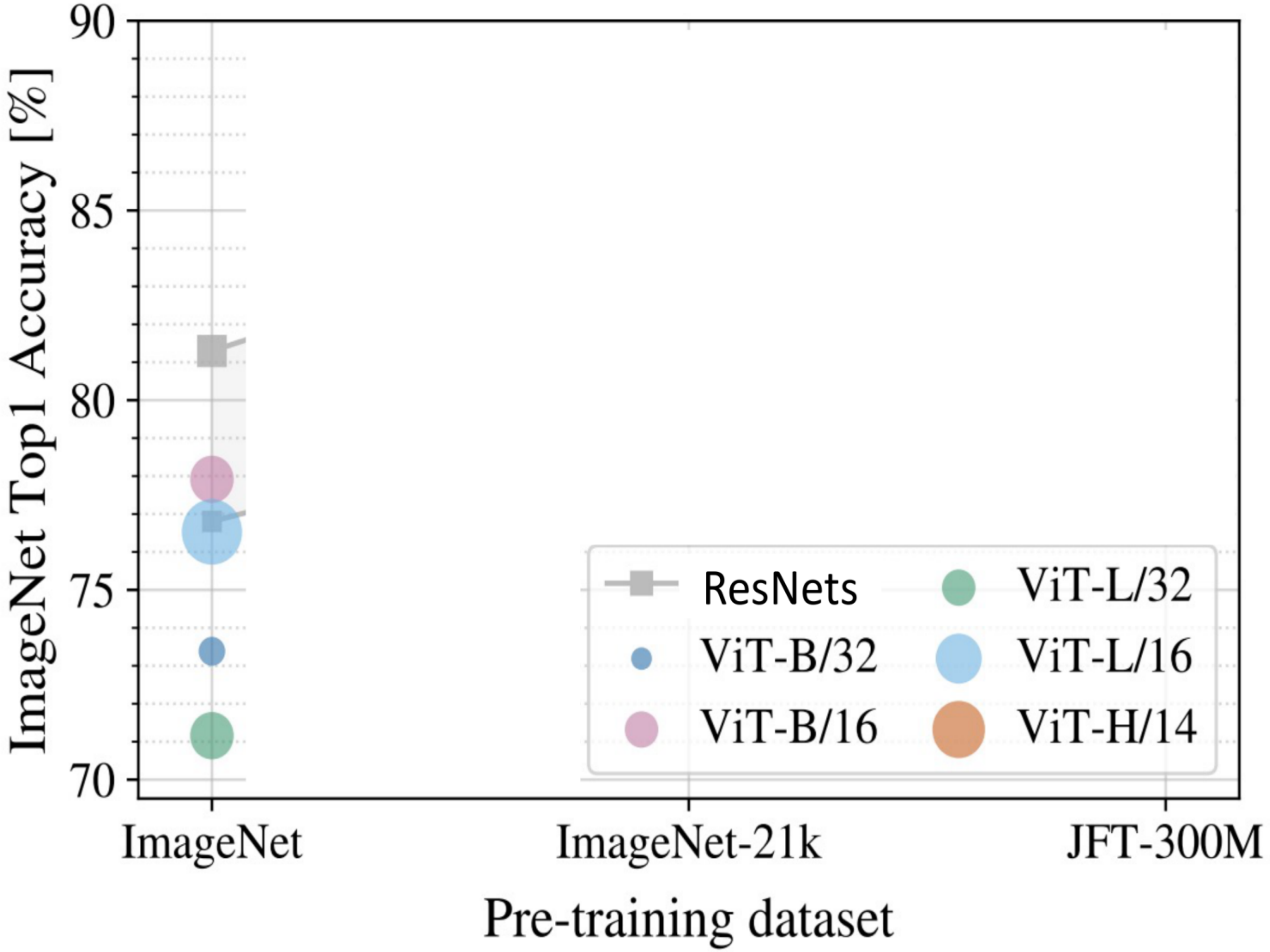
Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

Slide credit: Justin Johnson

Vision Transformer (ViT) vs ResNets

Recall: ImageNet dataset has 1k categories, 1.2M images

When trained on ImageNet, ViT models perform worse than ResNets



B = Base
L = Large
H = Huge

/32, /16, /14 is patch size; smaller patch size is a bigger model (more patches)

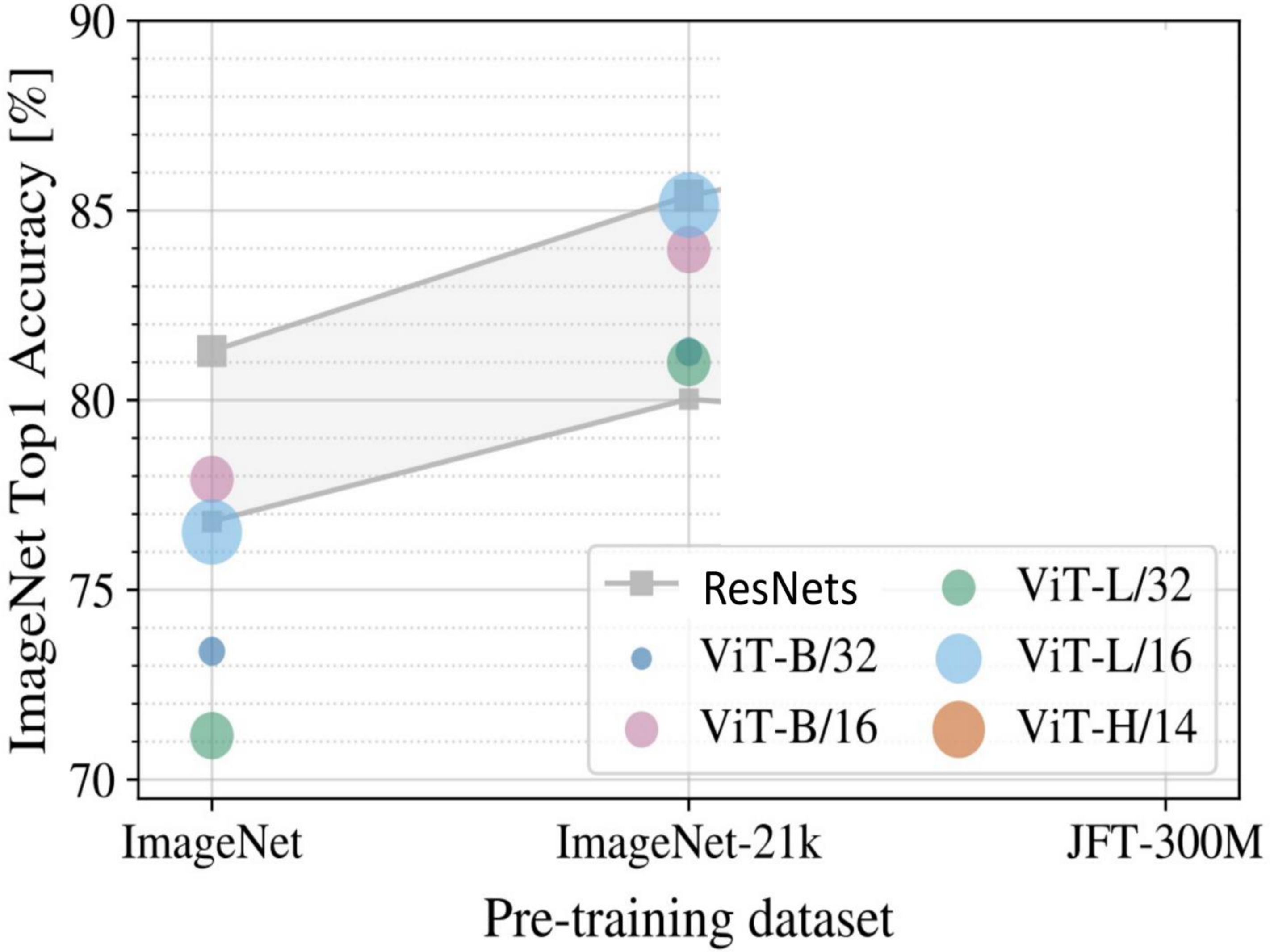
Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

Slide credit: Justin Johnson

Vision Transformer (ViT) vs ResNets

ImageNet-21k has 14M images with 21k categories

If you pretrain on ImageNet-21k and fine-tune on ImageNet, ViT does better: big ViTs match big ResNets



B = Base
L = Large
H = Huge

/32, /16, /14 is patch size; smaller patch size is a bigger model (more patches)

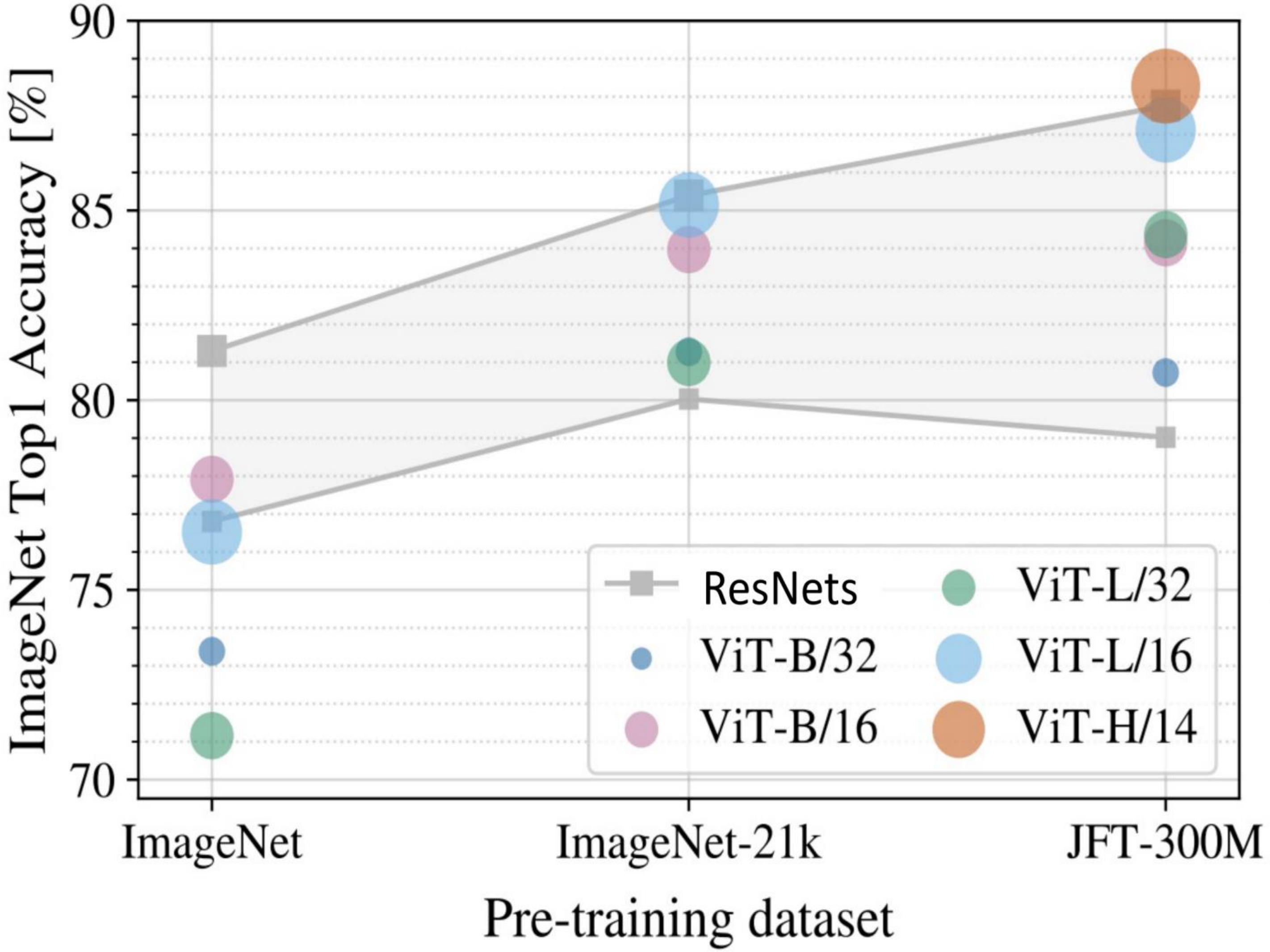
Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Slide credit: Justin Johnson

Vision Transformer (ViT) vs ResNets

JFT-300M is an internal Google dataset with 300M labeled images

If you pretrain on JFT and finetune on ImageNet, large ViTs outperform large ResNets



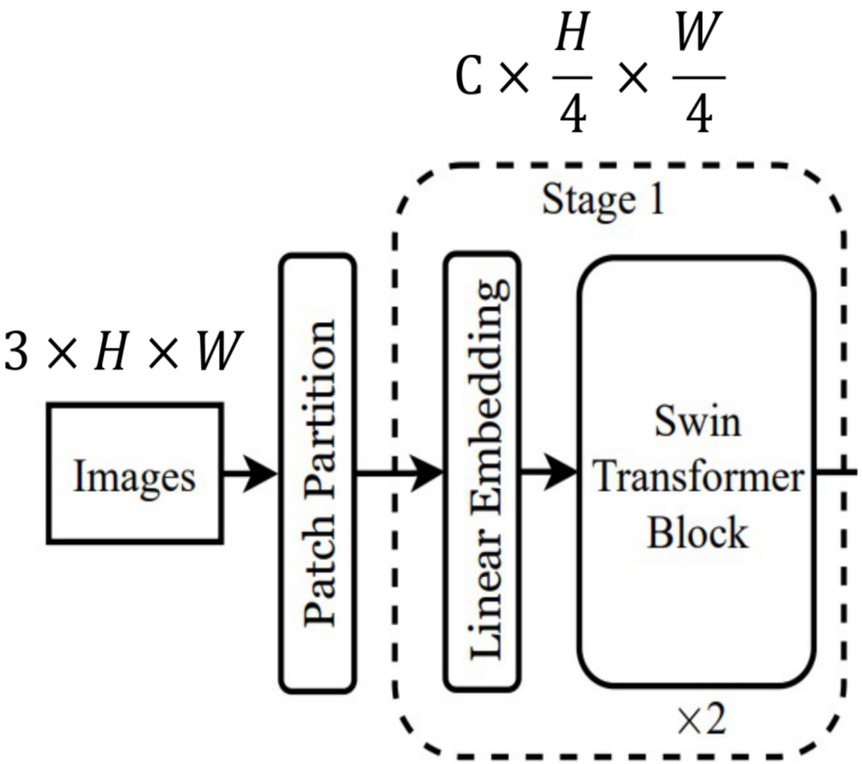
B = Base
L = Large
H = Huge

/32, /16, /14 is patch size; smaller patch size is a bigger model (more patches)

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

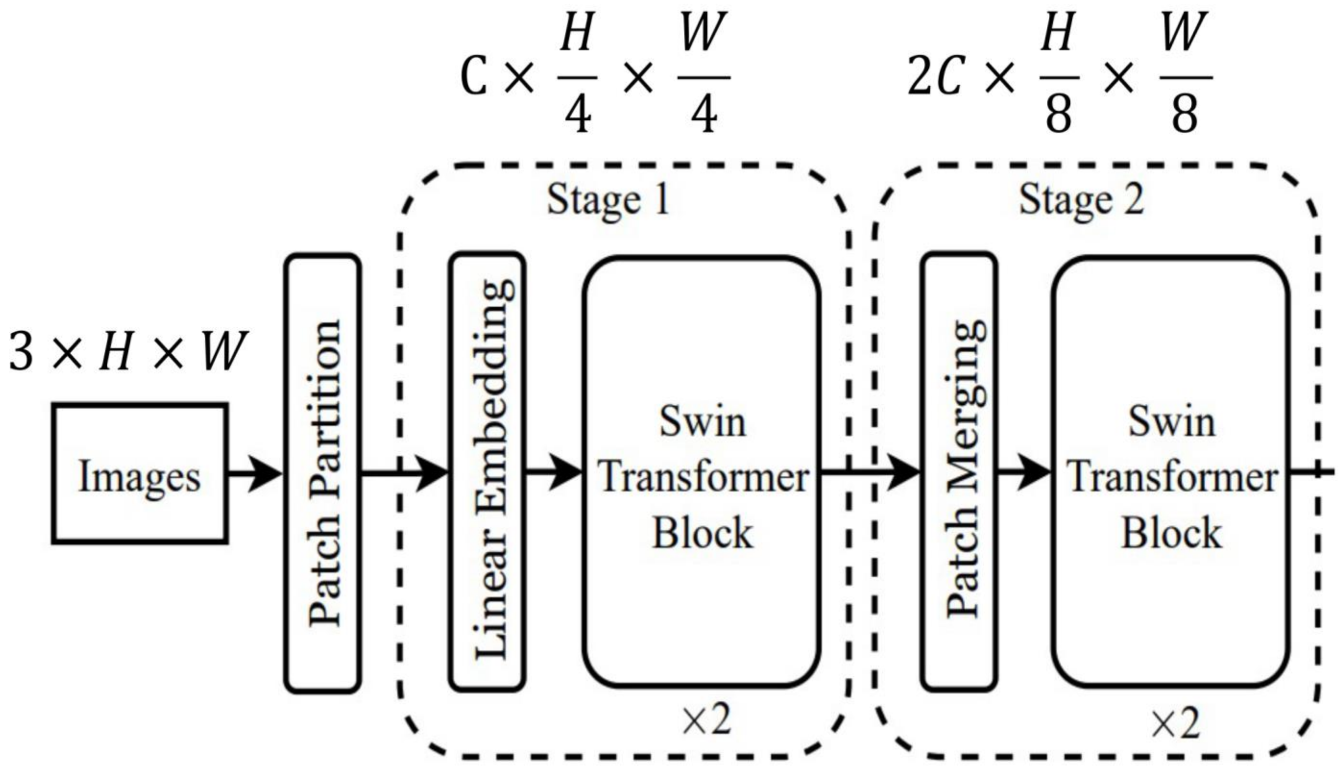
Slide credit: Justin Johnson

Hierarchical ViT: Swin Transformer



Divide image into 4x4 patches and project to C dimensions

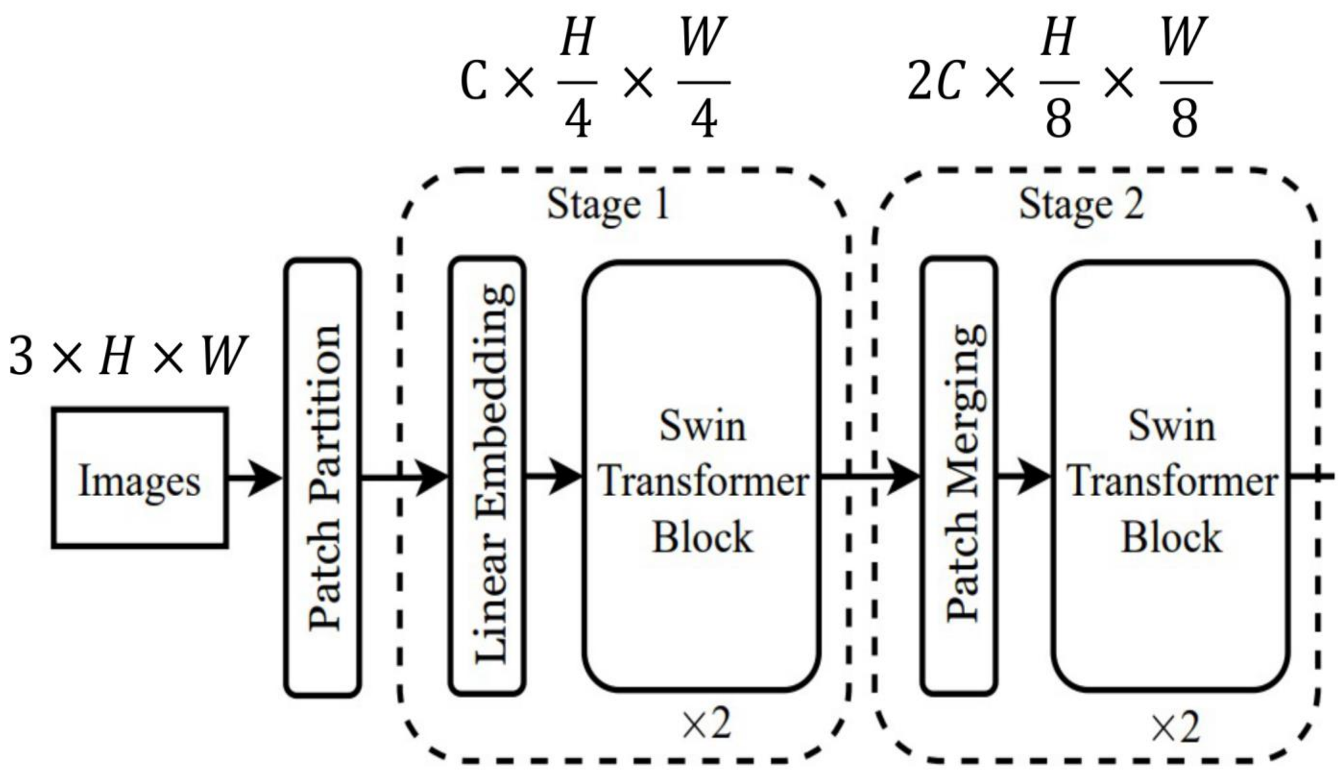
Hierarchical ViT: Swin Transformer

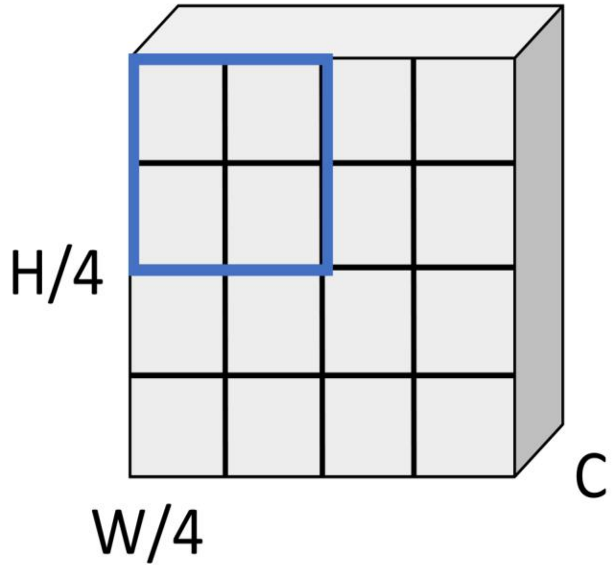


Divide image into 4x4 patches and project to C dimensions

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

Hierarchical ViT: Swin Transformer

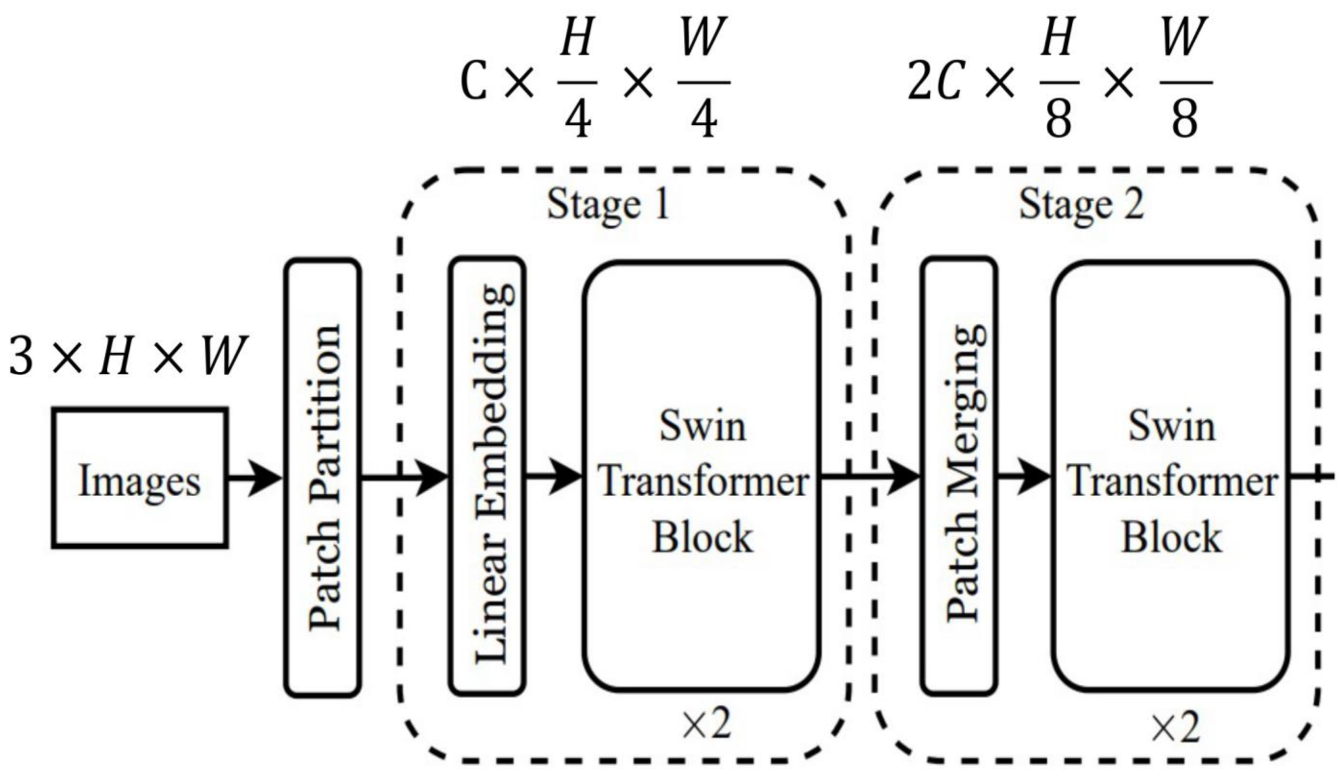


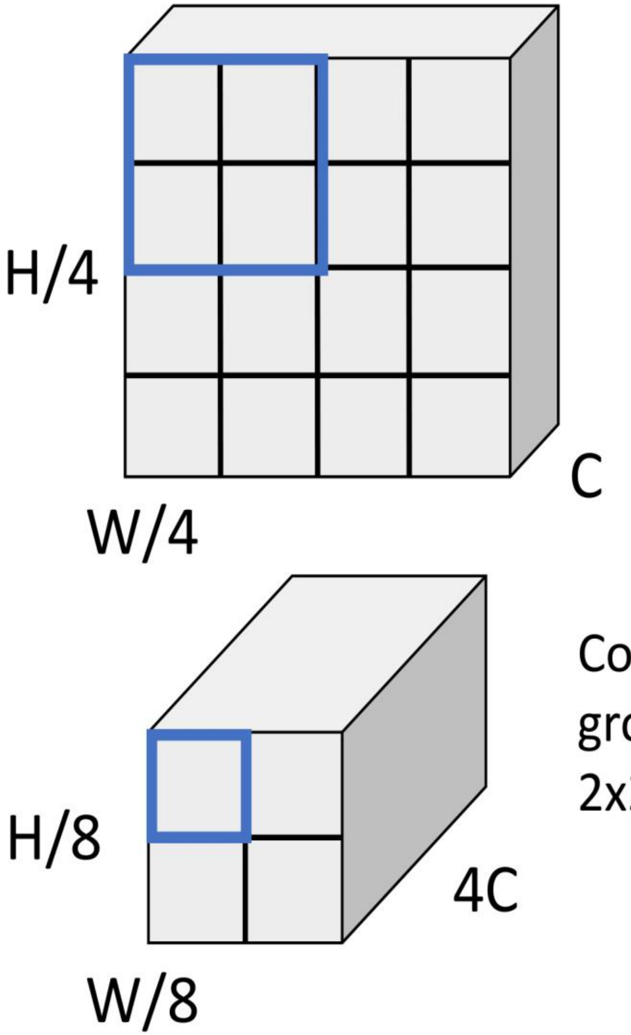


Divide image into 4x4 patches and project to C dimensions

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

Hierarchical ViT: Swin Transformer





Concatenate groups of 2x2 features

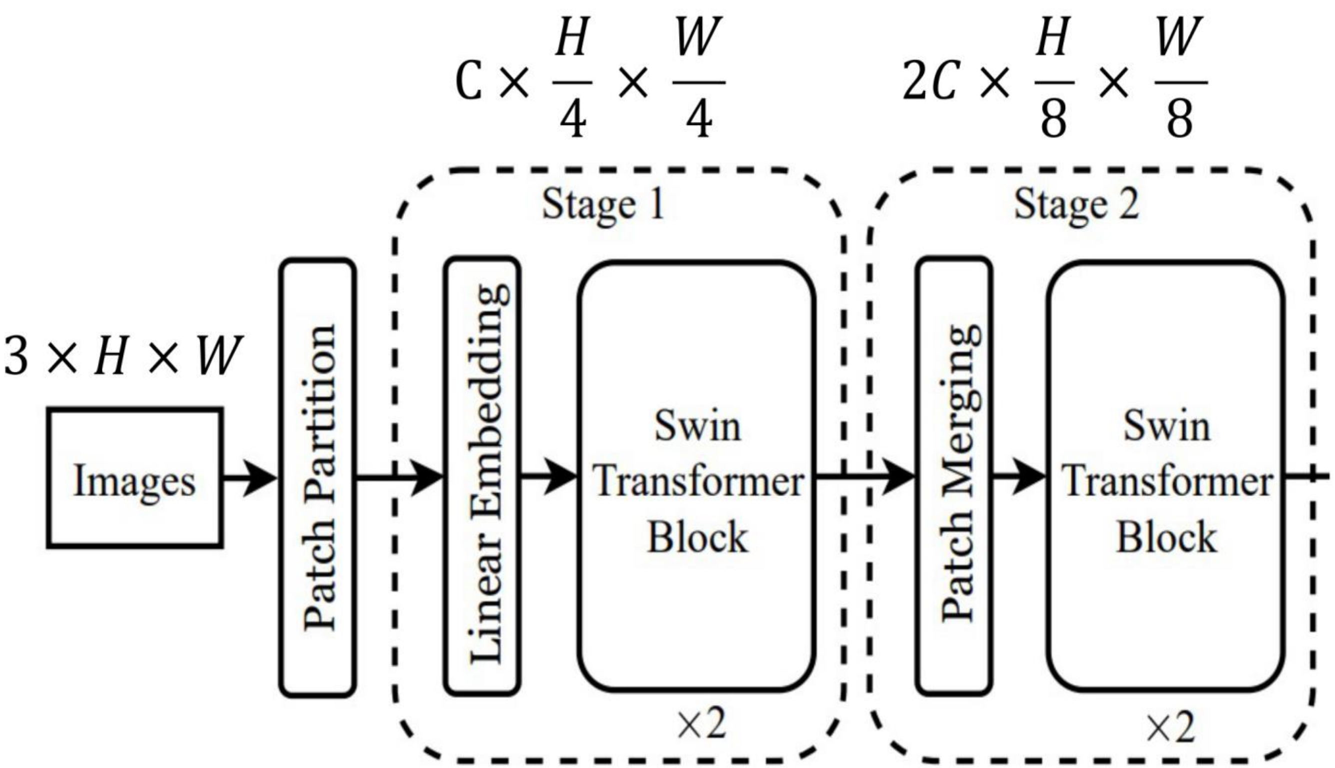
Divide image into 4x4 patches and project to C dimensions

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

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

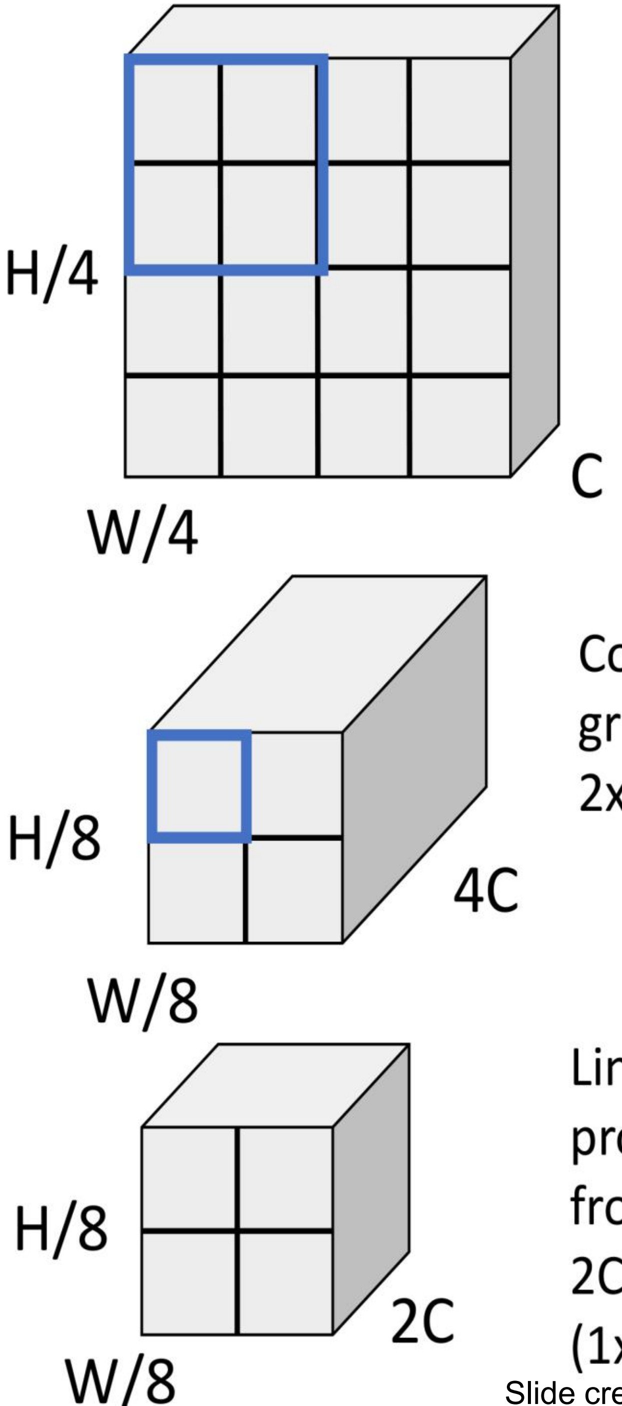
Slide credit: Justin Johnson

Hierarchical ViT: Swin Transformer



Divide image into 4x4 patches and project to C dimensions

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



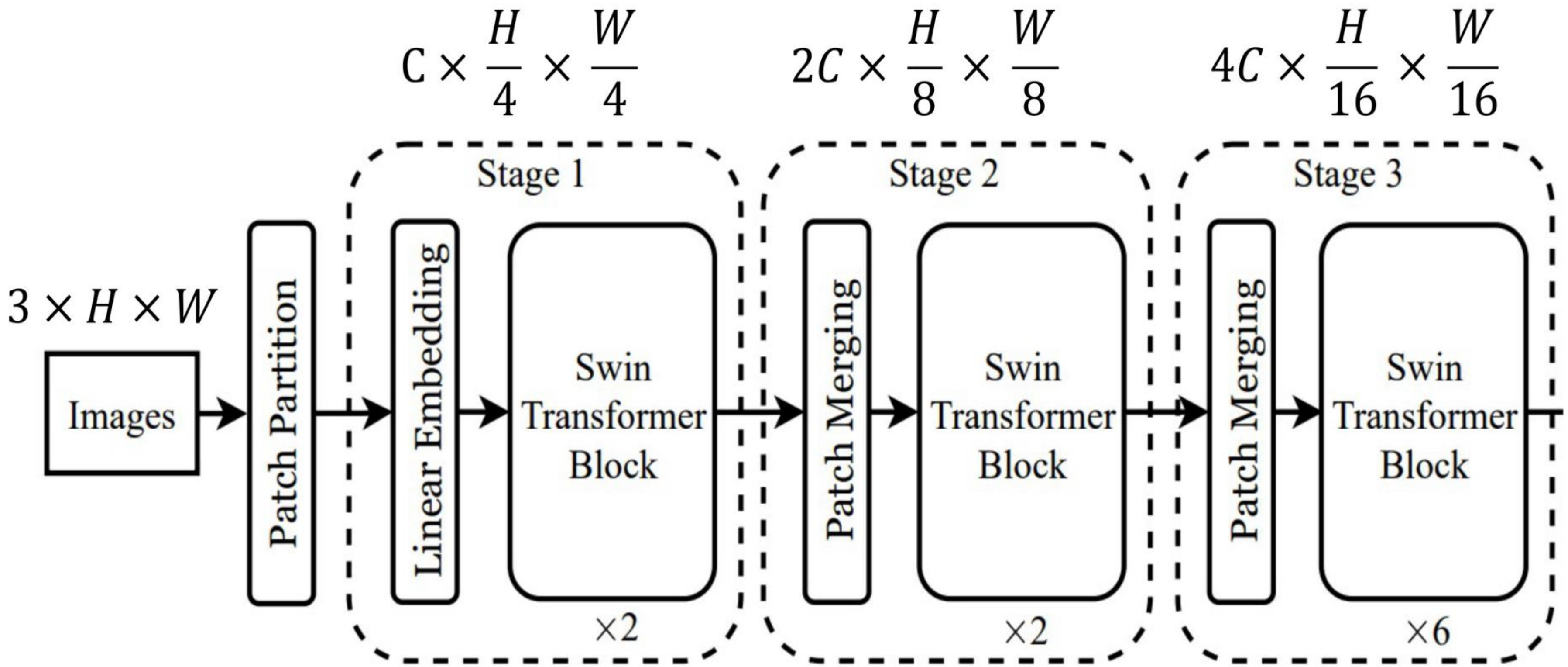
Concatenate groups of 2x2 features

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

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

Slide credit: Justin Johnson

Hierarchical ViT: Swin Transformer



Divide image into 4x4 patches and project to C dimensions

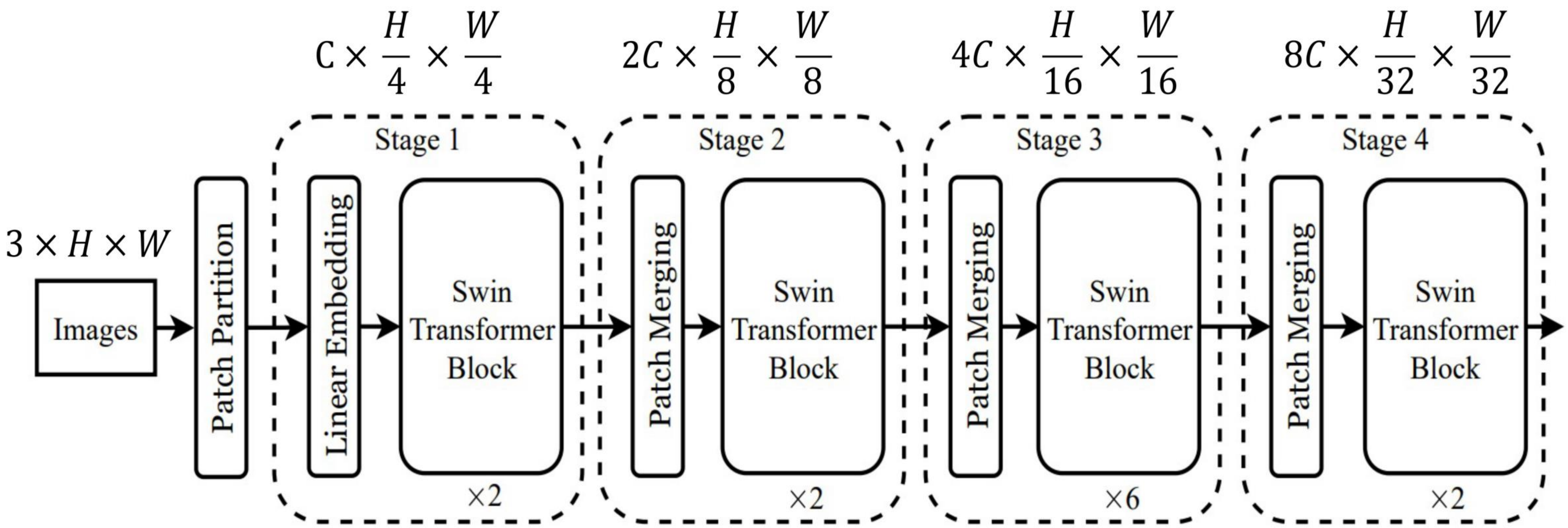
Merge 2x2 neighborhoods; now patches are (effectively) 8x8

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

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

Slide credit: Justin Johnson

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

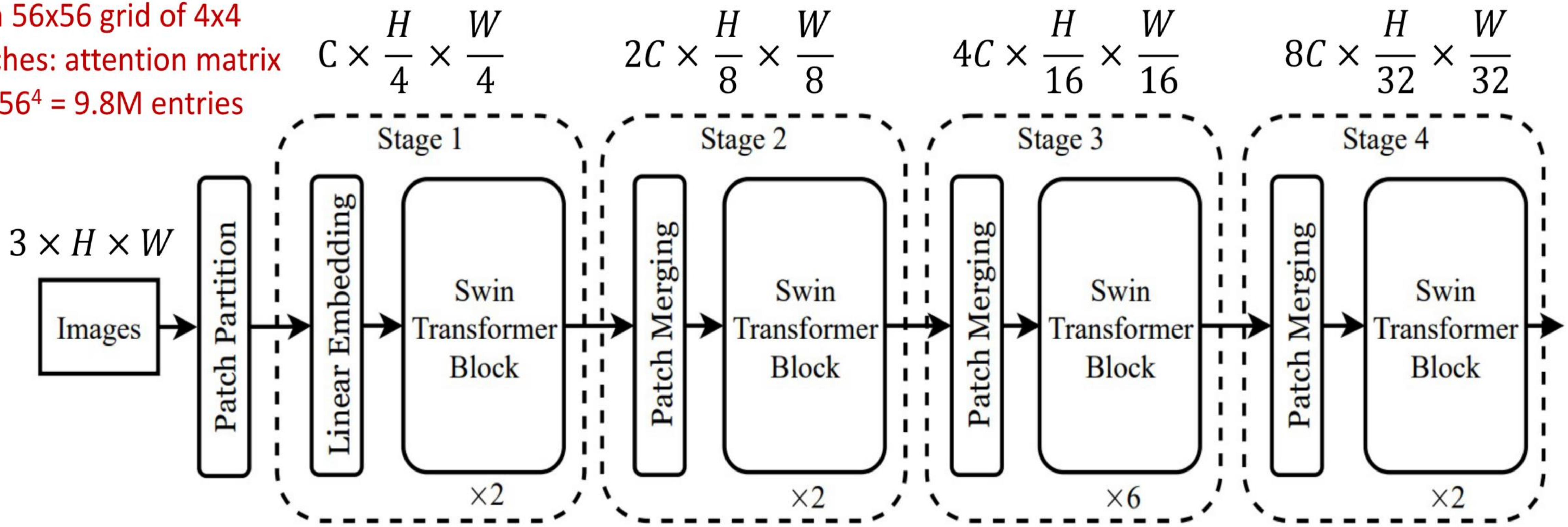
Merge 2x2 neighborhoods; now patches are (effectively) 32x32

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

Slide credit: Justin Johnson

Hierarchical ViT: Swin Transformer

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



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

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

Swin Transformer: Window Attention

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

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: $M^4(H/M)(W/M) = M^2HW$

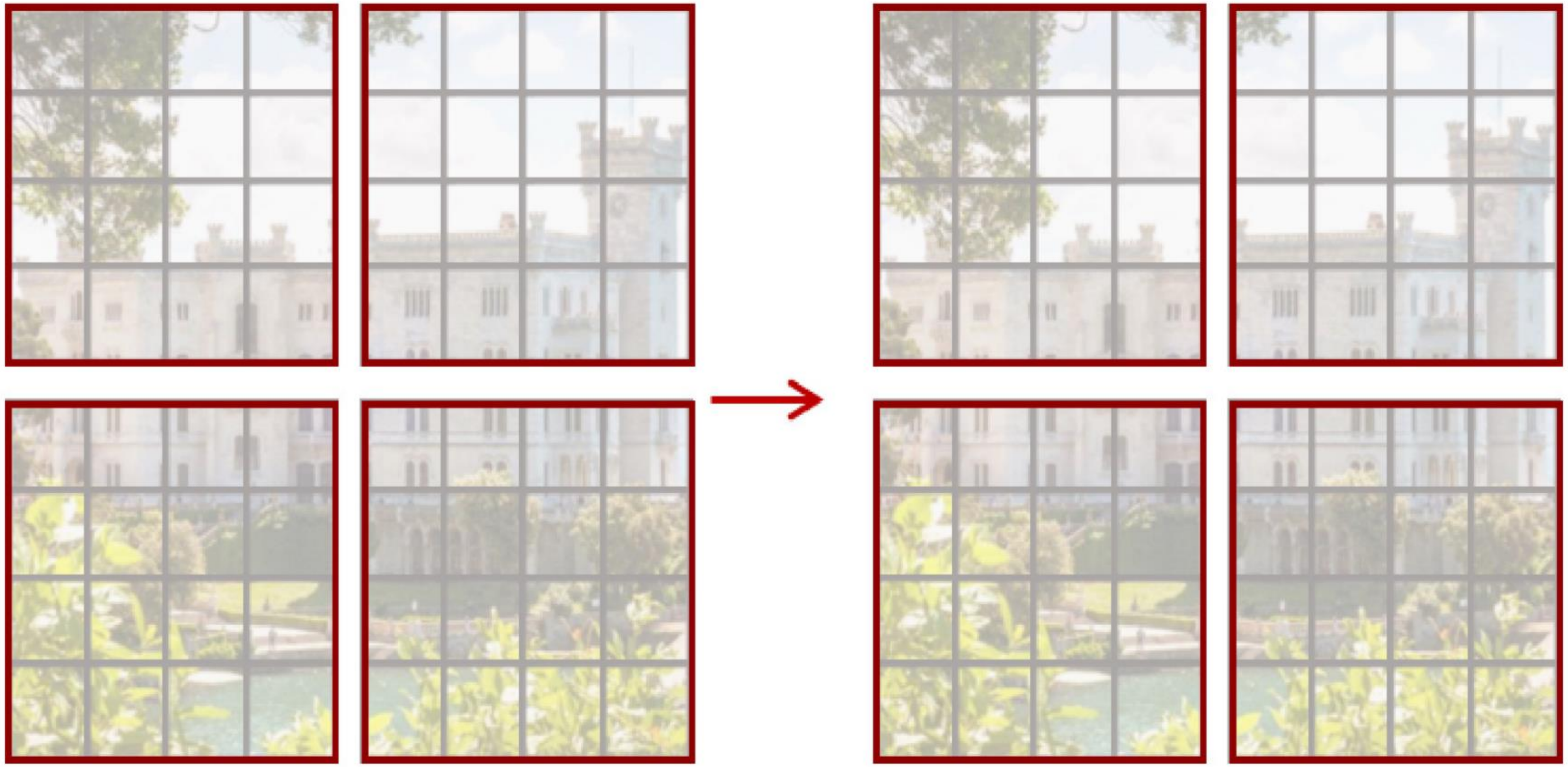
Linear in image size for fixed M!
Swin uses M=7 throughout the network

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

Slide credit: Justin Johnson

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

Slide credit: Justin Johnson

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

Slide credit: Justin Johnson

Swin Transformer: Shifted Window Attention

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

Detail: Relative Positional Bias

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

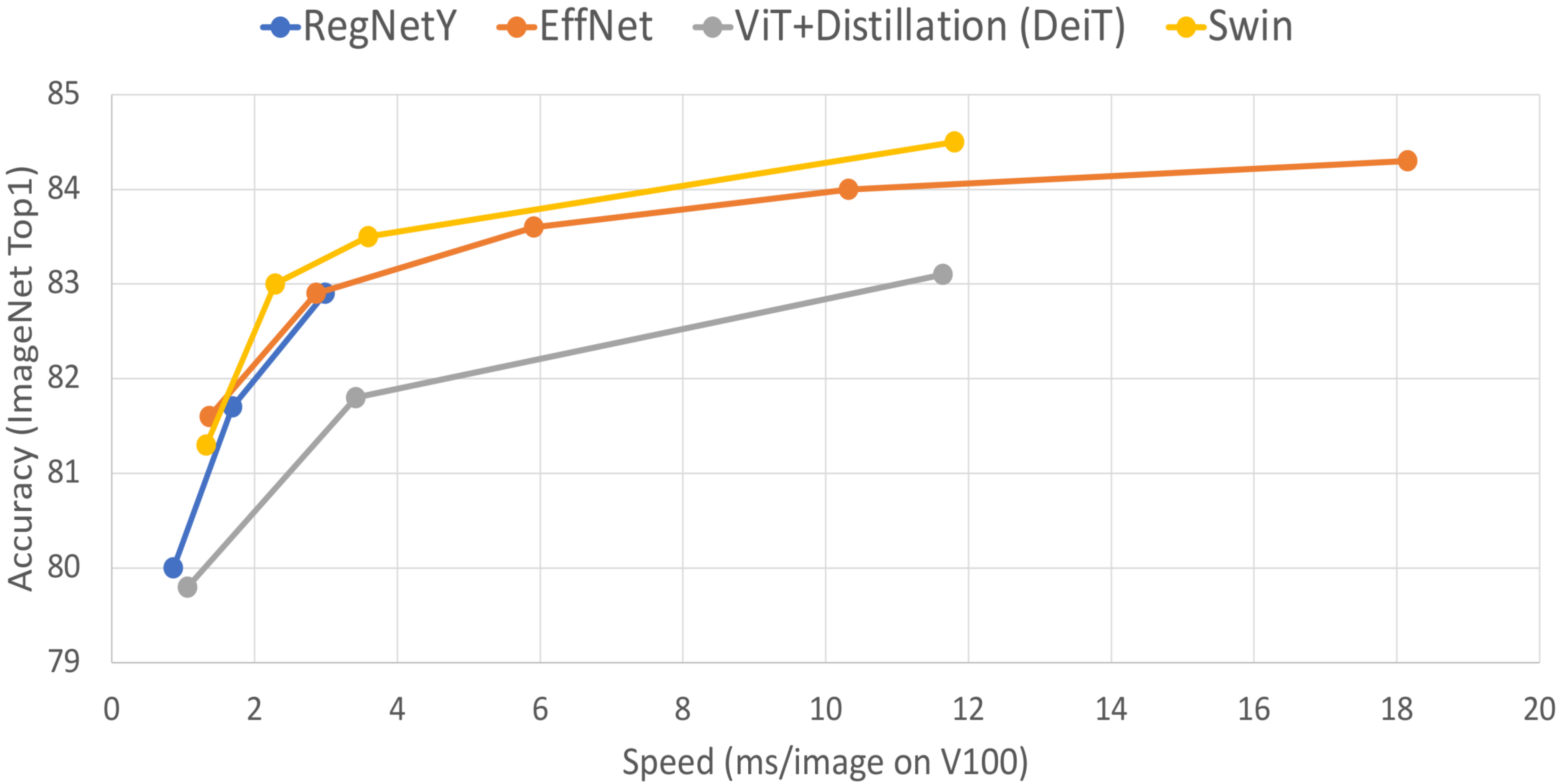
Block L: Normal windows

Block L+1: Shifted Windows

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

Slide credit: Justin Johnson

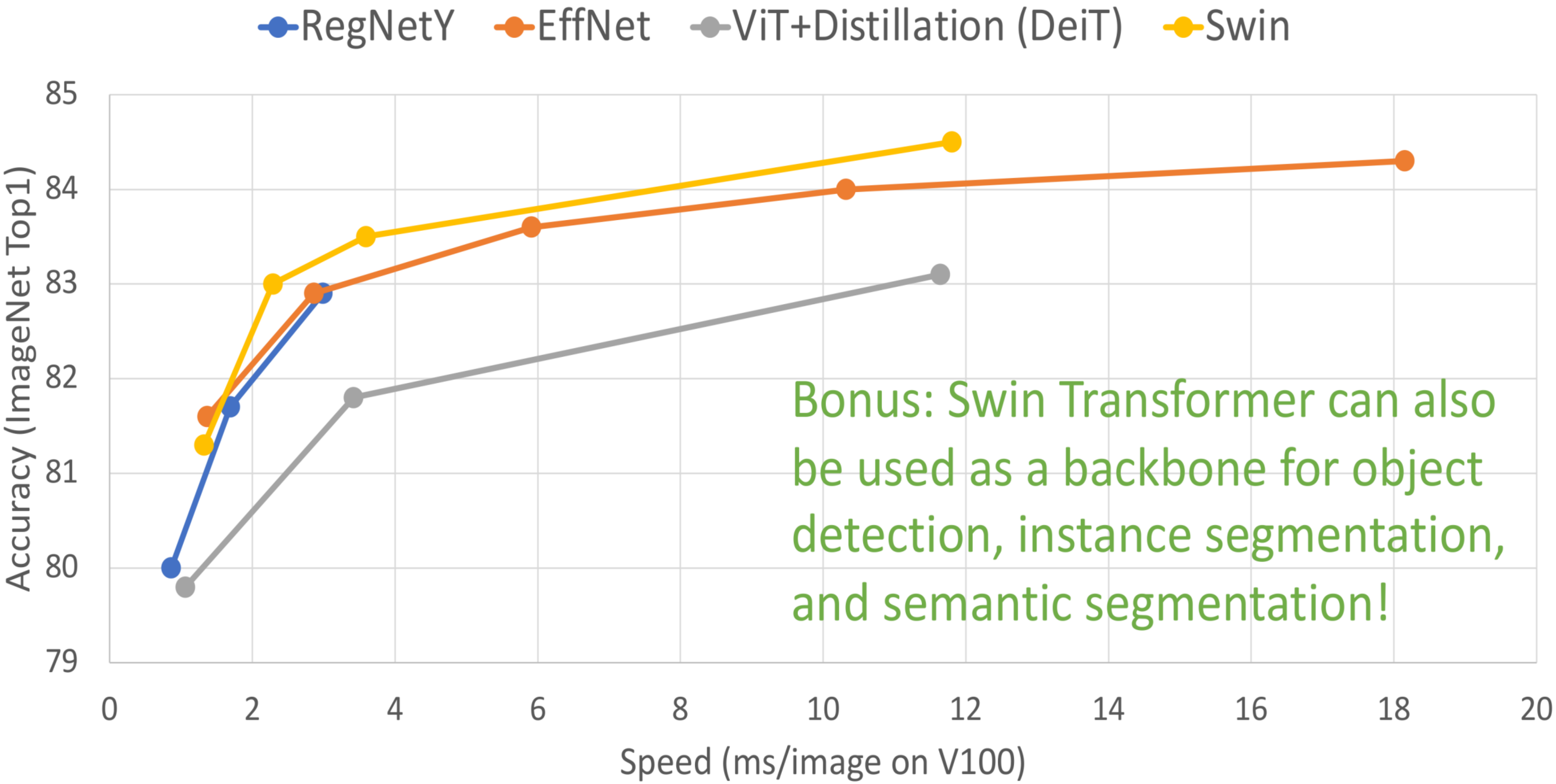
Swin Transformer: Speed vs Accuracy



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

Slide credit: Justin Johnson

Swin Transformer: Speed vs Accuracy

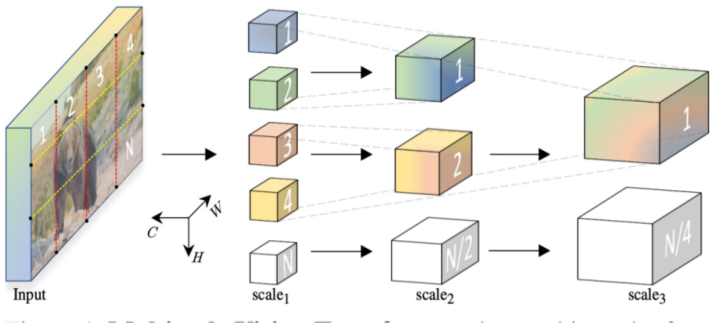


Liu et al. "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Slide credit: Justin Johnson

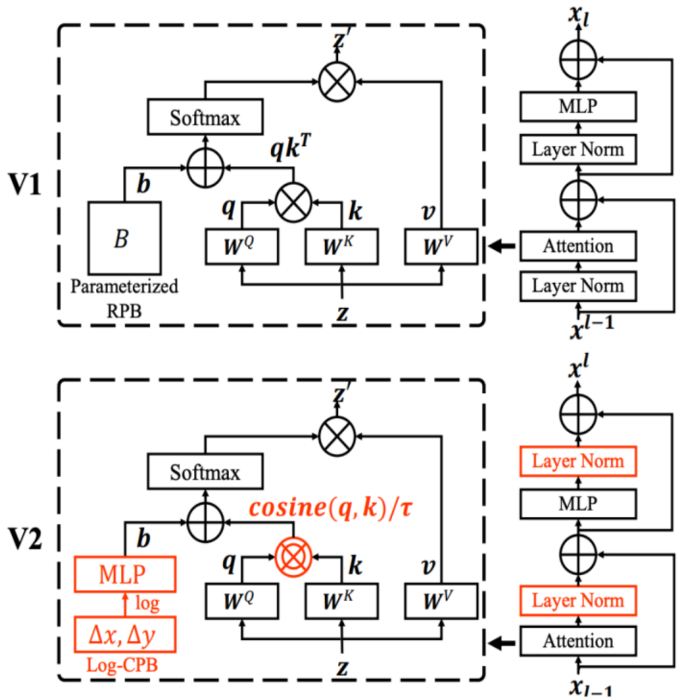
Other Hierarchical Vision Transformers

MViT



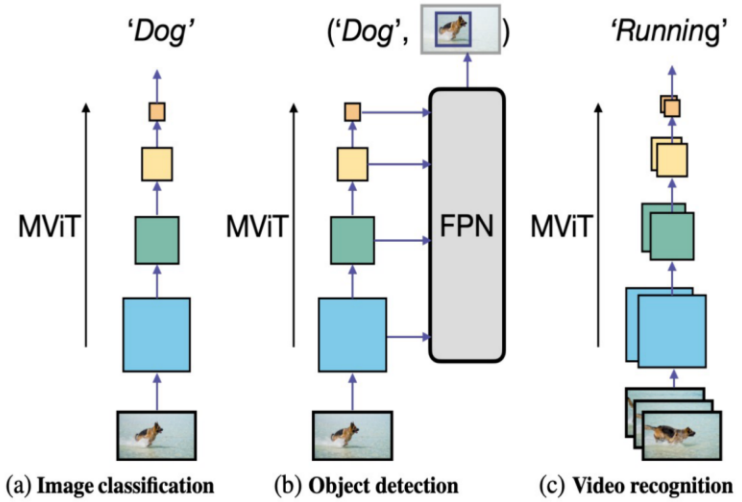
Fan et al, "Multiscale Vision Transformers", ICCV 2021

Swin-V2



Liu et al, "Swin Transformer V2: Scaling up Capacity and Resolution", CVPR 2022

Improved MViT

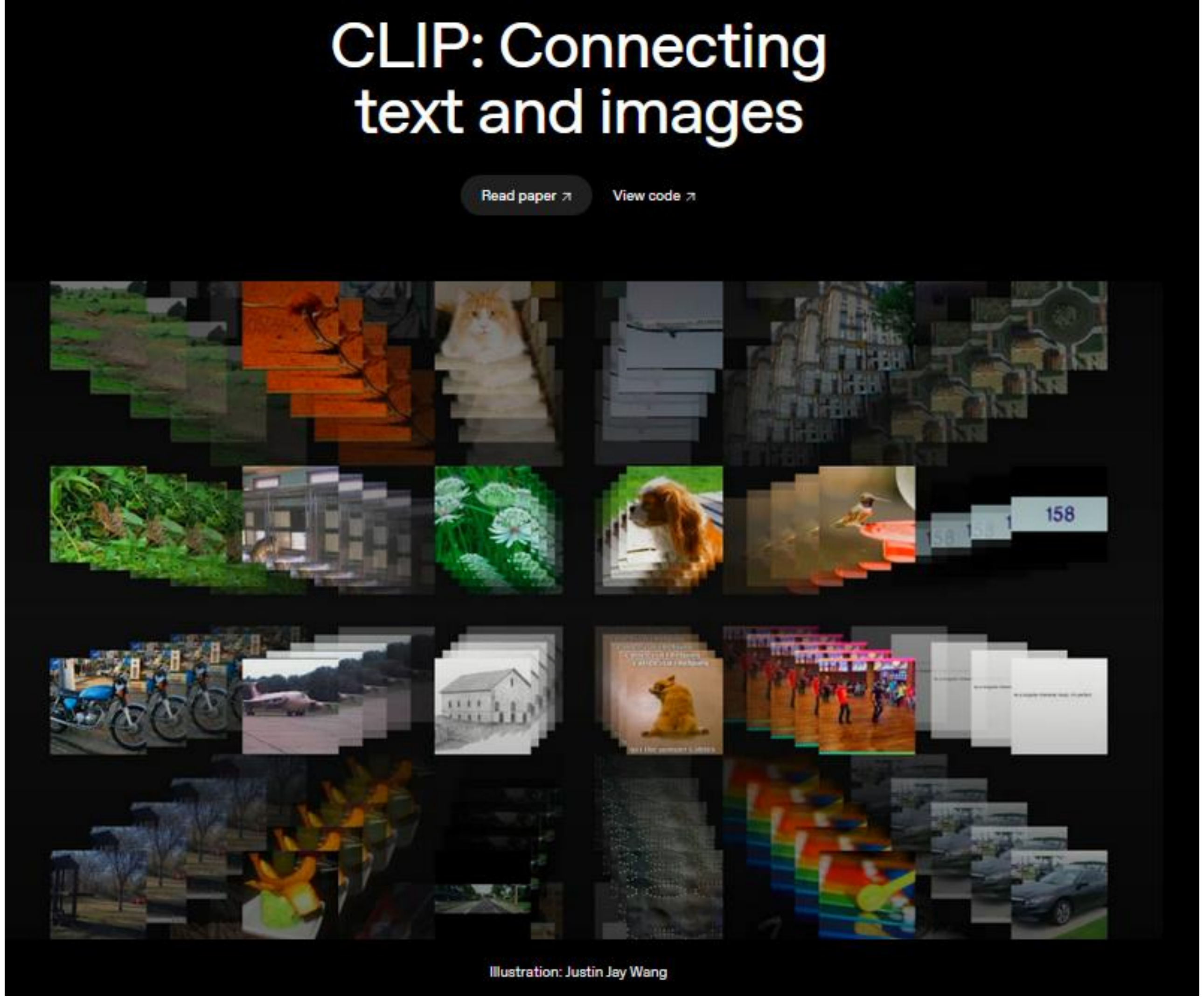


Li et al, "Improved Multiscale Vision Transformers for Classification and Detection", arXiv 2021

Slide credit: Justin Johnson

Introduction to Vision-Language Models


Pre-Trained Vision Model
?
Pre-Trained Language Model



Learning Transferable Visual Models From Natural Language Supervision

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, Ilya Sutskever

https://openai.com/index/clip/

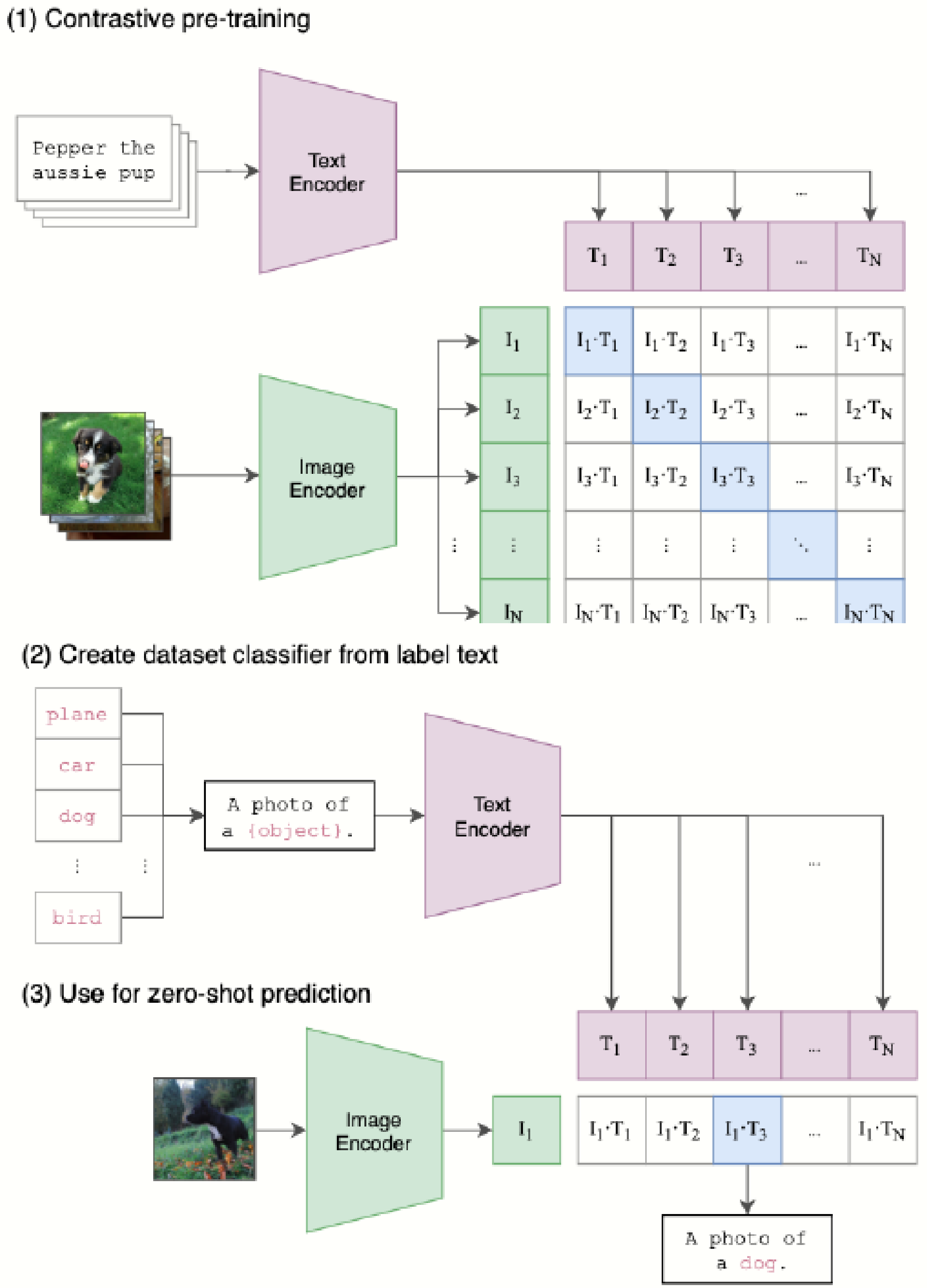
Problem Statement

- Create robust vision models with natural language supervision ⟷ - Involved training both a text encoder as well as an image encoder
- Go beyond previous limitations on models with specific labels ⟷ - ResNets/models were relatively limited to ImageNet classifications
- Enable zero-shot transfer to unseen tasks ⟷ - Utilize a metric ton of image-text data available without retraining and dependency on task-specific datasets

Slides by Gabriela Sanchez and Azeez Ishaqui

Approach

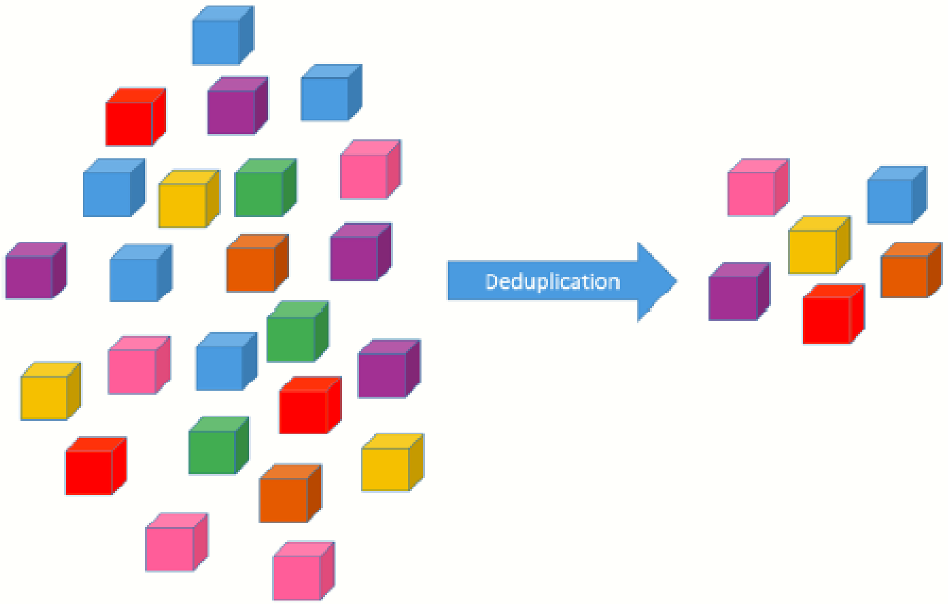
- What is CLIP? Contrastive Language-Image Pre-training
- 400M (image, text) pairs collected from various internet sources
- Image encoder piece: Modified ResNet or Vision Transformer (ViT)
  - Picked based on performance
- Text encoder: Transformer with 63M parameters



Slides by Gabriela Sanchez and Azeez Ishaqui

Approach (Data Collection)

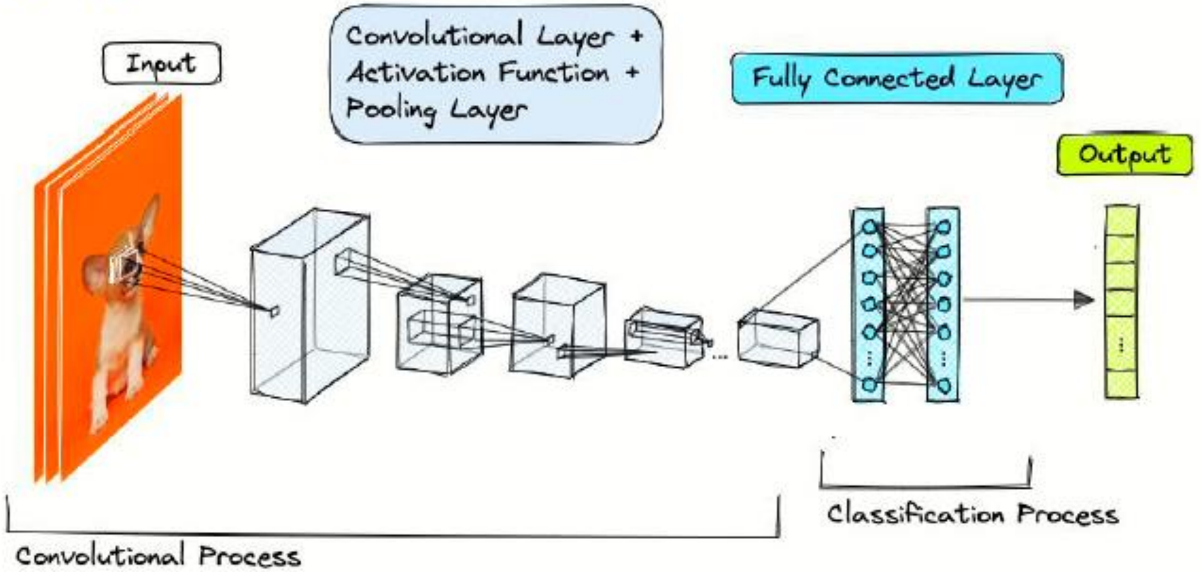
- Raw web pairs aren't going to be perfect
  - Plenty of noise and even mismatches, abstract pairs
  - Either way → CLIP gets stronger with weird stuff
- CLIP filtering
  - 500,000 unique internet queries to cover all domains
  - Pulled in captions, descriptions, comments any kind of data paired with images
  - 1 query could produce max most relevant 20k image-text pairs, ensuring diversity
- De-duplication
  - Image text pairs underwent de-duplication which just ensures overlap is minimal
  - Each sample should ideally be unique
  - Also lowers overlap with benchmarking datasets, → real evaluation and generalization capabilities

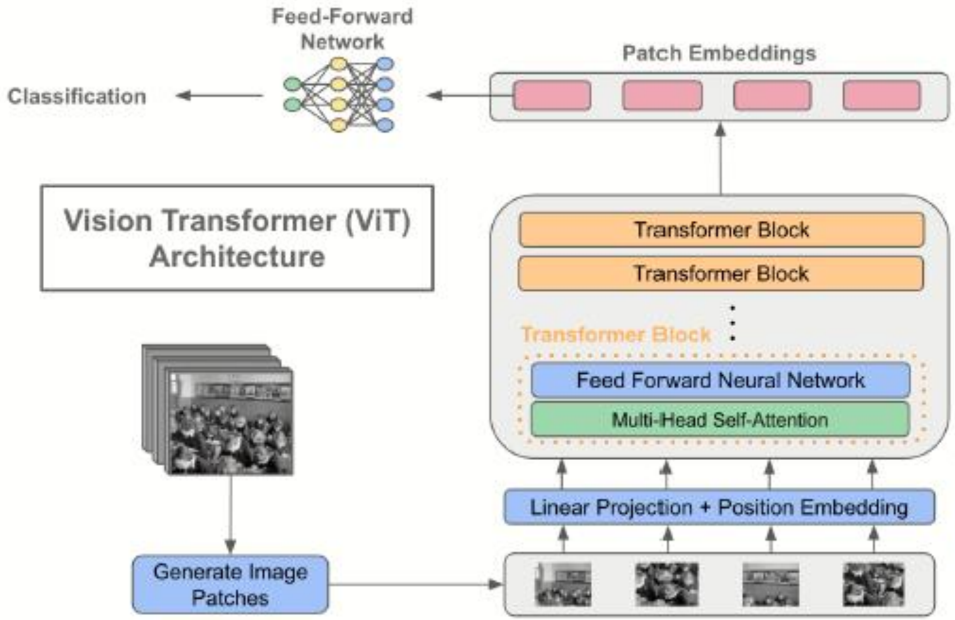


Slides by Gabriela Sanchez and Azeez Ishaqui

Approach (Image Encodings)

- ResNet encoder
  - CNN architecture, conv layers + pooling → feature vector
  - Linear layer for final embedding, L2 normed for ease of similarity
- ViT encoder
  - Patches over image, flattened and projected into embedding (like with text)
  - Positional encodings for those patches, multi-head self attention + feedforward neural nets are strong
  - A classification token is added onto the patch embeddings sequence, then normalized too

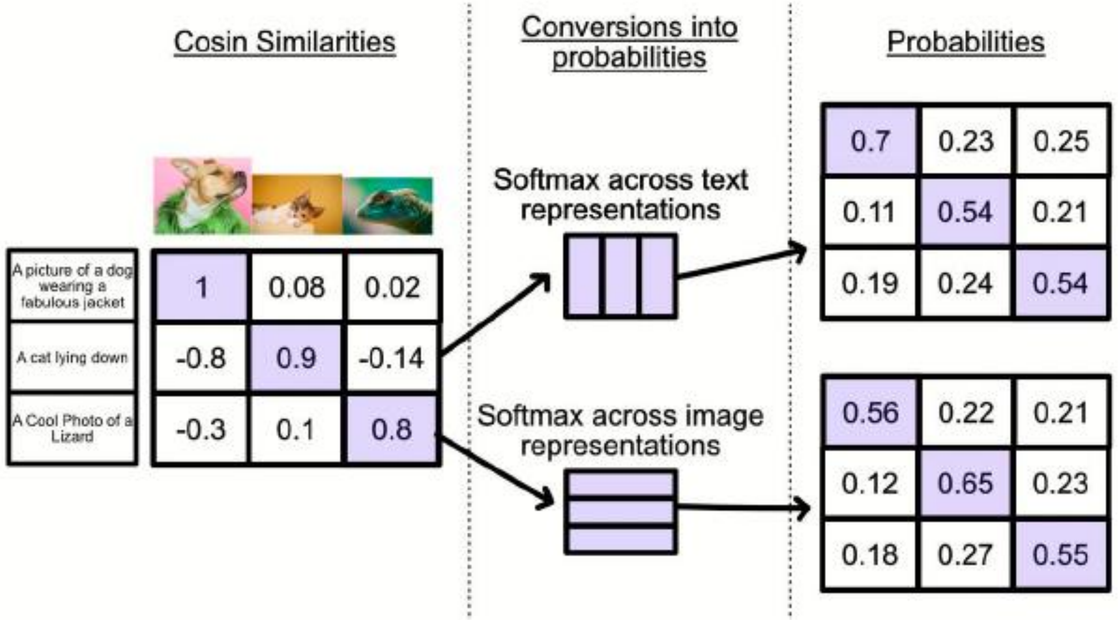


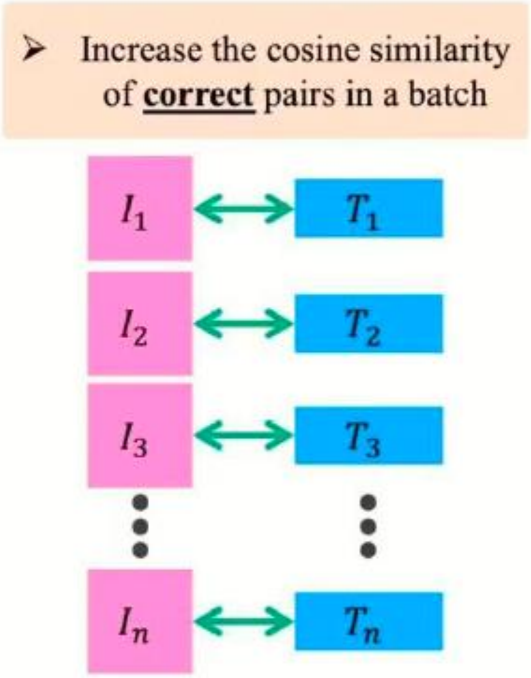


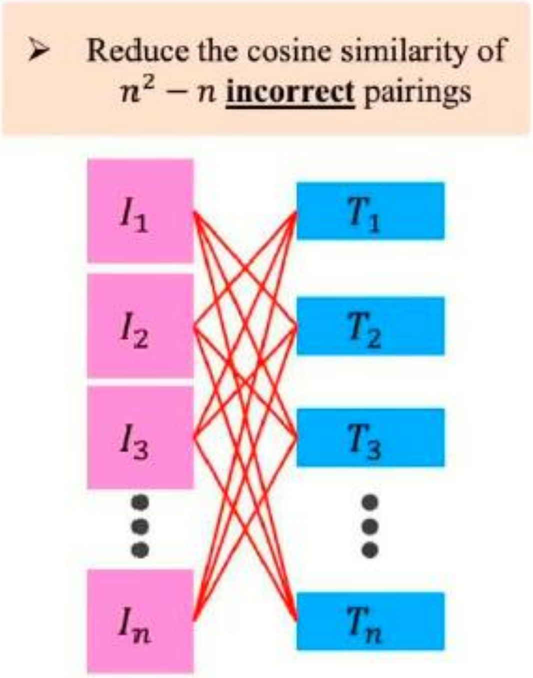
Slides by Gabriela Sanchez and Azeez Ishaqui

Approach (Similarity)

- Cosine similarity explanation





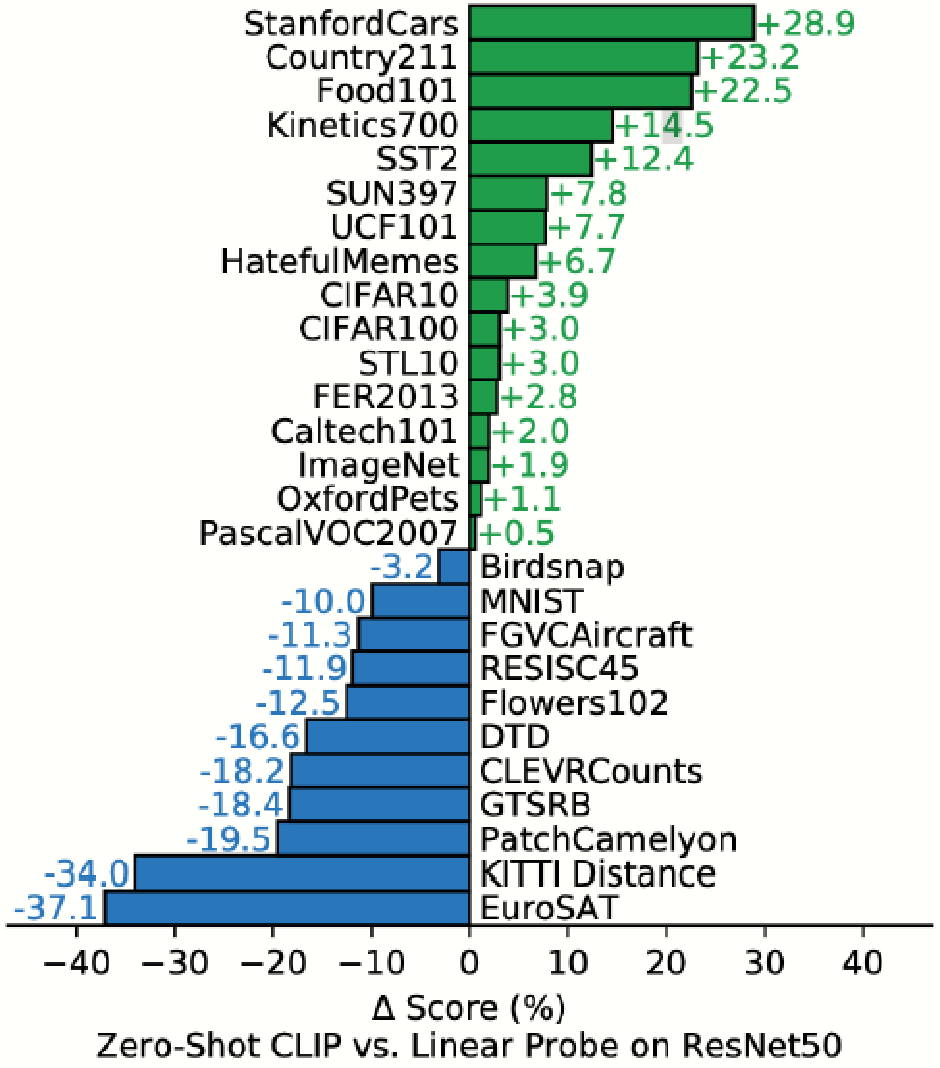


Georgia Tech

```
# 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
```

Experiments and Results (Linear Probe)

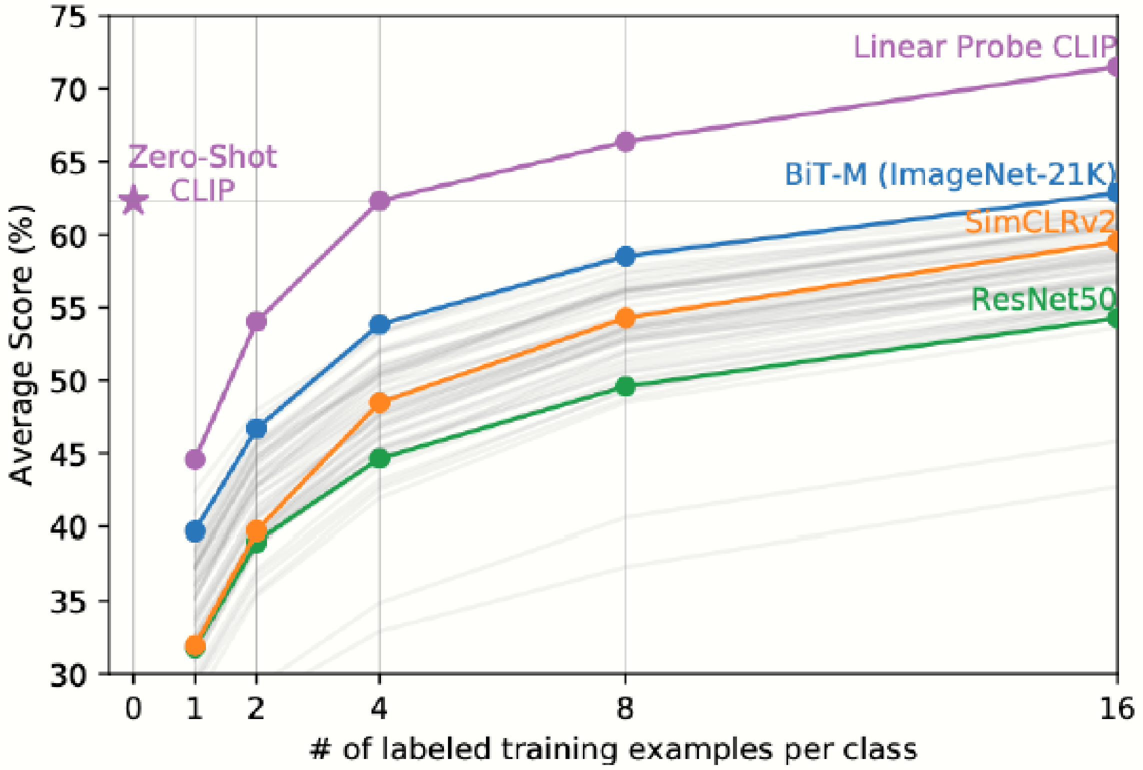
- Linear probe is a simple classifier (log reg) added to pre-trained features some labeled data
  - Beat logistic regression on ResNet50 features on 16/27 datasets – multimodal training power
  - Significance? ROBUST, no task-specific data or fine-tuning needed
  - Particularly good at general object recognition Food101, StandfordCars
  - Specific context-based understanding like EuroSAT and Satellite Imagery give CLIP more trouble



Experiments and Results (Few-Shot)

- Few-shot learning is training on a couple of examples per class

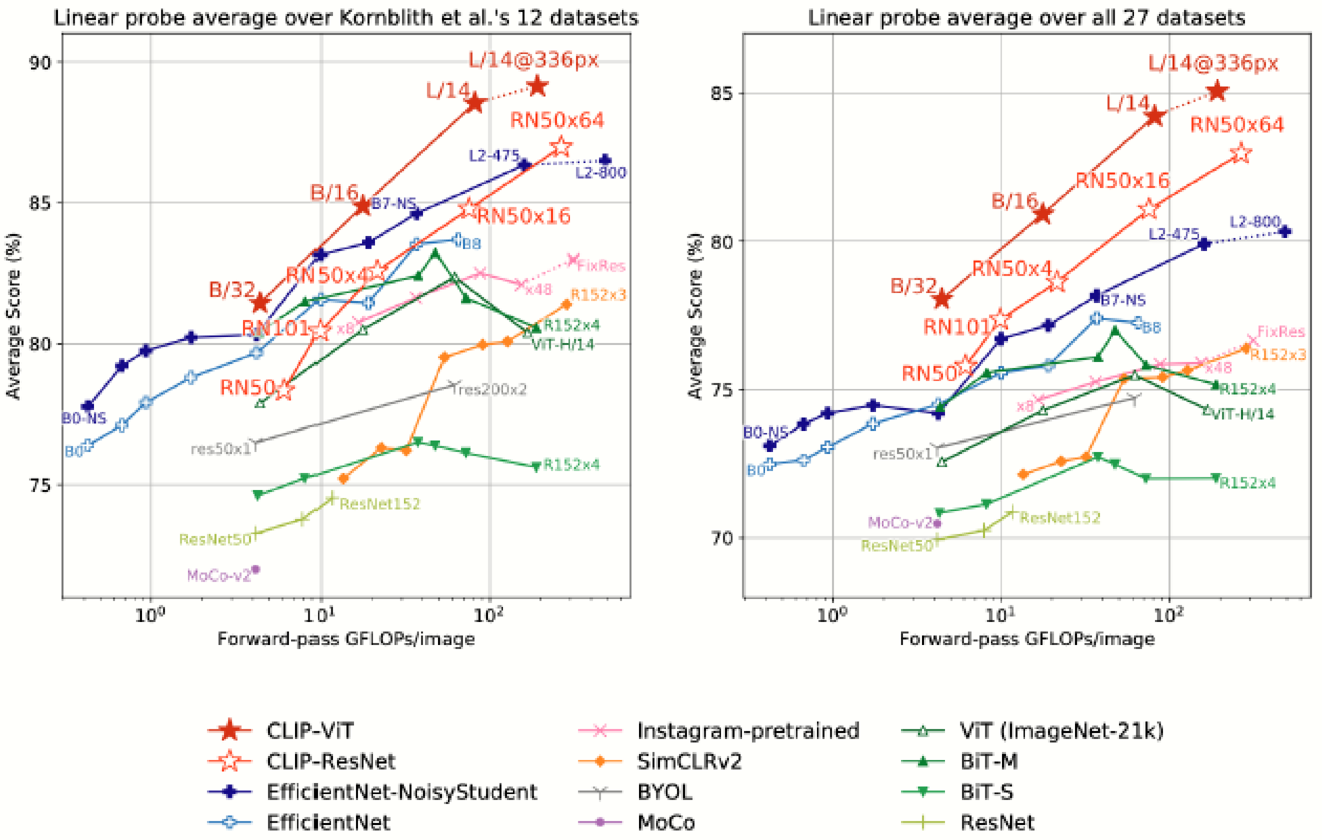
- Outperforms 16-shot classifiers using features from other models
  - Embeddings learned by CLIP capture a plenty of transferable knowledge and can generalize to out of domain concepts



Slides by Gabriela Sanchez and Azeez Ishaqui

Experiments and Results (Scaling)

- Scaling:
  - ViT’s scale well with compute + data
    - ResNets… not so much
  - Learned representations that are not just specific to one type of data
  - Largest CLIP model (ViT-L/14@336px) outperforms existing models by a significant margin
  - 2.6% average improvement
    - CLIP benefits from larger models but strong architectures too which better capture complex relationships



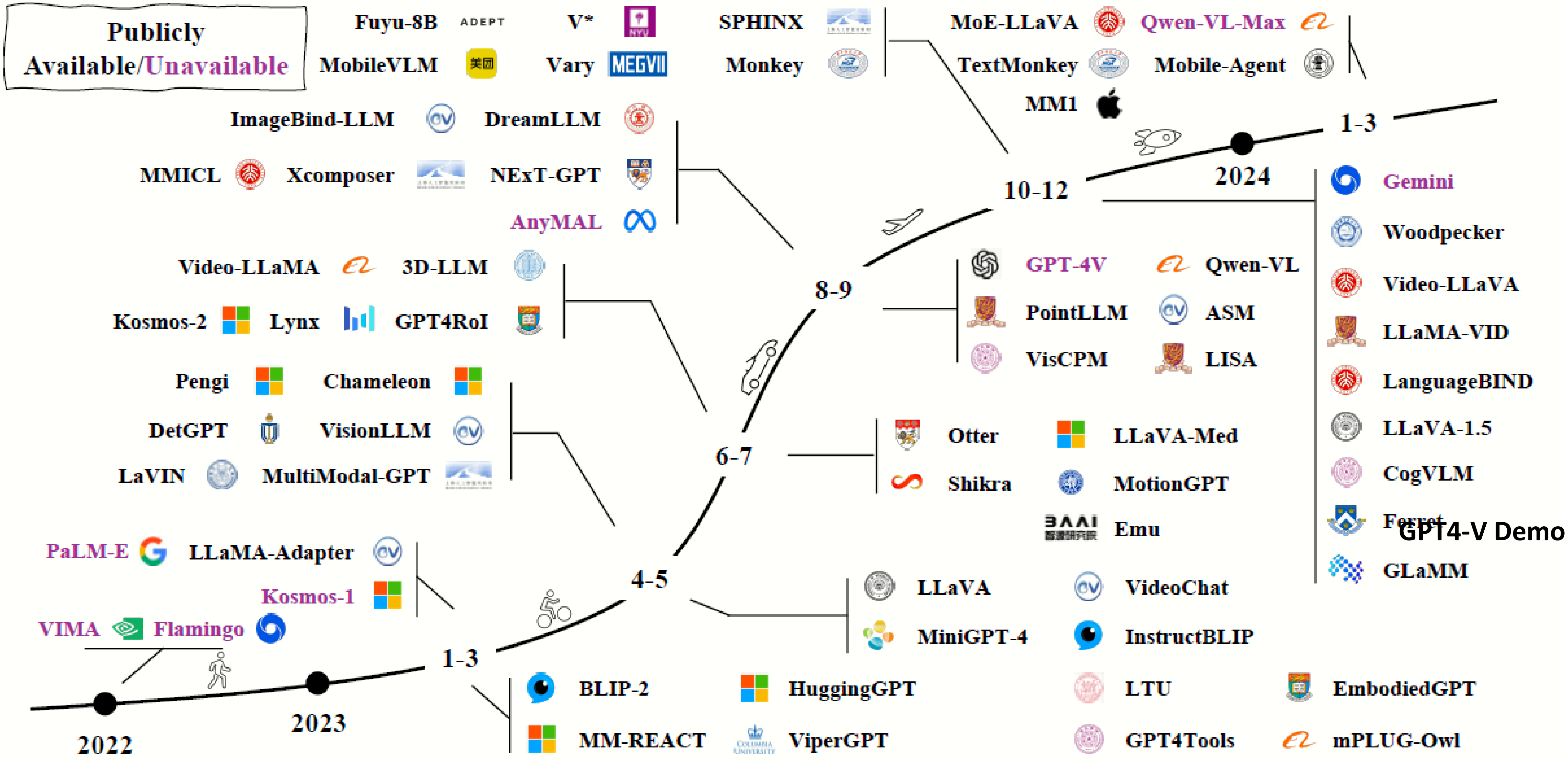
Slides by Gabriela Sanchez and Azeez Ishaqui

Strengths, Weaknesses, Relationships (including limitations)

- Pros:
  - Pre-trained robust zero-shot learning with encoder-backed supervision
  - Adaptable to distribution shifts
  - Scales well with compute
  - Many tasks learned and classifying classes without explicit supervision

- Cons:
  - Not SOTA on all tasks Satellite Imaging, (EuroSAT, RESISC45) etc
  - 1000x more compute to reach SOTA?!
  - “Prompt engineering” effects, like adding "child" to categories list reduced misclassification of young people into incorrect categories from 32.3% to 8.7%
  - Societal issues – surveillance/privacy
  - Data Overlap (3.2% testing dataset avg)

Introduction



Yin et al., A Survey on Multimodal Large Language Models