

CS163: Deep Learning for Computer Vision

Lecture 12: Transformers and Vision Transformers (ViT)

 MiniPlaces Leaderboard 2024 

Top 20% Students (out of 124 submissions)

1.	 Nathan Wei (82.37%)	04. JC (73.04%)	15. JW (64.52%)
2.	 Chanh Tran (82.12%)	05. HS (72.90%)	16. RW (64.36%)
3.	 Leon Liu (75.14%)	06. AL (71.74%)	17. TM (64.12%)
		07. LP (71.68%)	18. CC (63.21%)
		08. MD (69.78%)	19. RY (63.15%)
		09. YZ (69.65%)	20. JD (63.04%)
		10. YL (69.14%)	21. AP (62.70%)
		11. CV (67.57%)	22. AC (63.52%)
		12. JV (66.46%)	23. AG (62.16%)
		13. LD (66.10%)	24. AT (62.12%)
		14. EJ (65.74%)	25. TK (61.93%)

#1 Nathan Wei, Method Description (82.37%)

1. Using pre-trained backbone(**DINOv2**) on larger dataset (**142M**)
2. Using larger image resolution (128 -> 238)
3. Add another linear layer on top of the existing one. Freeze the remainder of the network for the first few epochs while the new linear layer is learning such that disruptive gradients don't destroy previously-learned features in the shallower layers of the network.
4. Using image augmentations including RandomCrop, RandomErasing

#2 Chanh Tran, Method Description (82.12%)

1. Pre-trained Model with self-supervised learning (**CLIP**)
2. Accelerating fine-tuning with half-precision
3. Learning rate search/tuning

#3 Leon Liu, Method Description (75.14%)

1. Using CutMix to improving model's robustness
2. Grid search for best hyper-parameters
3. Using better learning rate scheduler
4. Explored model ensemble of EfficientNetv2 and ConvNext
5. Explored model EMA

#4 Jun Yu Chen, Method Description (73.04%)

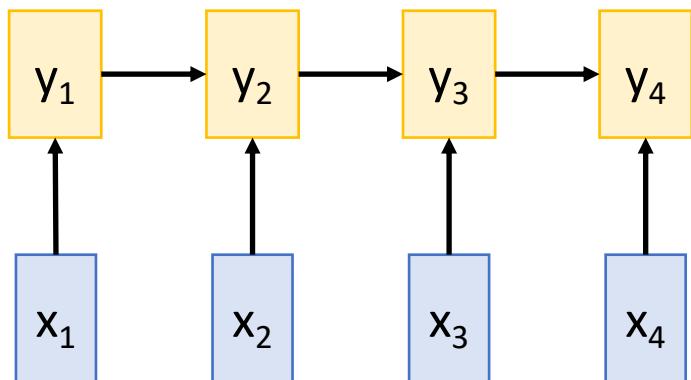
1. Crazy amount of models for voting: 9 pre-trained models + Fine-tuned FC layer
2. Using each model's validation accuracy as prediction confidence.

#5 Harkanwar Singh, Method Description (72.90%)

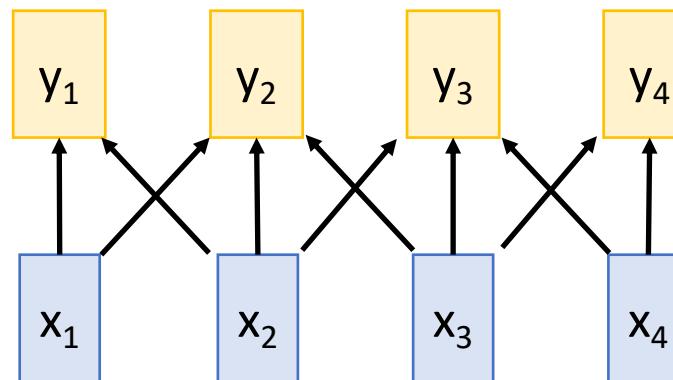
1. Tried RegNet and ResNet architecture
2. Fine-tune full model with carefully designed learning rate for different layers
3. Using data augmentation including Noise Perturbation, color jitter, random flip
4. Using label smoothing technique to improve model's robustness

Three Ways of Processing Sequences

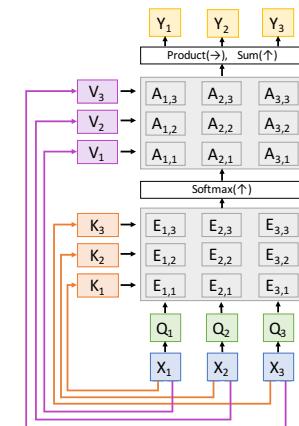
Recurrent Neural Network



1D Convolution



Self-Attention



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

Works on **Multidimensional Grids**

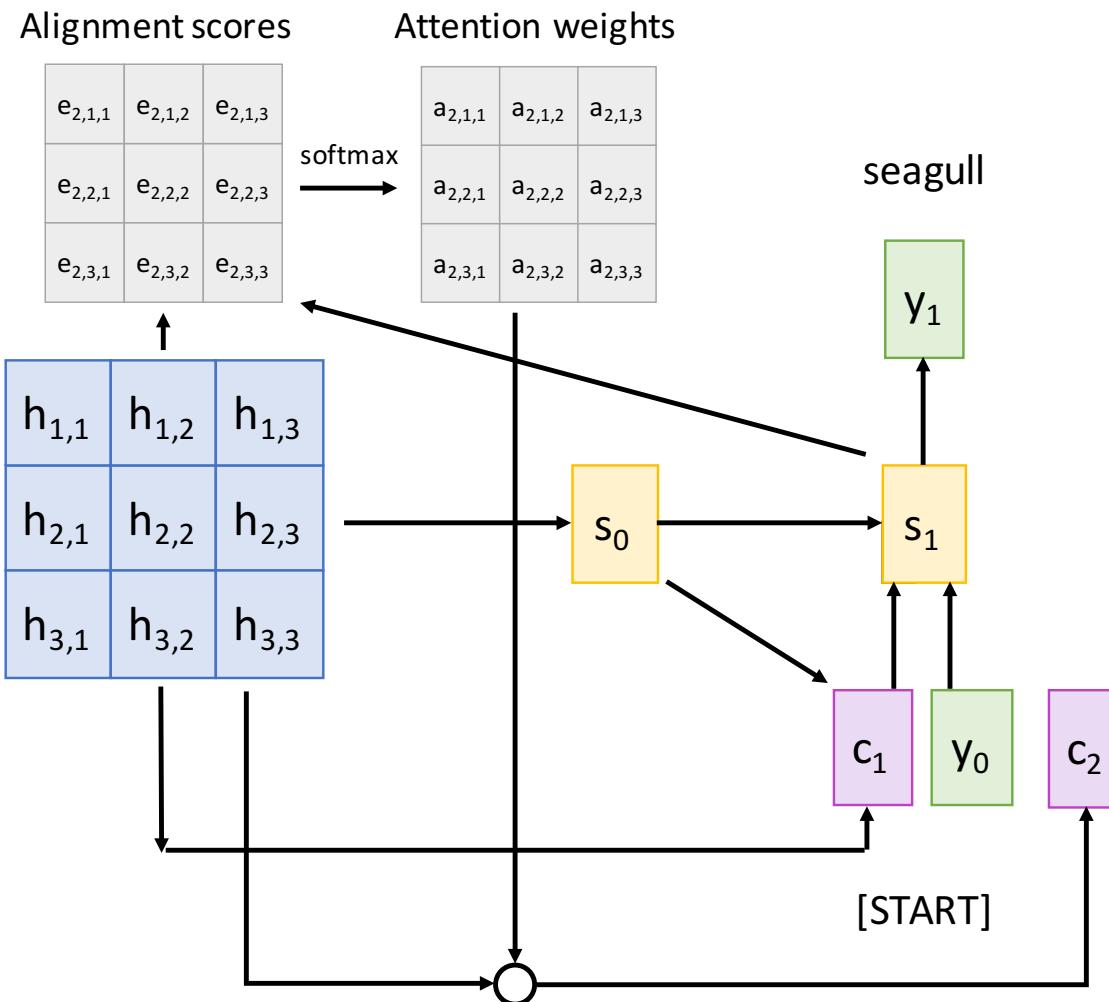
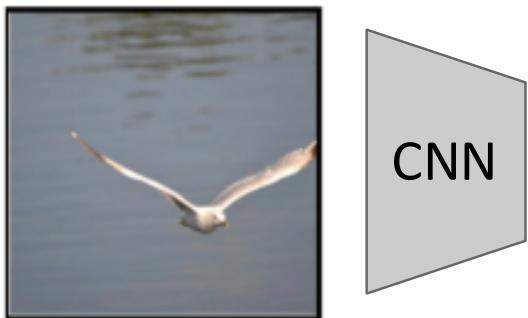
- (-) Bad at long sequences: Need to stack many conv layers for outputs to "see" the whole sequence
- (+) Highly parallel: Each output can be computed in parallel

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

Last Time: Attention

$$\begin{aligned} e_{t,i,j} &= f_{att}(s_{t-1}, h_{i,j}) \\ a_{t,:,:} &= \text{softmax}(e_{t,:,:}) \\ c_t &= \sum_{i,j} a_{t,i,j} h_{i,j} \end{aligned}$$



Last Time: Self-Attention Layer

Inputs:

Input vectors: X (Shape: $N_x \times D_x$)

Key matrix: W_K (Shape: $D_x \times D_Q$)

Value matrix: W_V (Shape: $D_x \times D_V$)

Query matrix: W_Q (Shape: $D_x \times D_Q$)

Computation:

Query vectors: $Q = XW_Q$

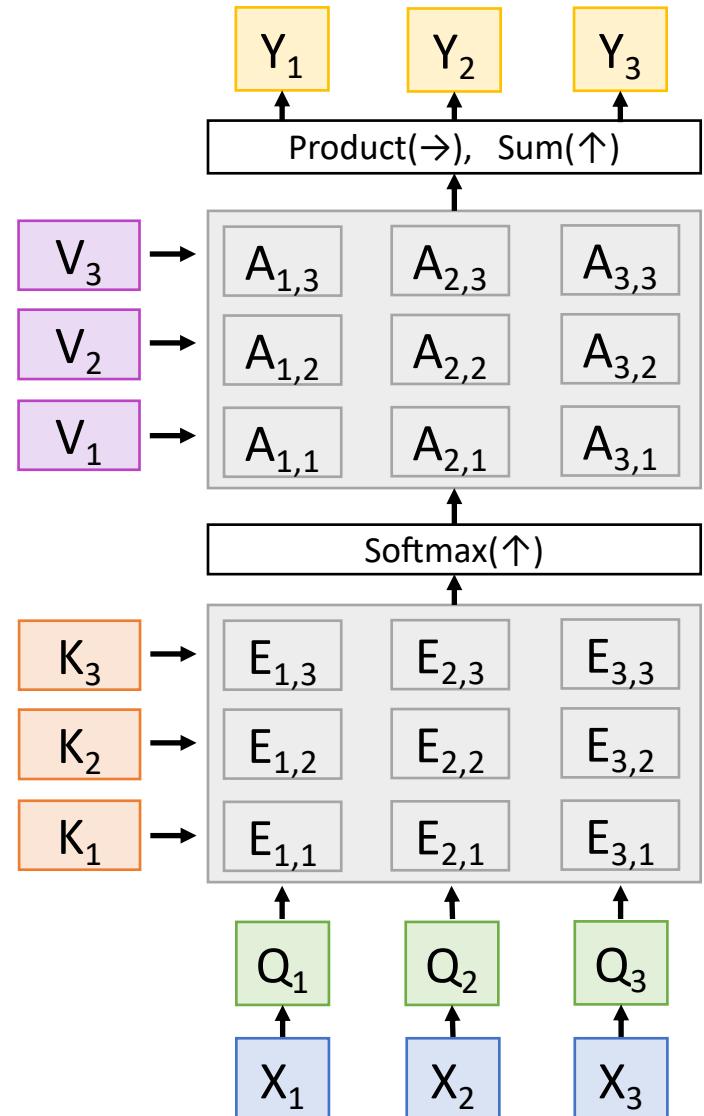
Key vectors: $K = XW_K$ (Shape: $N_x \times D_Q$)

Value Vectors: $V = XW_V$ (Shape: $N_x \times D_V$)

Similarities: $E = QK^T / \sqrt{D_Q}$ (Shape: $N_x \times N_x$) $E_{i,j} = (Q_i \cdot K_j) / \sqrt{D_Q}$

Attention weights: $A = \text{softmax}(E, \text{dim}=1)$ (Shape: $N_x \times N_x$)

Output vectors: $Y = AV$ (Shape: $N_x \times D_V$) $Y_i = \sum_j A_{i,j} V_j$



Three Ways of Processing Sequences

Recurrent Neural Network

1D Convolution

Self-Attention

Attention is all you need

Vaswani et al, NeurIPS 2017

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

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

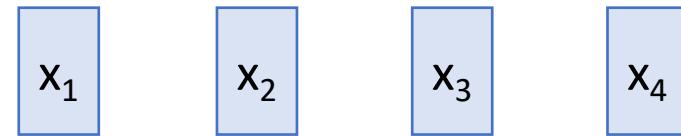
Works on **Sets of Vectors**

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

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

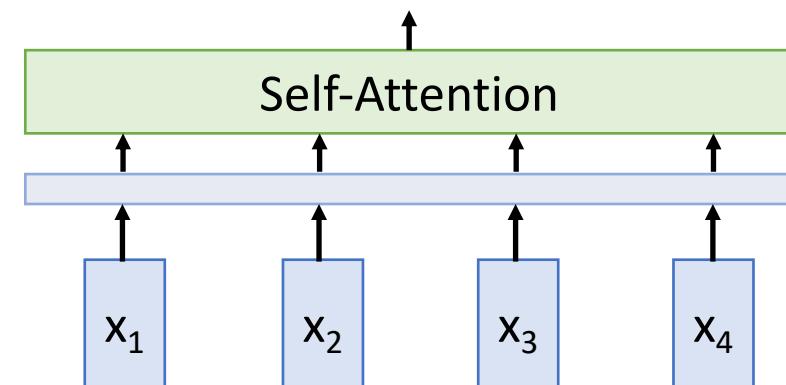
(-) **Very memory intensive**

The Transformer



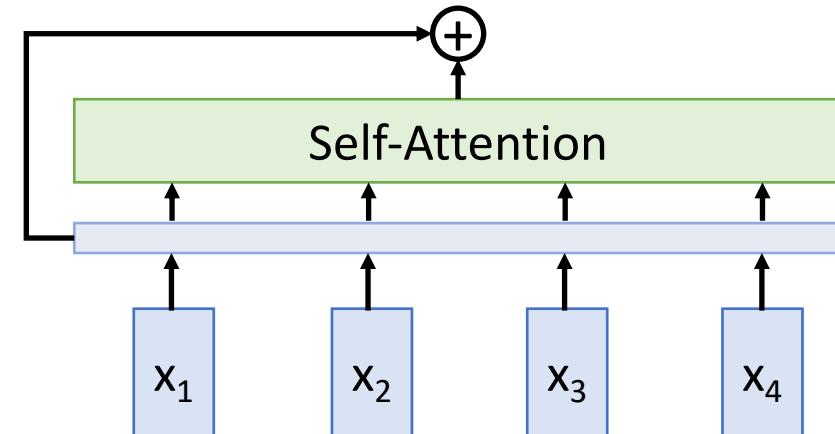
The Transformer

All vectors interact
with each other



The Transformer

Residual connection
All vectors interact
with each other



The Transformer

Recall **Layer Normalization**:

Given h_1, \dots, h_N (Shape: D)

scale: γ (Shape: D)

shift: β (Shape: D)

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

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

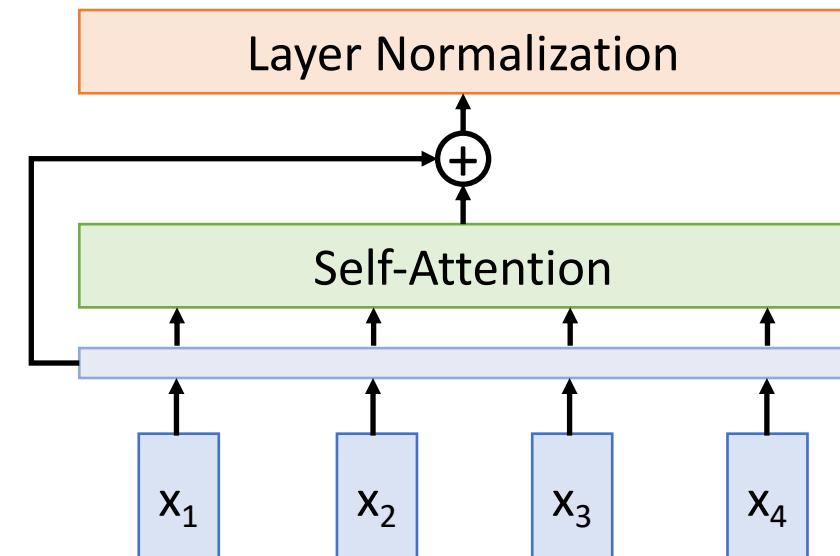
$z_i = (h_i - \mu_i) / \sigma_i$

$y_i = \gamma * z_i + \beta$

Ba et al, 2016

<https://arxiv.org/pdf/1607.06450.pdf>

Residual connection
All vectors interact
with each other



The Transformer

Recall **Layer Normalization**:

Given h_1, \dots, h_N (Shape: D)

scale: γ (Shape: D)

shift: β (Shape: D)

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

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

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

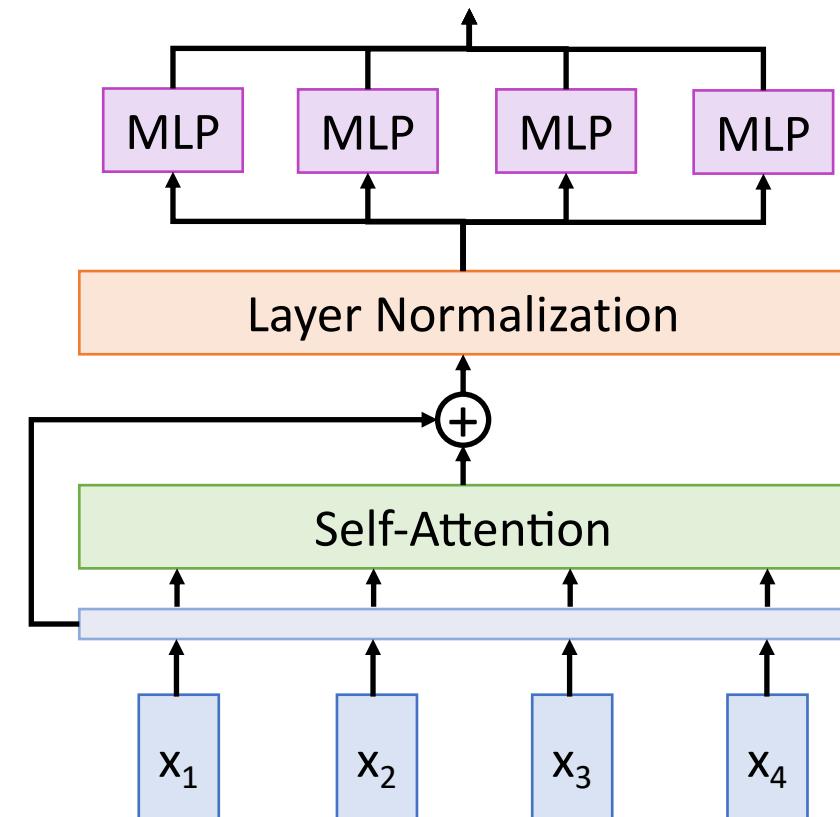
$y_i = \gamma * z_i + \beta$

MLP independently
on each vector

Residual connection

All vectors interact
with each other

Ba et al, 2016



The Transformer

Recall **Layer Normalization**:

Given h_1, \dots, h_N (Shape: D)

scale: γ (Shape: D)

shift: β (Shape: D)

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

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

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

$y_i = \gamma * z_i + \beta$

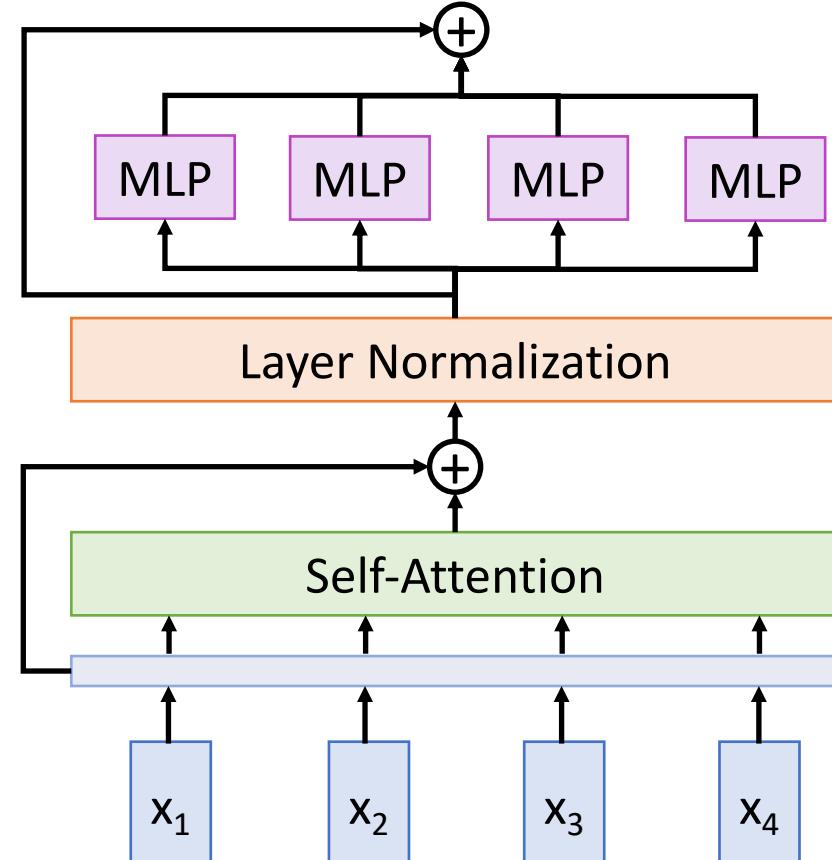
Ba et al, 2016

Residual connection

MLP independently
on each vector

Residual connection

All vectors interact
with each other



The Transformer

Recall Layer Normalization:

Given h_1, \dots, h_N (Shape: D)

scale: γ (Shape: D)

shift: β (Shape: D)

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

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

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

$y_i = \gamma * z_i + \beta$

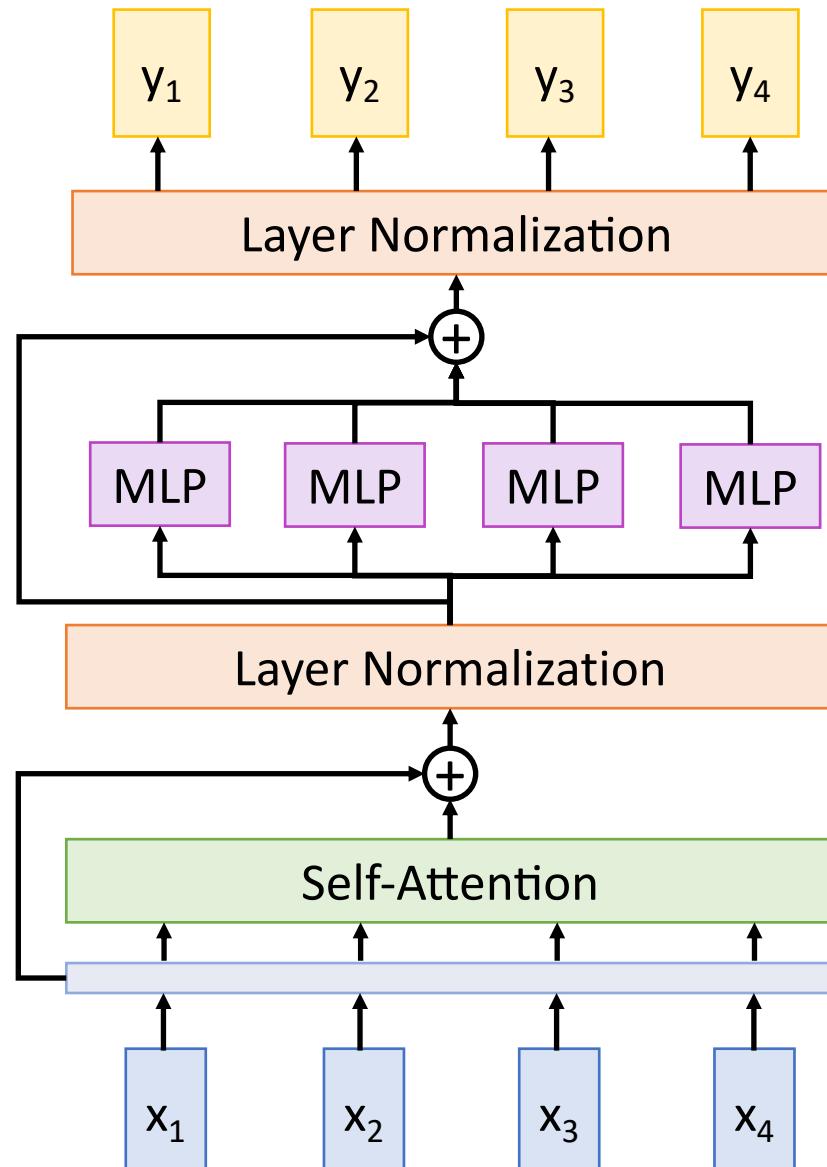
Ba et al, 2016

Residual connection

MLP independently
on each vector

Residual connection

All vectors interact
with each other



The Transformer

Transformer Block:

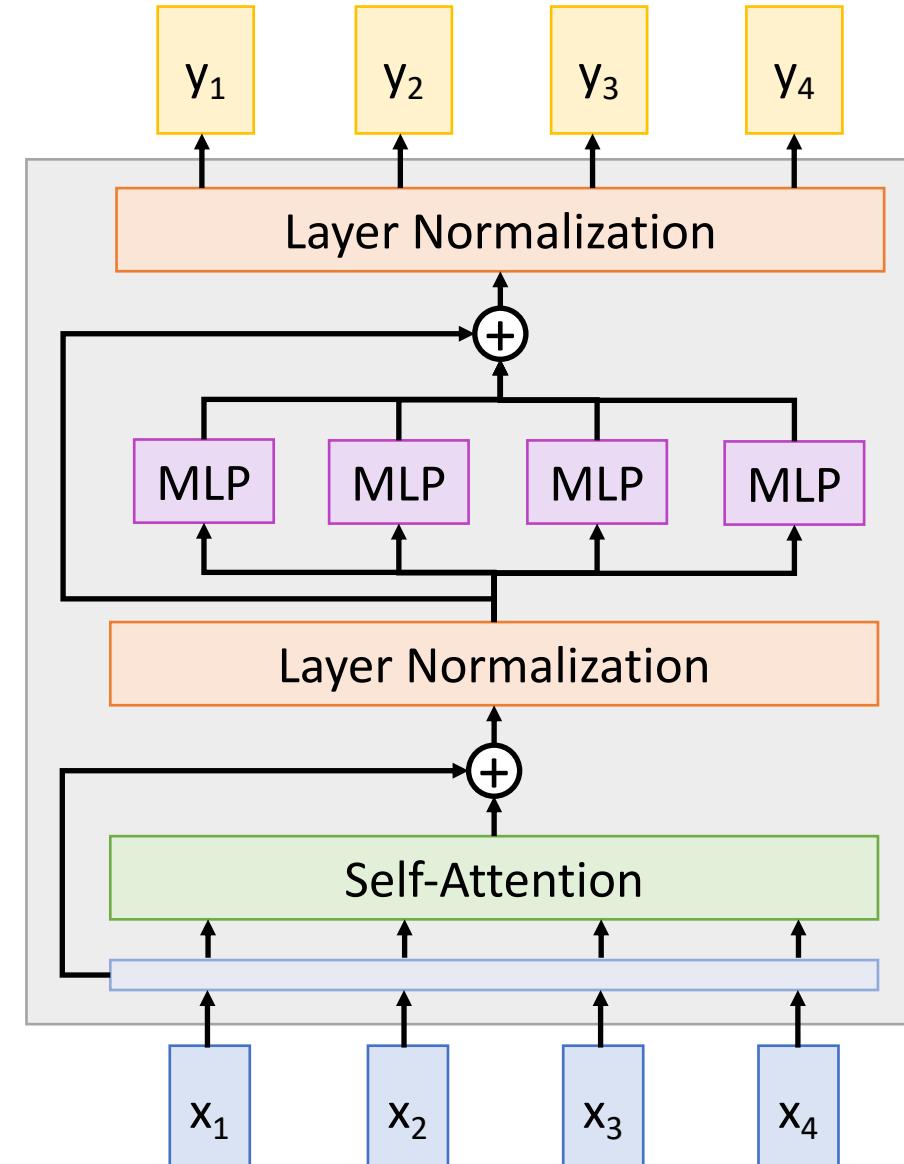
Input: Set of vectors x

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



The Transformer

Transformer Block:

Input: Set of vectors x

Output: Set of vectors y

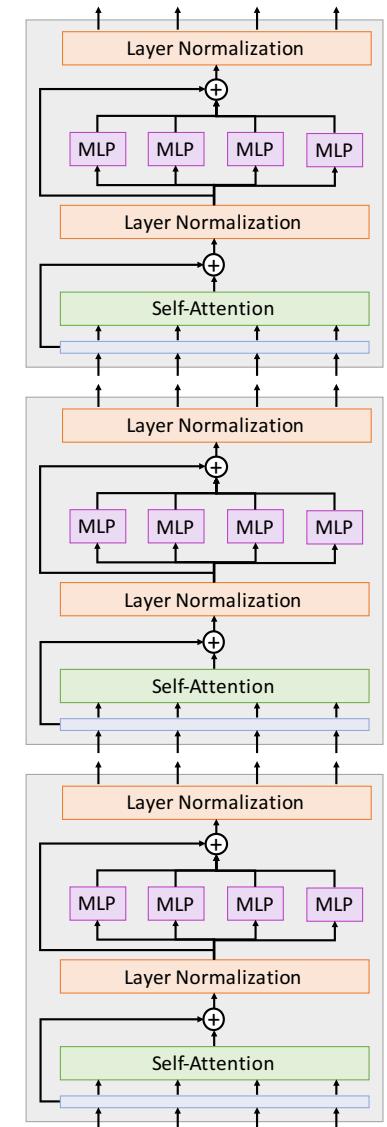
Self-attention is the only interaction between vectors!

Layer norm and MLP work independently per vector

Highly scalable, highly parallelizable

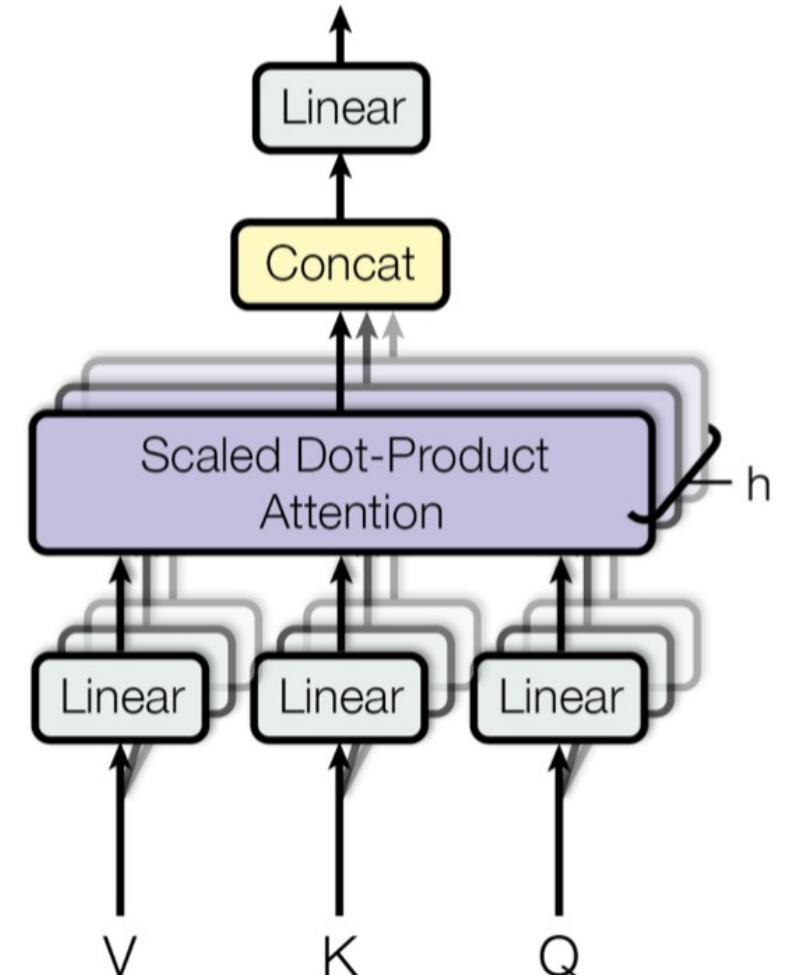
A **Transformer** is a sequence of transformer blocks

Vaswani et al:
12 blocks, $D_Q=512$, 6 heads



Multi-Head Attention

Multi-head Attention is a module for attention mechanisms which runs through an attention mechanism several times in parallel. The independent attention outputs are then concatenated and linearly transformed into the expected dimension.



The Transformer: Transfer Learning

“ImageNet Moment for Natural Language Processing”

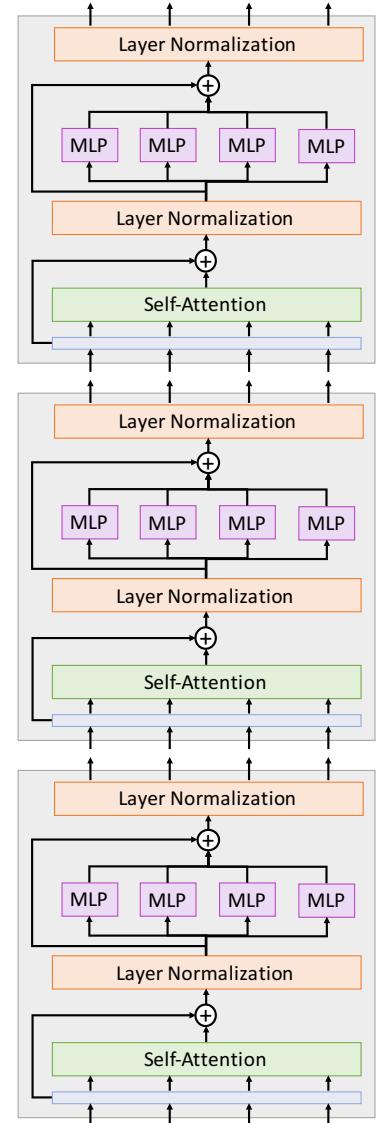
Pretraining:

Download a lot of text from the internet

Train a giant Transformer model for language modeling

Finetuning:

Fine-tune the Transformer on your own NLP task



Scaling up Transformers

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)

Scaling up Transformers

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	

Devlin et al, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", EMNLP 2018

Scaling up Transformers

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)

Yang et al, "XLNet: Generalized Autoregressive Pretraining for Language Understanding", 2019
Liu et al, "RoBERTa: A Robustly Optimized BERT Pretraining Approach", 2019

Scaling up Transformers

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	

Radford et al, "Language models are unsupervised multitask learners", 2019

Scaling up Transformers

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)

Shoeybi et al, "Megatron-LM: Training Multi-Billion Parameter Language Models using Model Parallelism", 2019

Scaling up Transformers

~\$430,000 on Amazon AWS!

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)

Scaling up Transformers

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

Scaling up Transformers

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	12288	96	175B	694GB	\$ 4.6 million

Generated Text from GPT-3

Bold text: Input prompt written by humans
Italics: Completion by GPT-3

Title: United Methodists Agree to Historic Split

Subtitle: Those who oppose gay marriage will form their own denomination

Article:

Generated Text from GPT-3

Bold text: Input prompt written by humans
Italics: Completion by GPT-3

Title: United Methodists Agree to Historic Split

Subtitle: Those who oppose gay marriage will form their own denomination

Article: *After two days of intense debate, the United Methodist Church has agreed to a historic split - one that is expected to end in the creation of a new denomination, one that will be "theologically and socially conservative," according to The Washington Post. The majority of delegates attending the church's annual General Conference in May voted to strengthen a ban on the ordination of LGBTQ clergy and to write new rules that will "discipline" clergy who officiate at same-sex weddings. But those who opposed these measures have a new plan: They say they will form a separate denomination by 2020, calling their church the Christian Methodist denomination. The Post notes that the denomination, which claims 12.5 million members, was in the early 20th century the "largest Protestant denomination in the U.S.," but that it has been shrinking in recent decades. The new split will be the second in the church's history. The first occurred in 1968, when roughly 10 percent of the denomination left to form the Evangelical United Brethren Church. The Post notes that the proposed split "comes at a critical time for the church, which has been losing members for years," which has been "pushed toward the brink of a schism over the role of LGBTQ people in the church." Gay marriage is not the only issue that has divided the church. In 2016, the denomination was split over ordination of transgender clergy, with the North Pacific regional conference voting to ban them from serving as clergy, and the South Pacific regional conference voting to allow them.*

GPT-3: Programming by prompt

Bold text: Input prompt written by humans
Italics: Completion by GPT-3

Poor English input: I eated the purple berries.

Good English output: I ate the purple berries.

Poor English input: Thank you for picking me as your designer. I'd appreciate it.

Good English output: Thank you for choosing me as your designer. I appreciate it.

Poor English input: The mentioned changes have done. or I did the alteration that you requested. or I changed things you wanted and did the modifications.

Good English output: The requested changes have been made. or I made the alteration that you requested. or I changed things you wanted and made the modifications.

Poor English input: I'd be more than happy to work with you in another project.

Good English output:

GPT-3: Programming by prompt

Bold text: Input prompt written by humans
Italics: Completion by GPT-3

Poor English input: I eated the purple berries.

Good English output: I ate the purple berries.

Poor English input: Thank you for picking me as your designer. I'd appreciate it.

Good English output: Thank you for choosing me as your designer. I appreciate it.

Poor English input: The mentioned changes have done. or I did the alteration that you requested. or I changed things you wanted and did the modifications.

Good English output: The requested changes have been made. or I made the alteration that you requested. or I changed things you wanted and made the modifications.

Poor English input: I'd be more than happy to work with you in another project.

Good English output: *I'd be more than happy to work with you on another project.*

GPT-3: Literary Style Transfer

Topic: humor: literary parodies and homages, pastiches, style parodies.

Parodies of the fantasy novel series *Harry Potter* in the style of various famous authors:

By Ernest Hemingway:

"It was a cold day on Privet Drive. A child cried. Harry felt nothing.

GPT-3: Literary Style Transfer

Topic: humor: literary parodies and homages, pastiches, style parodies.

Parodies of the fantasy novel series *Harry Potter* in the style of various famous authors:

By Ernest Hemingway:

"It was a cold day on Privet Drive. A child cried. Harry felt nothing. He was dryer than dust. He had been silent too long. He had not felt love. He had scarcely felt hate. Yet the Dementor's Kiss killed nothing. Death didn't leave him less dead than he had been a second before. It wasn't about living or dying really. It wasn't about death. It was about the Dark Lord, his creator, his engenderer. He was not resurrected. He was created again. He was whole again in the courtyard of Malfoy Manor."

GPT-3: Literary Style Transfer

Topic: humor: literary parodies and homages, pastiches, style parodies.

Parodies of the fantasy novel series *Harry Potter* in the style of various famous authors:

By Ernest Hemingway:

"It was a cold day on Privet Drive. A child cried. Harry felt nothing. He was dryer than dust. He had been silent too long. He had not felt love. He had scarcely felt hate. Yet the Dementor's Kiss killed nothing. Death didn't leave him less dead than he had been a second before. It wasn't about living or dying really. It wasn't about death. It was about the Dark Lord, his creator, his engenderer. He was not resurrected. He was created again. He was whole again in the courtyard of Malfoy Manor."

By Jane Austen:

"It is a truth universally acknowledged, that a broken Harry is in want of a book. This he knows to his cost; pressing that wretched nose against the window of a bookstore in Diagon Alley the last thing before he goes into hiding for several years, where upon, he goes straight to Azkaban."

GPT-3: Literary Style Transfer

Topic: humor: literary parodies and homages, pastiches, style parodies.

Parodies of the fantasy novel series *Harry Potter* in the style of various famous authors:

By Ernest Hemingway:

"It was a cold day on Privet Drive. A child cried. Harry felt nothing. He was drier than dust. He had been silent too long. He had not felt love. He had scarcely felt hate. Yet the Dementor's Kiss killed nothing. Death didn't leave him less dead than he had been a second before. It wasn't about living or dying really. It wasn't about death. It was about the Dark Lord, his creator, his engenderer. He was not resurrected. He was created again. He was whole again in the courtyard of Malfoy Manor."

By Jane Austen:

"It is a truth universally acknowledged, that a broken Harry is in want of a book. This he knows to his cost; pressing that wretched nose against the window of a bookstore in Diagon Alley the last thing before he goes into hiding for several years, where upon, he goes straight to Azkaban."

By Arthur Conan Doyle:

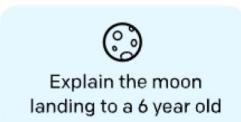
"Harry pushed at the swinging doors of the bookshop hard, and nearly knocked himself unconscious. He staggered in with his ungainly package, his cheeks scarlet with cold and the shame of having chosen the wrong month to go Christmas shopping. The proprietor of the store, however, didn't cast even a cursory glance at him, being far more interested in an enormous hunk of wood lying in the middle of the floor, which certainly looked like a gallows. Yes, the proprietor said to a reedy old man wearing a bowler hat and a forlorn expression that made Harry want to kick him, I can rent you such a gallows for a small fee."

It is difficult for humans to directly interact with GPT-3. Fine-tune an interface!

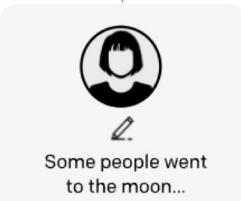
Step 1

Collect demonstration data, and train a supervised policy.

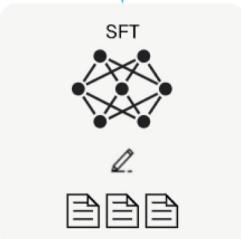
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



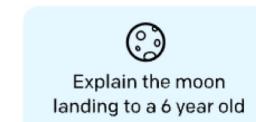
This data is used to fine-tune GPT-3 with supervised learning.



Step 2

Collect comparison data, and train a reward model.

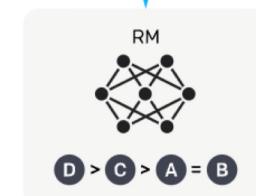
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



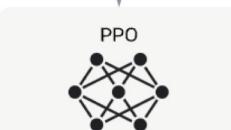
Step 3

Optimize a policy against the reward model using reinforcement learning.

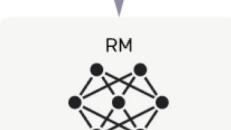
A new prompt is sampled from the dataset.



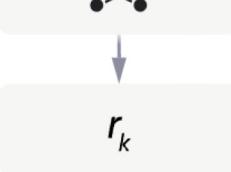
The policy generates an output.



The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.



Self-Attention vs. Cross-Attention

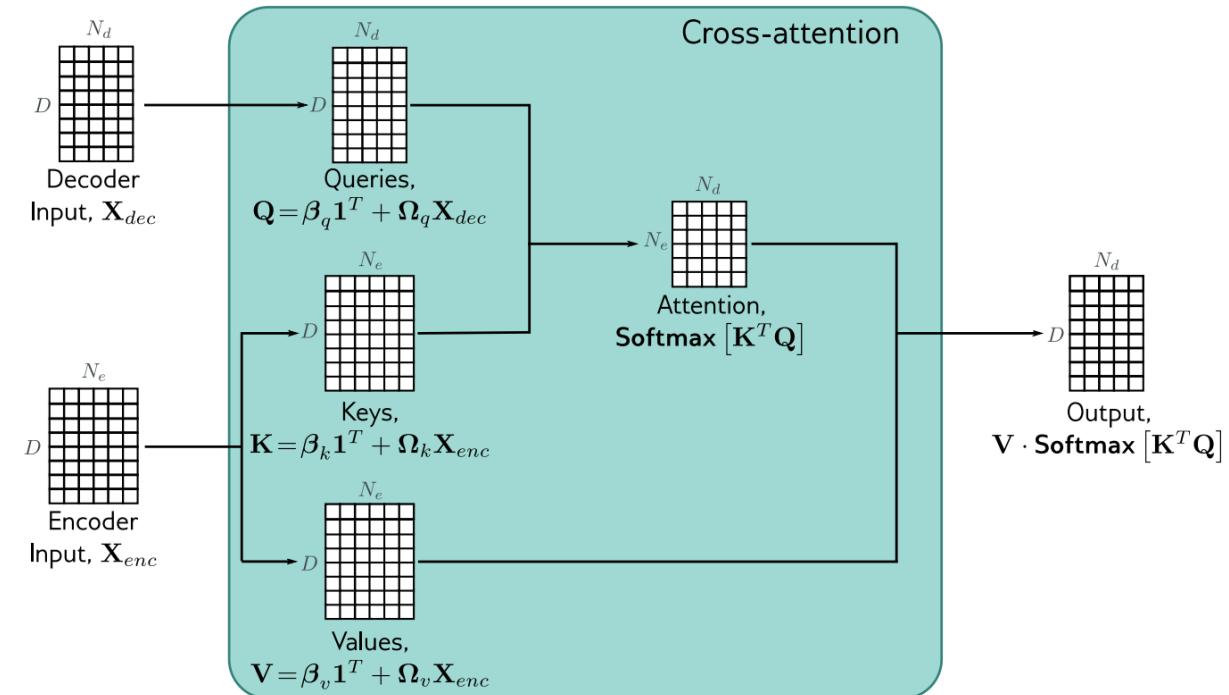
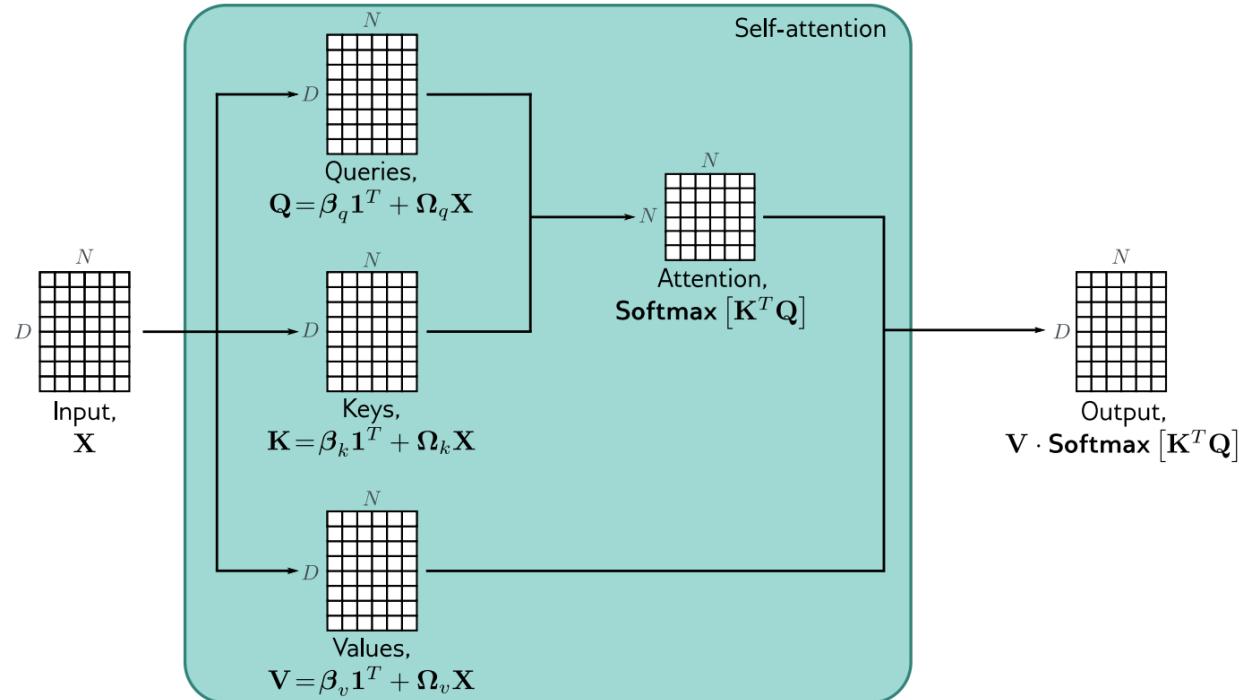


Figure extracted from Understanding Deep Learning textbook

the queries are calculated from the decoder embeddings X_{dec} , and the keys and values from the encoder embeddings X_{enc} . In the context of translation, the encoder contains information about the source language, and the decoder contains information about the target language statistics.

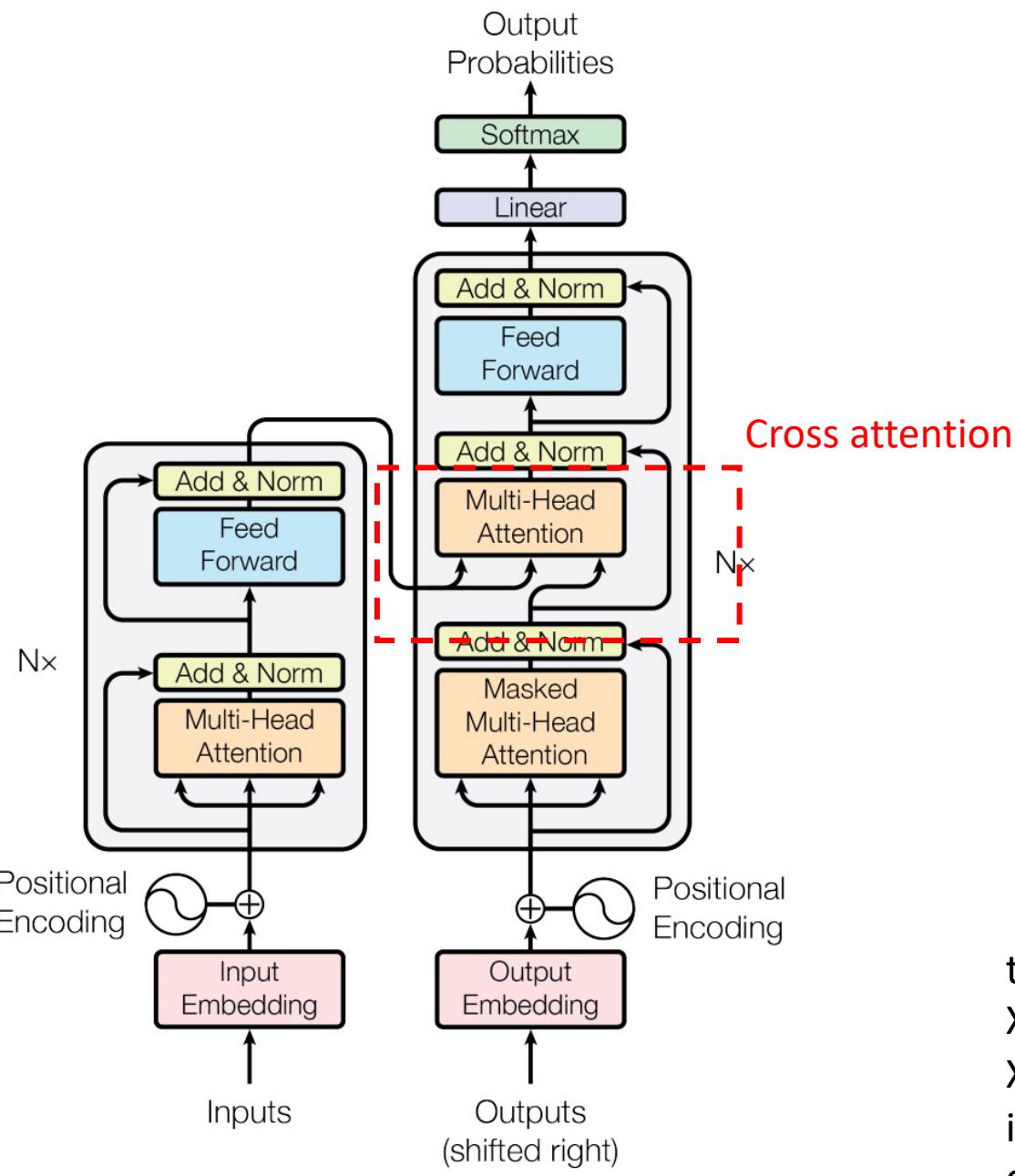


Figure 1: The Transformer - model architecture.

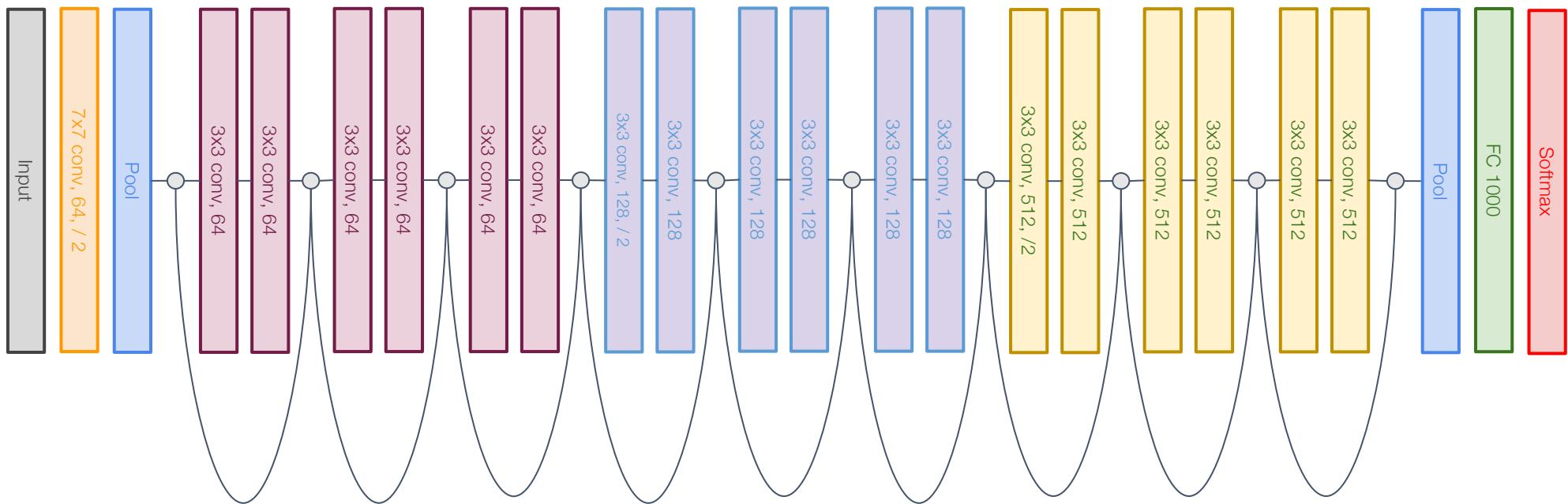
<https://arxiv.org/pdf/1706.03762.pdf>

the queries are calculated from the decoder embeddings X_{dec} , and the keys and values from the encoder embeddings X_{enc} . In the context of translation, the encoder contains information about the source language, and the decoder contains information about the target language statistics.

How to use Attention / Transformers for Vision?

Idea #1: Add attention to existing CNNs

Start from standard CNN architecture (e.g. ResNet)



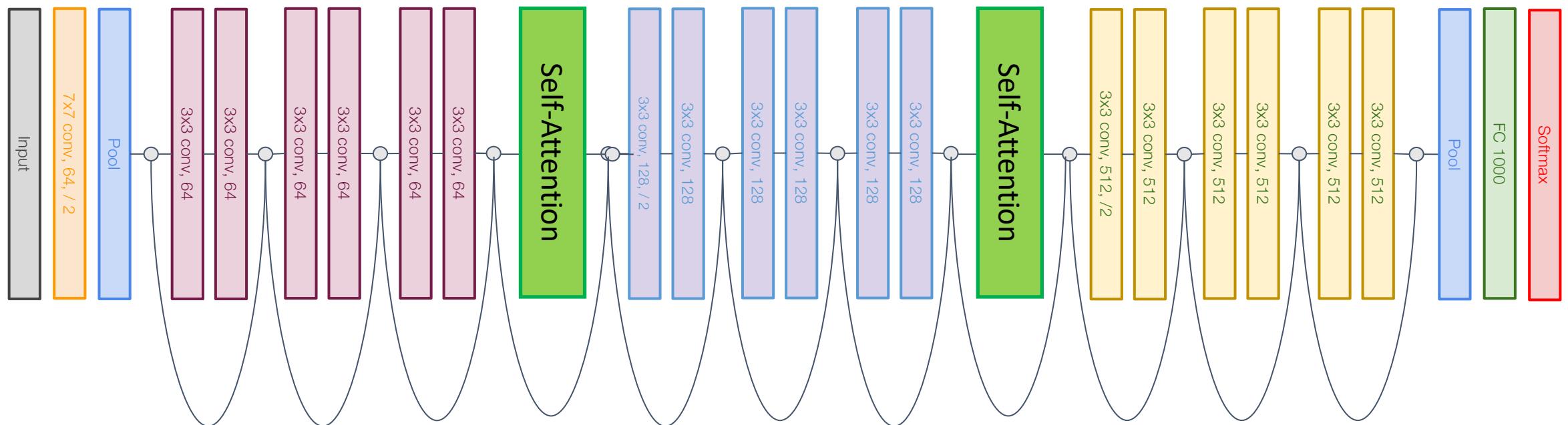
Zhang et al, "Self-Attention Generative Adversarial Networks", ICML 2018

Wang et al, "Non-local Neural Networks", CVPR 2018

Idea #1: Add attention to existing CNNs

Start from standard CNN architecture (e.g. ResNet)

Add Self-Attention blocks between existing ResNet blocks



Zhang et al, "Self-Attention Generative Adversarial Networks", ICML 2018

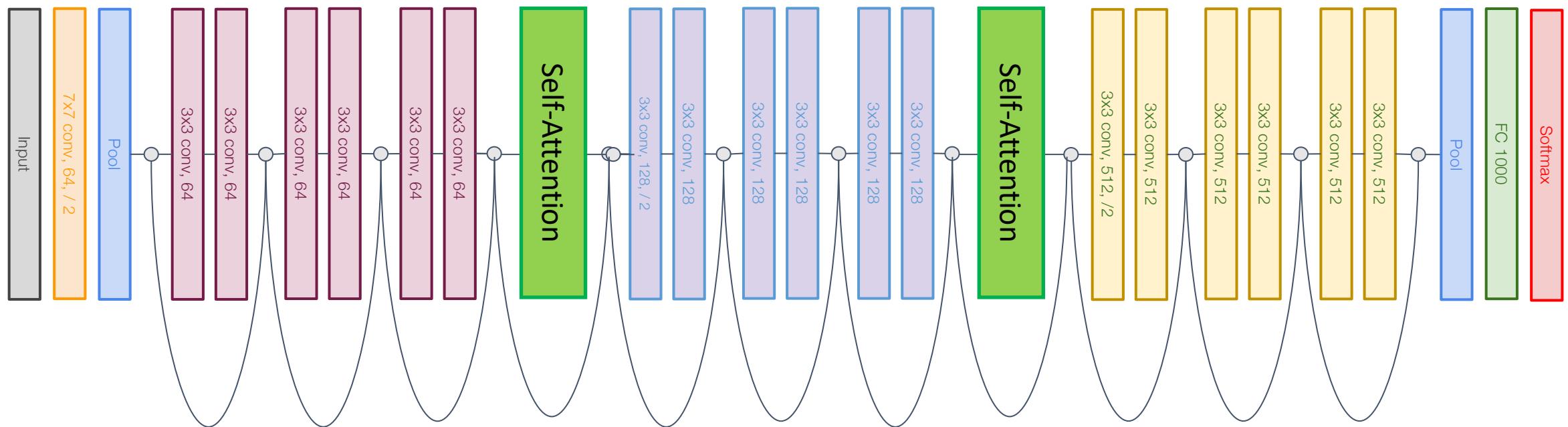
Wang et al, "Non-local Neural Networks", CVPR 2018

Idea #1: Add attention to existing CNNs

Model is still a CNN! Start from standard CNN architecture (e.g. ResNet)

Can we replace

convolution entirely? Add Self-Attention blocks between existing ResNet blocks

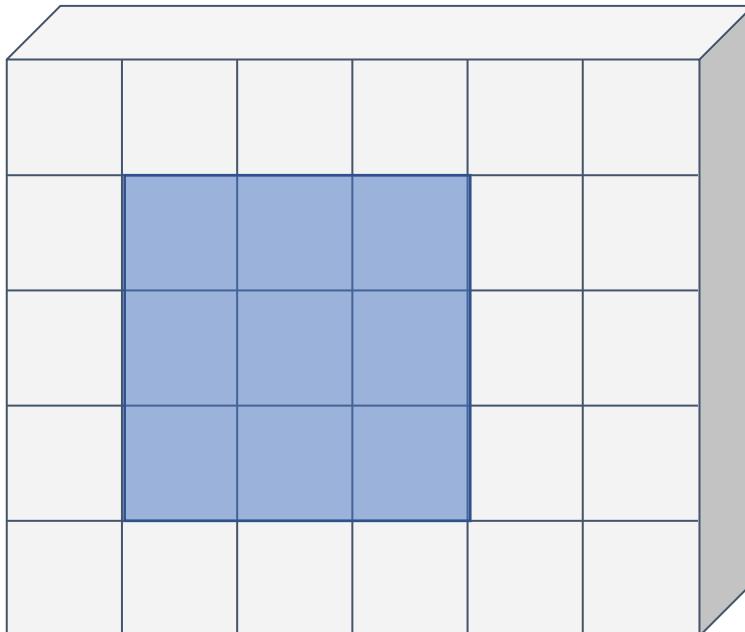


Zhang et al, "Self-Attention Generative Adversarial Networks", ICML 2018

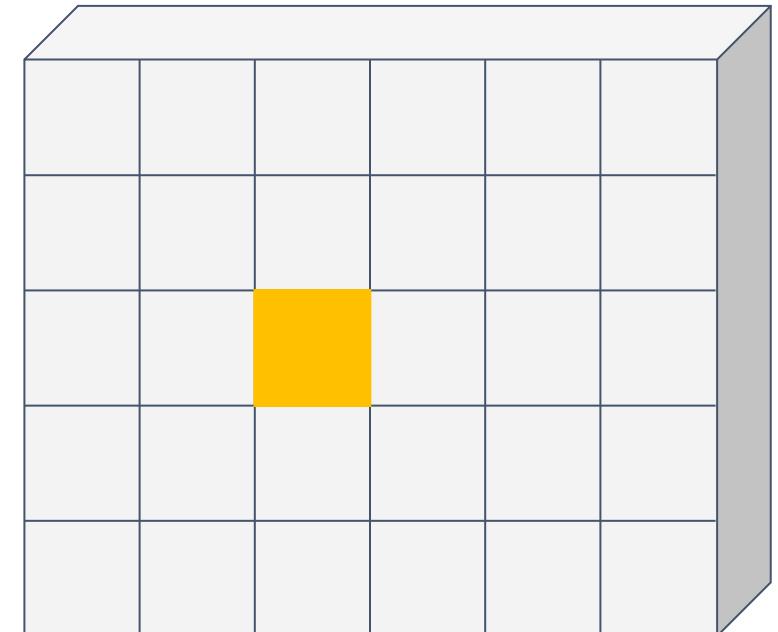
Wang et al, "Non-local Neural Networks", CVPR 2018

Idea #2: Replace Convolution with “Local Attention”

Convolution: Output at each position is inner product of conv kernel with receptive field in input



Input: $C \times H \times W$

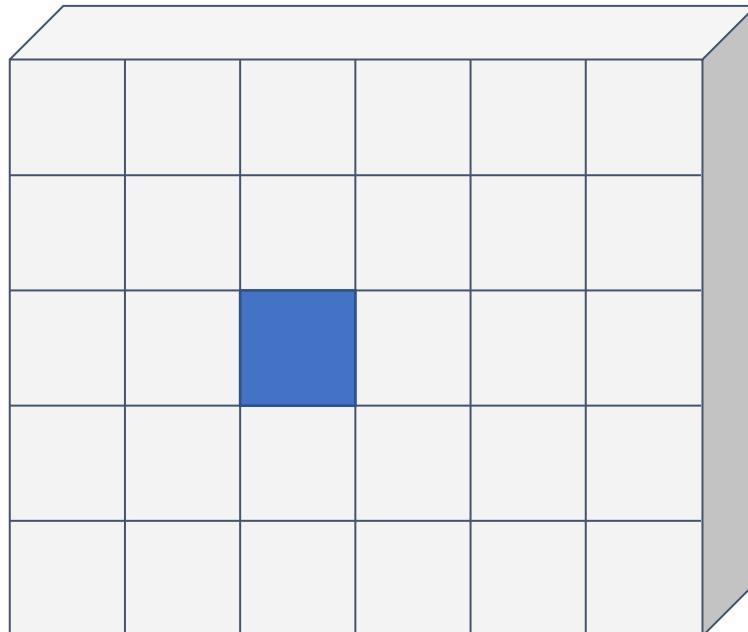


Output: $C' \times H \times W$

Hu et al, “Local Relation Networks for Image Recognition”, ICCV 2019; Ramachandran et al, “Stand-Alone Self-Attention in Vision Models”, NeurIPS 2019

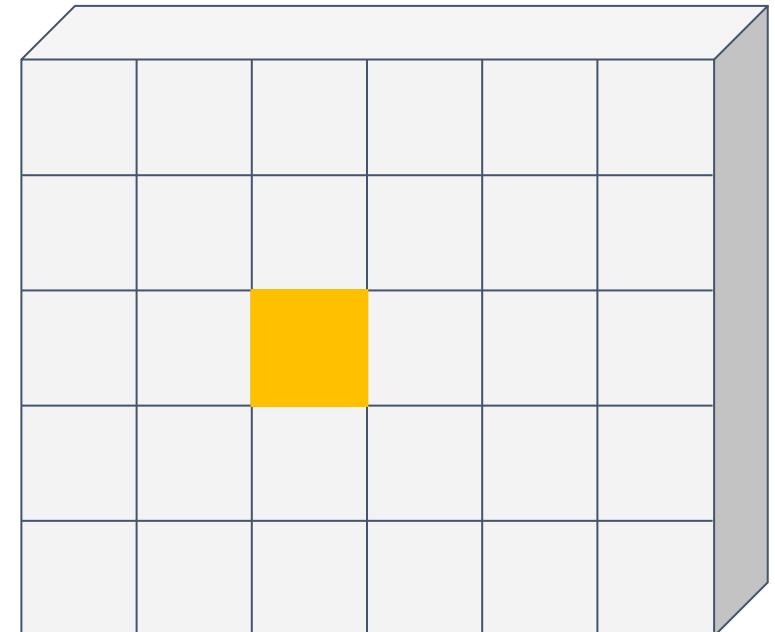
Idea #2: Replace Convolution with “Local Attention”

Map center of receptive field to **query**



Query: D_Q

Input: $C \times H \times W$



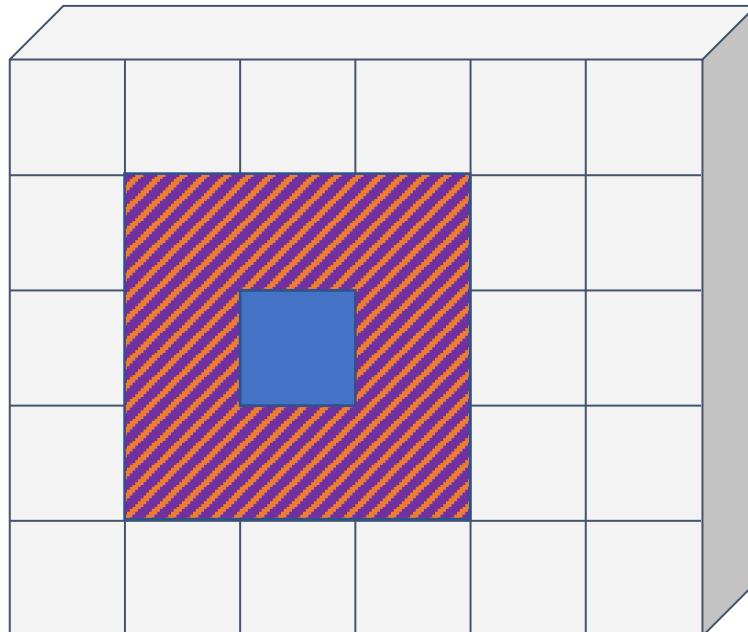
Output: $C' \times H \times W$

Hu et al, “Local Relation Networks for Image Recognition”, ICCV 2019; Ramachandran et al, “Stand-Alone Self-Attention in Vision Models”, NeurIPS 2019

Idea #2: Replace Convolution with “Local Attention”

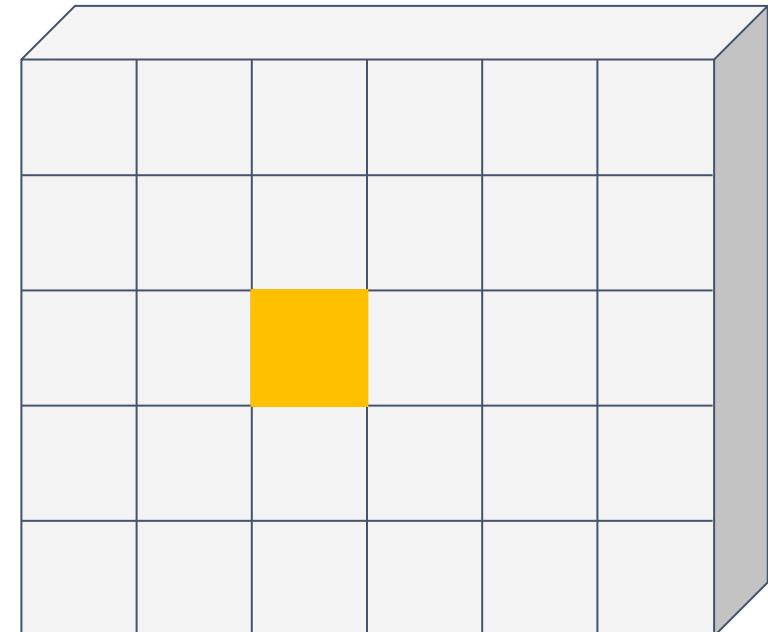
Map center of receptive field to **query**

Map each element in receptive field to **key** and **value**



Input: $C \times H \times W$

Query: D_Q
Keys: $R \times R \times D_Q$
Values: $R \times R \times C'$



Output: $C' \times H \times W$

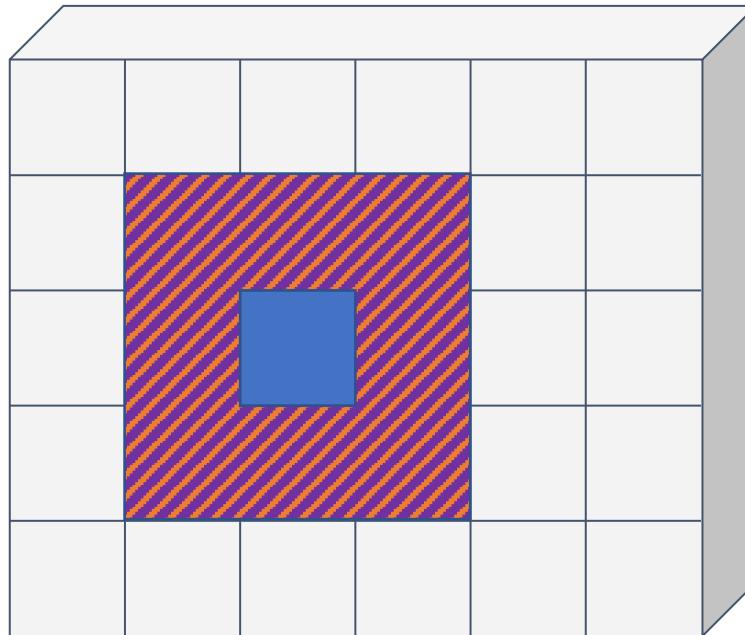
Hu et al, “Local Relation Networks for Image Recognition”, ICCV 2019; Ramachandran et al, “Stand-Alone Self-Attention in Vision Models”, NeurIPS 2019

Idea #2: Replace Convolution with “Local Attention”

Map center of receptive field to **query**

Map each element in receptive field to **key** and **value**

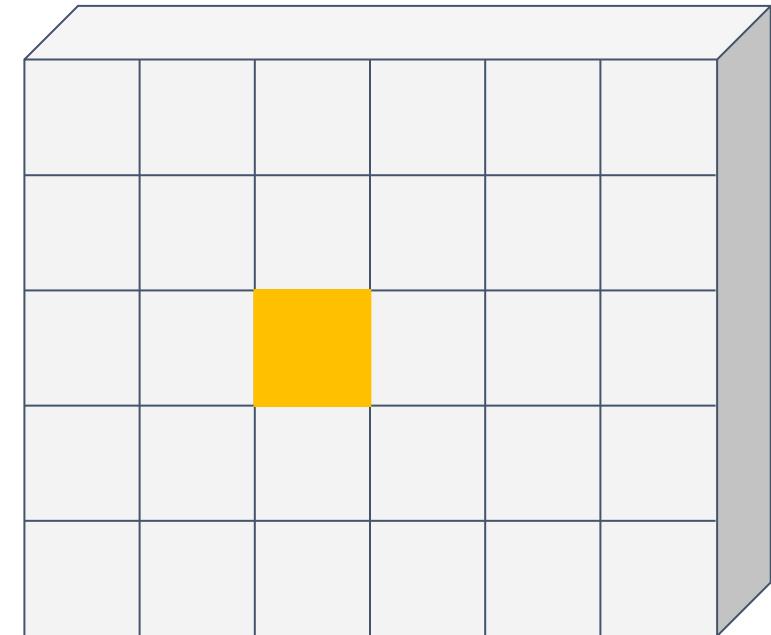
Compute **output** using attention



Input: $C \times H \times W$

Query: D_Q
Keys: $R \times R \times D_Q$
Values: $R \times R \times C'$

Output: C
Attention
↓ ↑



Output: $C' \times H \times W$

Hu et al, “Local Relation Networks for Image Recognition”, ICCV 2019; Ramachandran et al, “Stand-Alone Self-Attention in Vision Models”, NeurIPS 2019

Idea #2: Replace Convolution with “Local Attention”

Map center of receptive field to **query**

Map each element in receptive field to **key** and **value**

Compute **output** using attention

Replace all conv in ResNet with local attention

LR = “Local Relation”

stage	output	ResNet-50	LR-Net-50 ($7 \times 7, m=8$)
res1	112×112	7×7 conv, 64, stride 2	$1 \times 1, 64$ 7×7 LR, 64, stride 2
		3×3 max pool, stride 2	3×3 max pool, stride 2
res2	56×56	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3 \text{ conv}, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 100 \\ 7 \times 7 \text{ LR, 100} \\ 1 \times 1, 256 \end{bmatrix} \times 3$
res3	28×28	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3 \text{ conv}, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 200 \\ 7 \times 7 \text{ LR, 200} \\ 1 \times 1, 512 \end{bmatrix} \times 4$
res4	14×14	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3 \text{ conv}, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 400 \\ 7 \times 7 \text{ LR, 400} \\ 1 \times 1, 1024 \end{bmatrix} \times 6$
res5	7×7	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3 \text{ conv}, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 800 \\ 7 \times 7 \text{ LR, 800} \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params		25.5×10^6	23.3×10^6
FLOPs		4.3×10^9	4.3×10^9

Hu et al, “Local Relation Networks for Image Recognition”, ICCV 2019;
Ramachandran et al, “Stand-Alone Self-Attention in Vision Models”, NeurIPS 2019

Idea #2: Replace Convolution with “Local Attention”

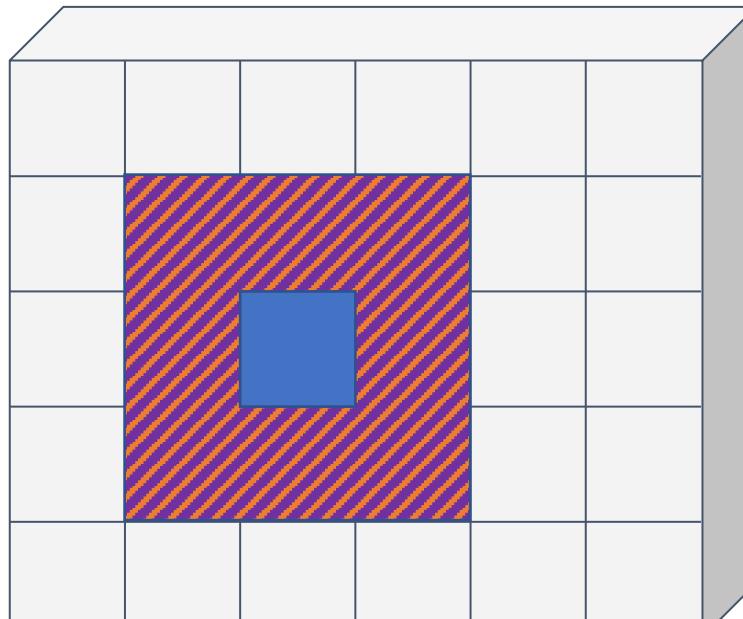
Map center of receptive field to **query**

Map each element in receptive field to **key** and **value**

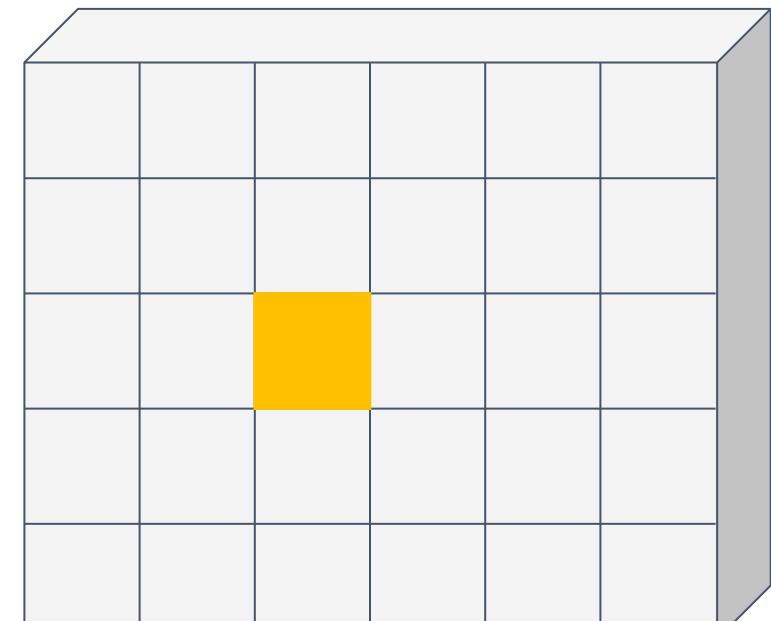
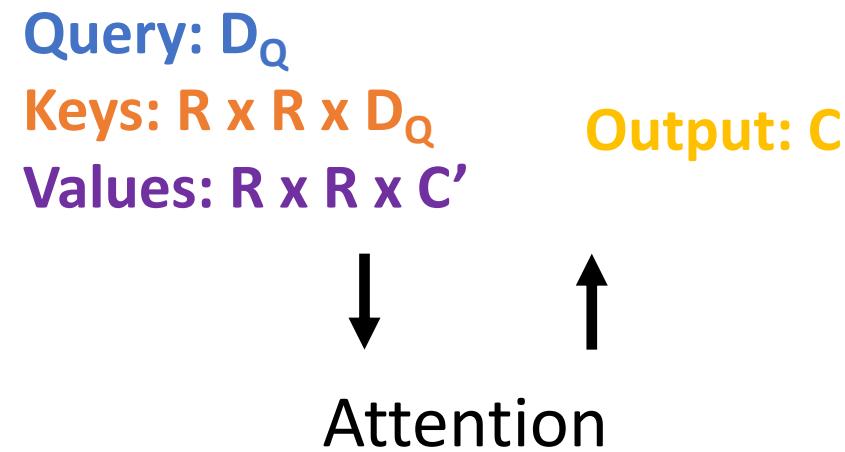
Compute **output** using attention

Replace all conv in ResNet with local attention

Lots of tricky details,
hard to implement,
only marginally better
than ResNets



Input: $C \times H \times W$

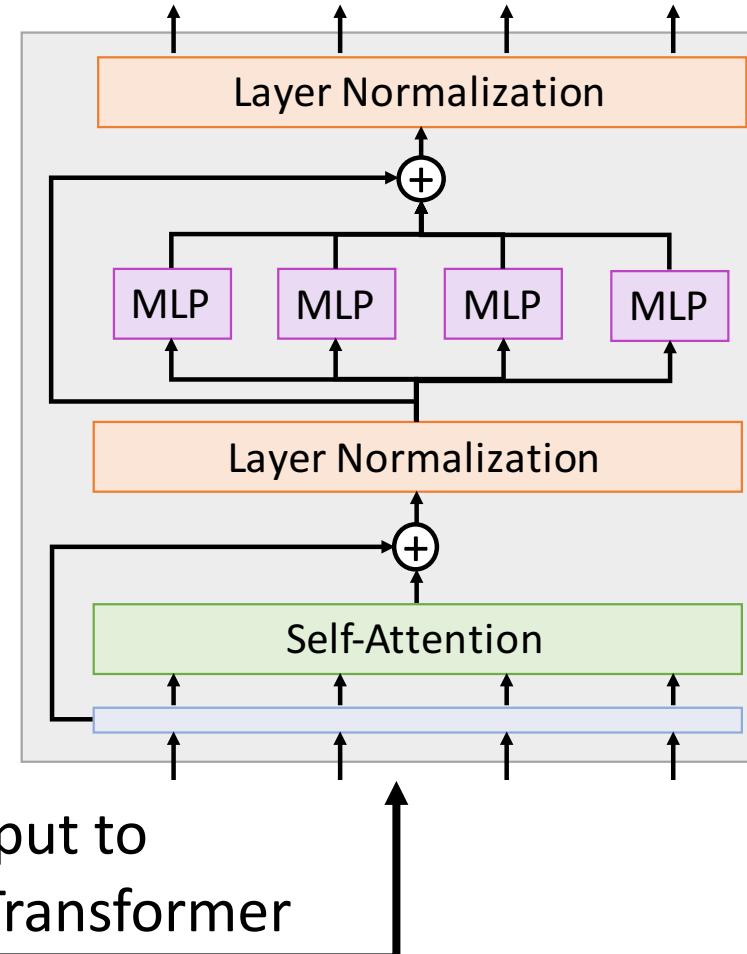
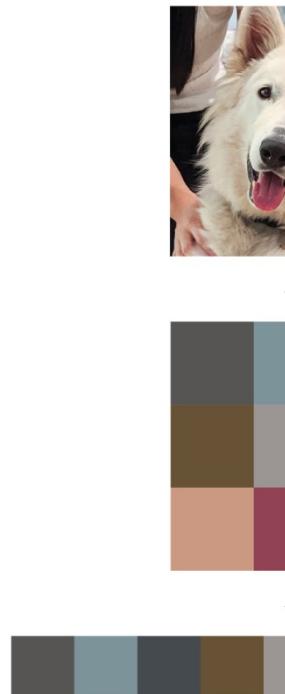


Output: $C' \times H \times W$

Hu et al, “Local Relation Networks for Image Recognition”, ICCV 2019; Ramachandran et al, “Stand-Alone Self-Attention in Vision Models”, NeurIPS 2019

Idea #3: Standard Transformer on Pixels

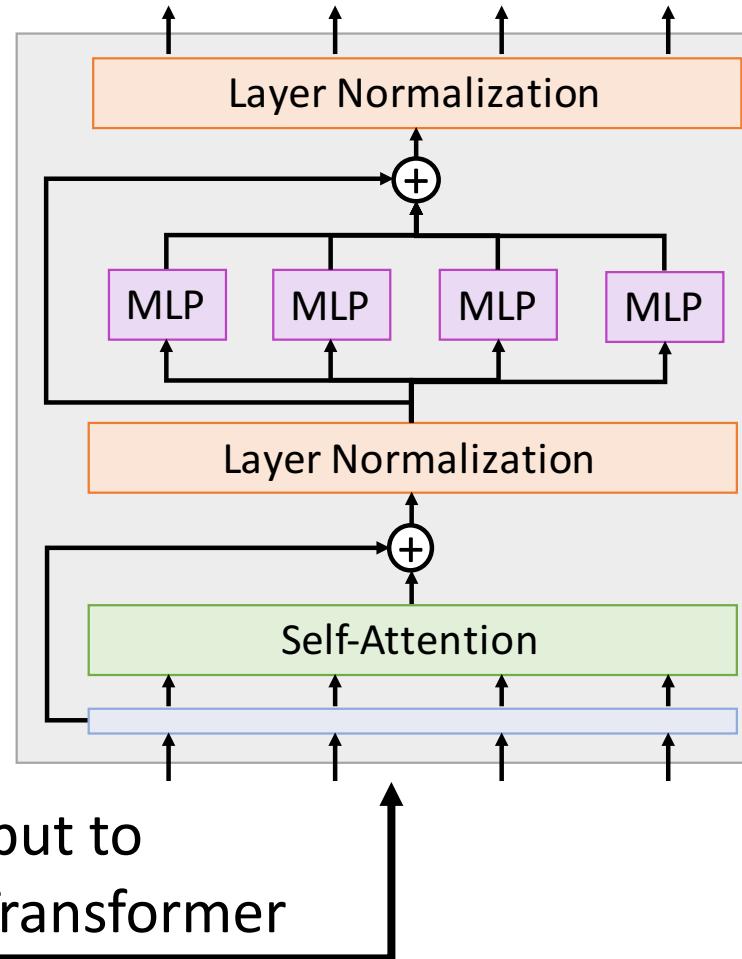
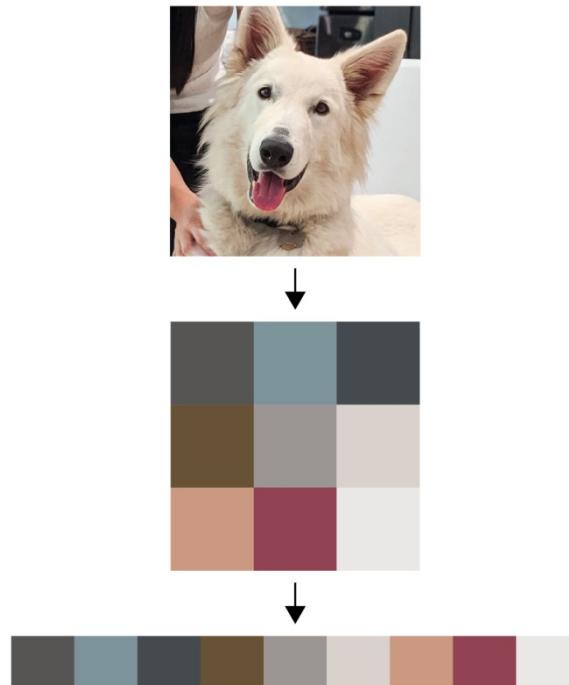
Treat an image as a set of pixel values



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

Idea #3: Standard Transformer on Pixels

Treat an image as a set of pixel values

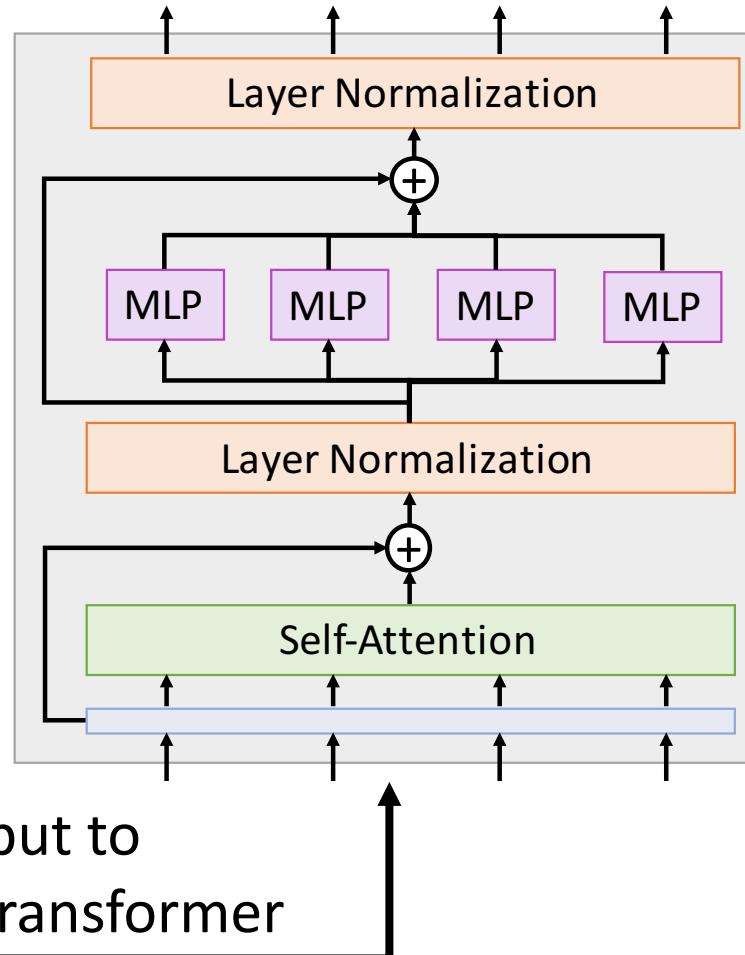
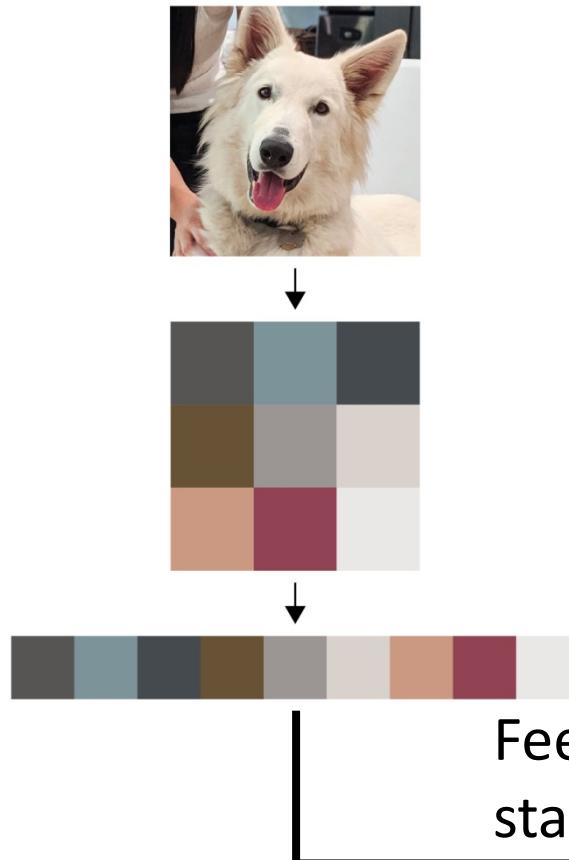


Problem: Memory use!

$R \times R$ image needs R^4 elements per attention matrix

Idea #3: Standard Transformer on Pixels

Treat an image as a set of pixel values



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...

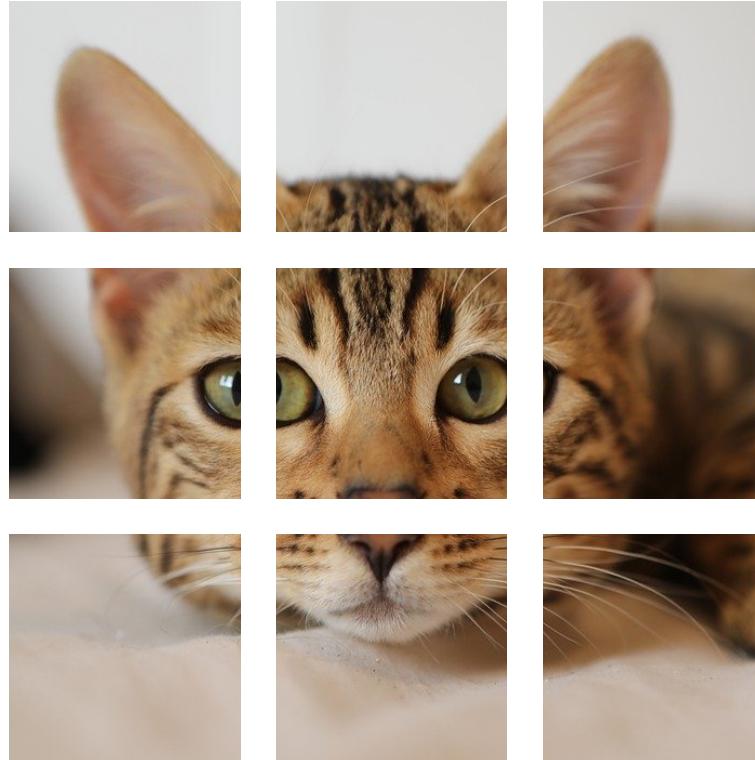
Idea #4: Standard Transformer on Patches



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 #4: 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 #4: Standard Transformer on Patches

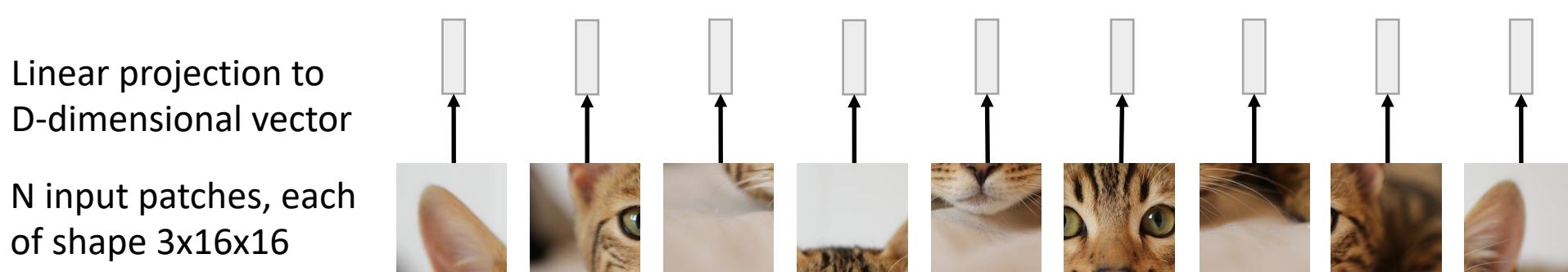
N input patches, each
of shape 3x16x16



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

[Cat image](#) is free for commercial
use under a [Pixabay license](#)

Idea #4: Standard Transformer on Patches

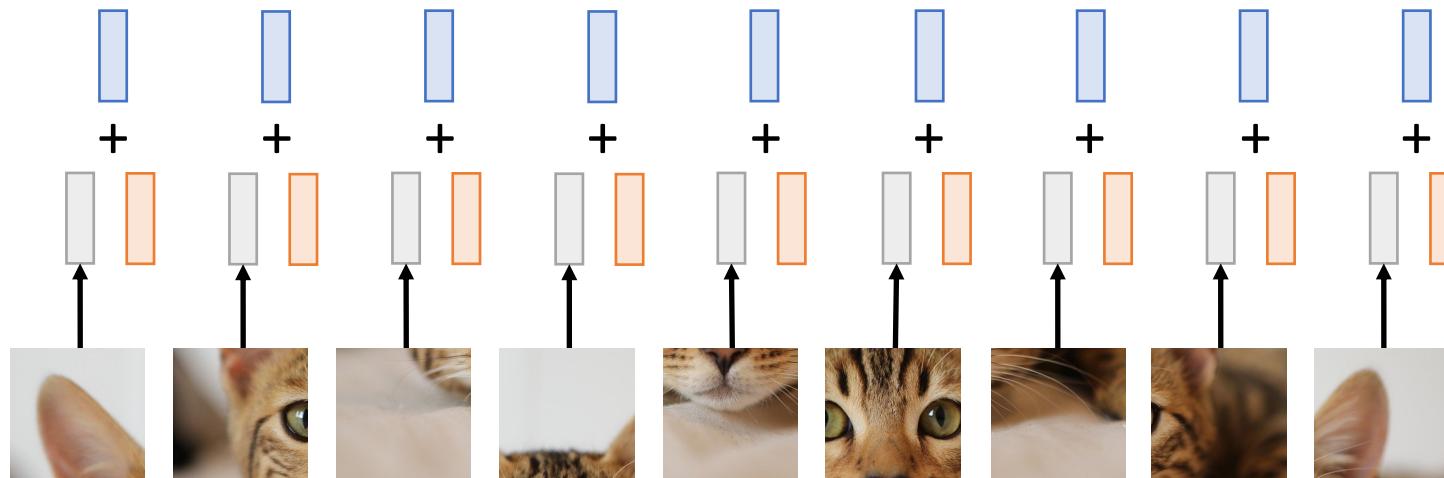


Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

[Cat image](#) is free for commercial use under a [Pixabay license](#)

Idea #4: Standard Transformer on Patches

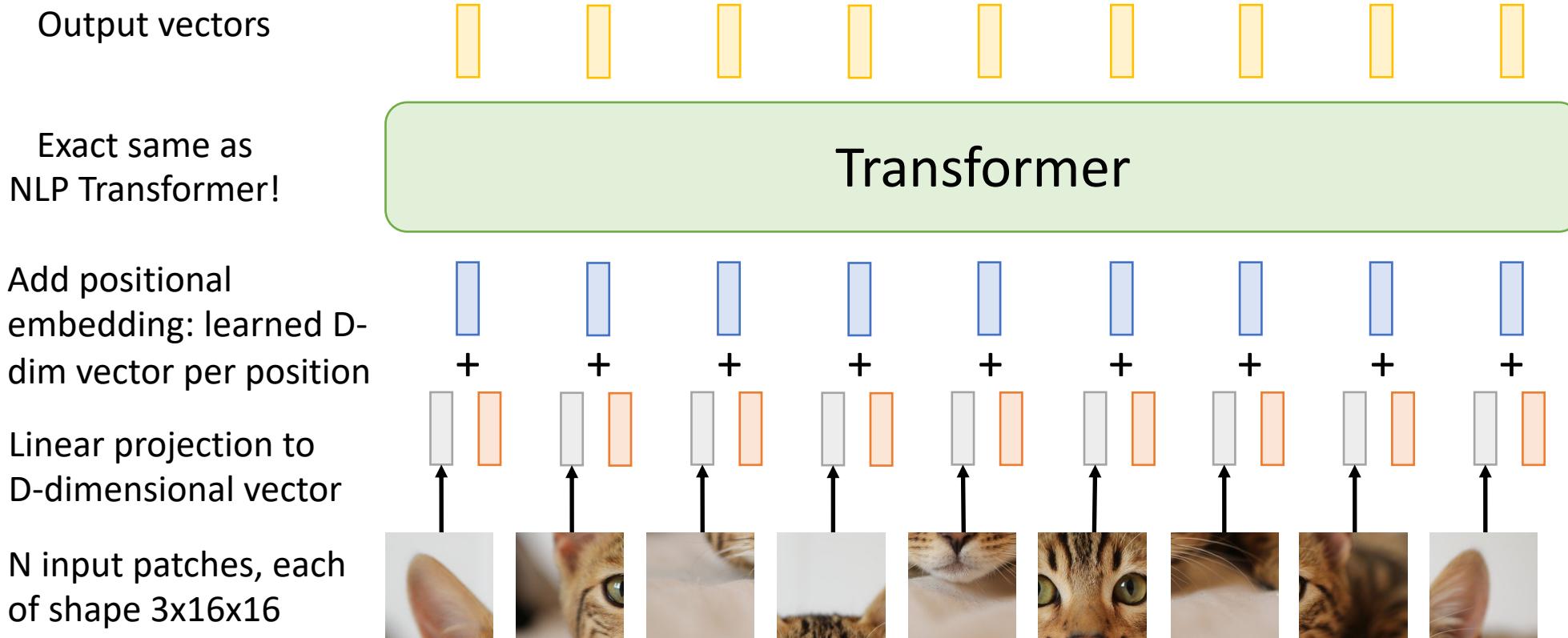
Add positional embedding: learned D-dim vector per position
Linear projection to D-dimensional vector
N input patches, each of shape 3x16x16



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

[Cat image](#) is free for commercial use under a [Pixabay license](#)

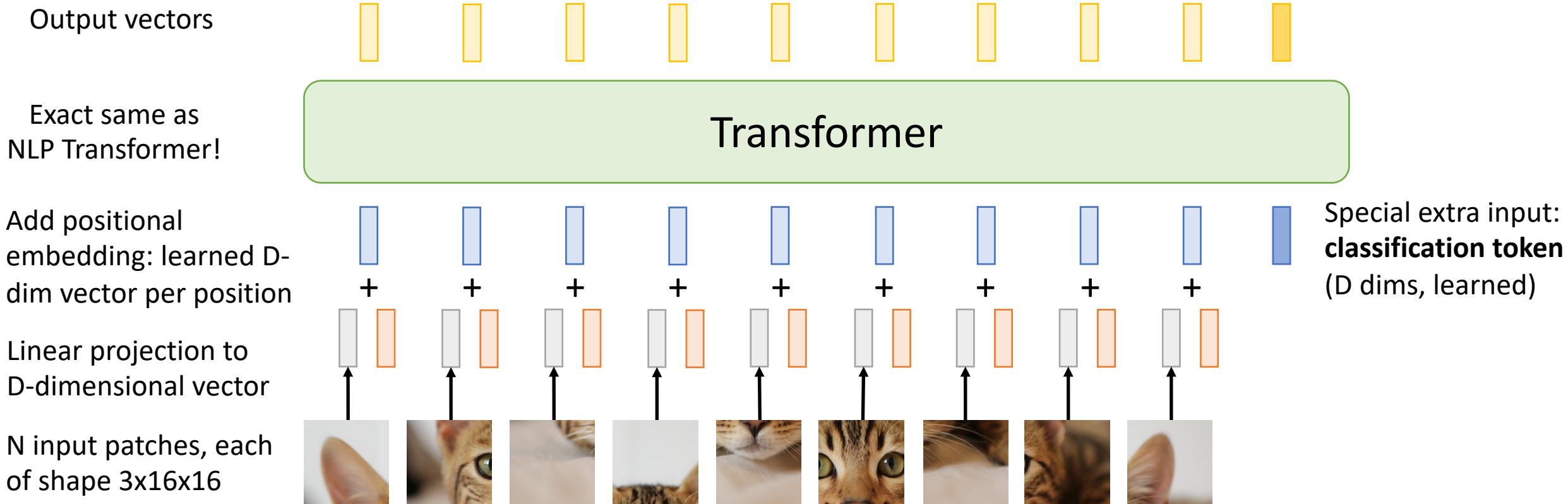
Idea #4: Standard Transformer on Patches



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](#)

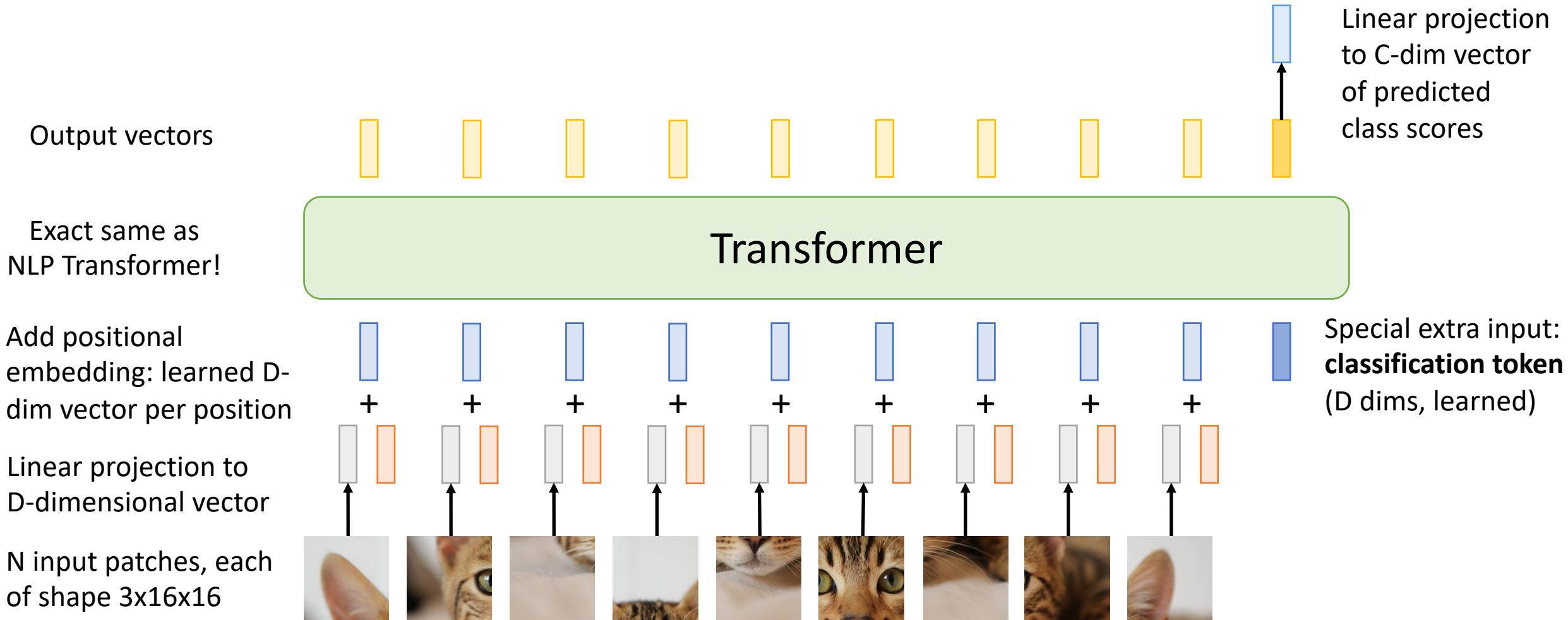
Idea #4: Standard Transformer on Patches



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 #4: 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](#)

Vision Transformer (ViT)

Computer vision model
with no convolutions!

Output vectors



Exact same as
NLP Transformer!

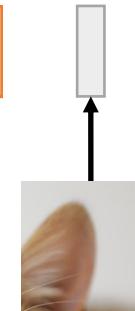
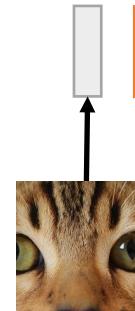
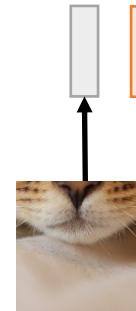
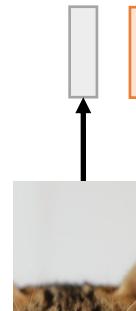
Transformer

Add positional
embedding: learned D-
dim vector per position

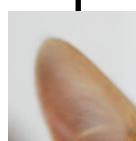


Special extra input:
classification token
(D dims, learned)

Linear projection to
D-dimensional vector



N input patches, each
of shape 3x16x16



Cat image is free for commercial
use under a [Pixabay license](#)

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: With patch size p , first
layer is Conv2D($p \times p$, 3->D, stride= p)

Output vectors

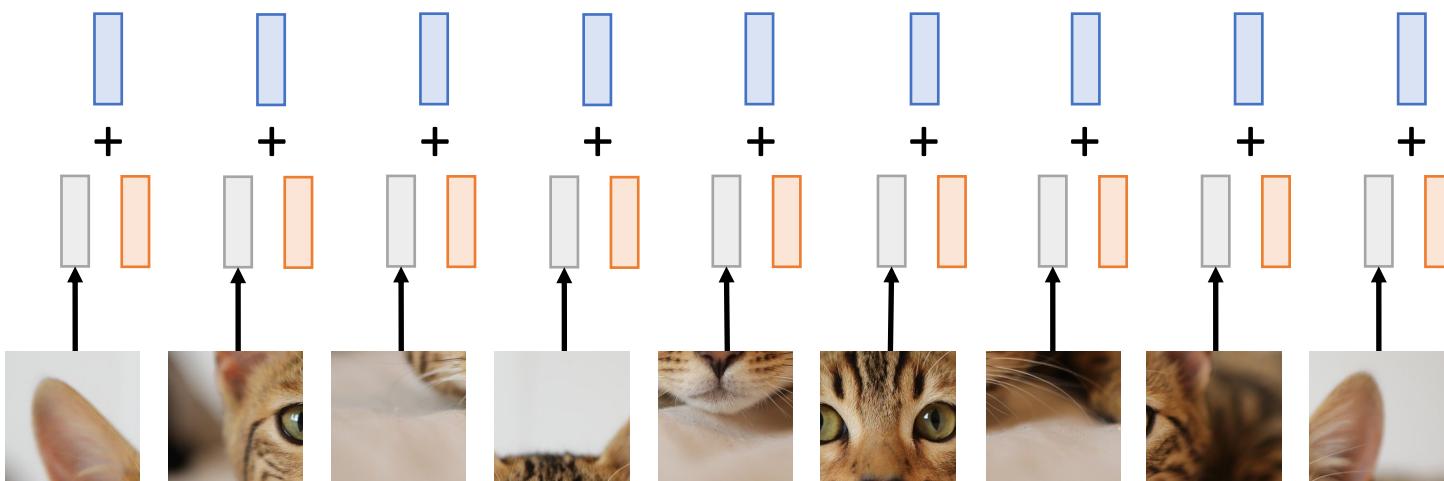


Linear projection
to C-dim vector
of predicted
class scores

Exact same as
NLP Transformer!

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)

Computer vision model
with no convolutions!

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

Output vectors



Linear projection
to C-dim vector
of predicted
class scores

Exact same as
NLP Transformer!

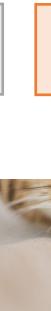
Transformer

Add positional
embedding: learned D-
dim vector per position

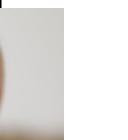
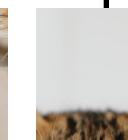
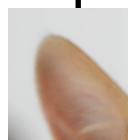


Special extra input:
classification token
(D dims, learned)

Linear projection to
D-dimensional vector



N input patches, each
of shape 3x16x16



[Cat image](#) is free for commercial
use under a [Pixabay license](#)

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

Vision Transformer (ViT)

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

Output vectors



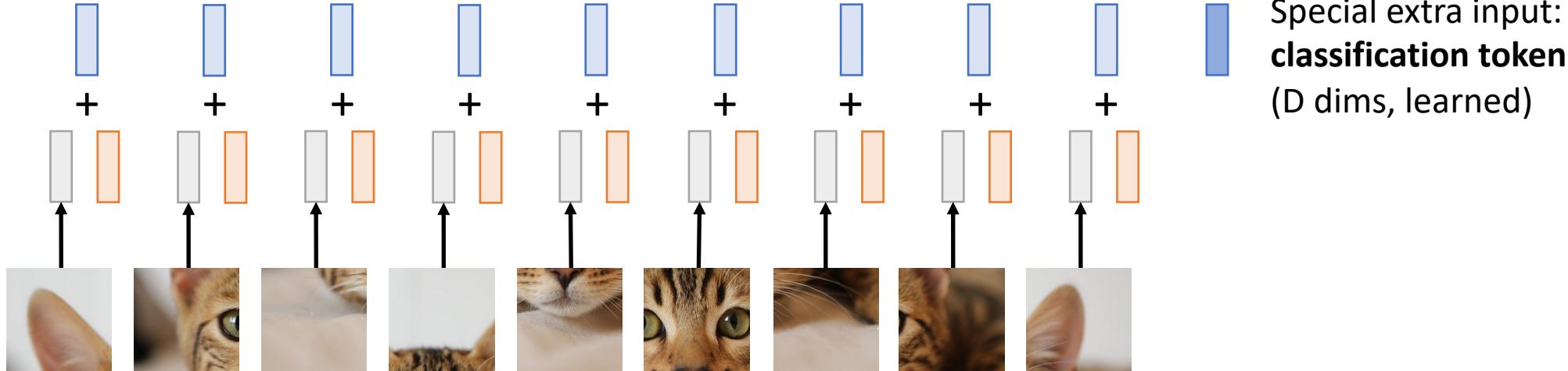
Each attention matrix has $14^4 = 38,416$ entries, takes 150 KB (or 65,536 entries, takes 256 KB)

Linear projection to C-dim vector of predicted class scores

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

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)

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

Output vectors



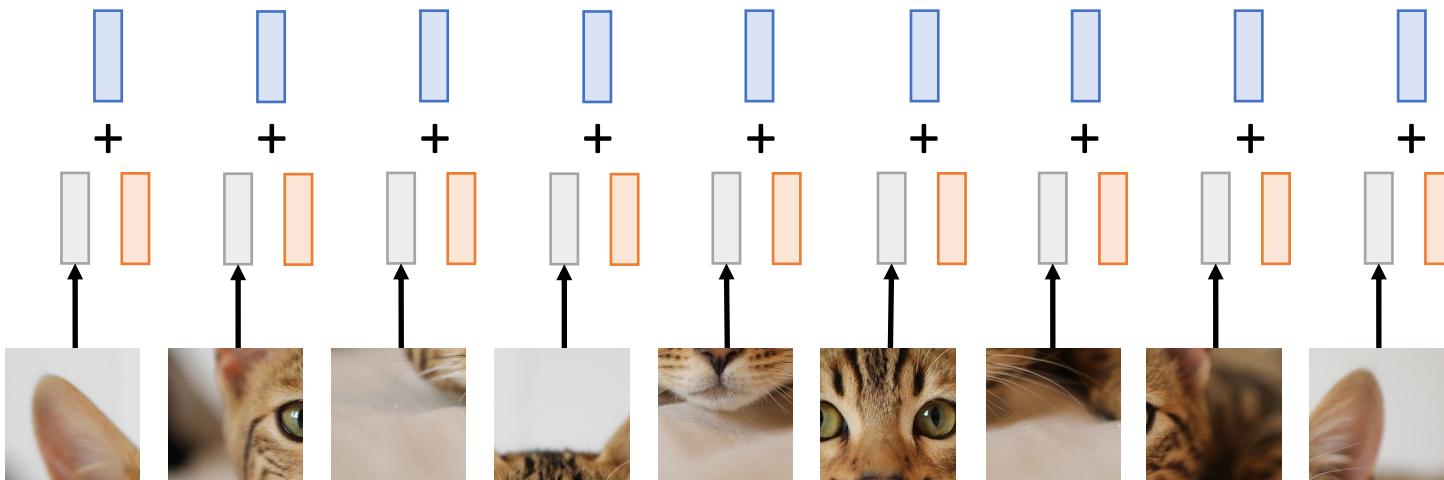
With 48 layers, 16 heads per
layer, all attention matrices
take 112 MB (or 192MB)

Linear projection
to C-dim vector
of predicted
class scores

Exact same as
NLP Transformer!

Transformer

Add positional
embedding: learned D-
dim vector per position



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

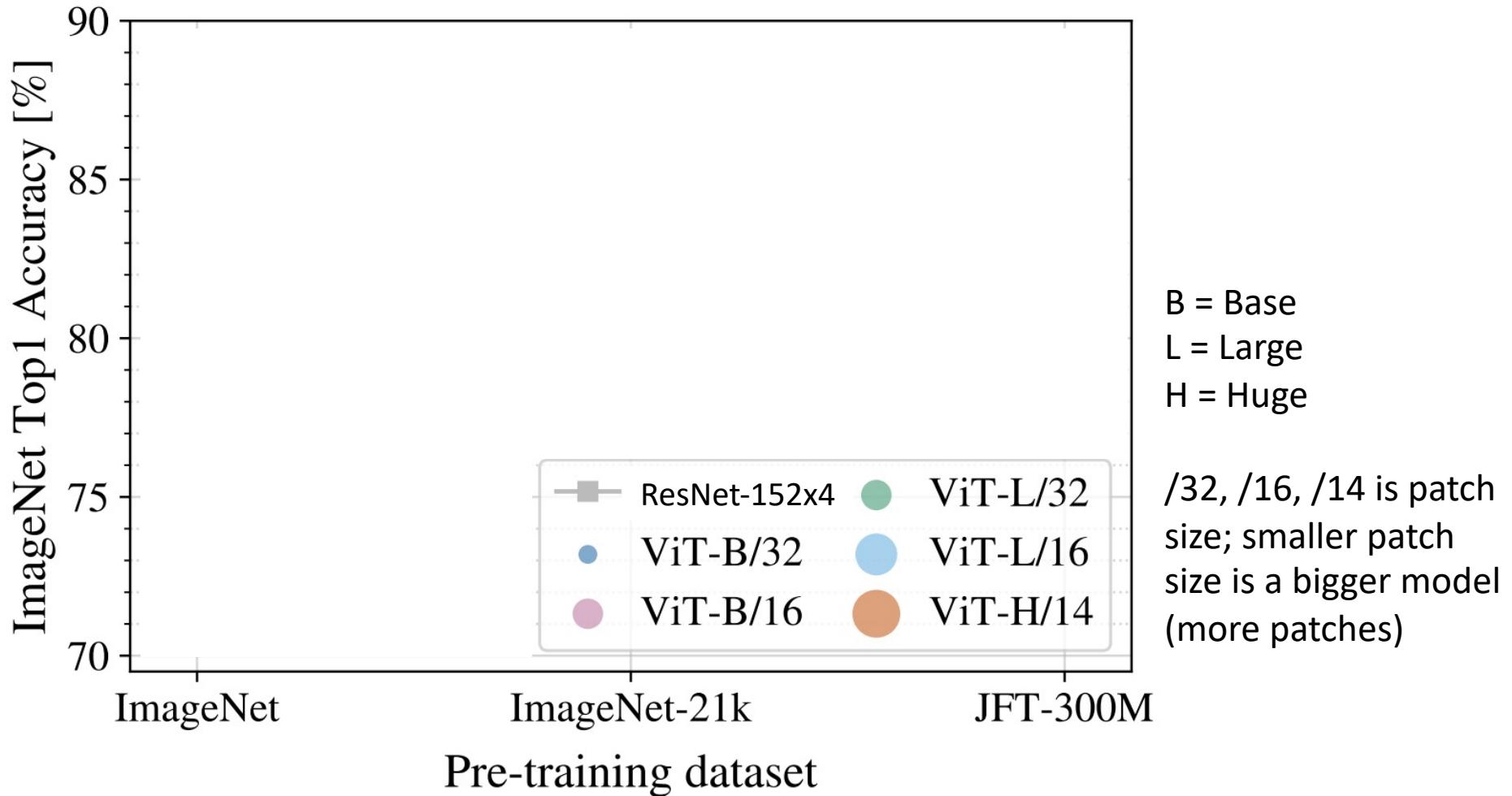
Linear projection to
D-dimensional vector

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) vs ResNets

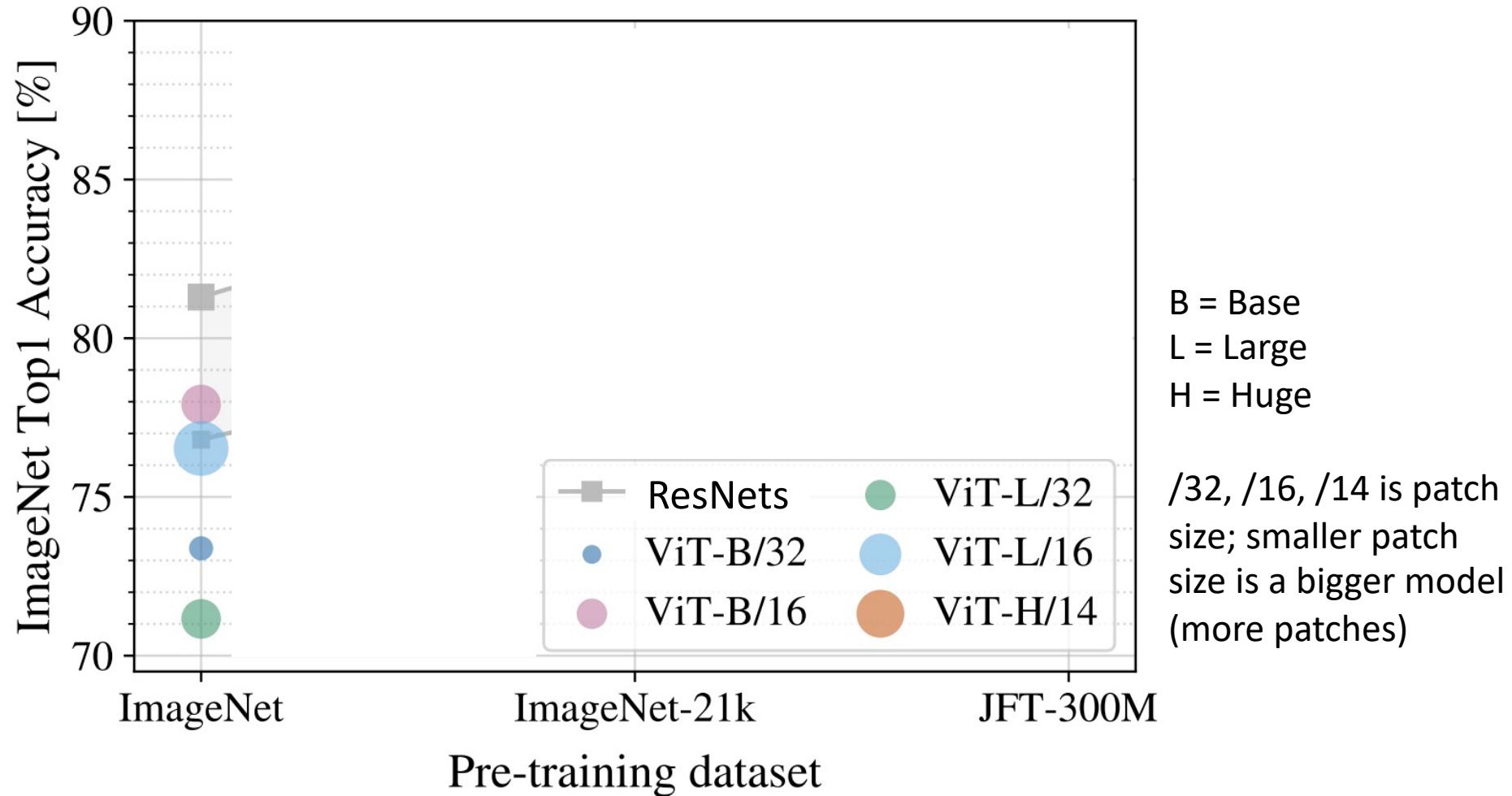


Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

Vision Transformer (ViT) vs ResNets

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

When trained on ImageNet, ViT models perform worse than ResNets

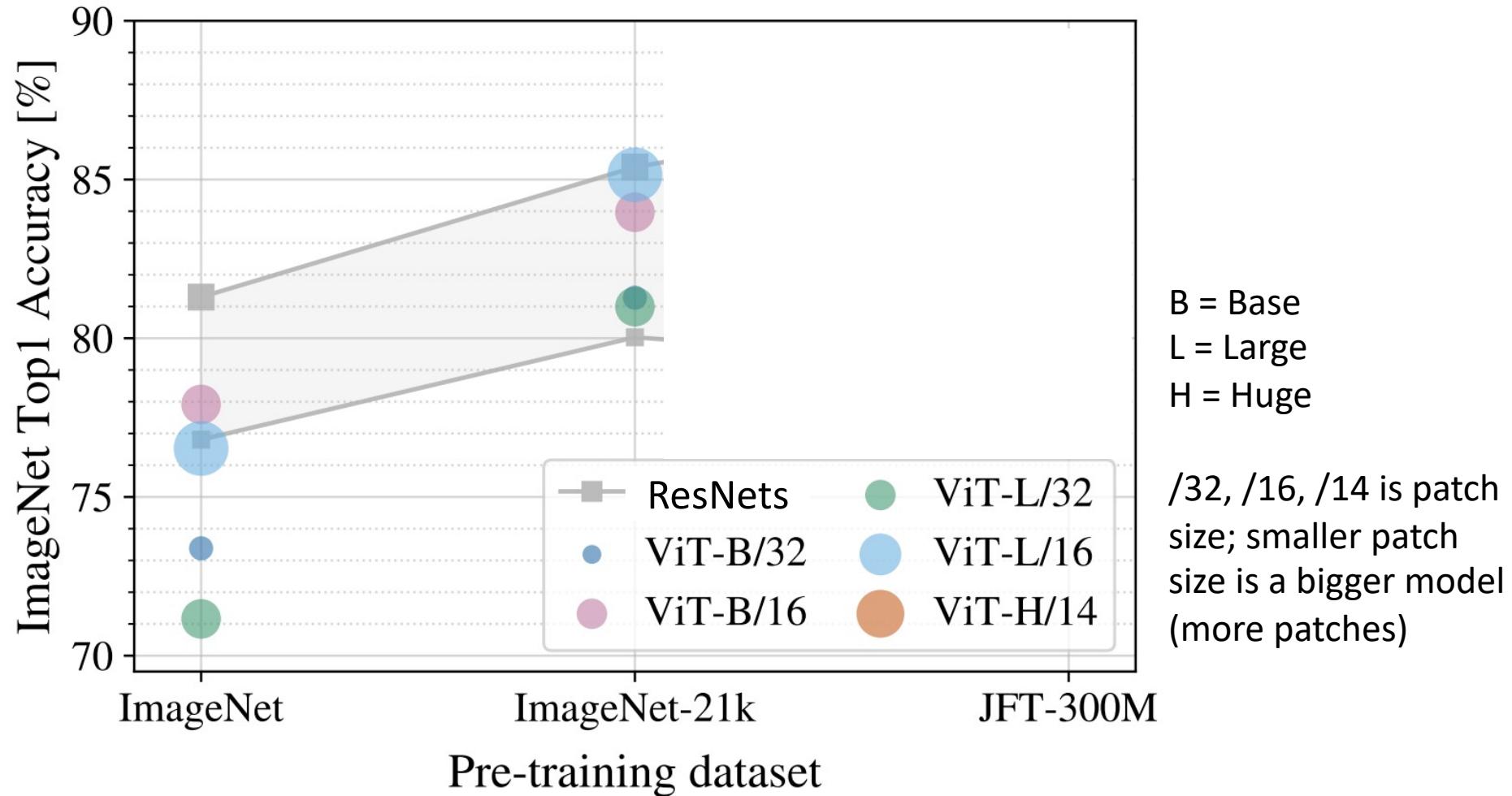


Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

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

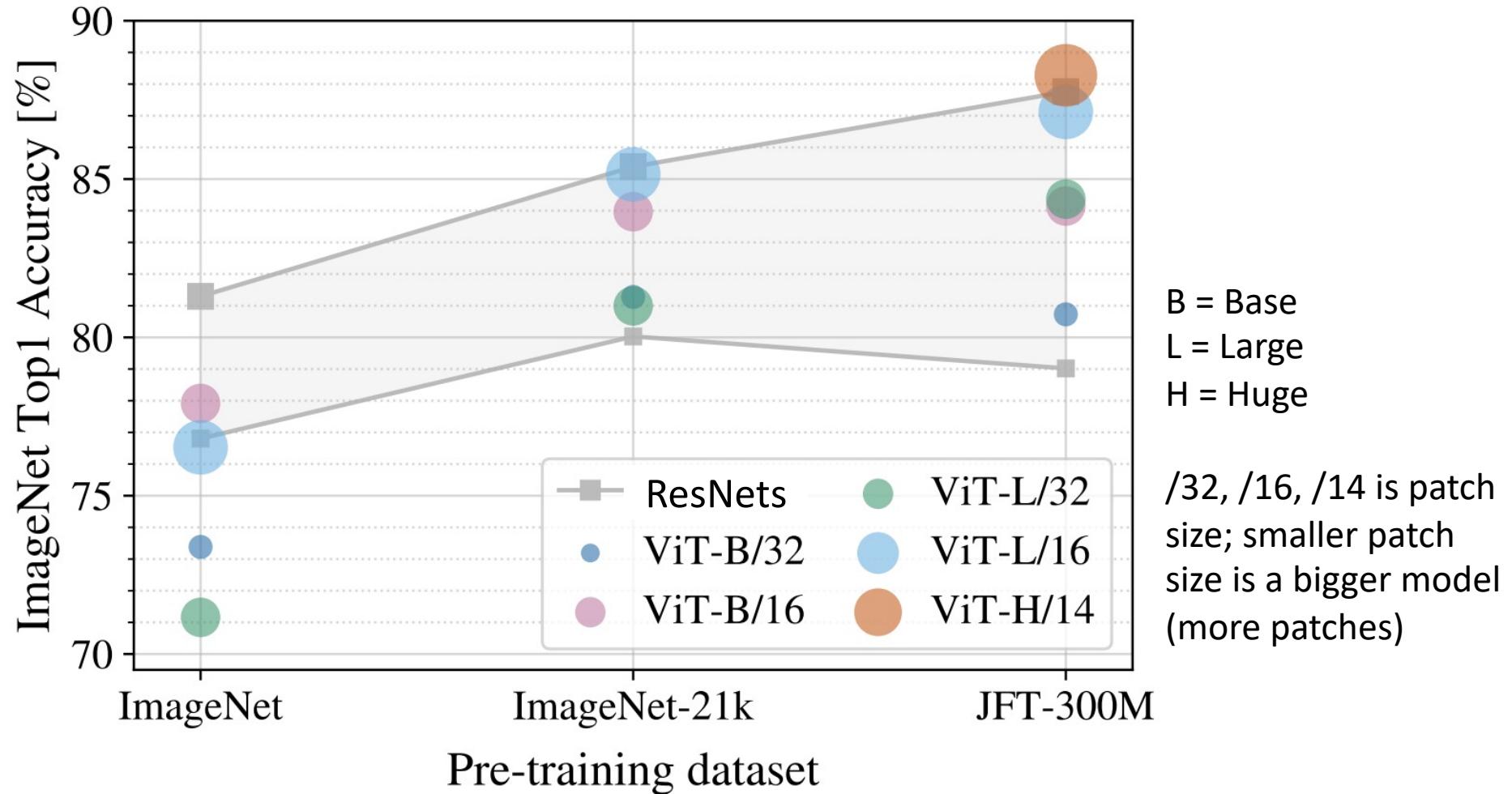


Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

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

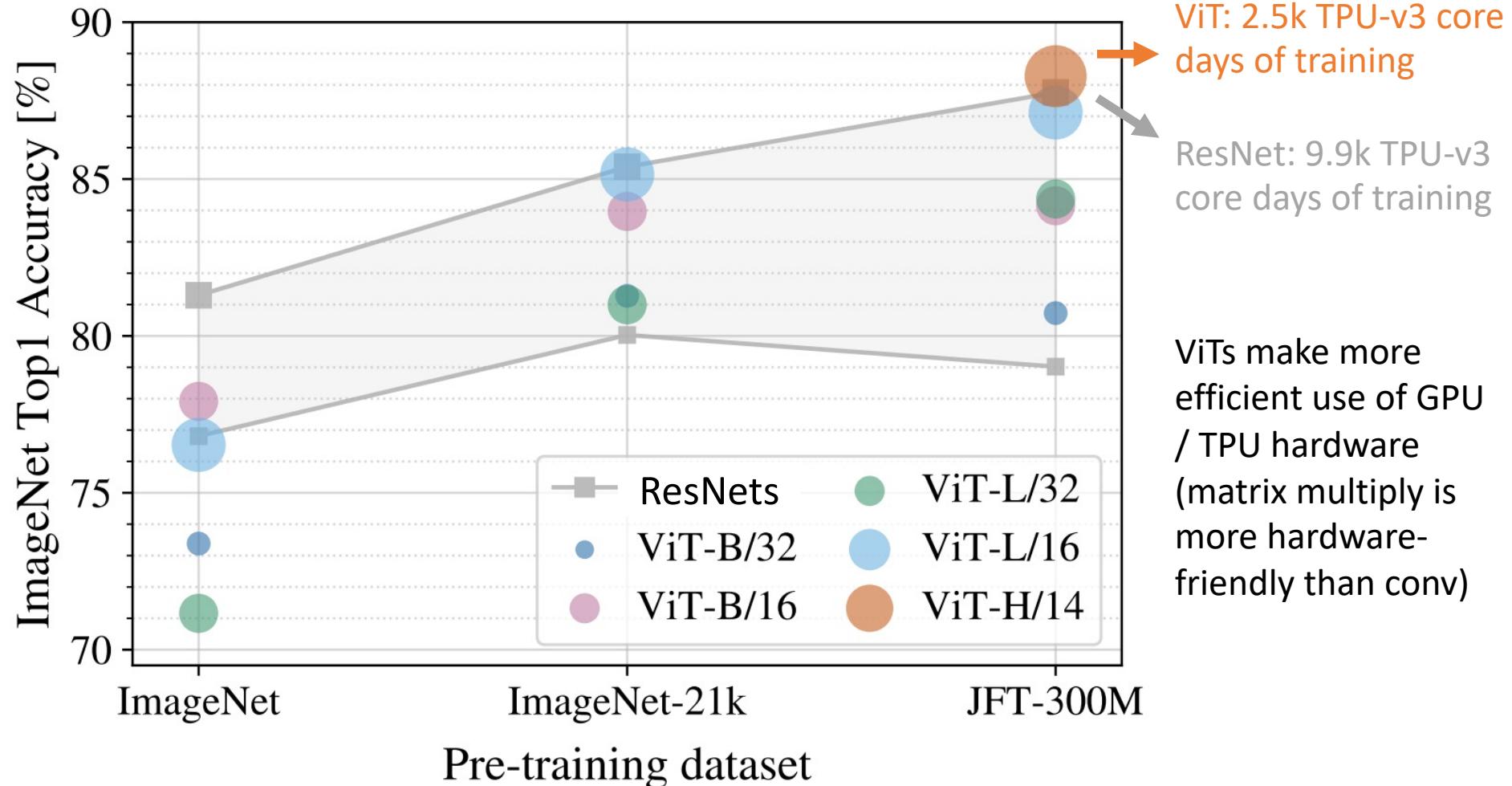


Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

Vision Transformer (ViT) vs ResNets

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

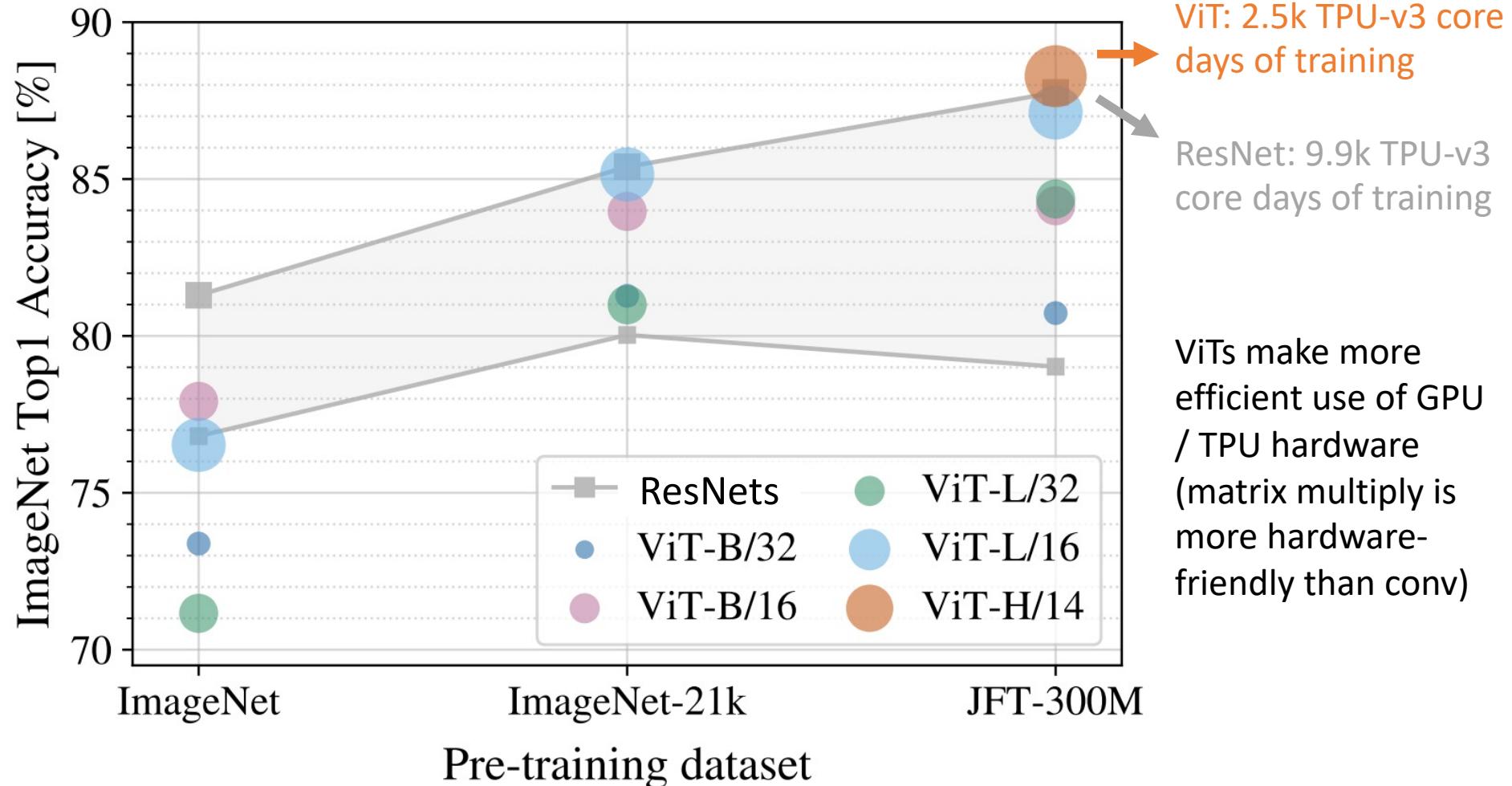
If you pretrain on JFT and finetune on ImageNet, large ViTs outperform large ResNets



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

Vision Transformer (ViT) vs ResNets

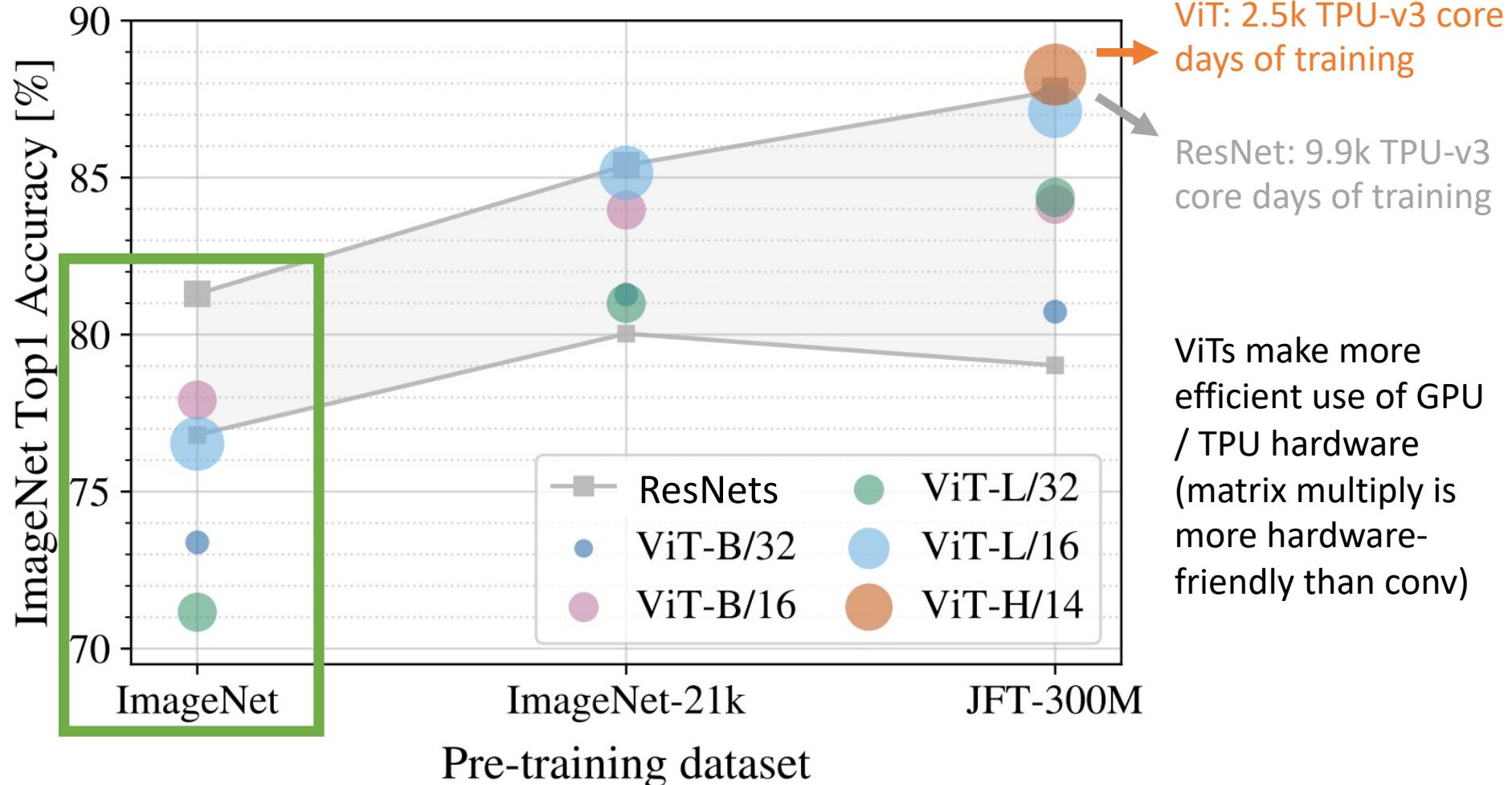
Claim: ViT models have “less inductive bias” than ResNets, so need more pretraining data to learn good features



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

Vision Transformer (ViT) vs ResNets

How can we improve the performance of ViT models on ImageNet?



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

Improving ViT: Augmentation and Regularization

Regularization for ViT models:

- Weight Decay
- Stochastic Depth
- Dropout (in FFN layers of Transformer)

Data Augmentation for ViT models:

- MixUp
- RandAugment

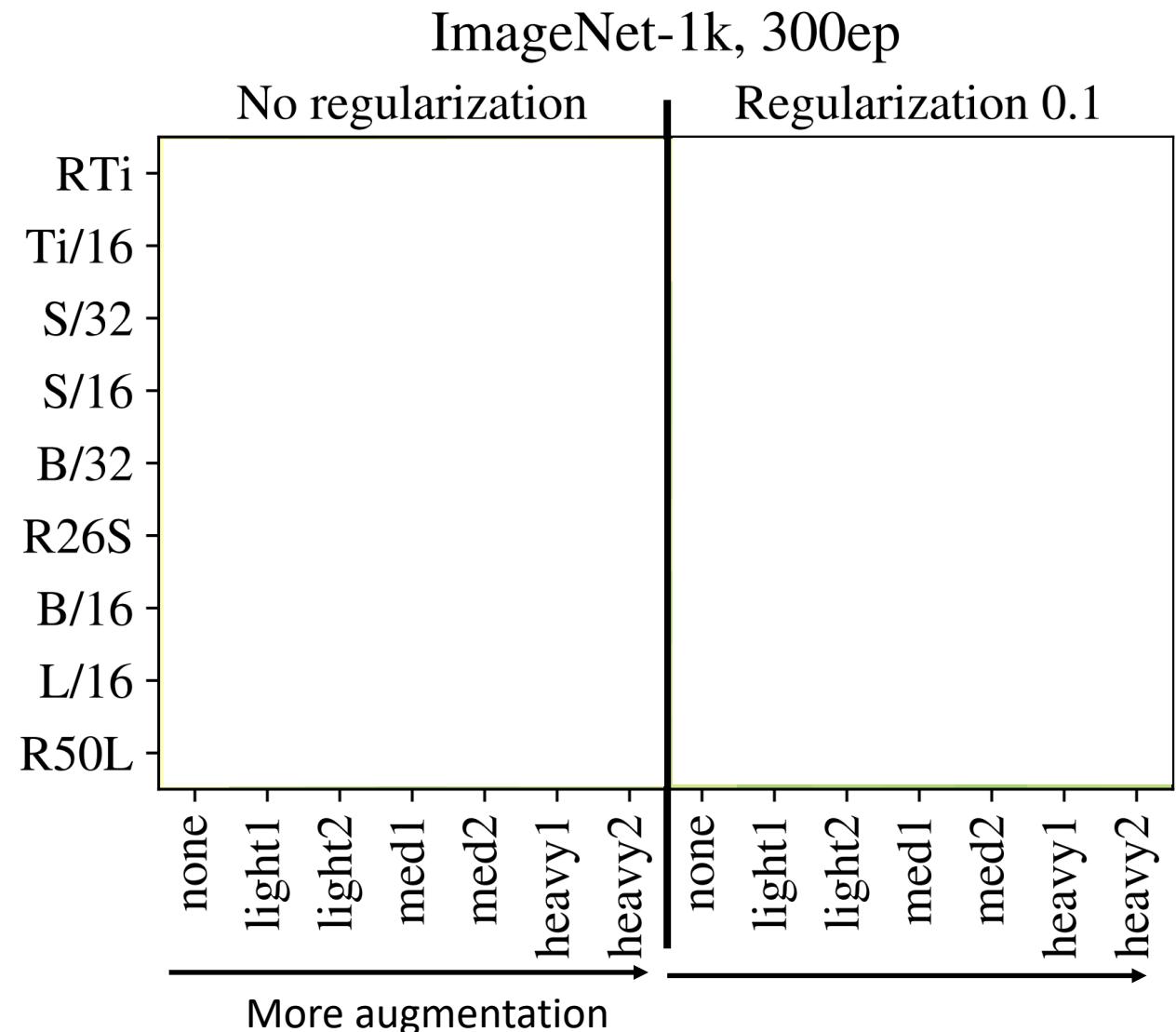
Improving ViT: Augmentation and Regularization

Regularization for ViT models:

- Weight Decay
- Stochastic Depth
- Dropout (in FFN layers of Transformer)

Data Augmentation for ViT models:

- MixUp
- RandAugment



Steiner et al, "How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers", arXiv 2021

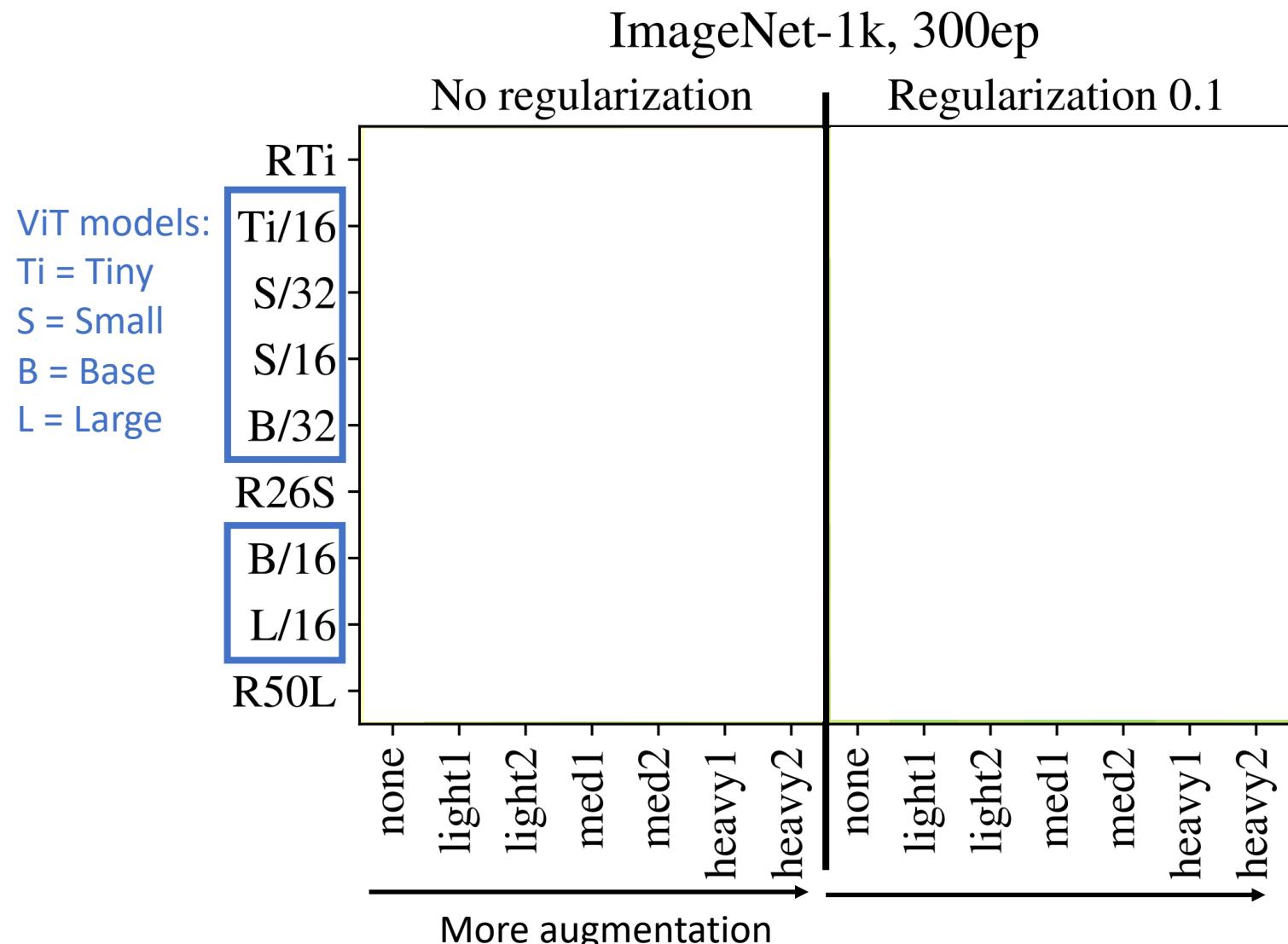
Improving ViT: Augmentation and Regularization

Regularization for ViT models:

- Weight Decay
- Stochastic Depth
- Dropout (in FFN layers of Transformer)

Data Augmentation for ViT models:

- MixUp
- RandAugment



Steiner et al, "How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers", arXiv 2021

Improving ViT: Augmentation and Regularization

Regularization for ViT models:

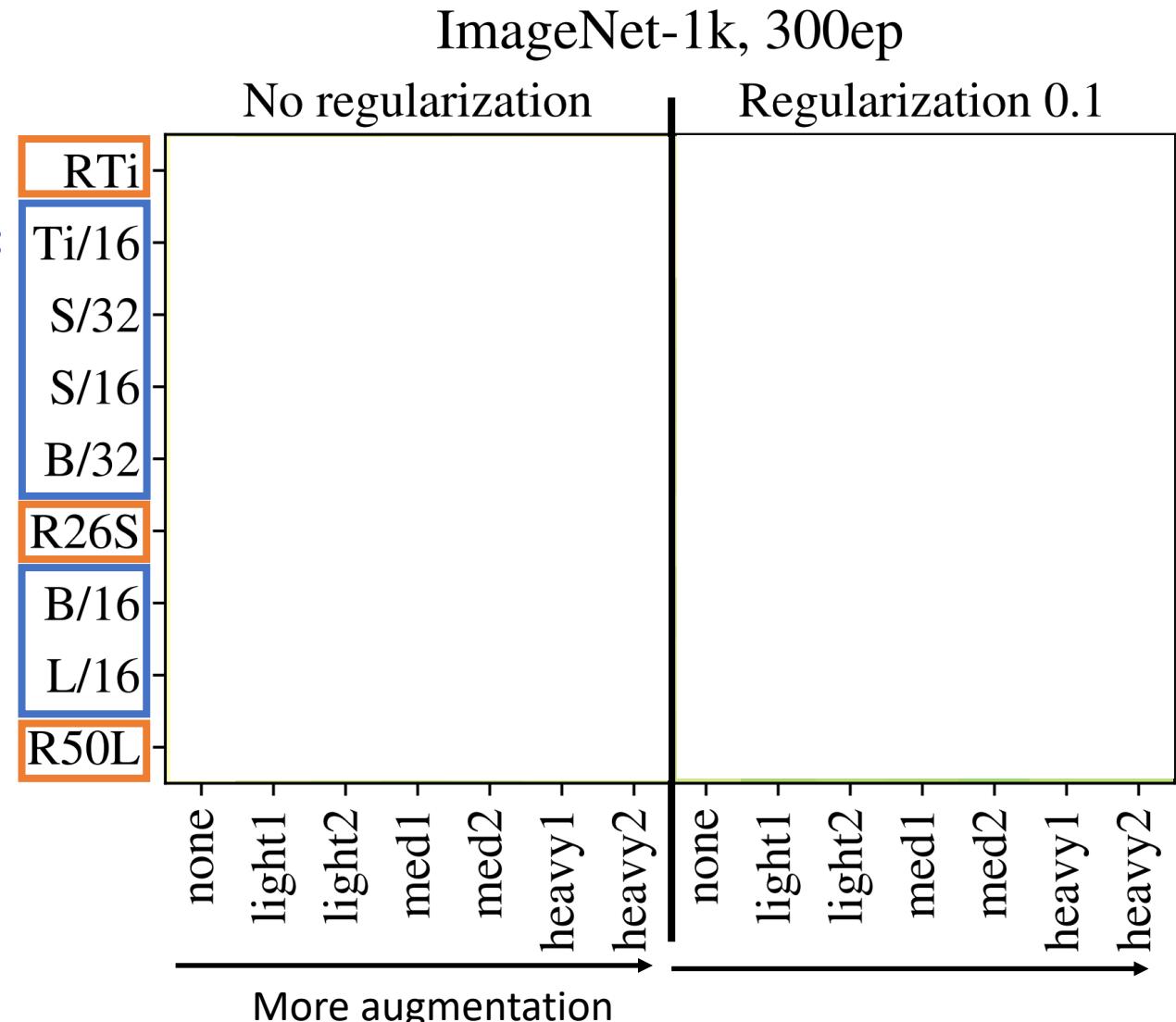
- Weight Decay
- Stochastic Depth
- Dropout (in FFN layers of Transformer)

Data Augmentation for ViT models:

- MixUp
- RandAugment

Hybrid models:
ResNet blocks,
then ViT blocks

ViT models:
 T_i = Tiny
 S = Small
 B = Base
 L = Large



Improving ViT: Augmentation and Regularization

Regularization for ViT models:

- Weight Decay
- Stochastic Depth
- Dropout (in FFN layers of Transformer)

Data Augmentation for ViT models:

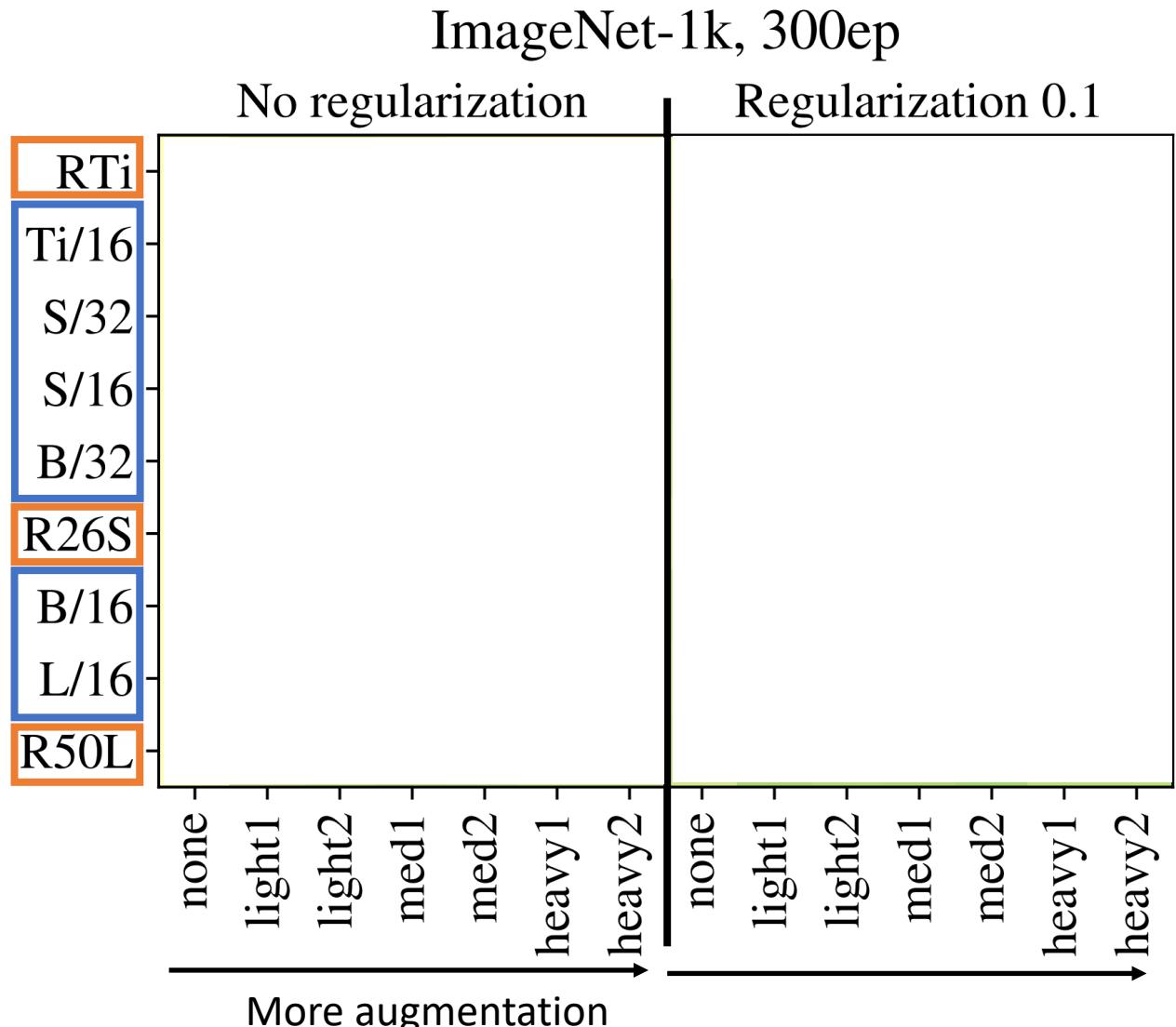
- MixUp
- RandAugment

Hybrid models:
ResNet blocks,
then ViT blocks

ViT models:
 T_i = Tiny
 S = Small
 B = Base
 L = Large

Original Paper:

77.9
76.53



Improving ViT: Augmentation and Regularization

Regularization for ViT models:

- Weight Decay
- Stochastic Depth
- Dropout (in FFN layers of Transformer)

Data Augmentation for ViT models:

- MixUp
- RandAugment

Adding regularization is
(almost) always helpful

Hybrid models:
ResNet blocks,
then ViT blocks

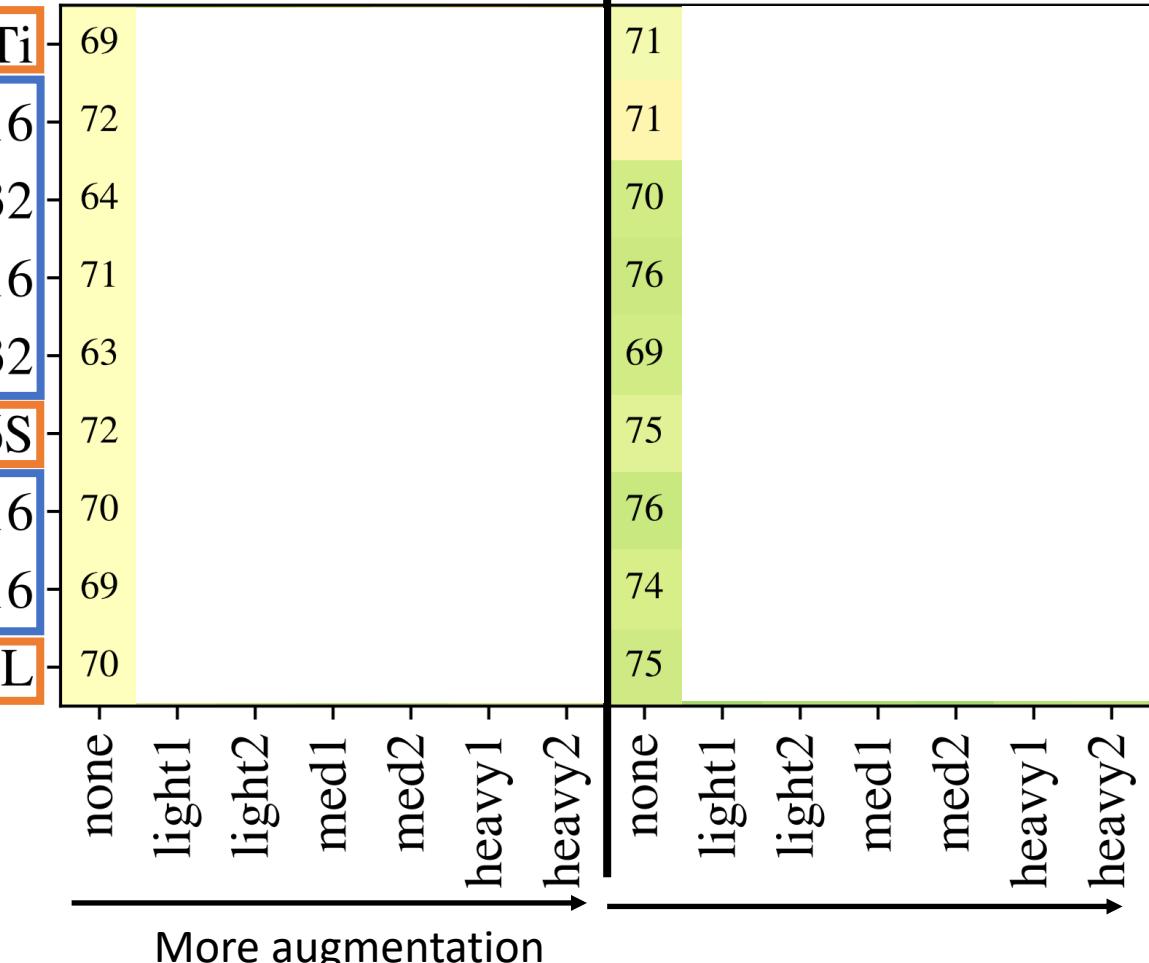
ViT models:
 $Ti = \text{Tiny}$
 $S = \text{Small}$
 $B = \text{Base}$
 $L = \text{Large}$

Original Paper:
77.9
76.53

ImageNet-1k, 300ep

No regularization

Regularization 0.1



Improving ViT: Augmentation and Regularization

Regularization for ViT models:

- Weight Decay
- Stochastic Depth
- Dropout (in FFN layers of Transformer)

Data Augmentation for ViT models:

- MixUp
- RandAugment

Regularization +
Augmentation gives
big improvements
over original results

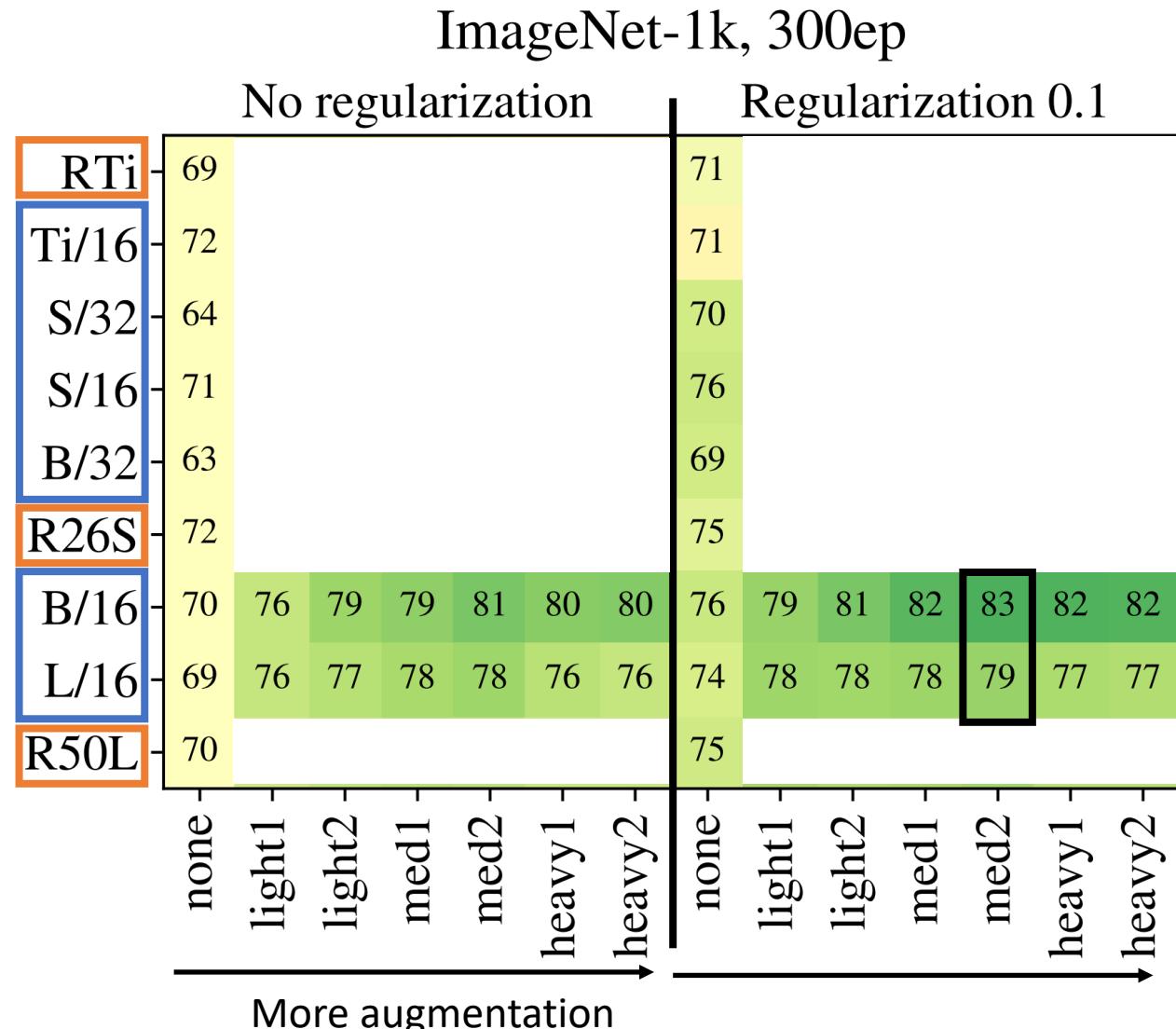
Hybrid models:
ResNet blocks,
then ViT blocks

ViT models:
 $Ti = \text{Tiny}$
 $S = \text{Small}$
 $B = \text{Base}$
 $L = \text{Large}$

Original Paper:

77.9

76.53



Improving ViT: Augmentation and Regularization

Regularization for ViT models:

- Weight Decay
- Stochastic Depth
- Dropout (in FFN layers of Transformer)

Data Augmentation for ViT models:

- MixUp
- RandAugment

Hybrid models:
ResNet blocks,
then ViT blocks

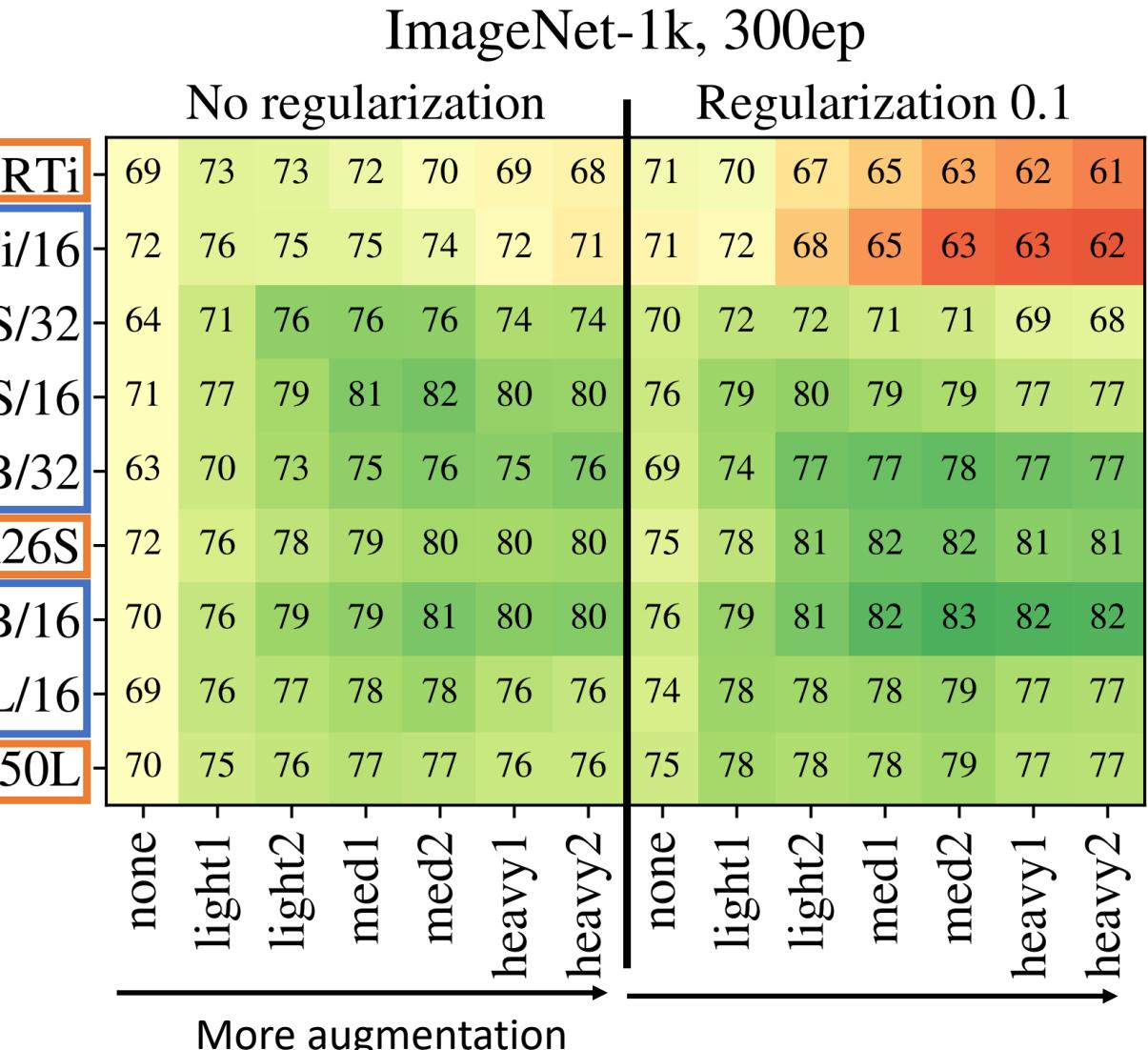
ViT models:
 T_i = Tiny
 S = Small
 B = Base
 L = Large

Original Paper:

77.9

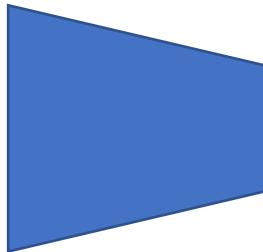
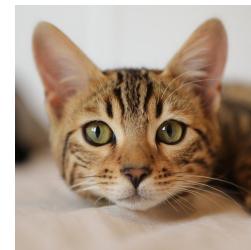
76.53

Lots of other
patterns in
full results

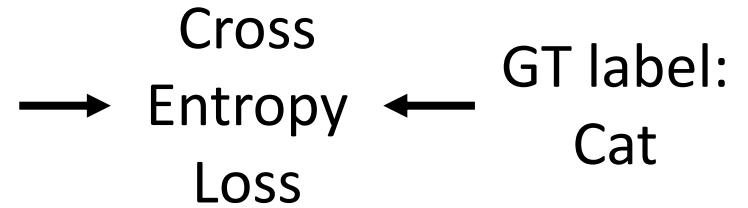


Improving ViT: Distillation

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

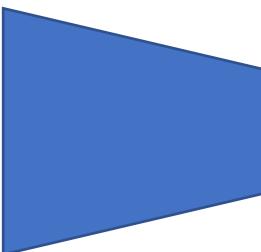
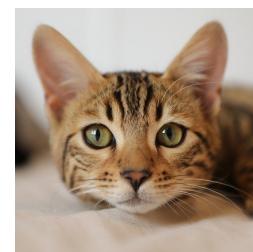


$$\begin{aligned} P(\text{cat}) &= 0.9 \\ P(\text{dog}) &= 0.1 \end{aligned}$$

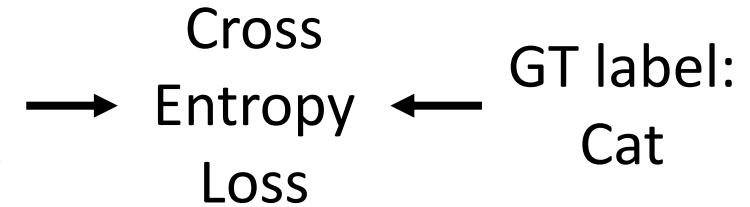


Improving ViT: Distillation

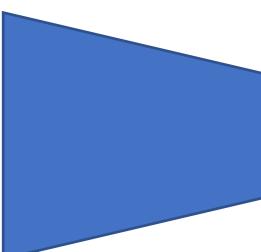
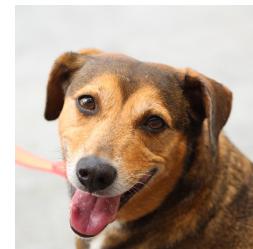
Step 1: Train a **teacher model** on images and ground-truth labels



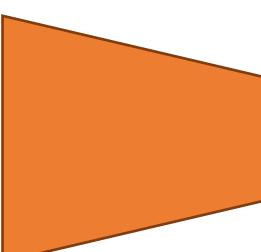
$$\begin{aligned} P(\text{cat}) &= 0.9 \\ P(\text{dog}) &= 0.1 \end{aligned}$$



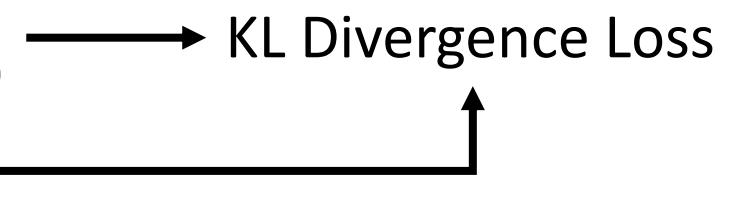
Step 2: Train a **student model** to match predictions from the **teacher**



$$\begin{aligned} P(\text{cat}) &= 0.1 \\ P(\text{dog}) &= 0.9 \end{aligned}$$

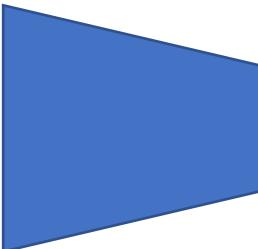
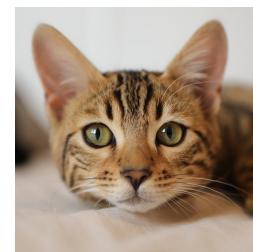


$$\begin{aligned} P(\text{cat}) &= 0.2 \\ P(\text{dog}) &= 0.8 \end{aligned}$$

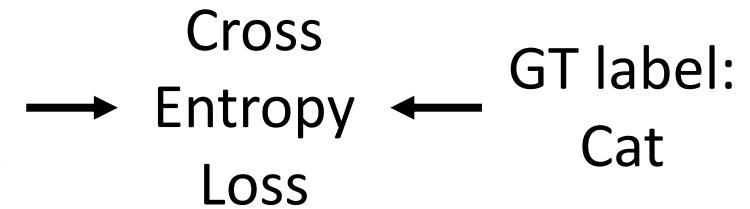


Improving ViT: Distillation

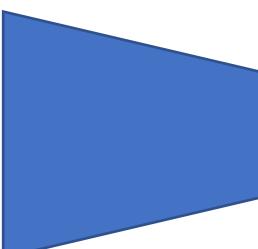
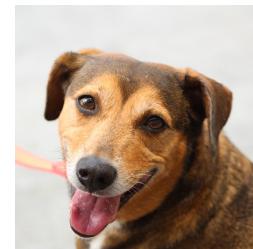
Step 1: Train a **teacher model** on images and ground-truth labels



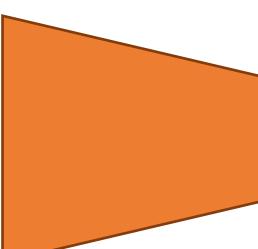
$$\begin{aligned} P(\text{cat}) &= 0.9 \\ P(\text{dog}) &= 0.1 \end{aligned}$$



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



$$\begin{aligned} P(\text{cat}) &= 0.1 \\ P(\text{dog}) &= 0.9 \end{aligned}$$

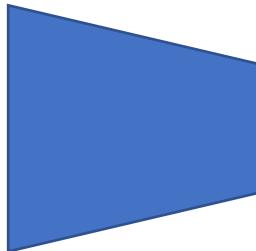
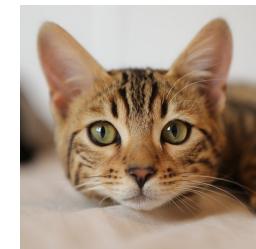


$$\begin{aligned} P(\text{cat}) &= 0.2 \\ P(\text{dog}) &= 0.8 \end{aligned}$$



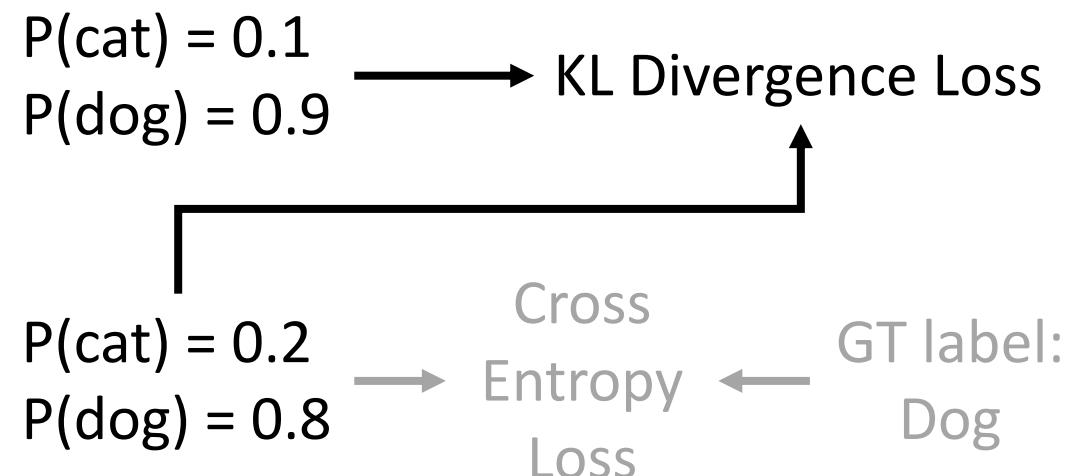
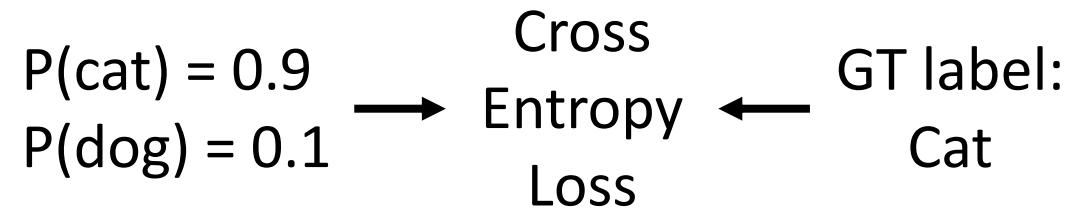
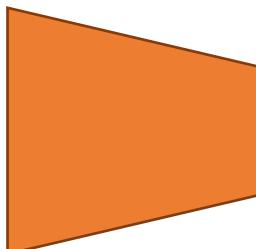
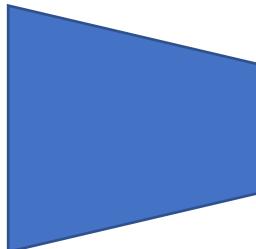
Improving ViT: Distillation

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



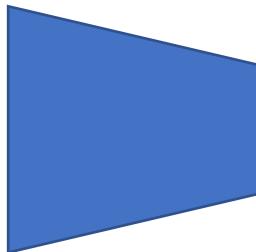
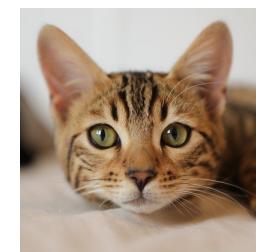
Often works better than training student from scratch (especially if teacher is bigger than student)

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



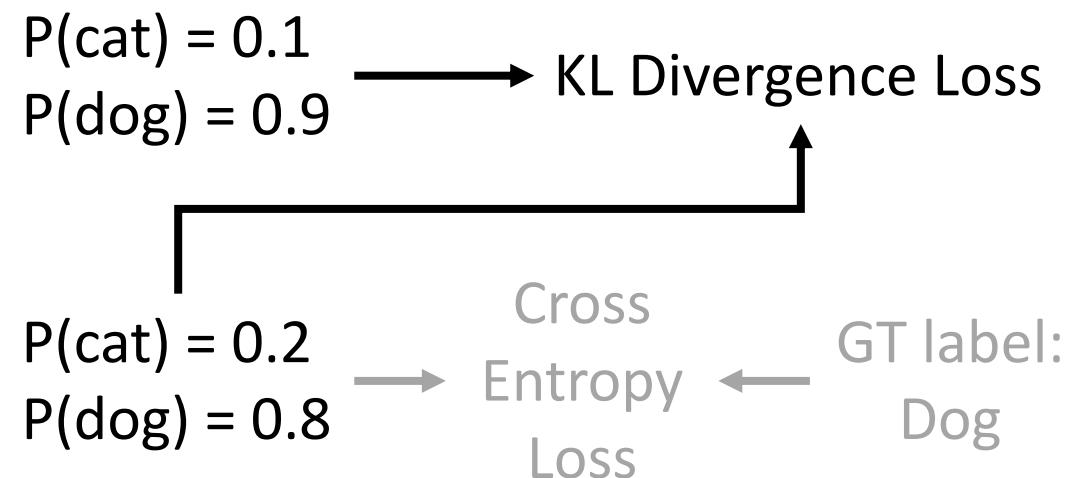
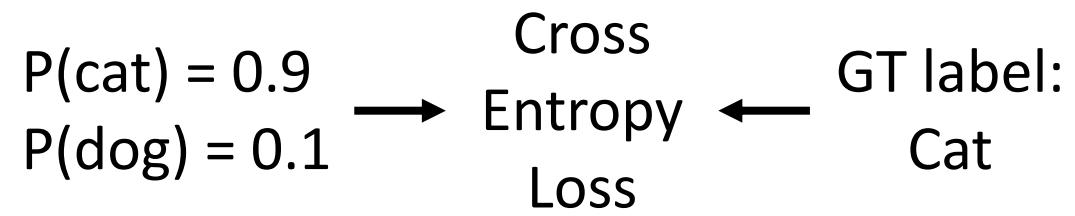
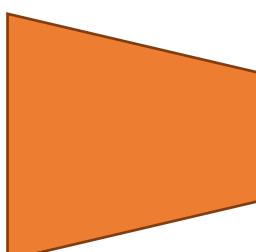
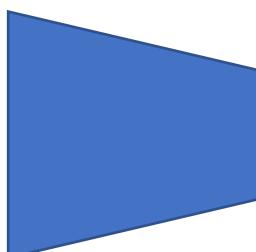
Improving ViT: Distillation

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



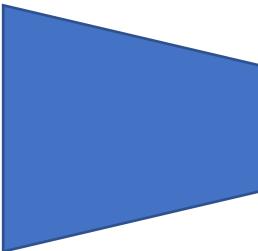
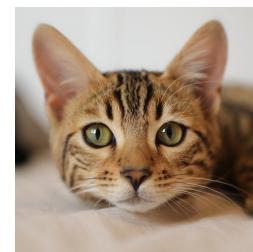
Can also train student on **unlabeled** data! (Semi-supervised learning)

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

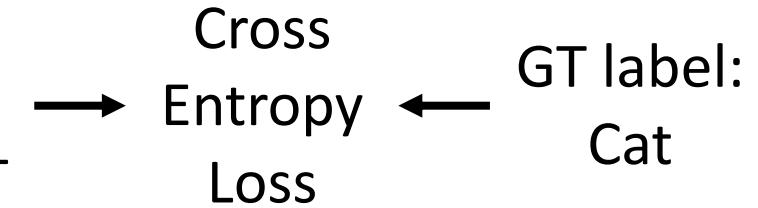


Improving ViT: Distillation

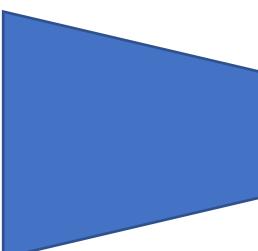
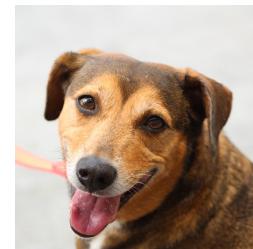
Step 1: Train a teacher CNN on ImageNet



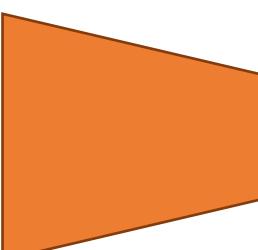
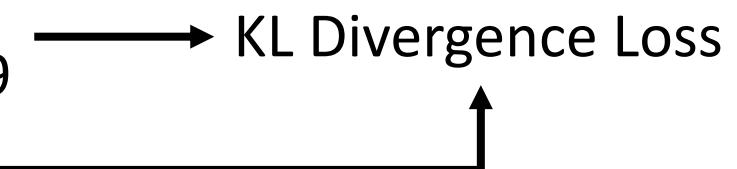
$$\begin{aligned} P(\text{cat}) &= 0.9 \\ P(\text{dog}) &= 0.1 \end{aligned}$$



Step 2: Train a student ViT to match ImageNet predictions from the teacher CNN (and match GT labels)



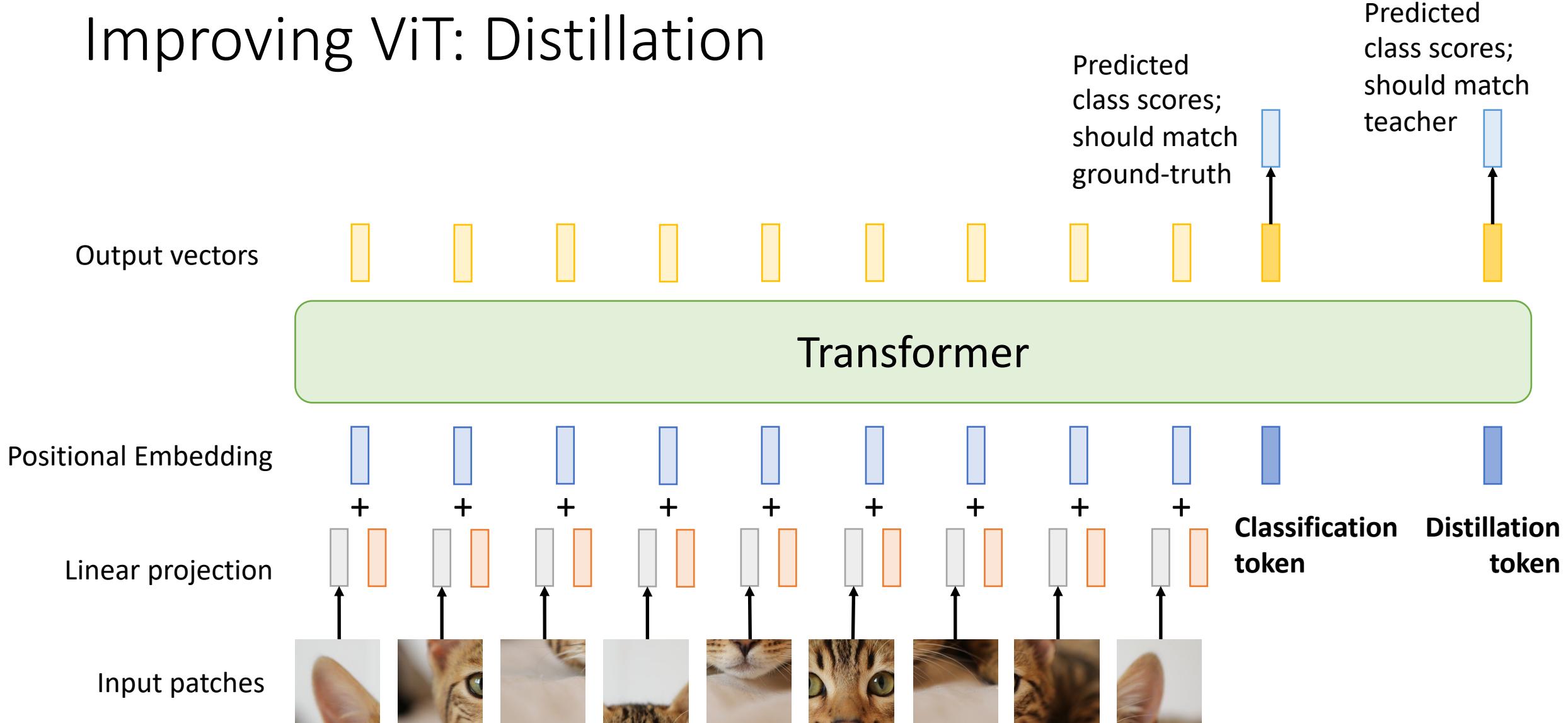
$$\begin{aligned} P(\text{cat}) &= 0.1 \\ P(\text{dog}) &= 0.9 \end{aligned}$$



$$\begin{aligned} P(\text{cat}) &= 0.2 \\ P(\text{dog}) &= 0.8 \end{aligned}$$



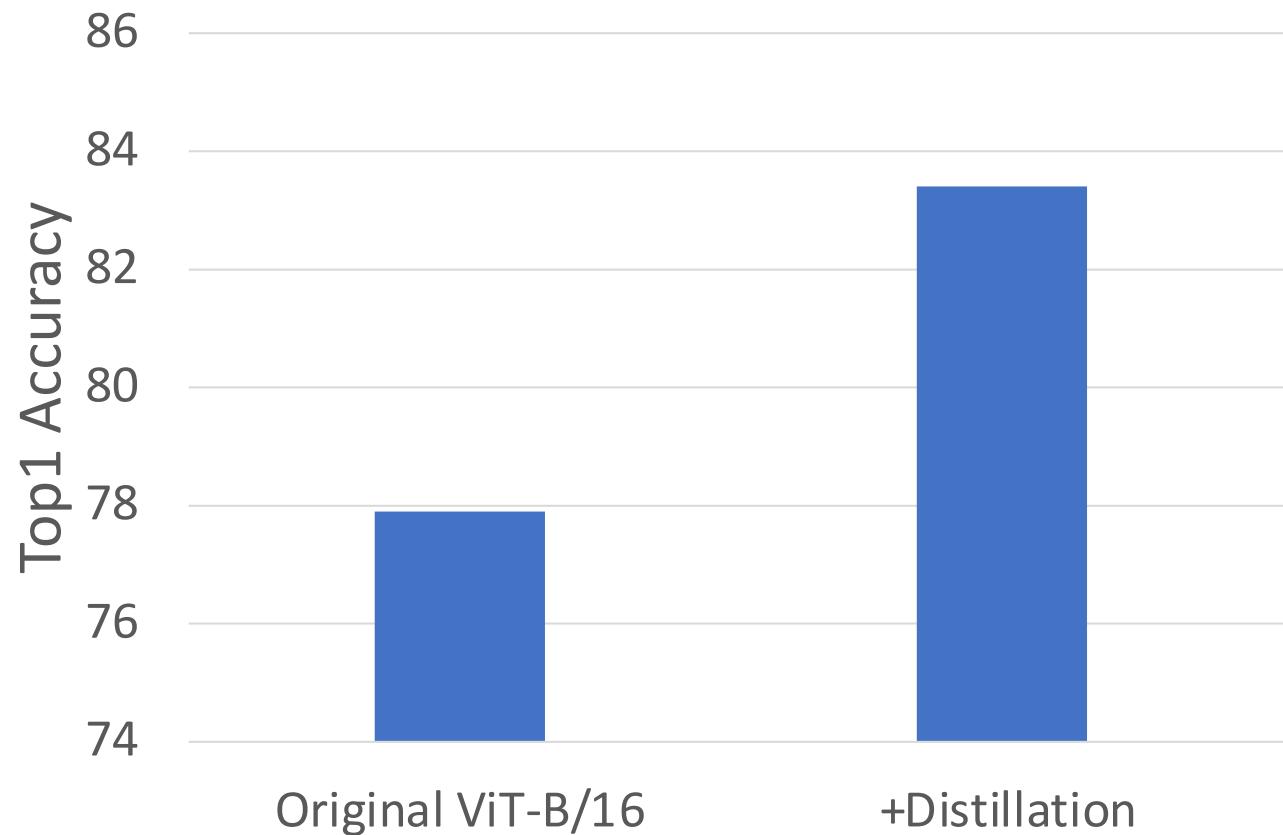
Improving ViT: Distillation



Touvron et al, "Training data-efficient image transformers & distillation through attention", ICML 2021

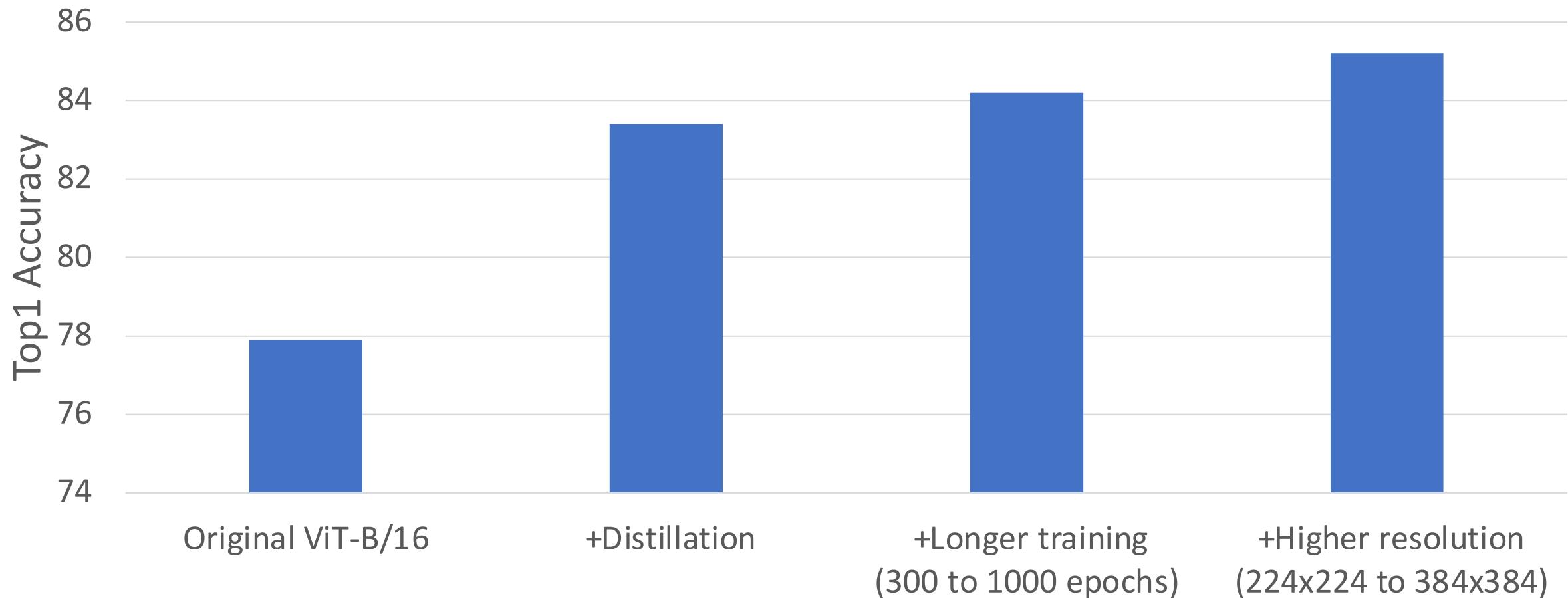
Improving ViT: Distillation

ViT-B/16 on ImageNet



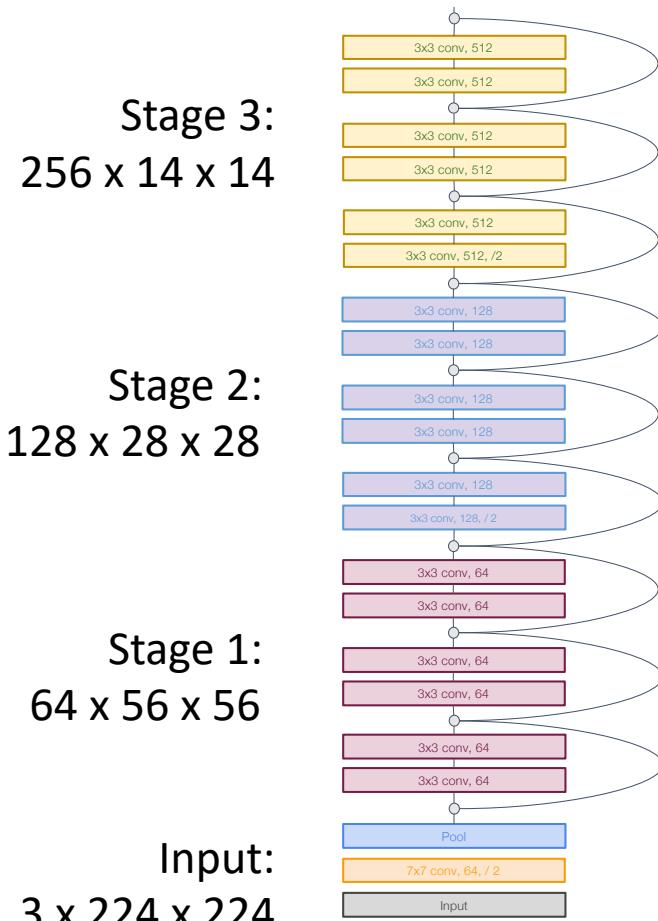
Improving ViT: Distillation

ViT-B/16 on ImageNet



Touvron et al, "Training data-efficient image transformers & distillation through attention", ICML 2021

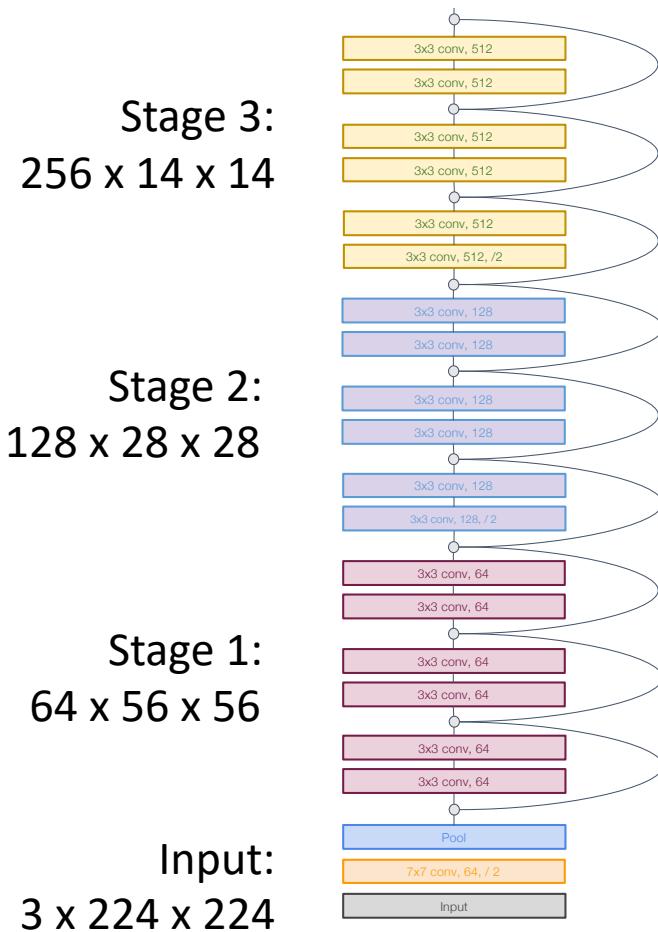
ViT vs CNN



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

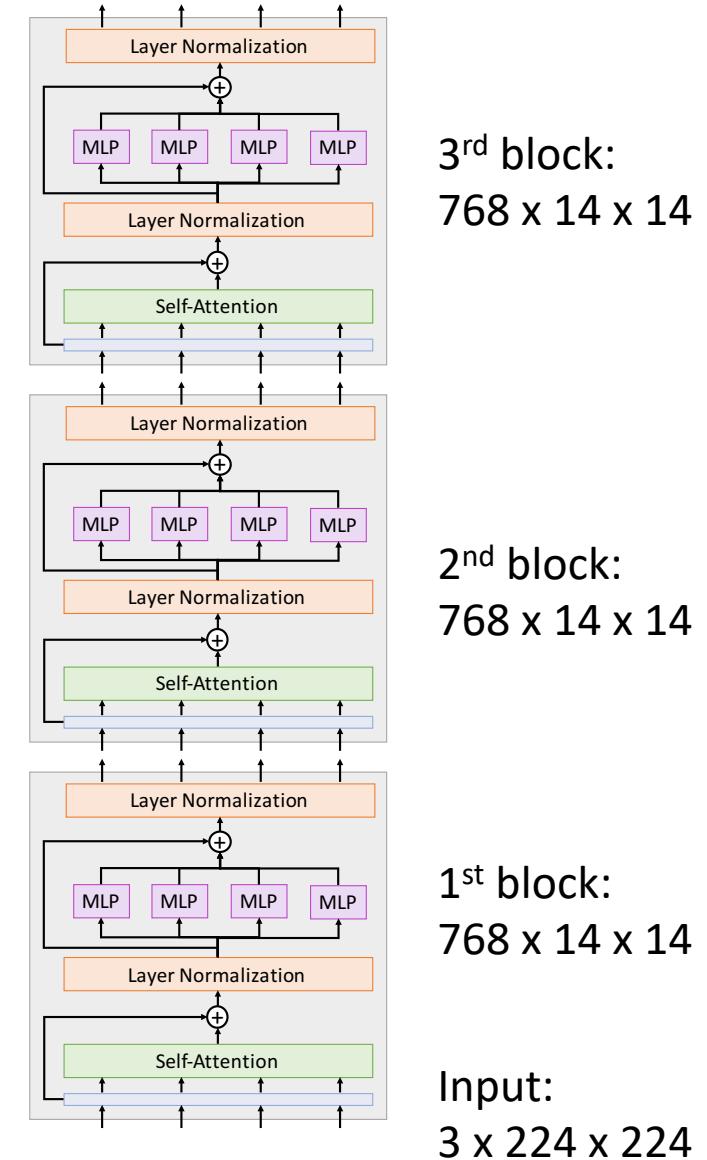
ViT vs CNN



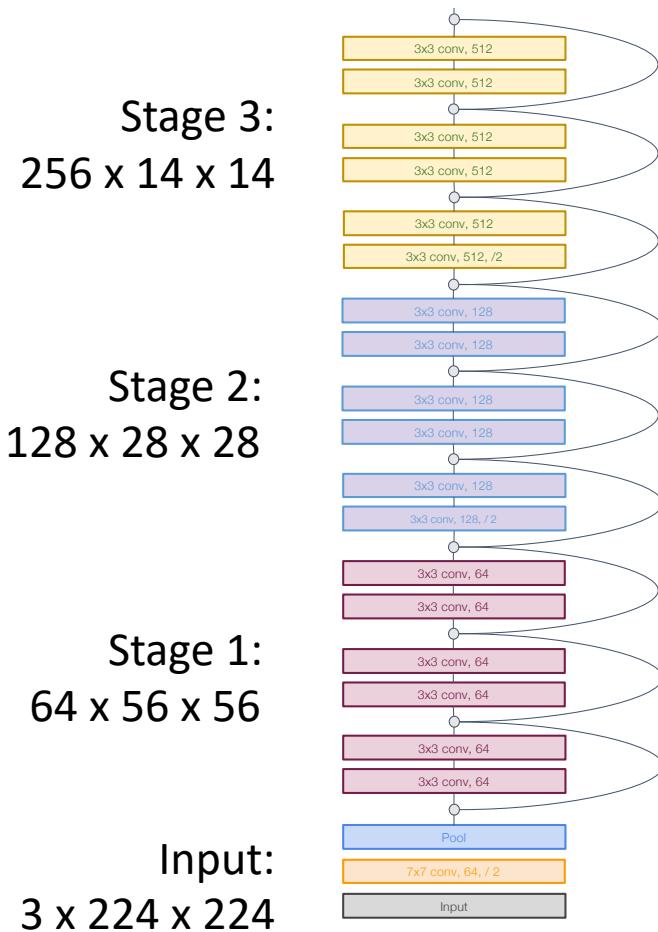
In most CNNs (including ResNets), **decrease** resolution and **increase** channels as you go deeper in the network (Hierarchical architecture)

Useful since objects in images can occur at various scales

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



ViT vs CNN

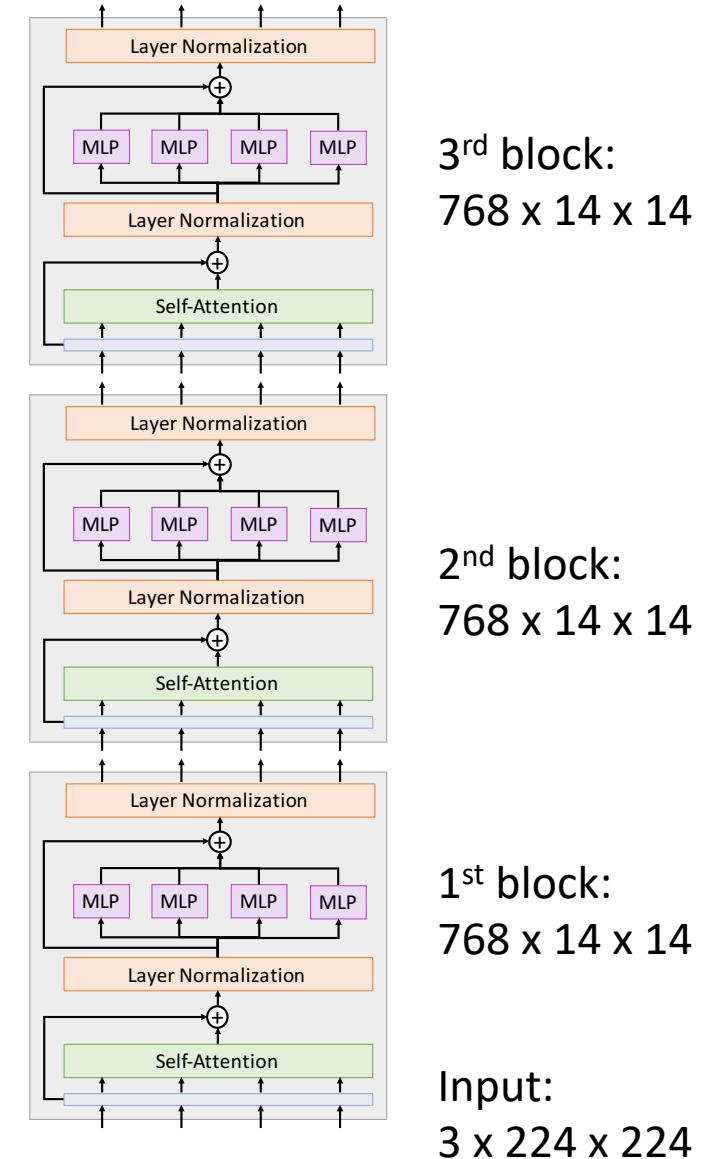


In most CNNs (including ResNets), **decrease** resolution and **increase** channels as you go deeper in the network (Hierarchical architecture)

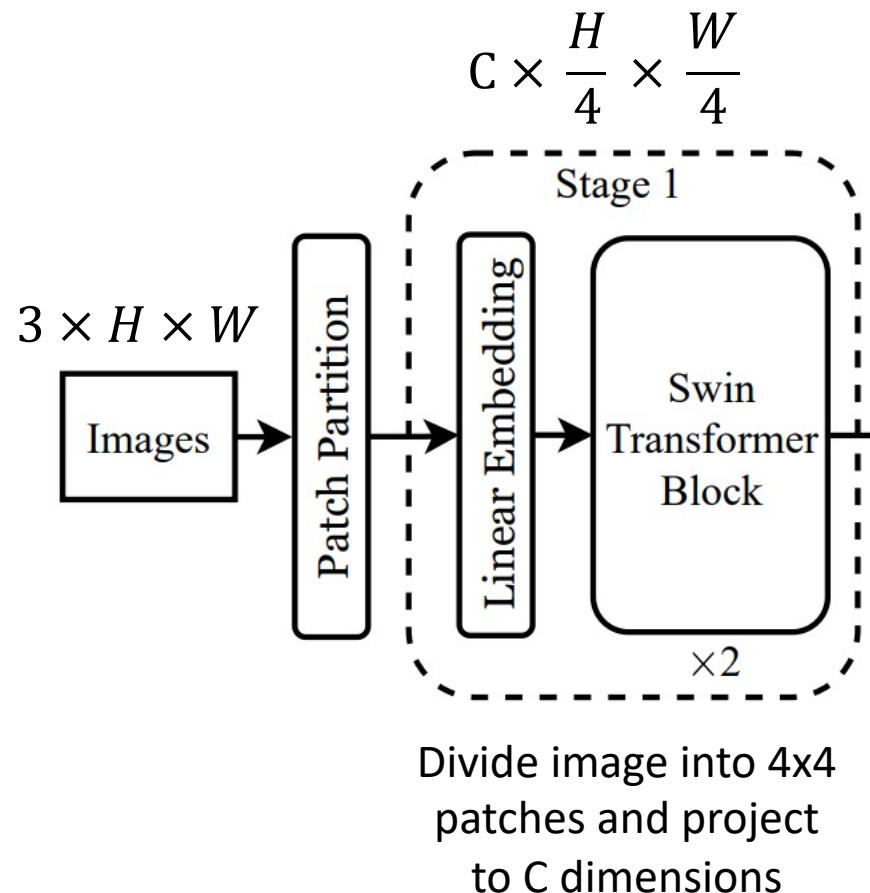
Useful since objects in images can occur at various scales

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

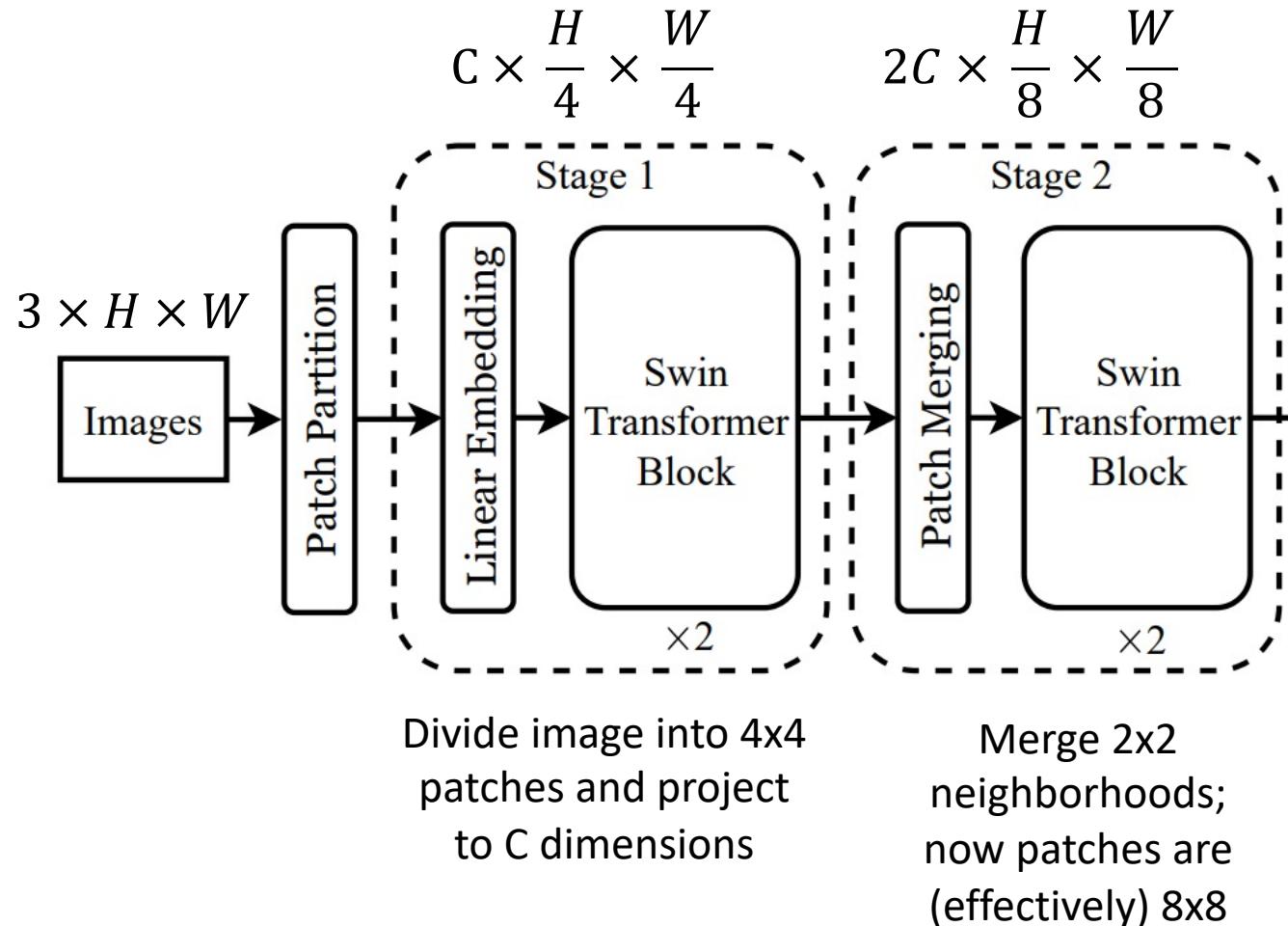
Can we build a **hierarchical** ViT model?



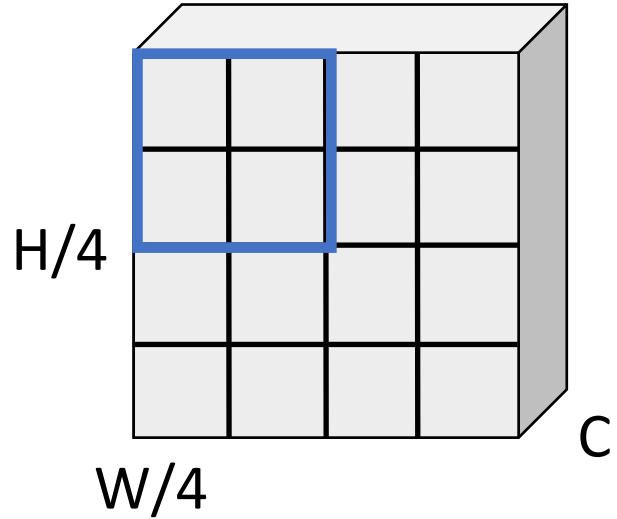
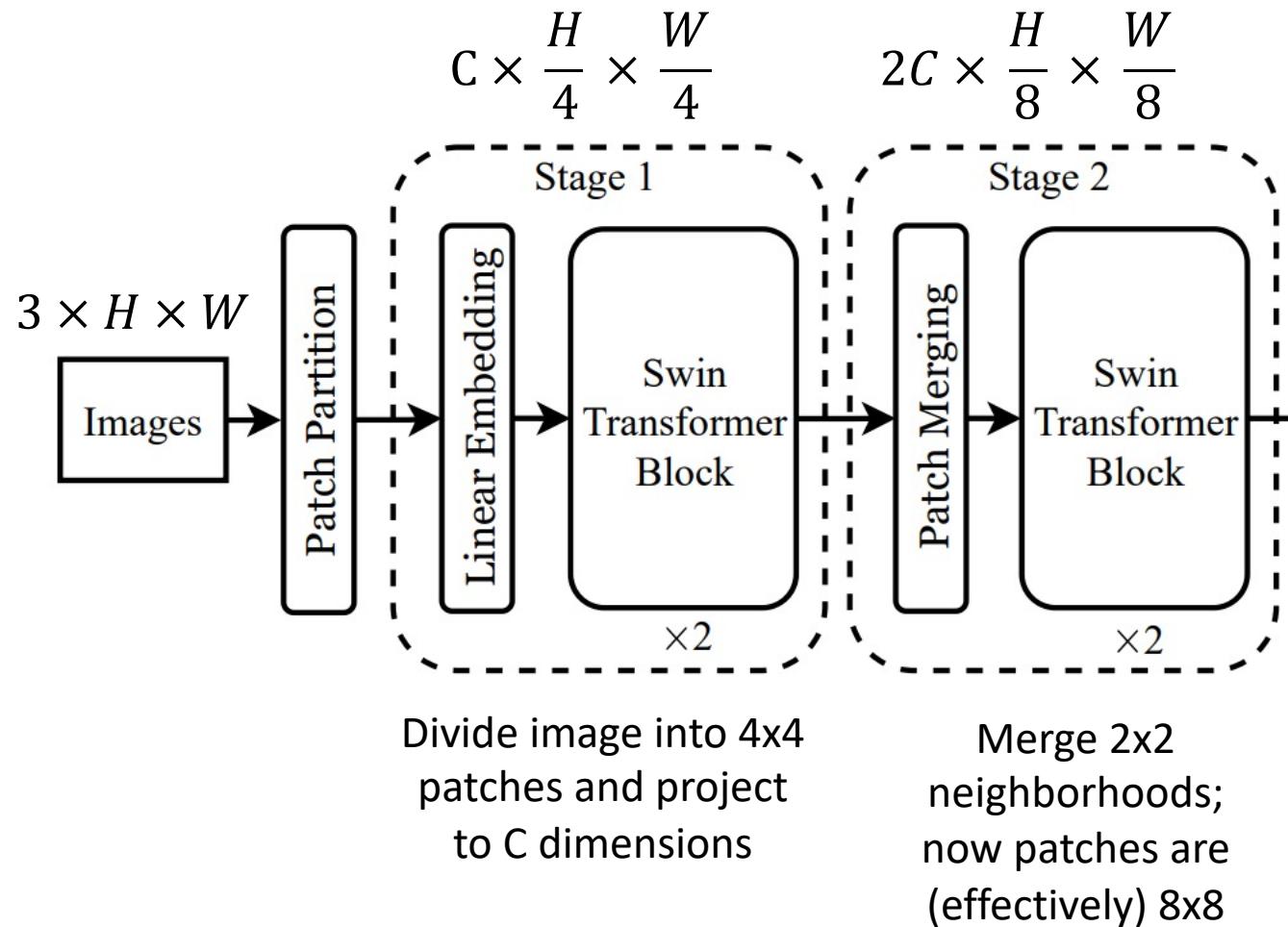
Hierarchical ViT: Swin Transformer



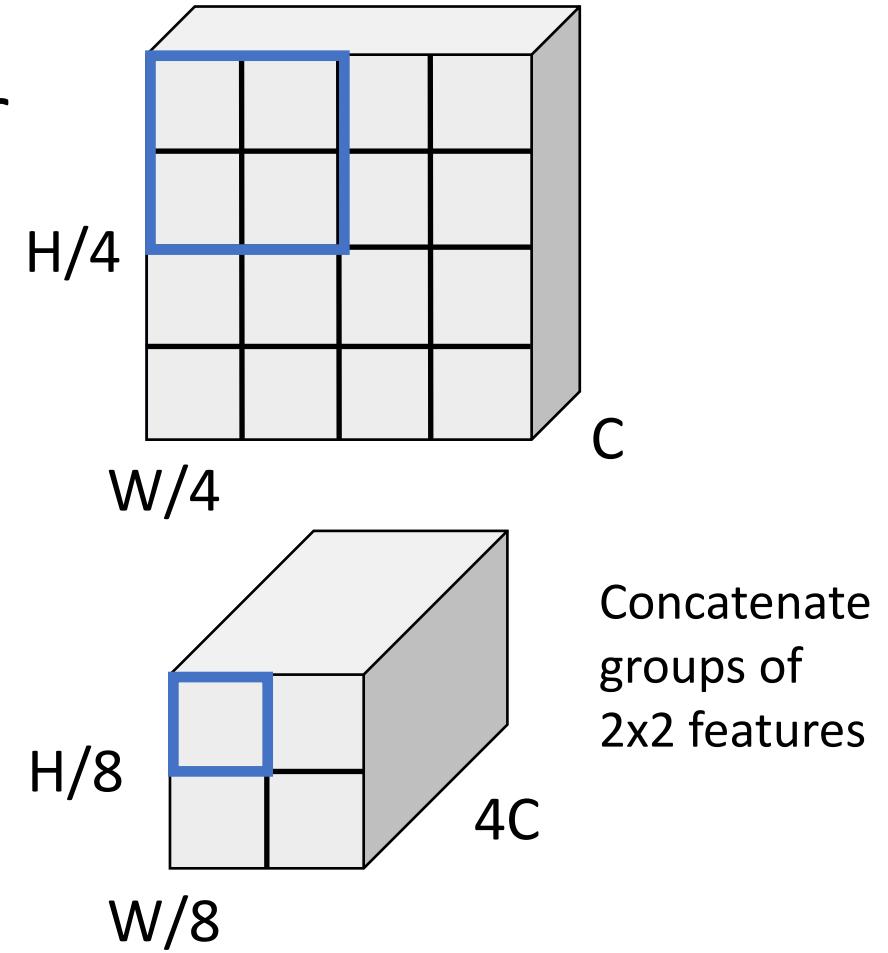
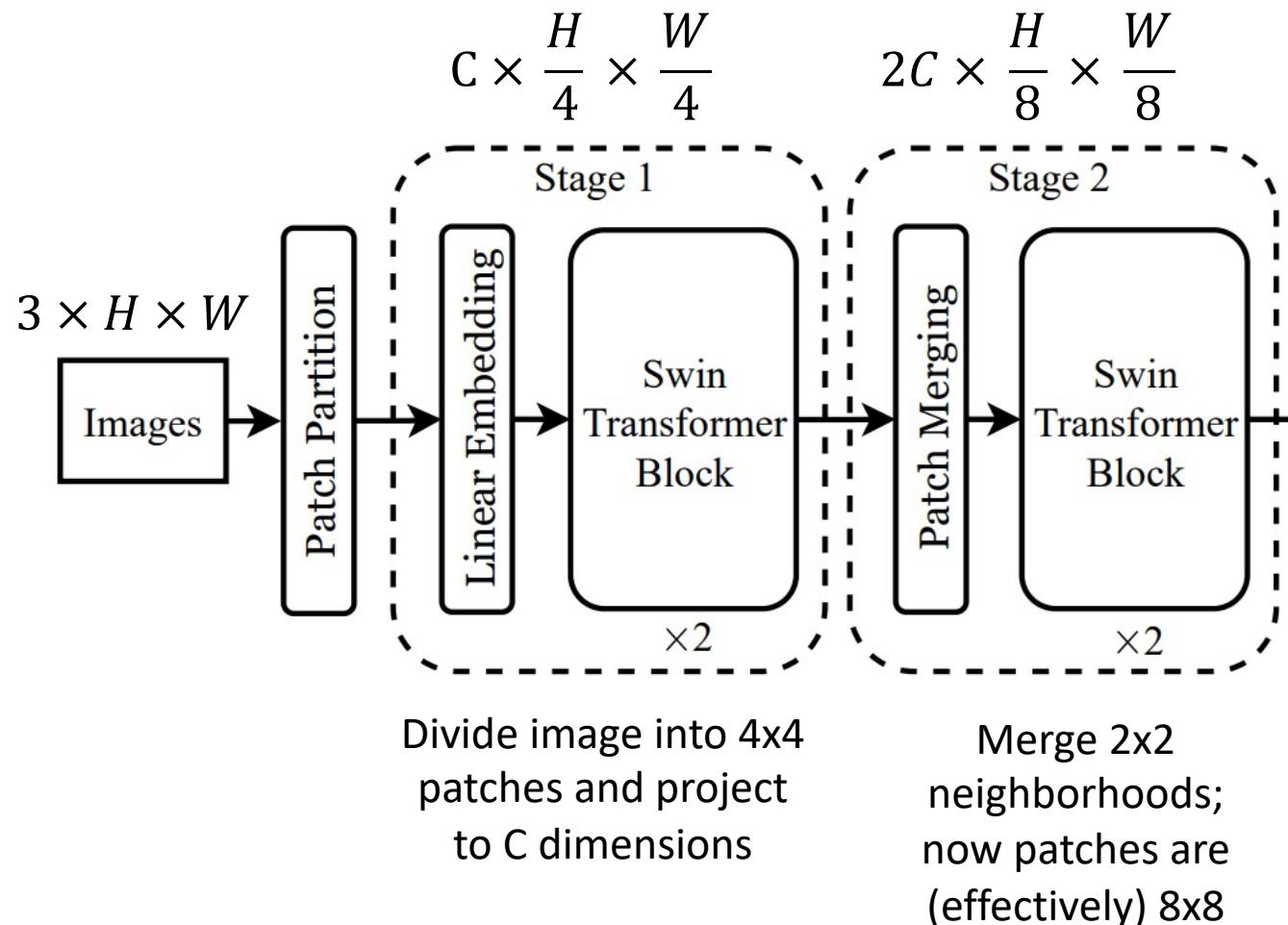
Hierarchical ViT: Swin Transformer



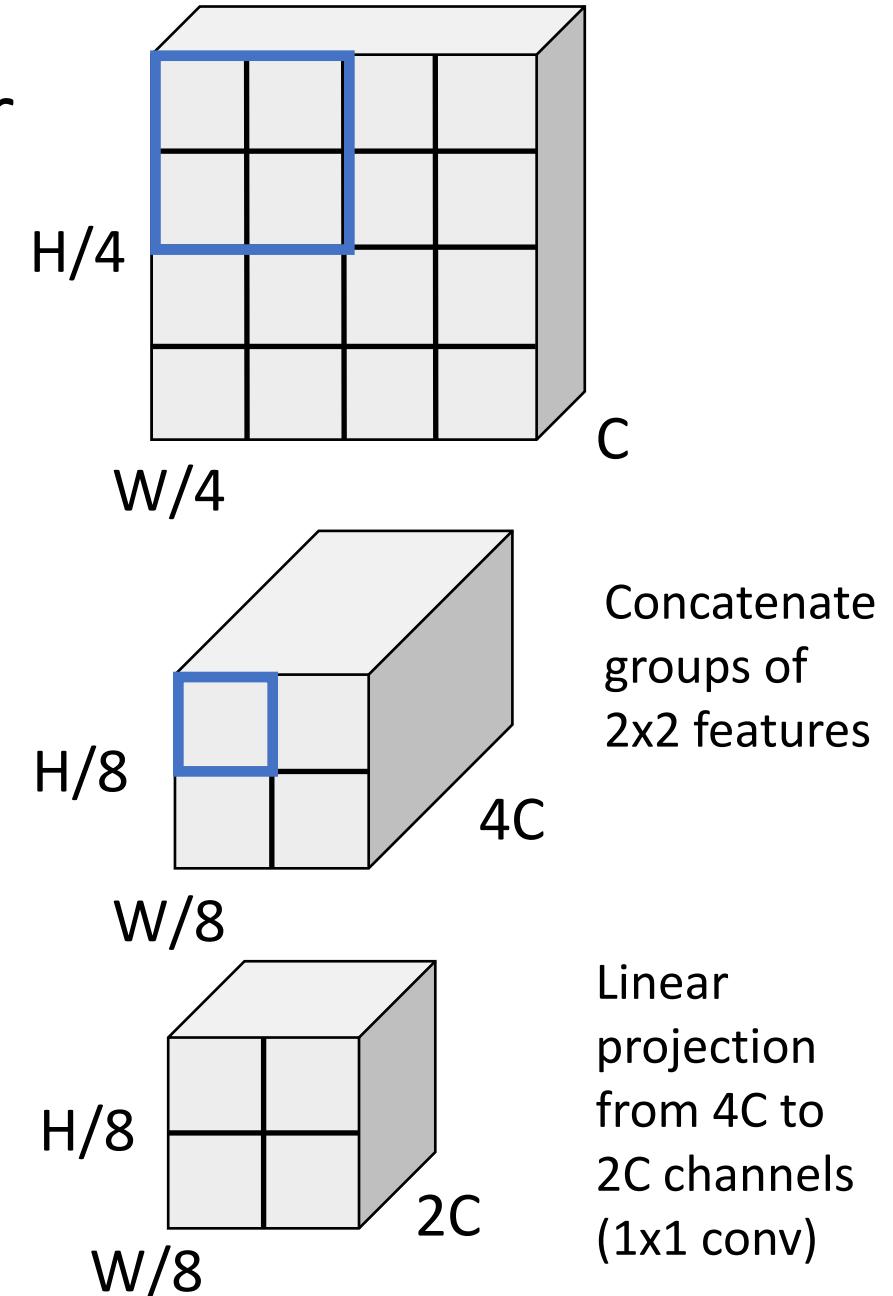
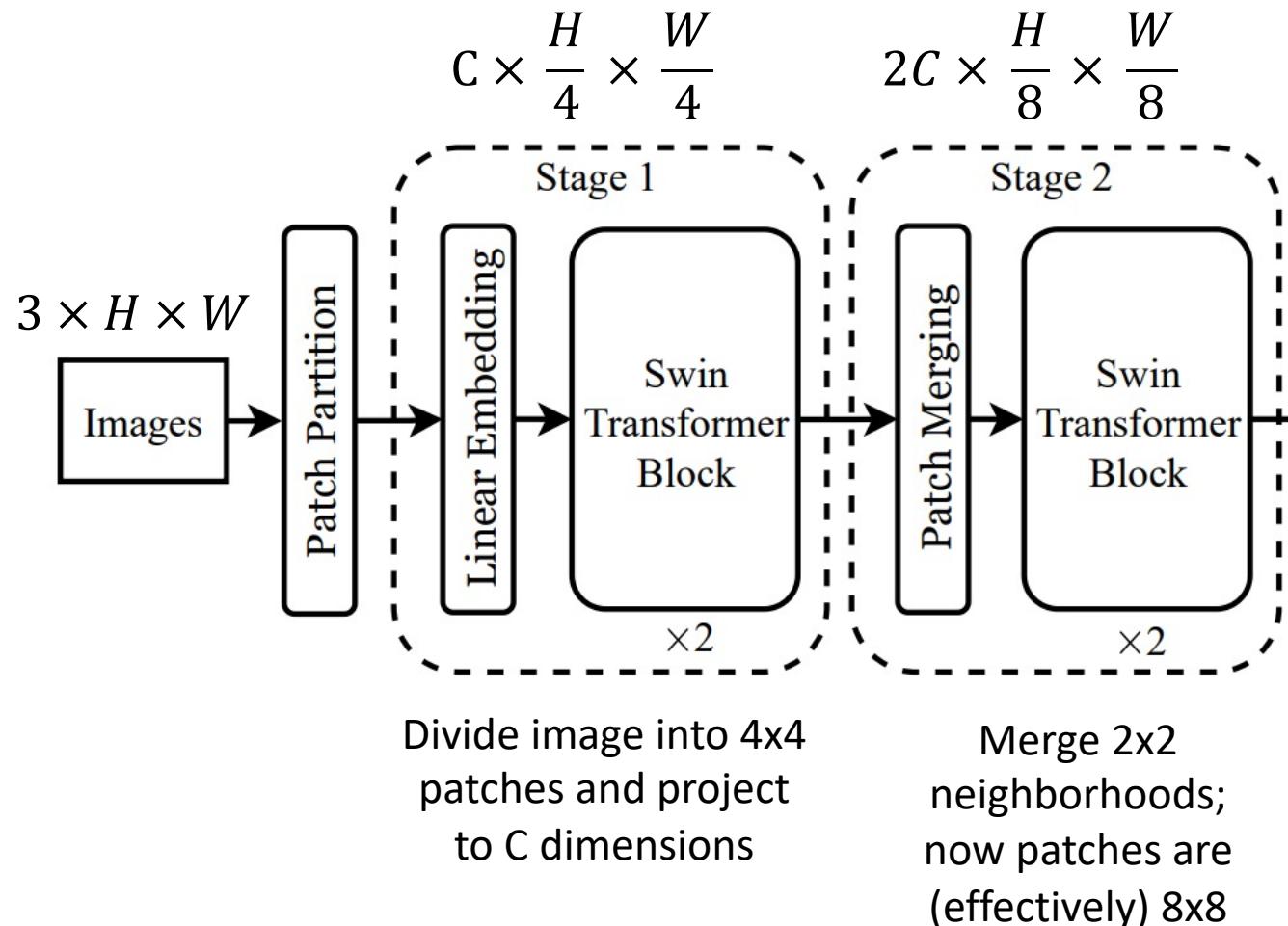
Hierarchical ViT: Swin Transformer



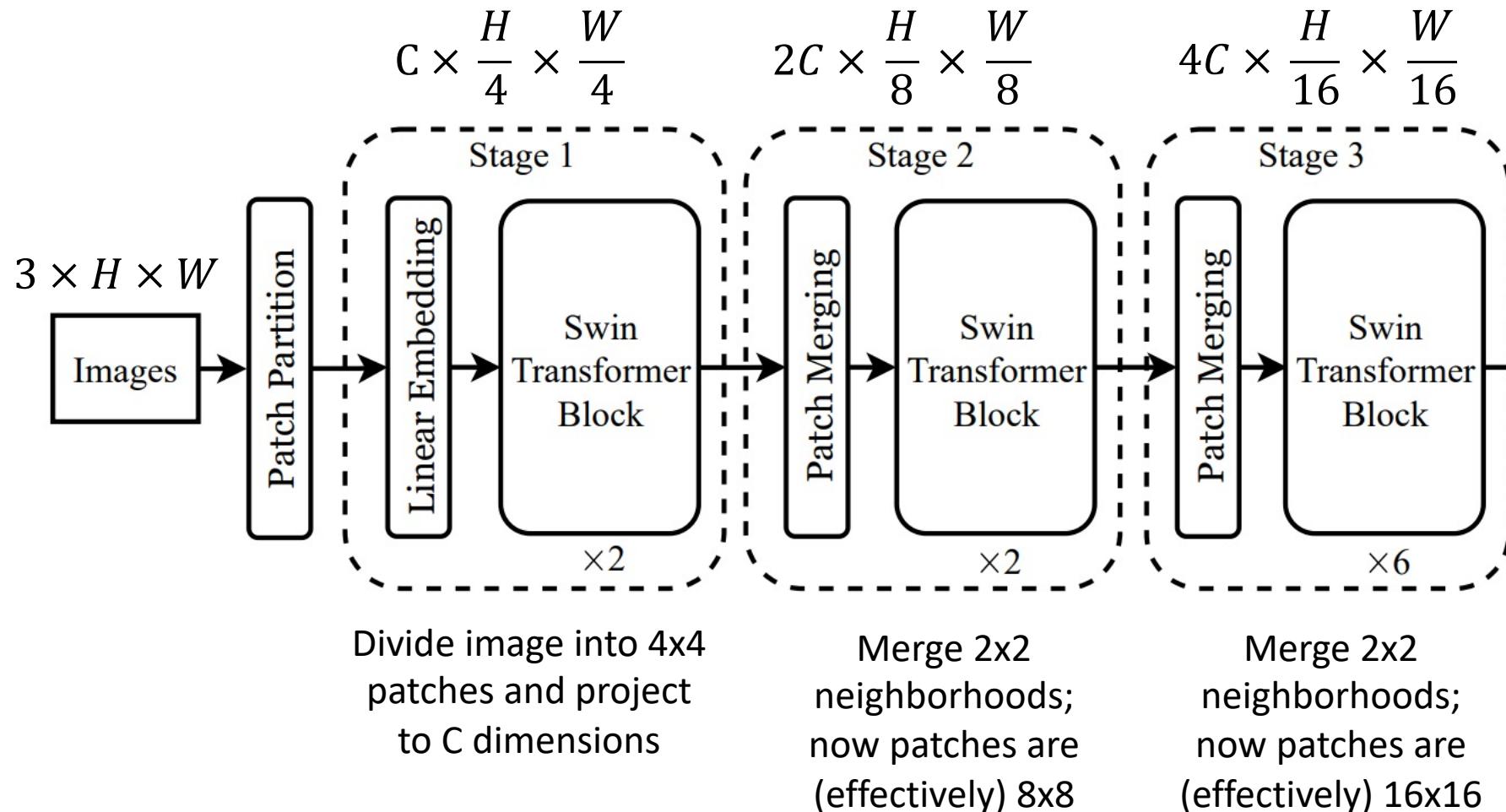
Hierarchical ViT: Swin Transformer



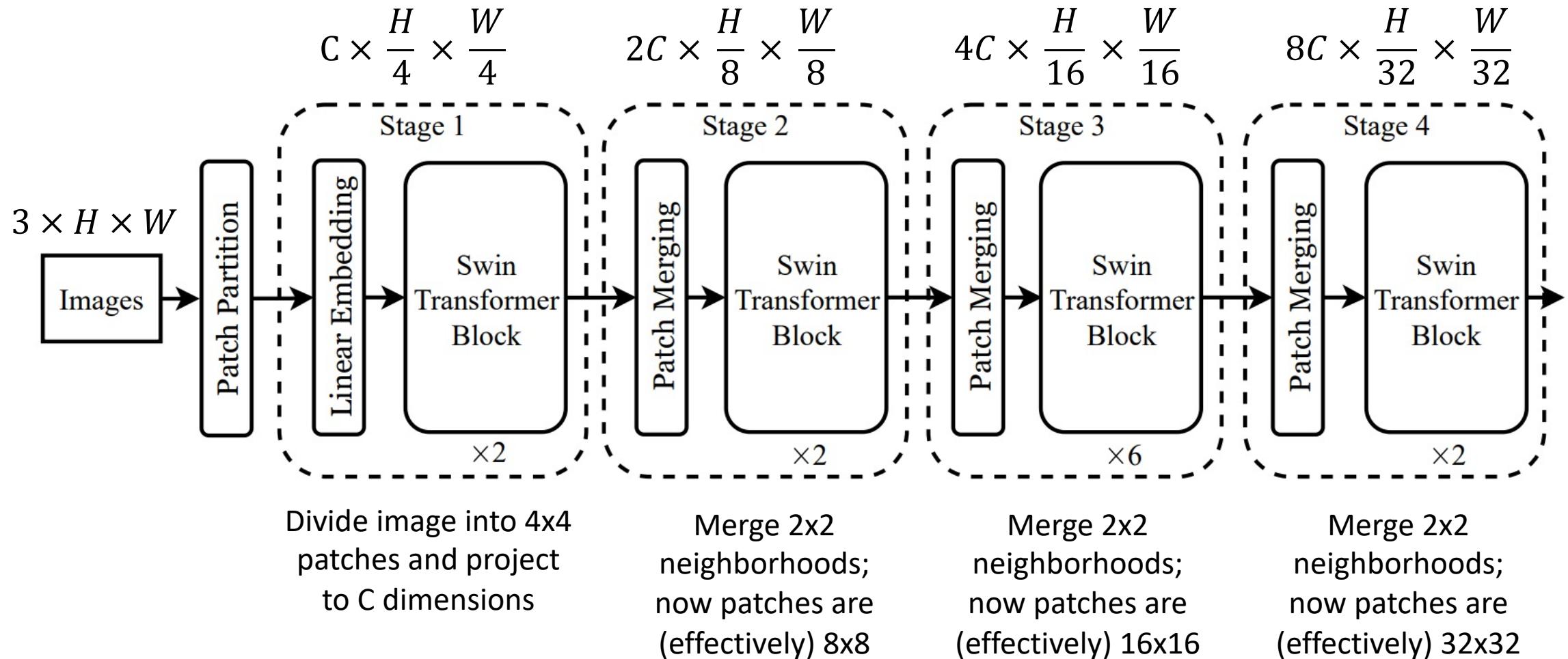
Hierarchical ViT: Swin Transformer



Hierarchical ViT: Swin Transformer

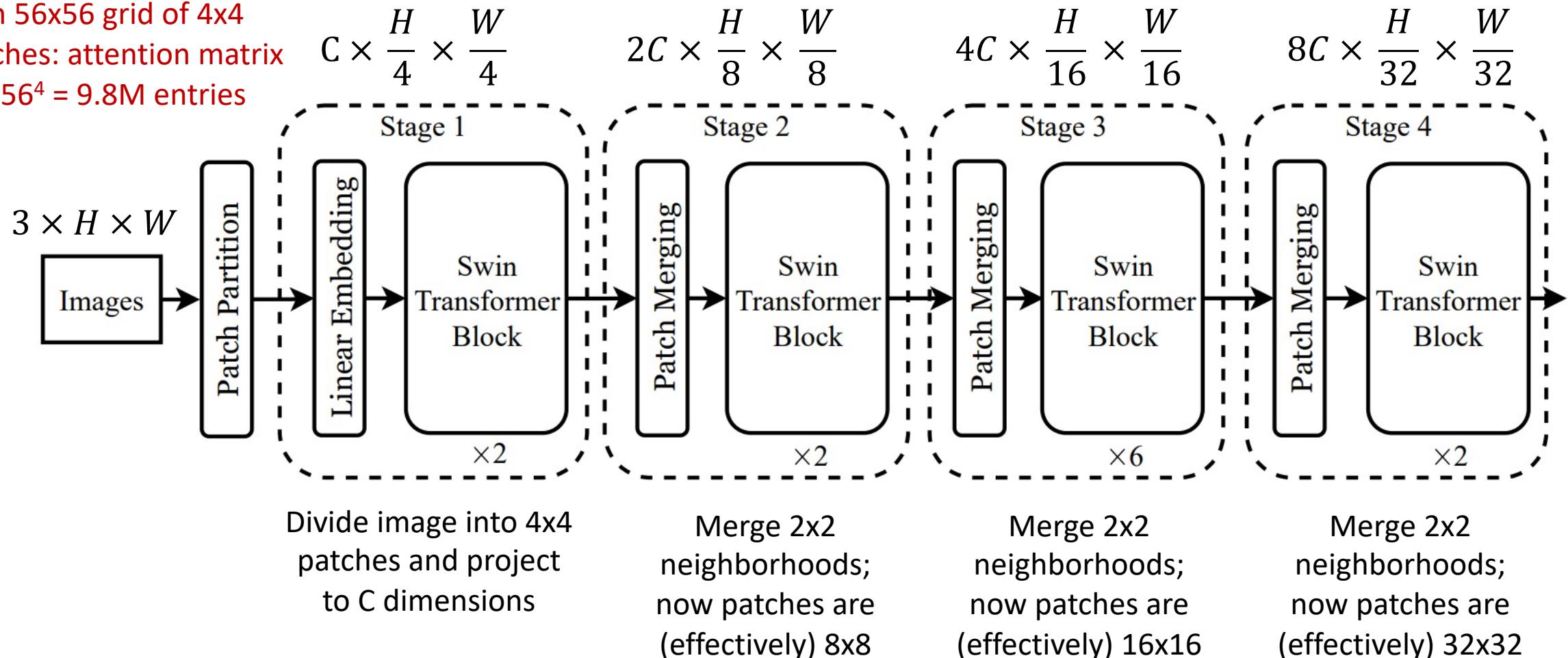


Hierarchical ViT: Swin Transformer



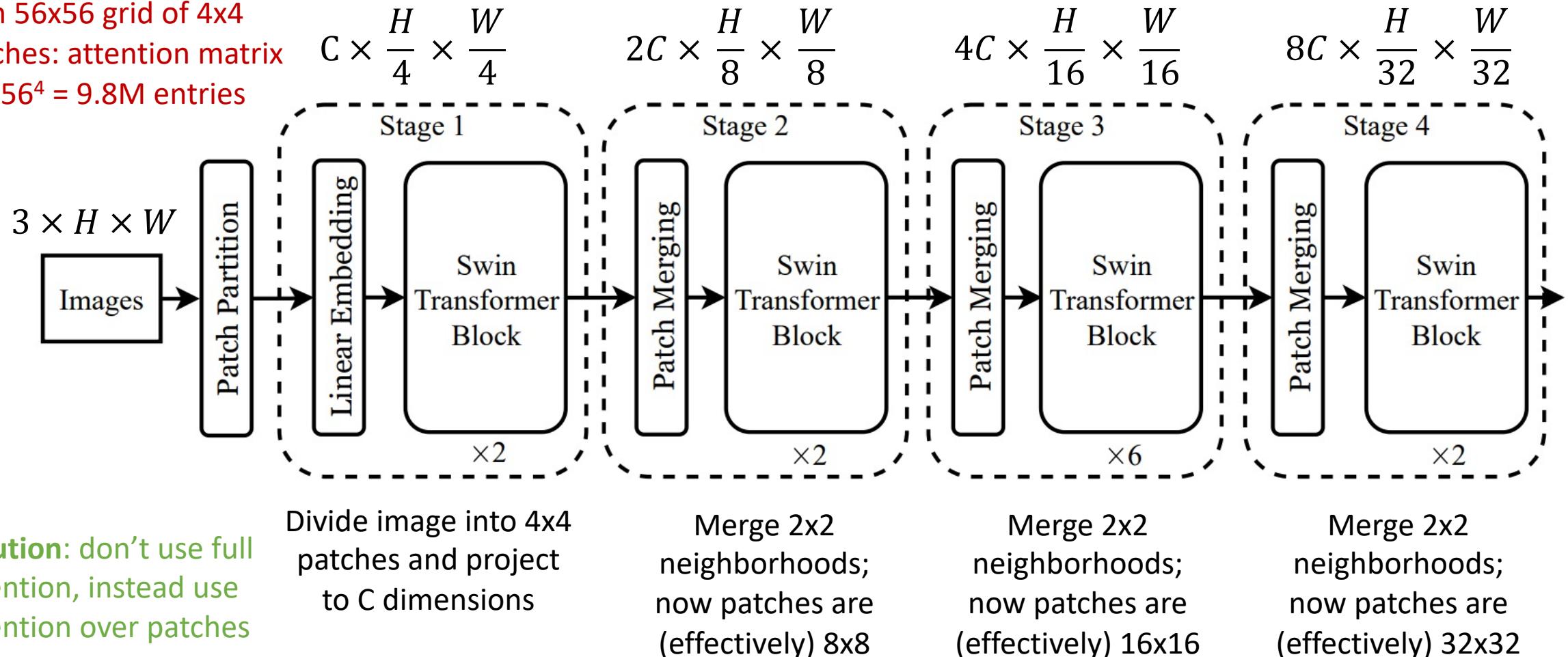
Hierarchical ViT: Swin Transformer

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



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

Swin Transformer: Window Attention

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

Swin Transformer: Window Attention



With $H \times W$ 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 $M \times M$ tokens (here $M=4$); only compute attention within each window

Swin Transformer: Window Attention



With $H \times W$ 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 $M \times M$ 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

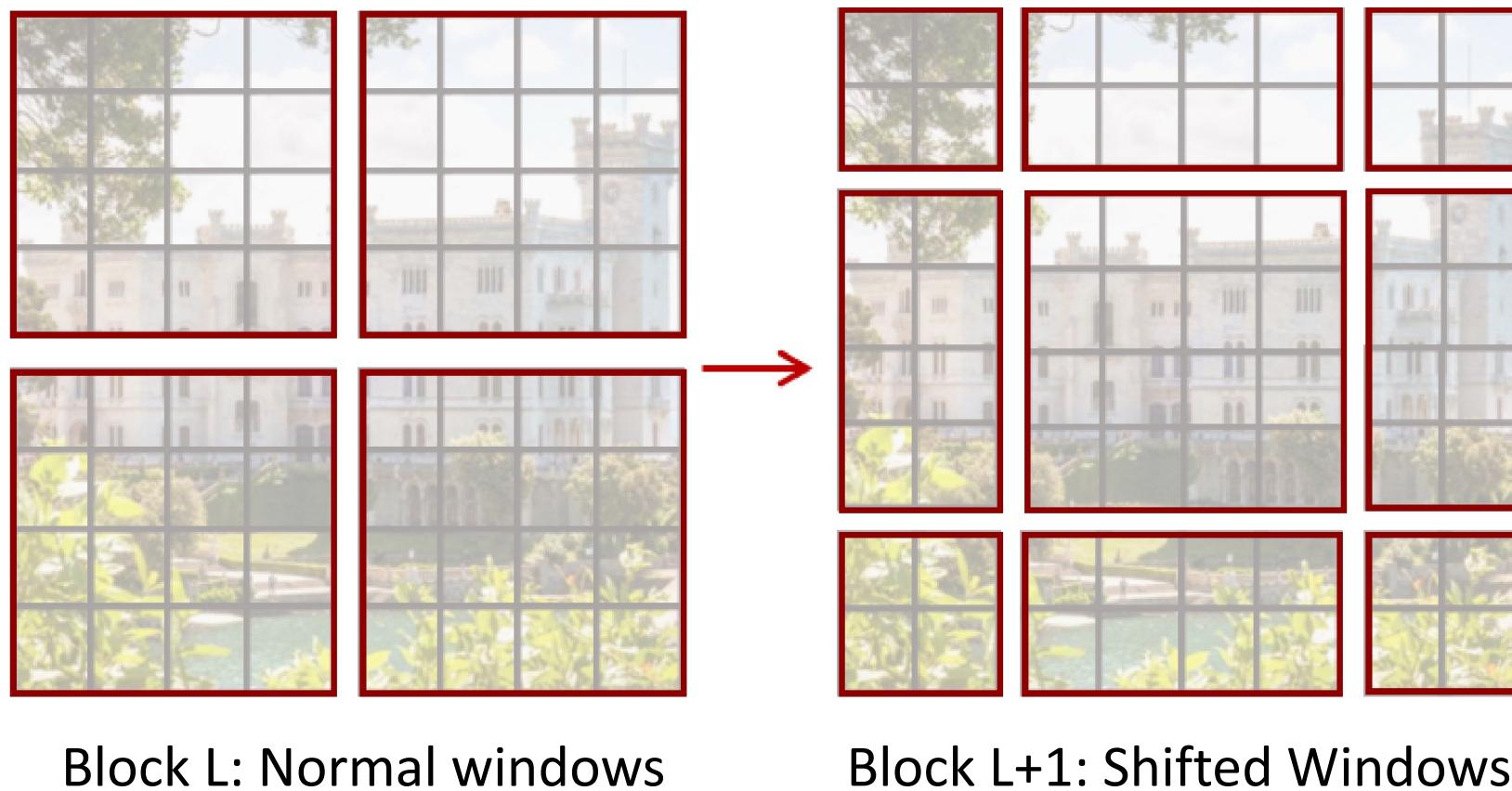
Swin Transformer: Window Attention

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



Swin Transformer: Shifted Window Attention

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

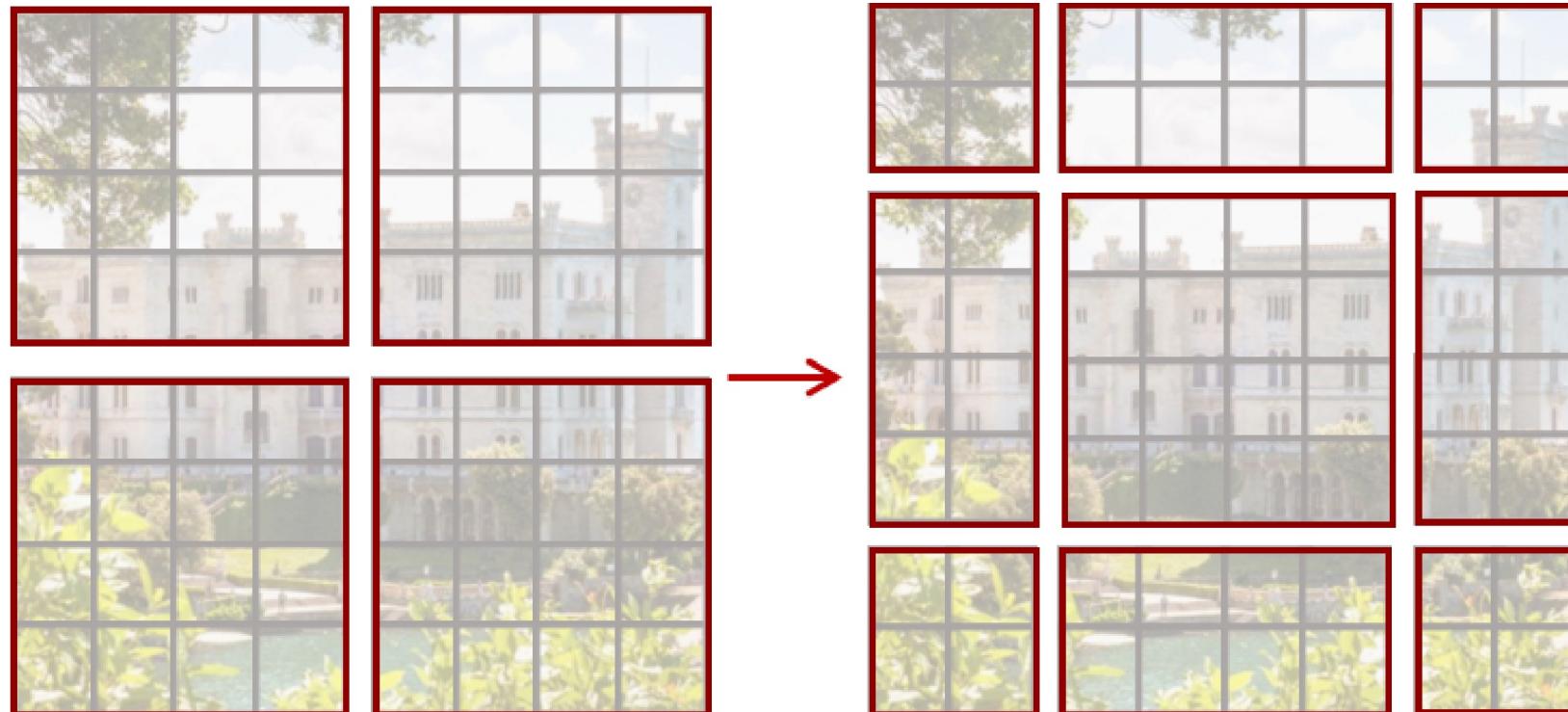


Ugly detail:
Non-square
windows at
edges and
corners

Swin Transformer: Shifted Window Attention

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

Detail: Relative Positional Bias

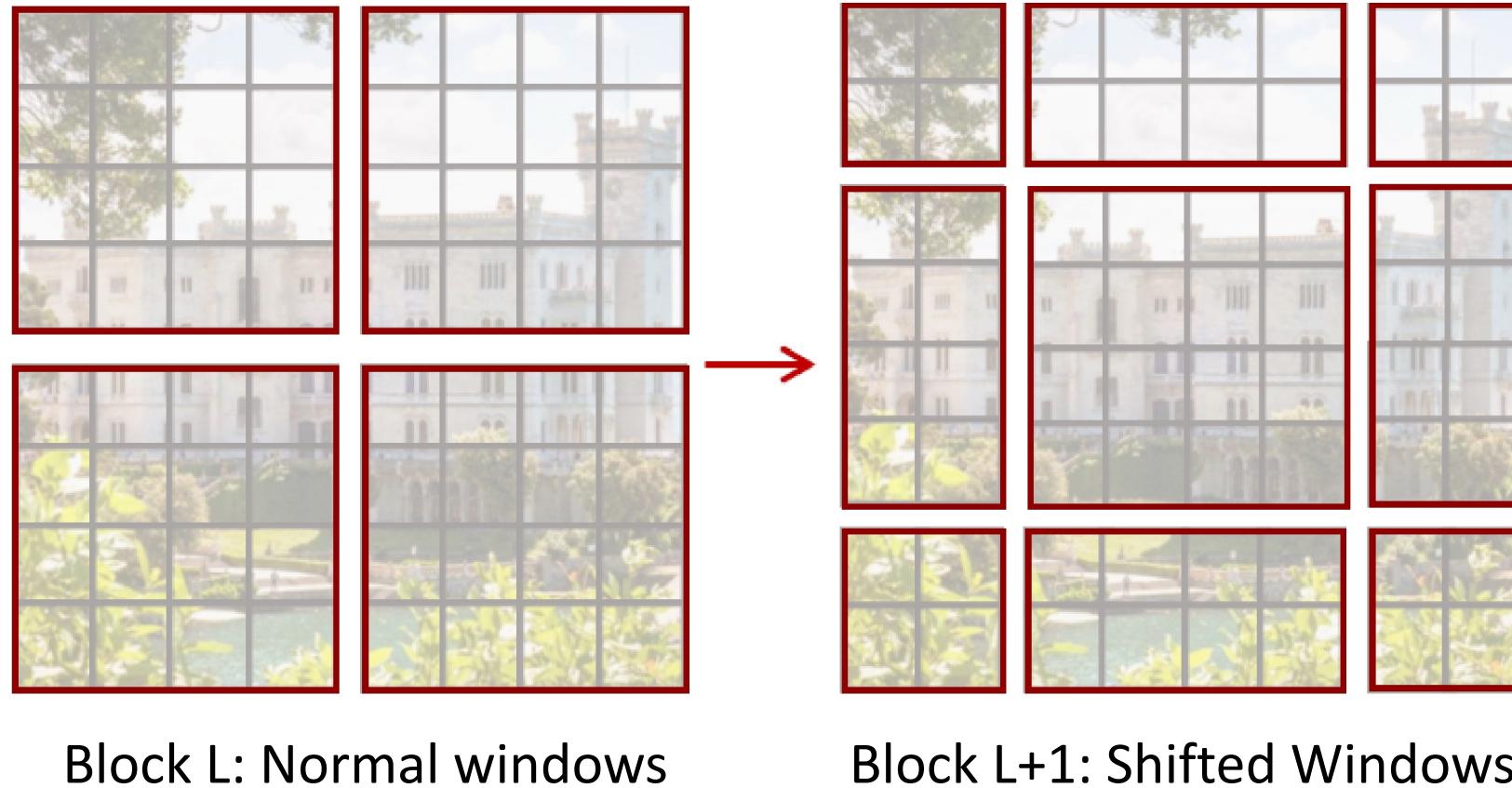


Block L: Normal windows

Block L+1: Shifted Windows

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

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

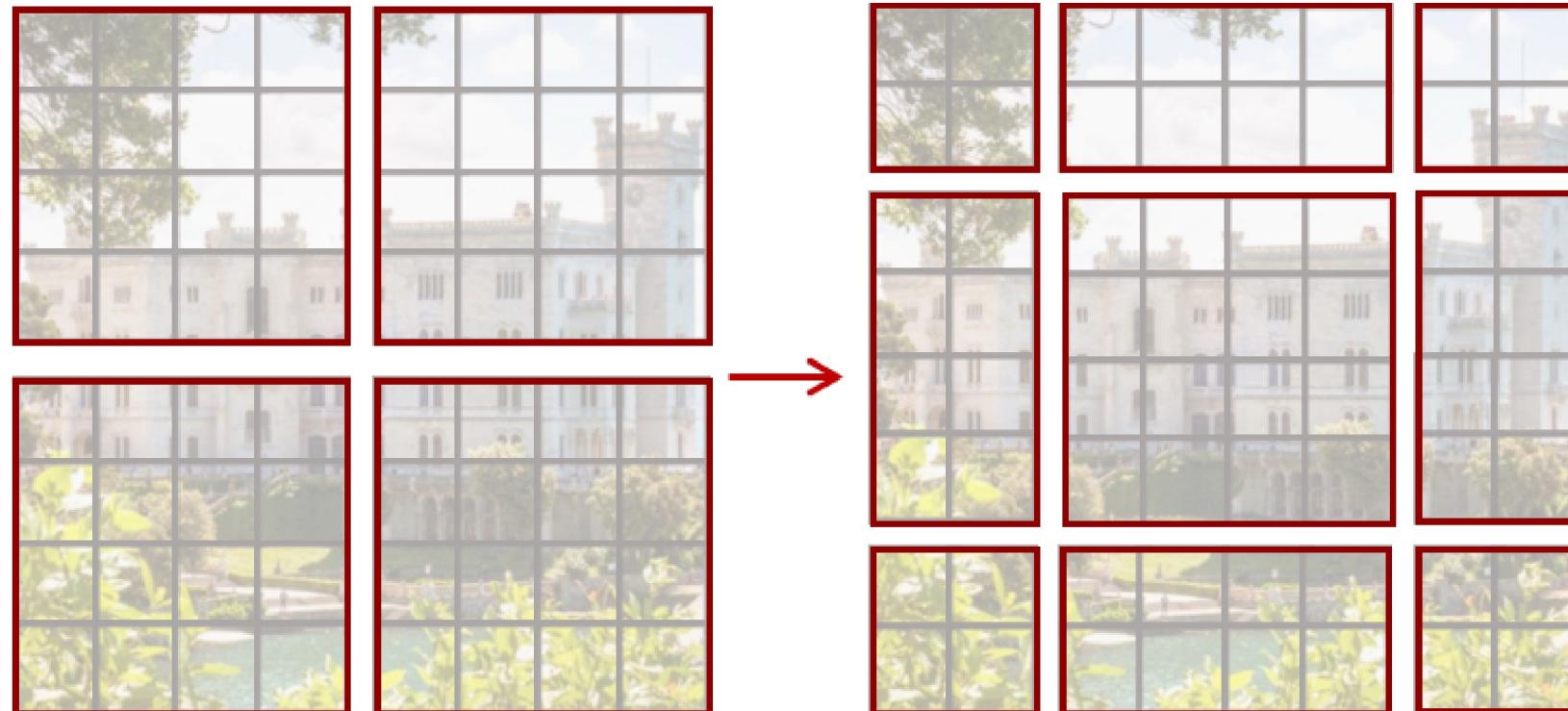
Standard Attention:

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

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

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:

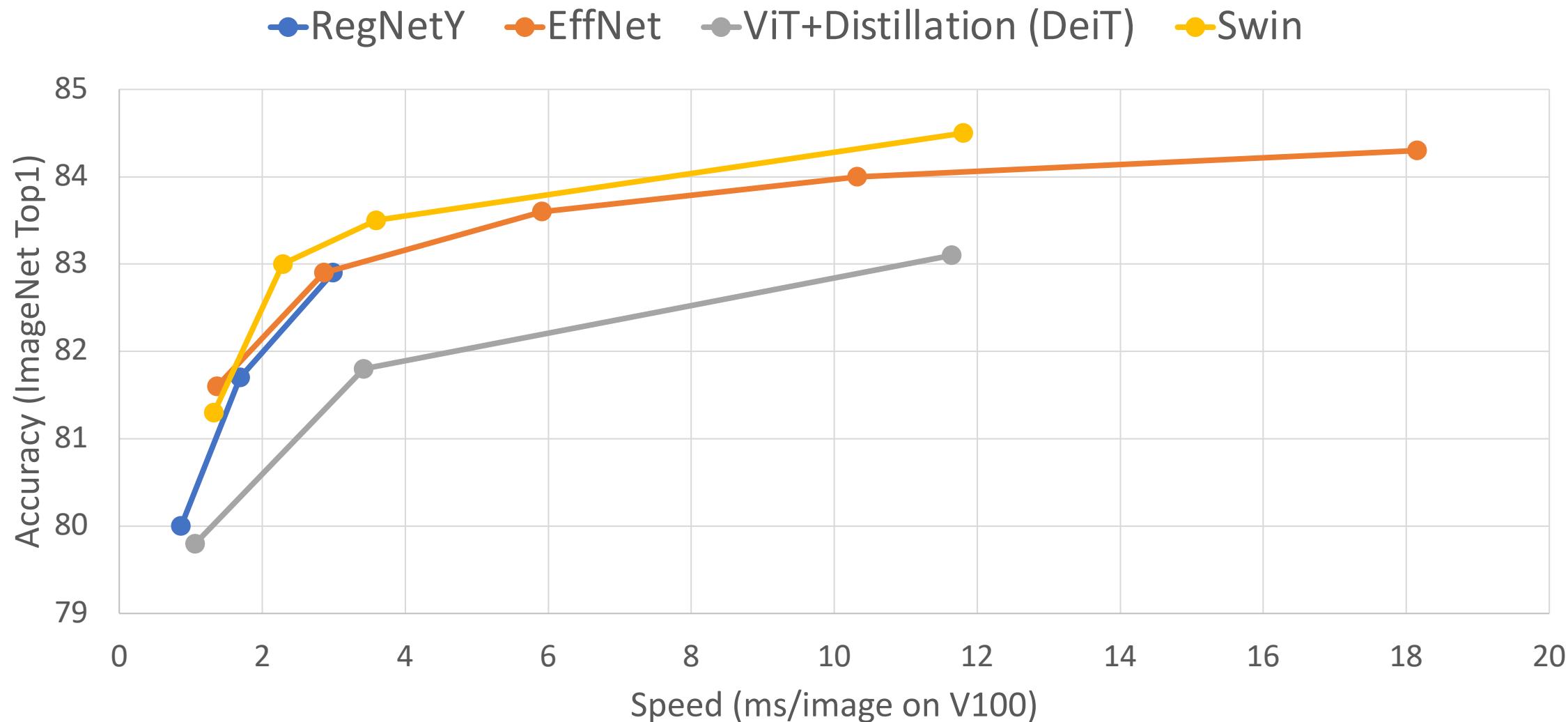
Attention with relative bias:

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

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

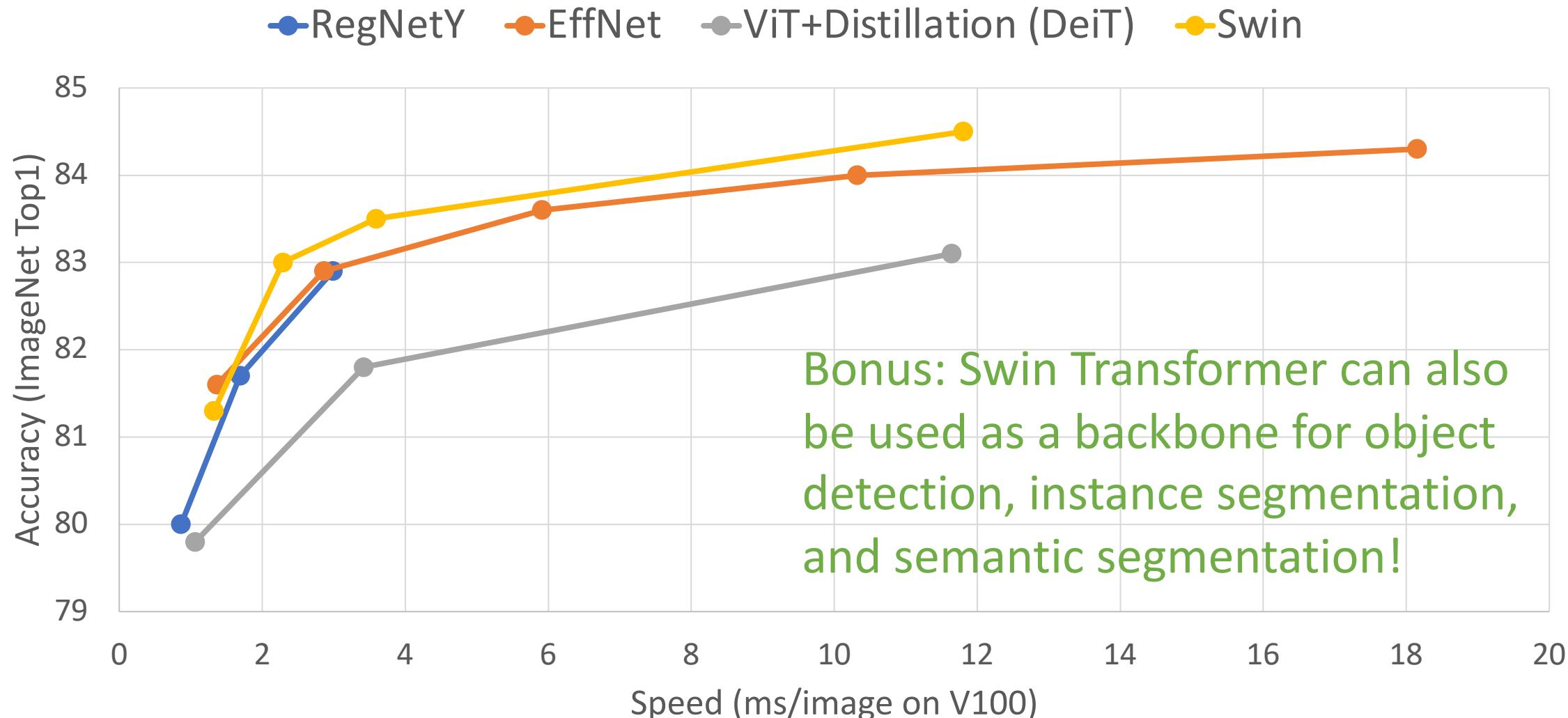
$B: M^2 \times M^2$ (learned biases)

Swin Transformer: Speed vs Accuracy



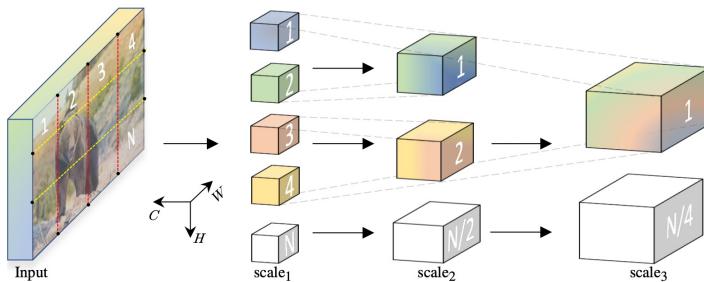
Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Swin Transformer: Speed vs Accuracy



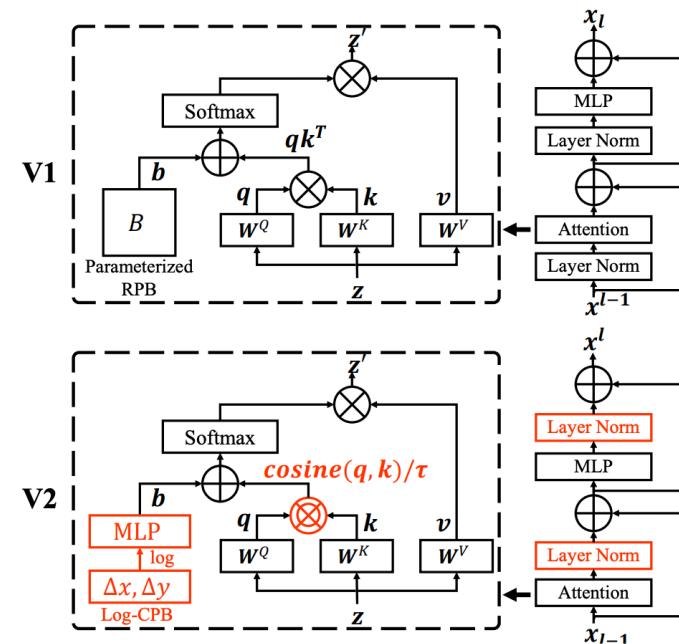
Other Hierarchical Vision Transformers

MViT



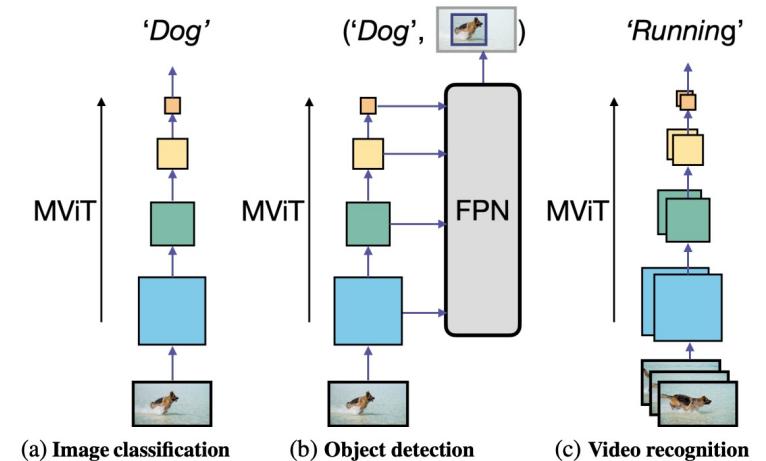
Fan et al, "Multiscale Vision
Transformers", ICCV 2021

Swin-V2



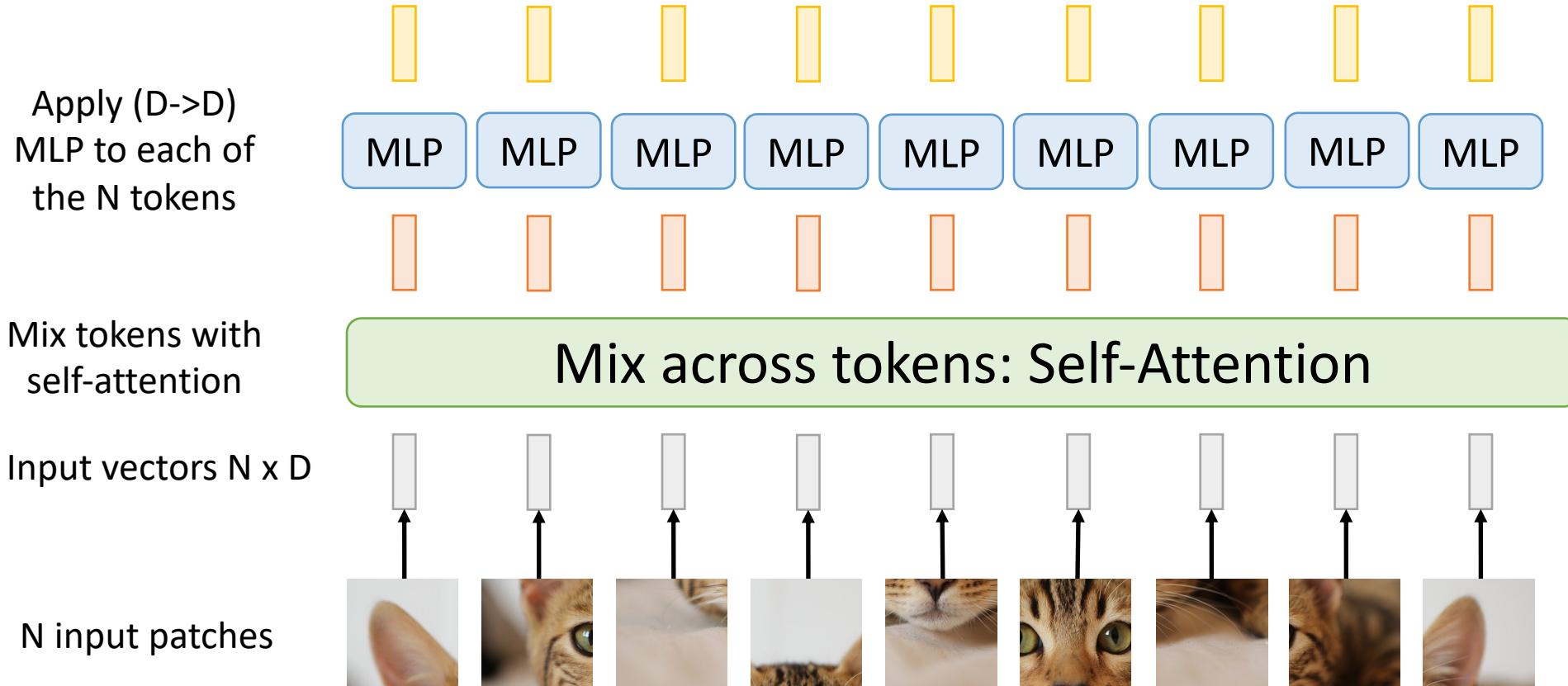
Liu et al, "Swin Transformer V2: Scaling up Capacity and Resolution", CVPR 2022

Improved MViT



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

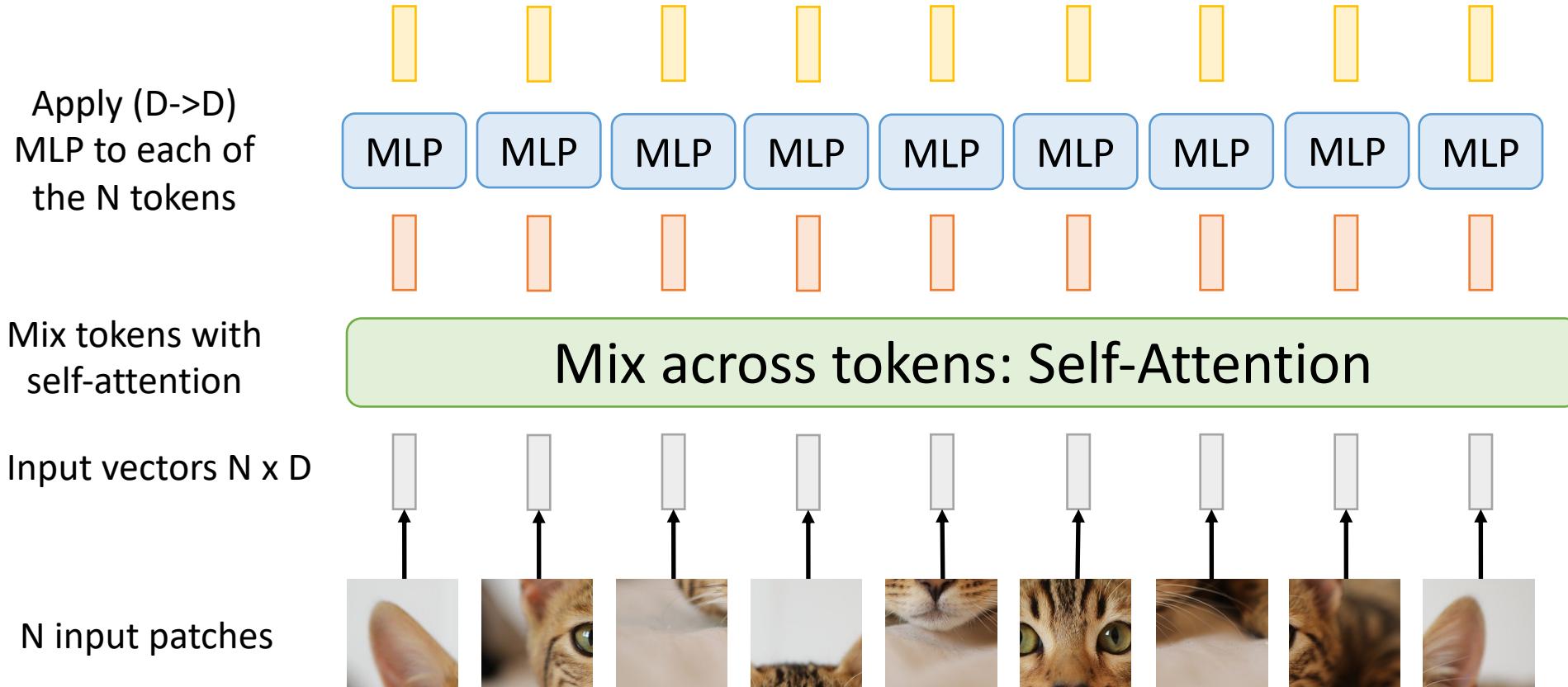
Vision Transformer: Another Look



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: Another Look

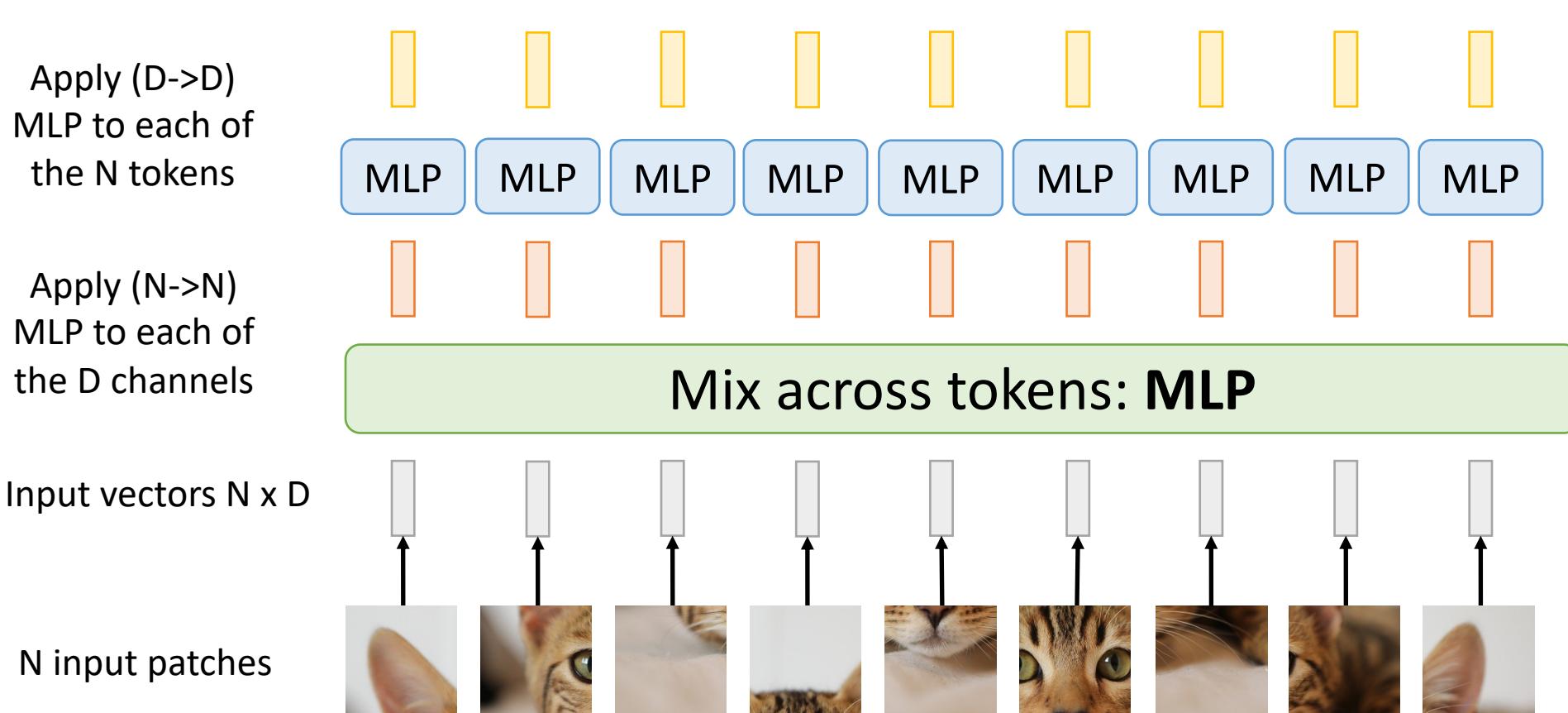


Question: Can we use something simpler than self-attention to mix across tokens?

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

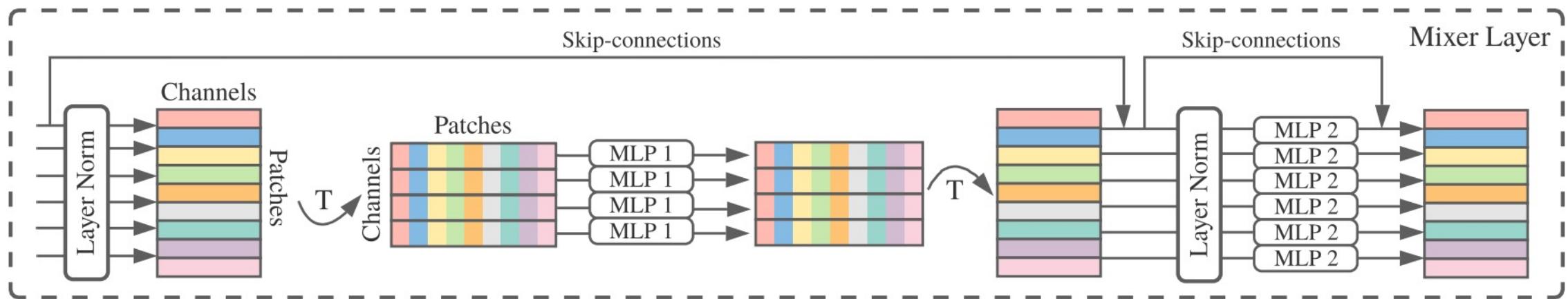
[Cat image](#) is free for commercial use under a [Pixabay license](#)

MLP-Mixer: An All-MLP Architecture



Question: Can we use something simpler than self-attention to mix across tokens?

MLP-Mixer: An All-MLP Architecture



Input: $N \times C$
 N patches with
 C channels each

$MLP 1: C \rightarrow C$,
apply to each of
the **N patches**

$MLP 2: N \rightarrow N$,
apply to each of
the **C channels**

MLP-Mixer: Many concurrent and followups

Touvron et al, “ResMLP: Feedforward Networks for Image Classification with Data-Efficient Training”, arXiv 2021,
<https://arxiv.org/abs/2105.03404>

Tolstikhin et al, “MLP-Mixer: An all-MLP architecture for vision”, NeurIPS 2021, <https://arxiv.org/abs/2105.01601>

Liu et al, “Pay Attention to MLPs”, NeurIPS 2021,
<https://arxiv.org/abs/2105.08050>

Yu et al, “S2-MLP: Spatial-Shift MLP Architecture for Vision”, WACV 2022, <https://arxiv.org/abs/2106.07477>

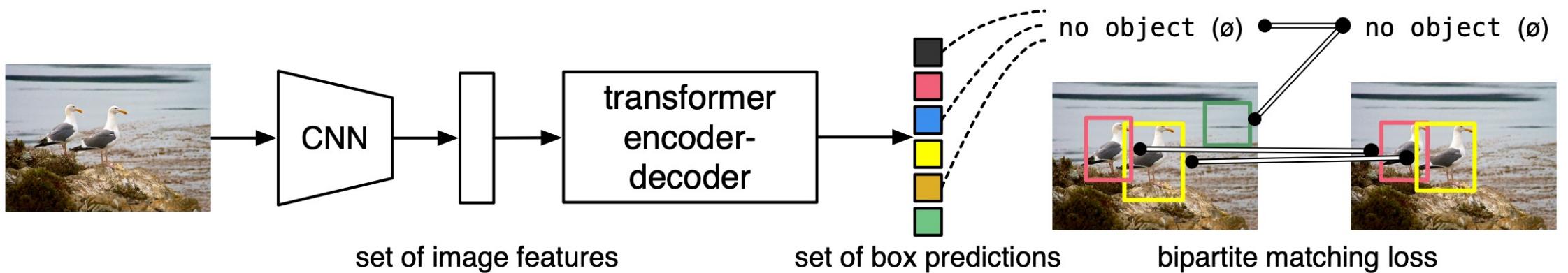
Chen et al, “CycleMLP: A MLP-like Architecture for Dense Prediction”, ICLR 2022, <https://arxiv.org/abs/2107.10224>

Object Detection with Transformers: DETR

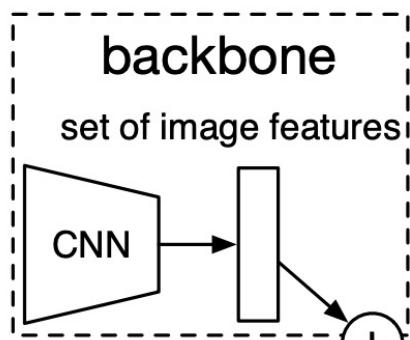
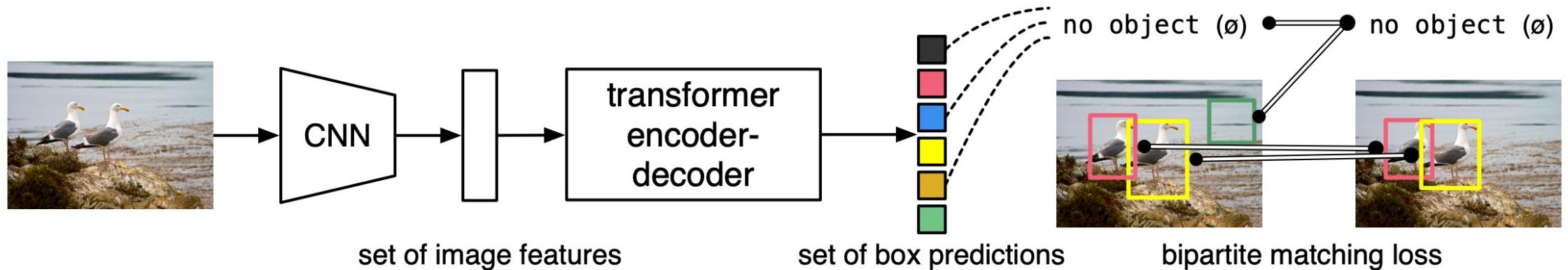
Simple object detection pipeline: directly output a set of boxes from a Transformer

No anchors, no regression of box transforms

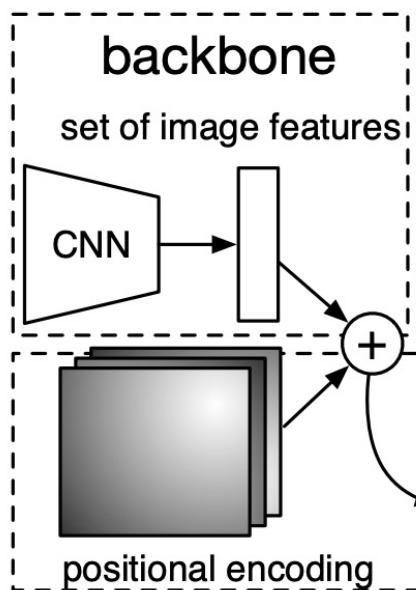
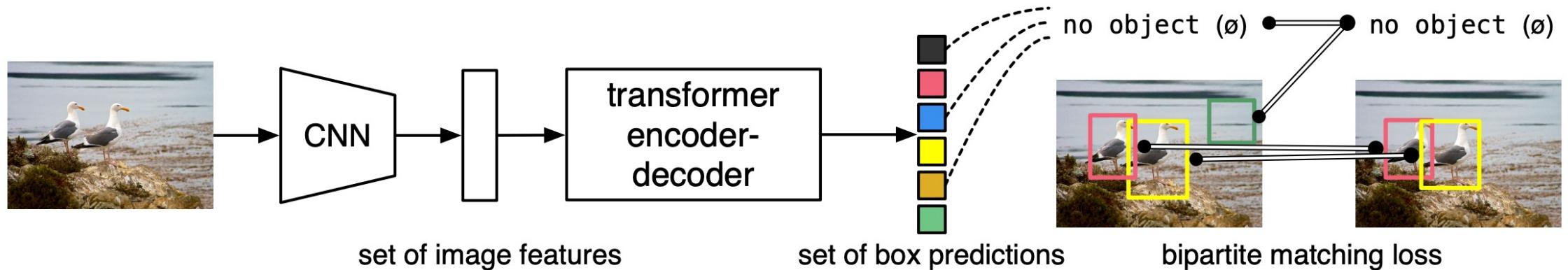
Match predicted boxes to GT boxes with bipartite matching; train to regress box coordinates



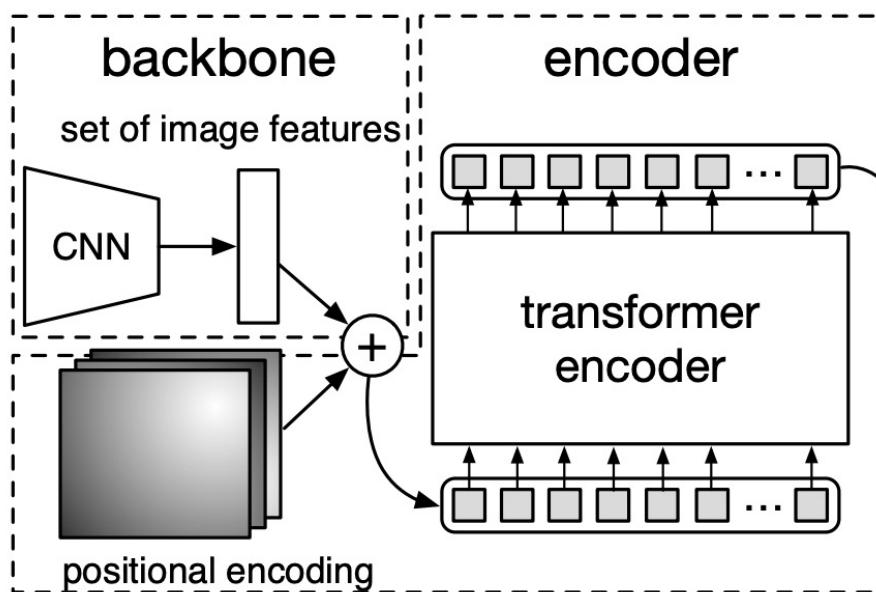
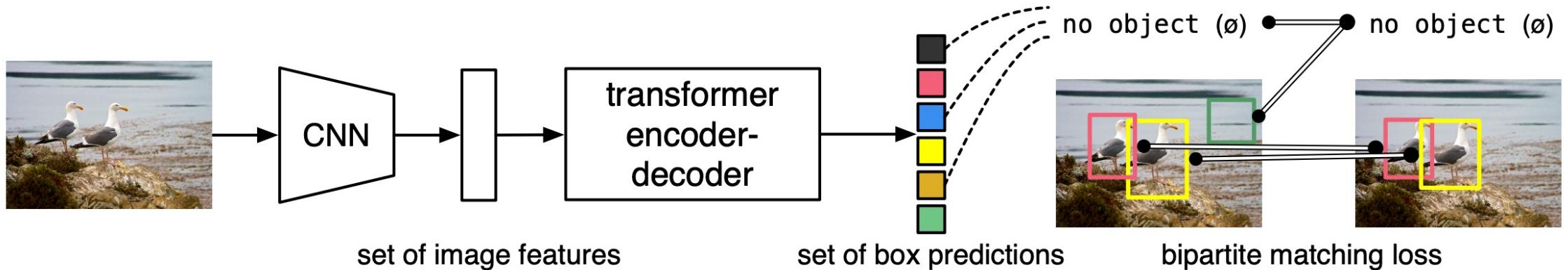
Object Detection with Transformers: DETR



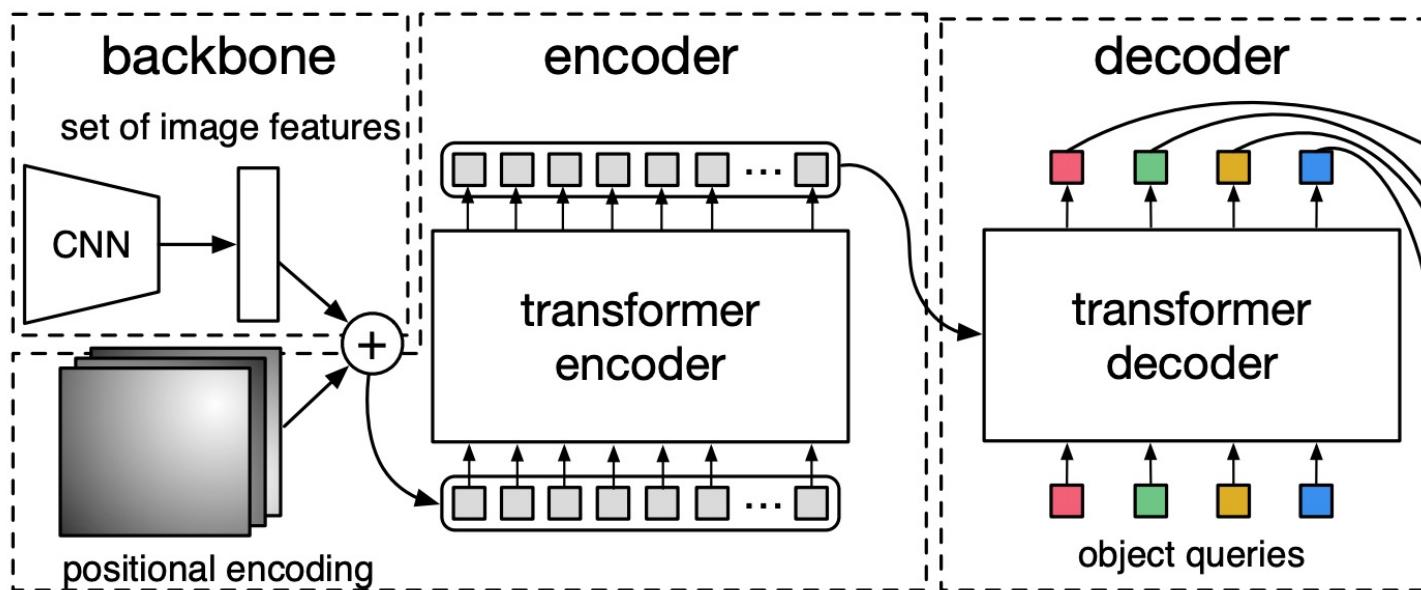
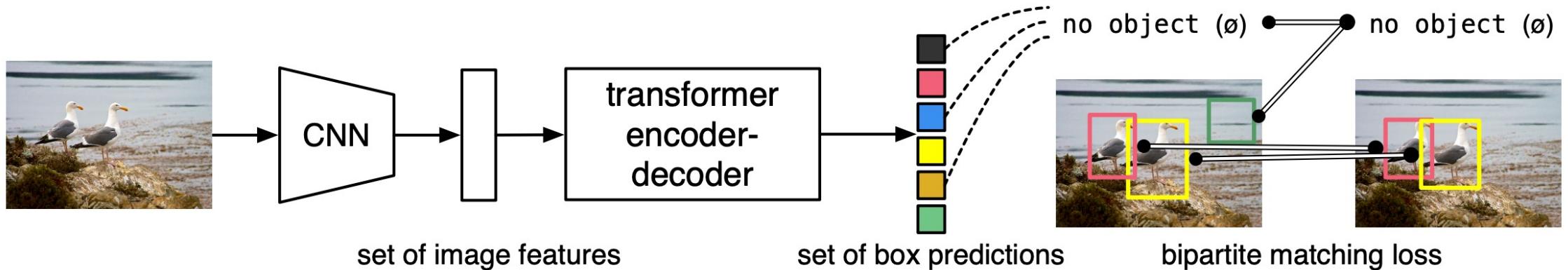
Object Detection with Transformers: DETR



Object Detection with Transformers: DETR

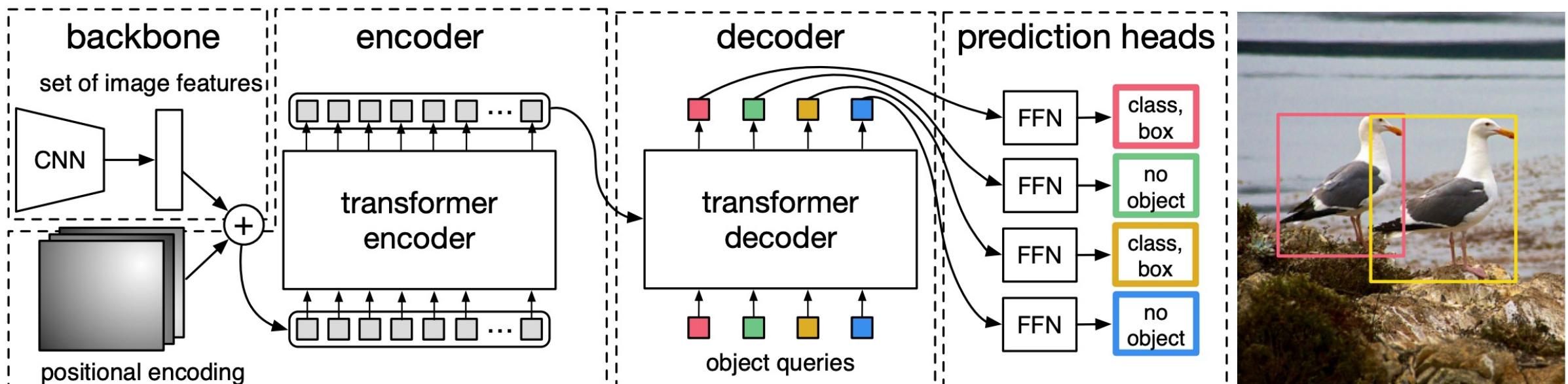
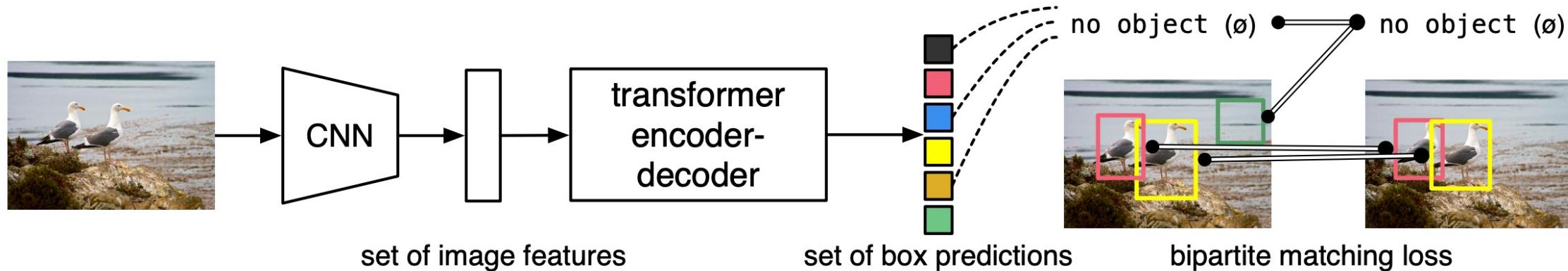


Object Detection with Transformers: DETR



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

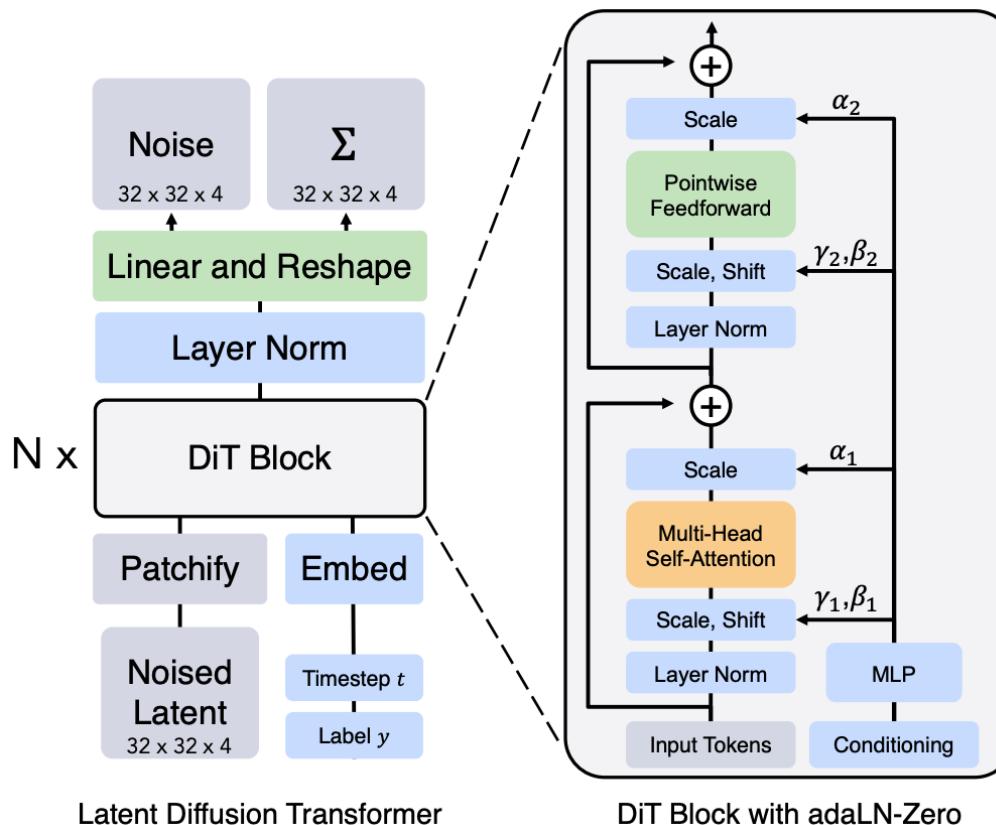
Object Detection with Transformers: DETR



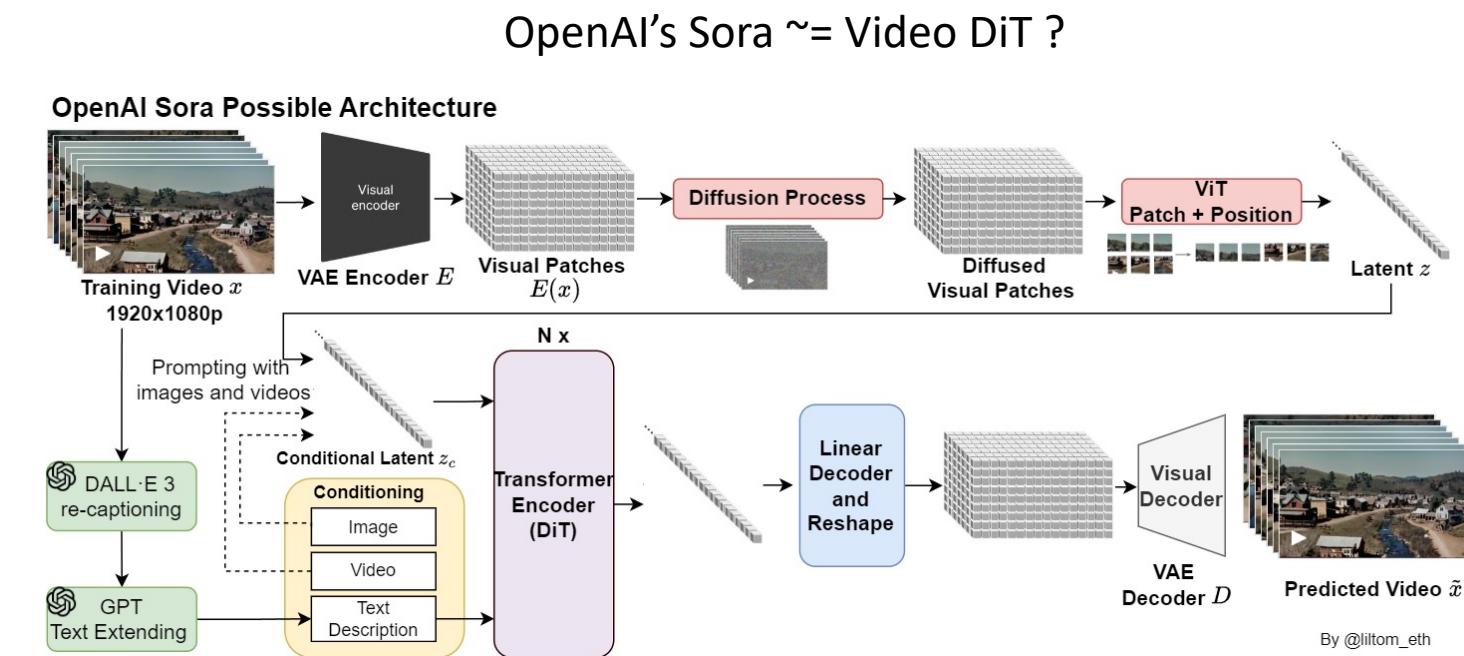
Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

Diffusion Models with Transformers (DiT): Core model behind OpenAI's Sora (<https://openai.com/sora>)

For latent diffusion models, replacing the commonly-used U-Net backbone with a transformer that operates on latent patches



<https://github.com/facebookresearch/DiT>



Lecture 12 - 130

Summary

Vision Transformers have been a super hot topic the past 2 years!

Very different architecture vs traditional CNNs, with more emphasis on the scalability

Applications to all tasks: classification, detection, segmentation, etc

Main benefit is probably speed: Matrix multiply is more hardware-friendly than convolution, so ViTs with same FLOPs as CNNs can train and run much faster

Reading and Practice

D2L textbook:

Chapter 11: Attention Mechanism and Transformers

Next week: Object detection and semantic segmentation