Transformers and ViT

CSE 849 Deep Learning
Spring 2025

Zijun Cui

Partial slides are adapted from CMU 11-785 Introduction to Deep Learning by Bhiksha Raj and partial slides are revised based on 598 University of Michigan by Justin Johnson

Project 3

- It requires considerable time from your end.
- **Plan ahead and Start early.**

Class Schedule Update

- We will begin *Generative Modeling* immediately after the *Transformer* lectures. Topics on *Graph Neural Networks* will be moved to the end of the course. This change ensures that you learn about *Diffusion Models* in time for Project 4.

3/24	17 Transformer and ViT We are here	18 Transformer Case Study: LLMs
3/31	19 Probabilistic Deep Learning and Generative modeling	20 Generative Modeling
4/7	21 Diffusion Models 1	22 Diffusion Models 2 **Project 4 Out; Project 3 Due**
4/14	23 Diffusion Models Applications and Bayesian Deep Learning	24 Bayesian Deep Learning and Deep PGM
4/21	25 Graph Neural Network	No Class
4/28	Final Exam Week	Final Exam Week **Project 4 Due on Wednesday, April 30th.**

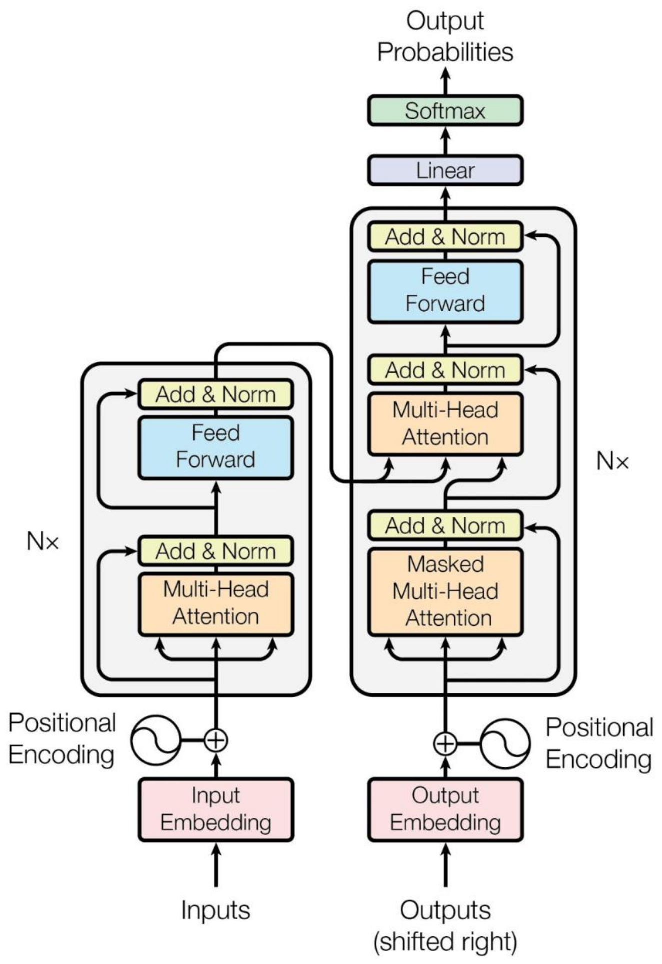
I plan to have **no class on Thursday, April 24**, to give you more time to focus on Project 4

- I have updated syllabus in Piazza accordingly.

Now let’s continue from the last lecture and finish Transformer Architecture

Transformers

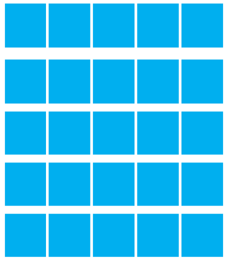
- ✓ Tokenization
- ✓ Input Embeddings
- ✓ Position Encodings
- ✓ Query, Key, & Value
- ✓ Attention
- ✓ Self Attention
- ✓ Multi-Head Attention
- ✓ Feed Forward
- ✓ Add & Norm
- ✓ Encoders
- ✓ Masked Attention
- Encoder Decoder Attention
- Linear
- Softmax
- Decoders
- Encoder-Decoder Models



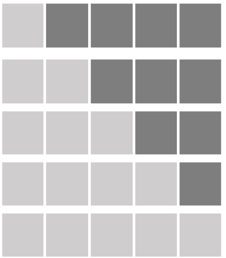
Encoder Decoder Attention

Encoder Decoder Attention ?

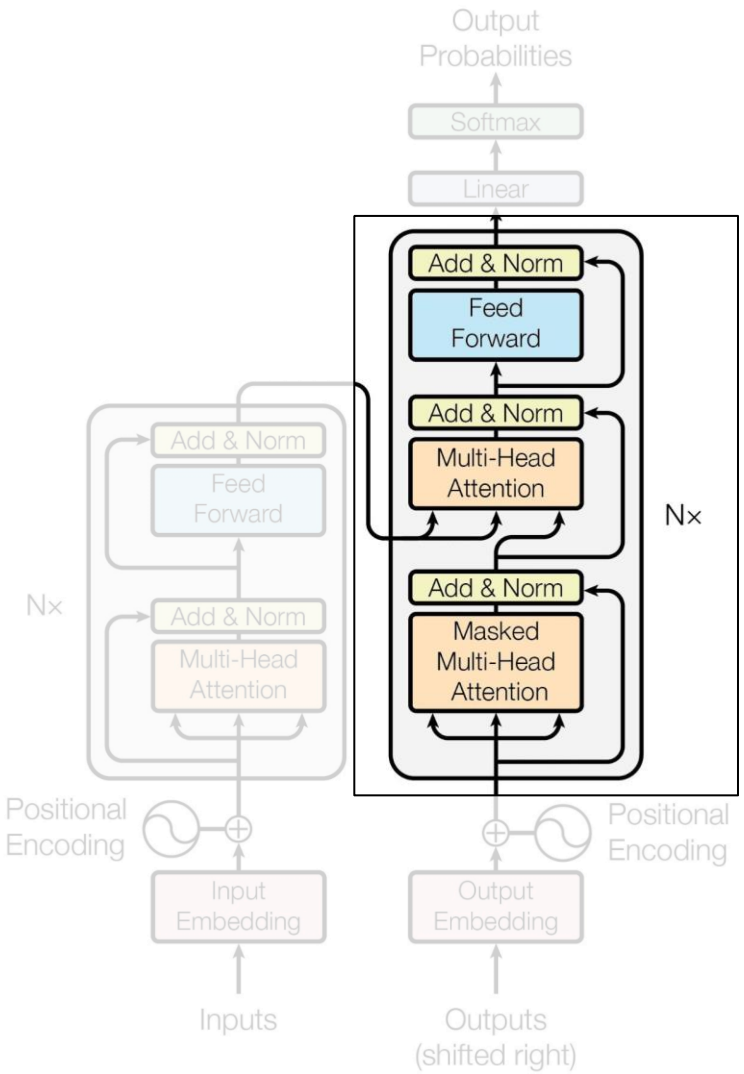
Add & Norm



Input



Norm(Z’)



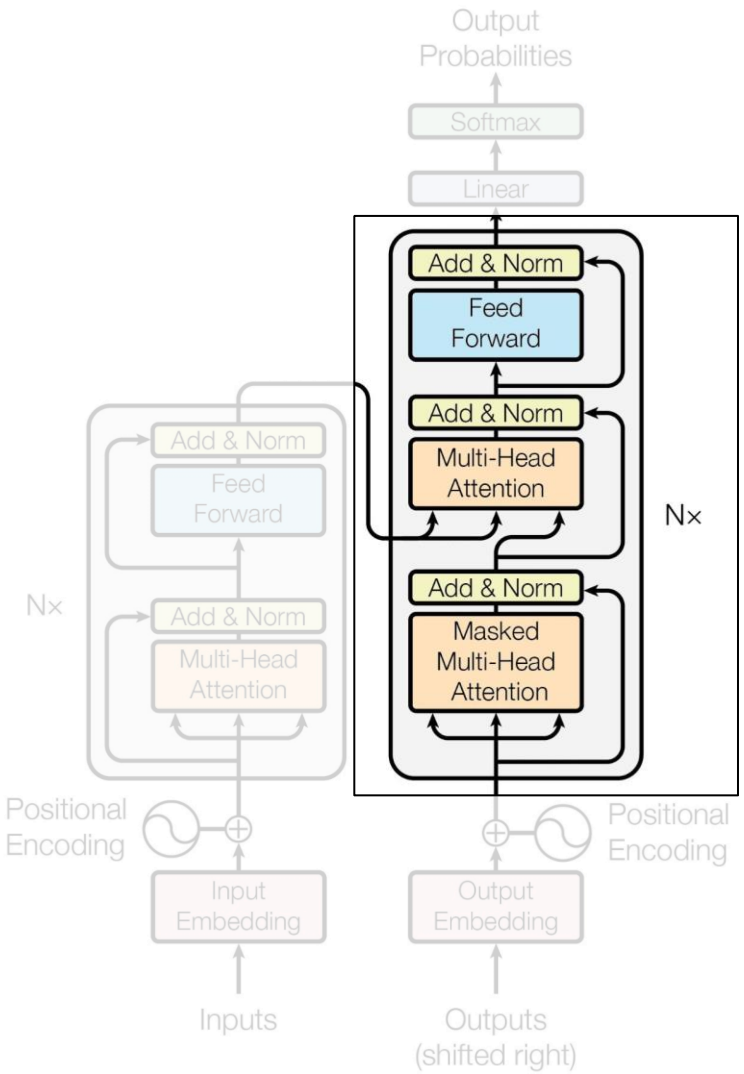
Encoder Decoder Attention

Encoder

Keys from **Encoder Outputs**
Values from **Encoder Outputs**

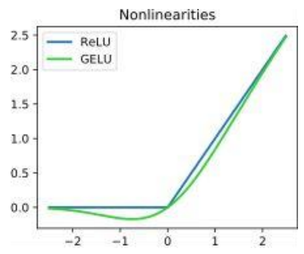
Decoder

Queries from **Decoder Inputs**

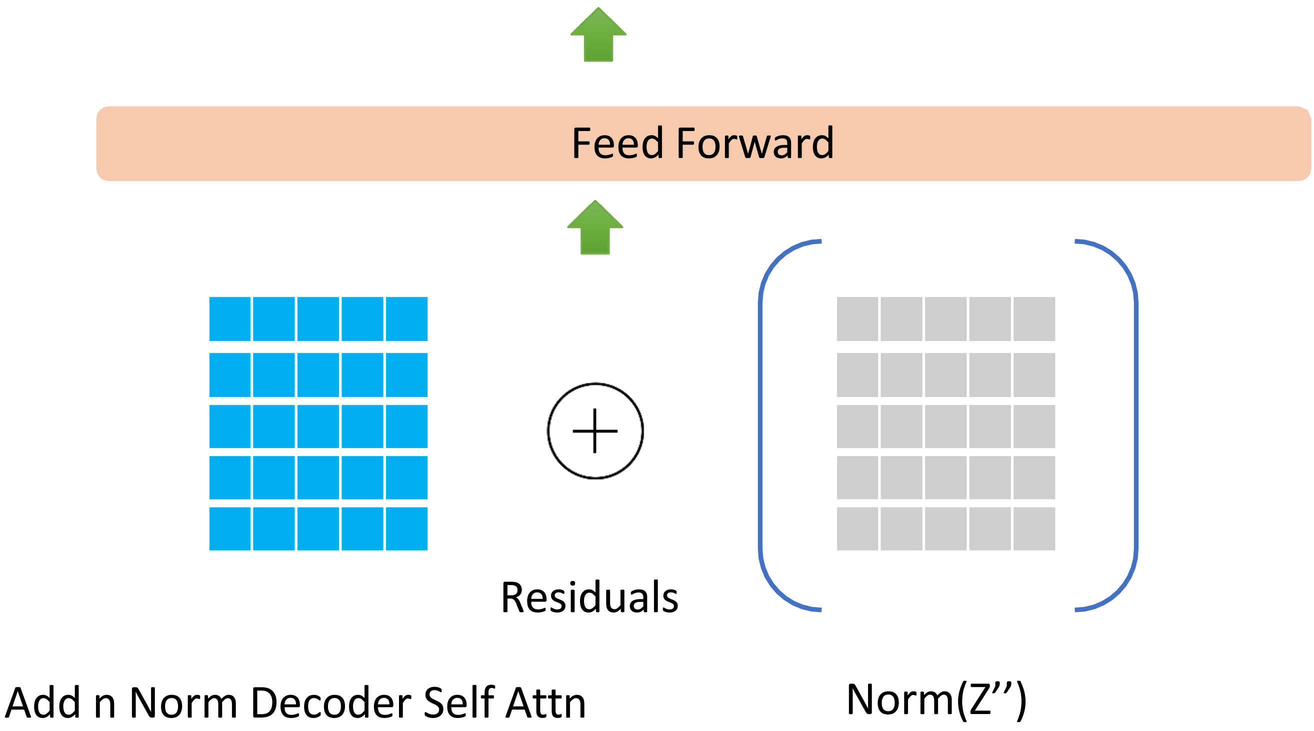


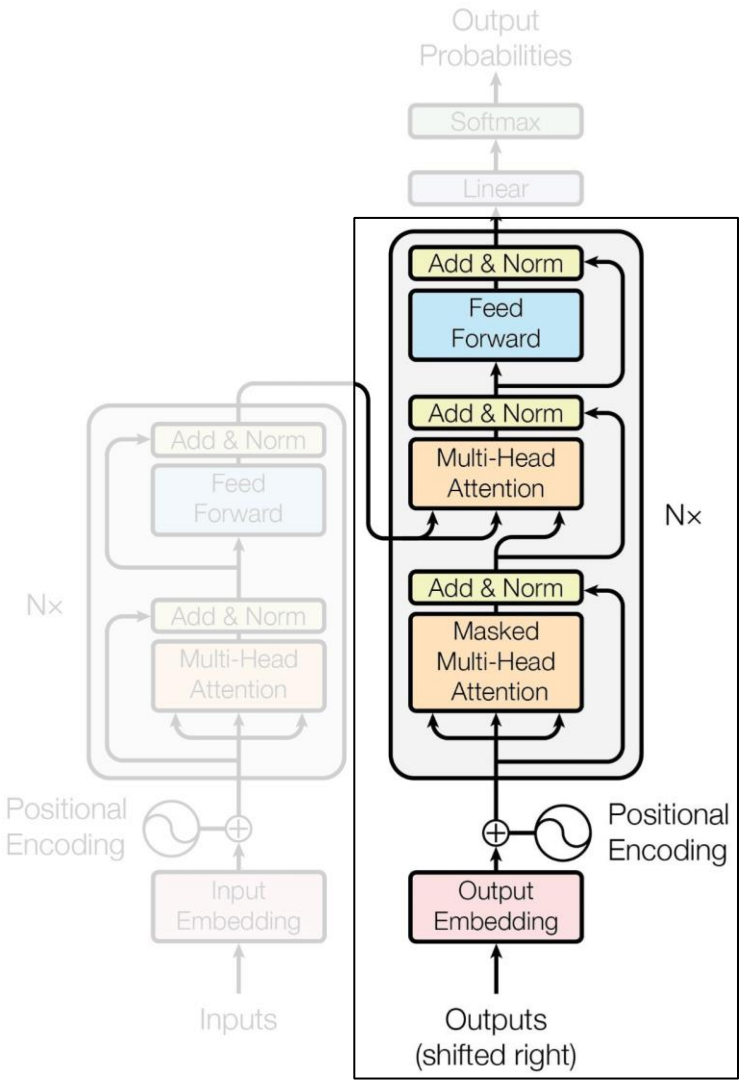
NOTE: Every decoder block receives the same FINAL encoder output

Encoder Decoder Attention

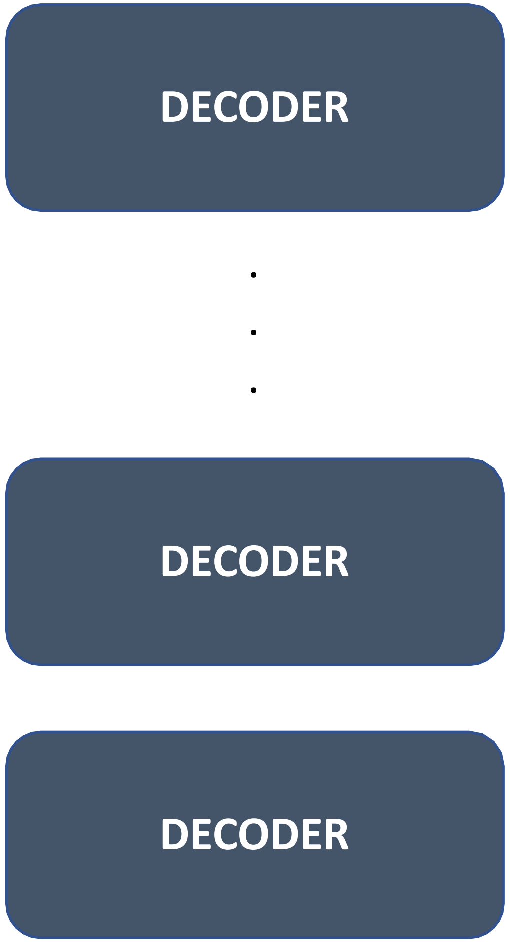


- Non Linearity
- Complex Relationships
- Learn from each other



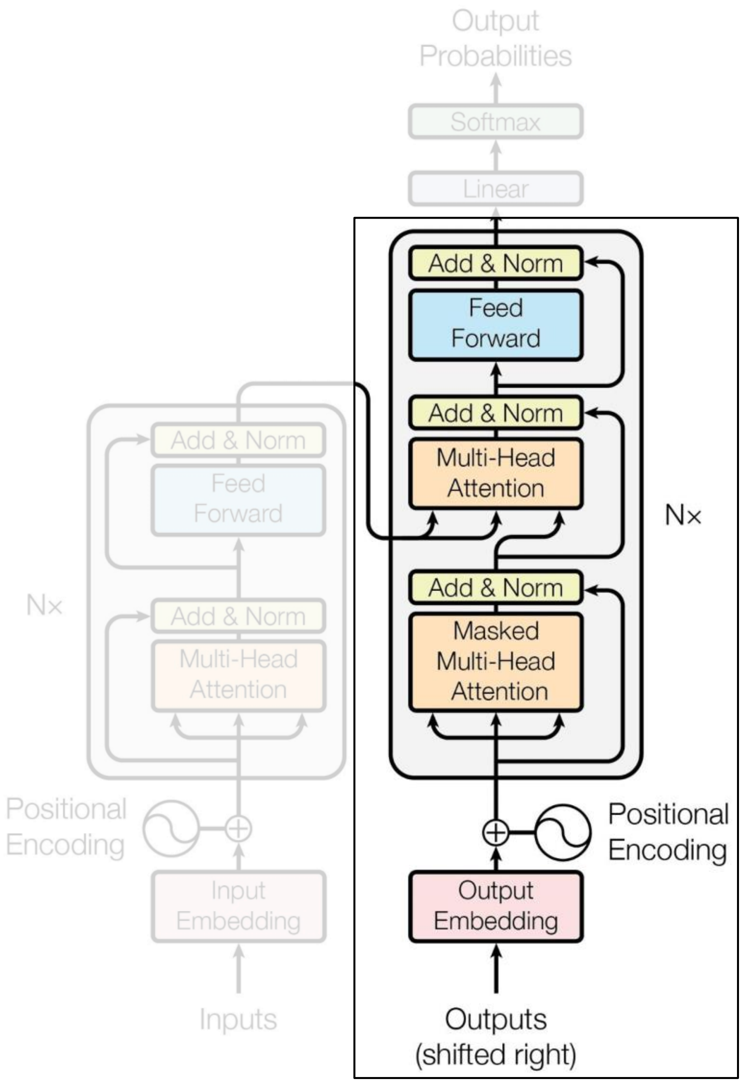


Decoder

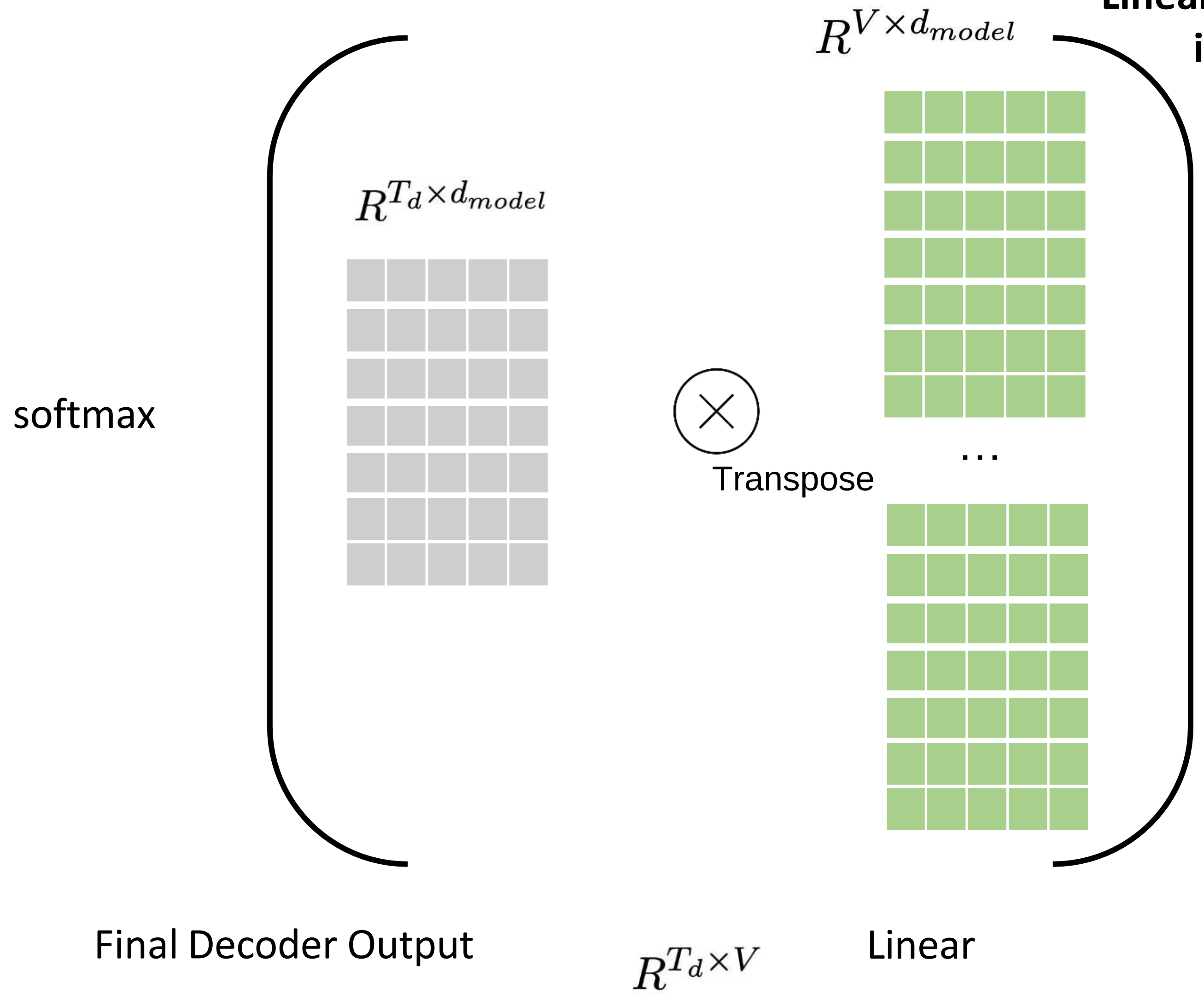


$R^{T_d \times d_{model}}$

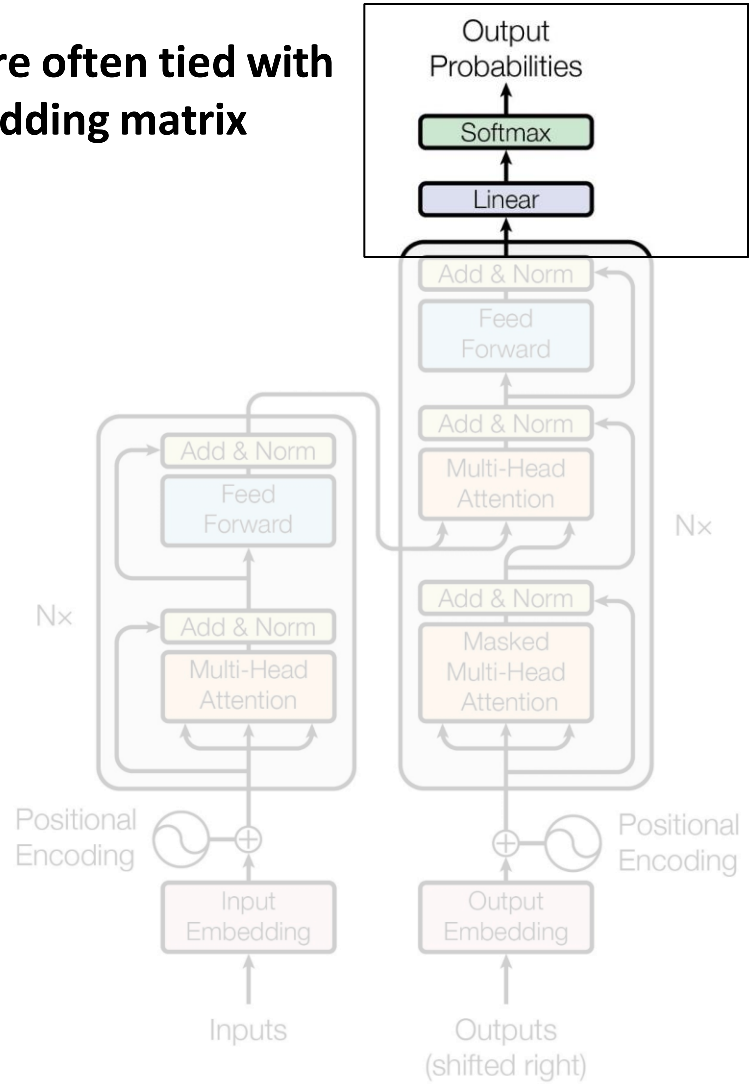
Decoder output



Linear

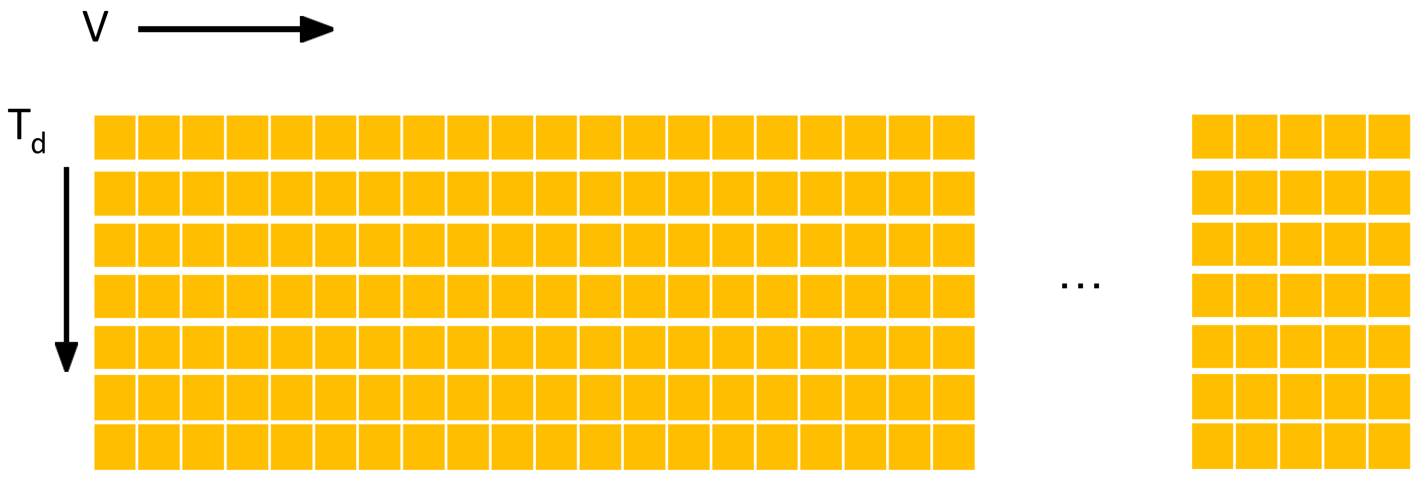


Linear weights are often tied with input embedding matrix

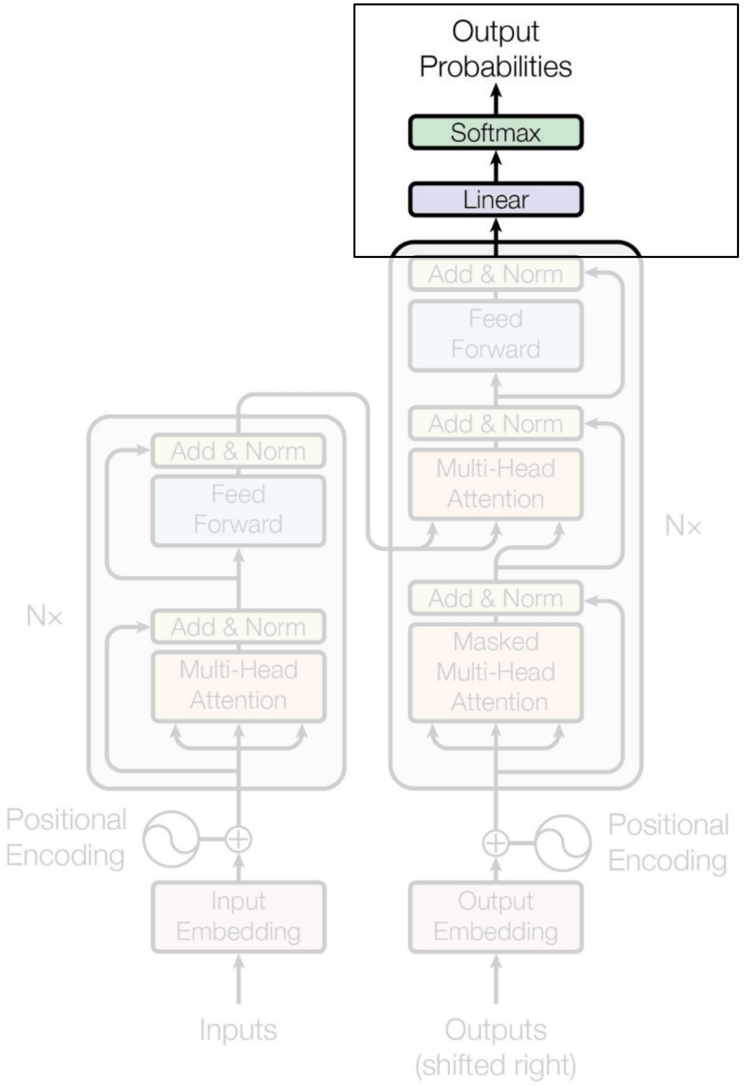


Softmax

Output Probabilities

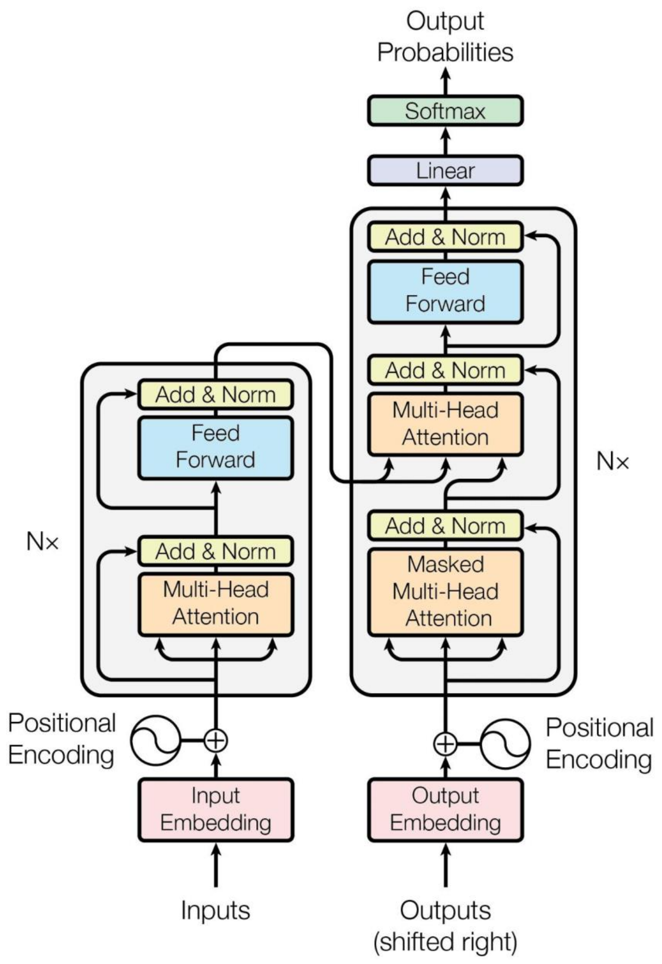


$$R^{T_d \times V}$$

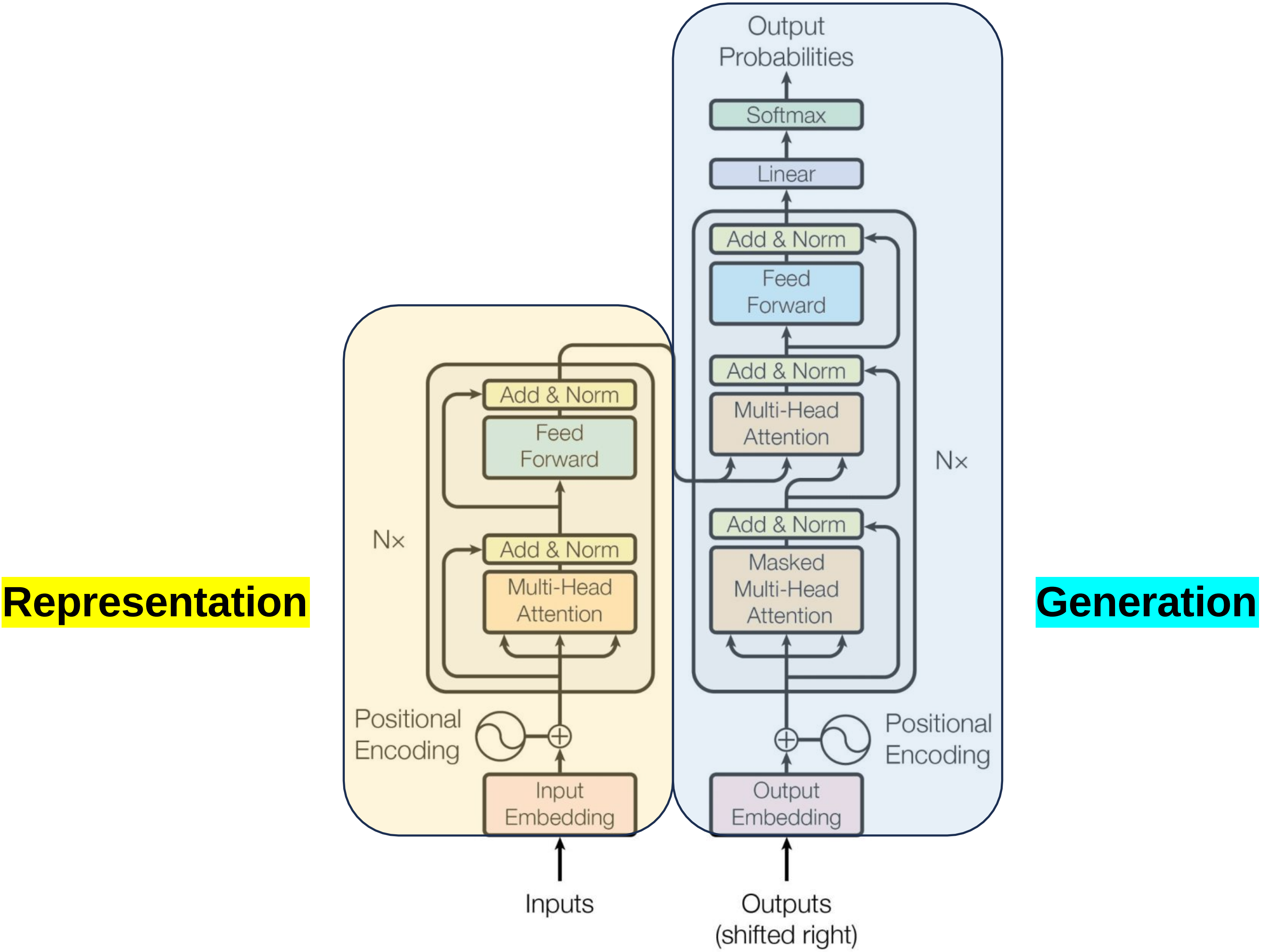


Transformers

- ✓ Tokenization
- ✓ Input Embeddings
- ✓ Position Encodings
- ✓ Query, Key, & Value
- ✓ Attention
- ✓ Self Attention
- ✓ Multi-Head Attention
- ✓ Feed Forward
- ✓ Add & Norm
- ✓ Encoders
- ✓ Masked Attention
- ✓ Encoder Decoder Attention
- ✓ Linear
- ✓ Softmax
- ✓ Decoders
- Encoder-Decoder Models



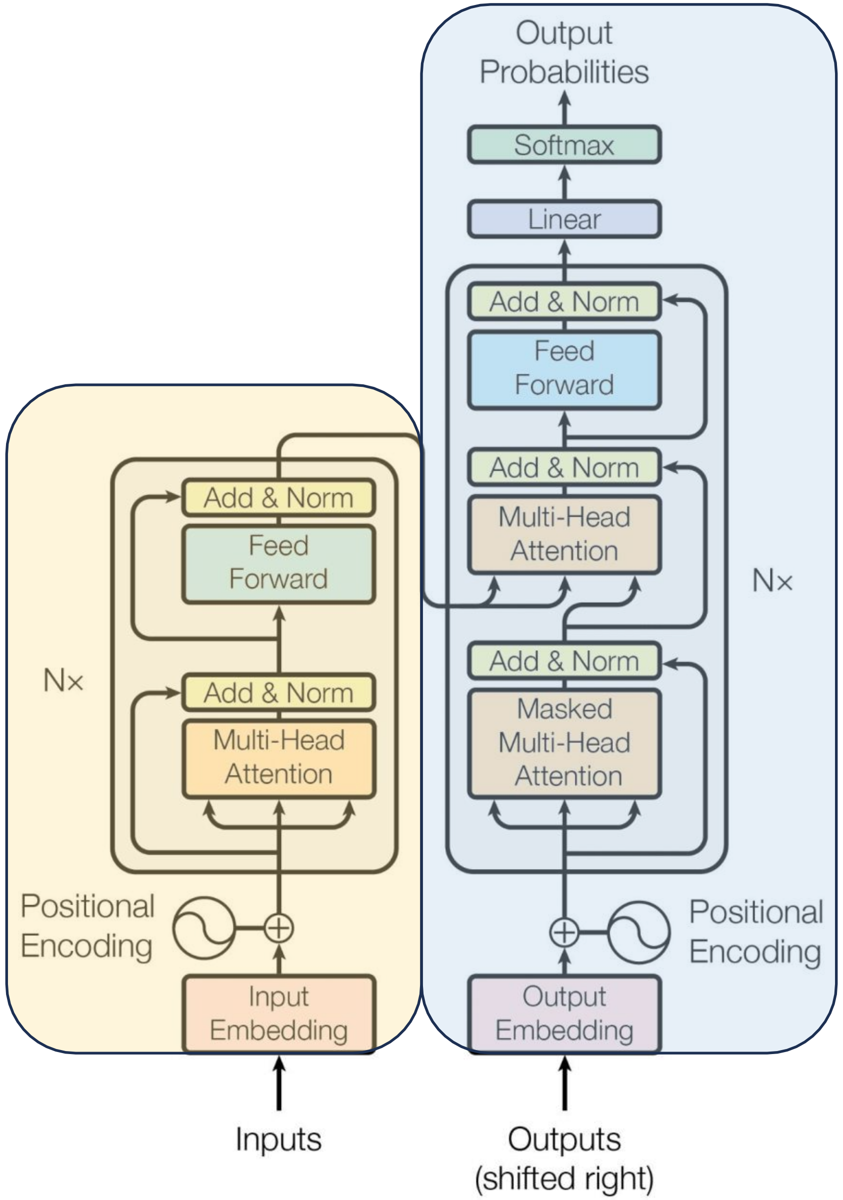
Transformers



Transformers

Input – input tokens
Output – hidden states

Representation



Input – output tokens and hidden states
Output – output tokens

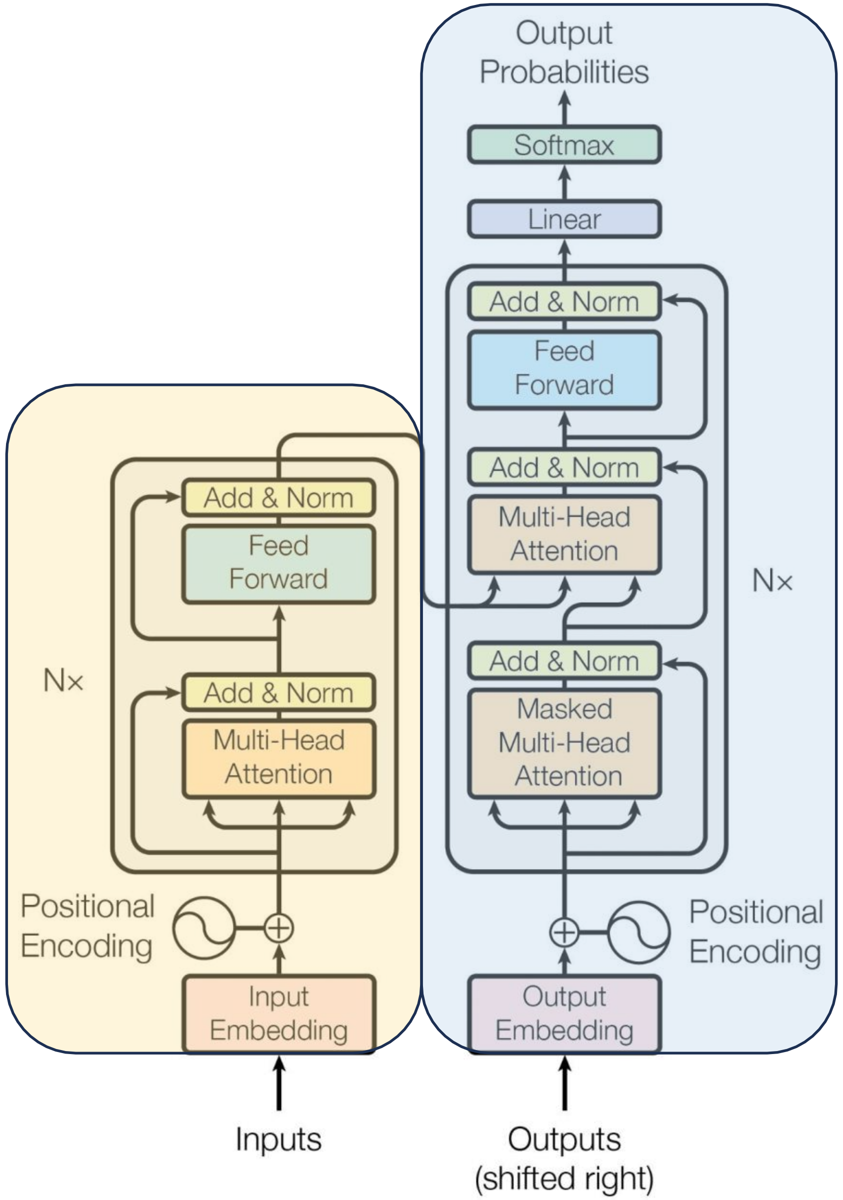
Generation

Transformers

Input – input tokens
Output – hidden states

Model can see all timesteps

Representation



Input – output tokens and hidden states
Output – output tokens

Model can only see previous timesteps

Generation

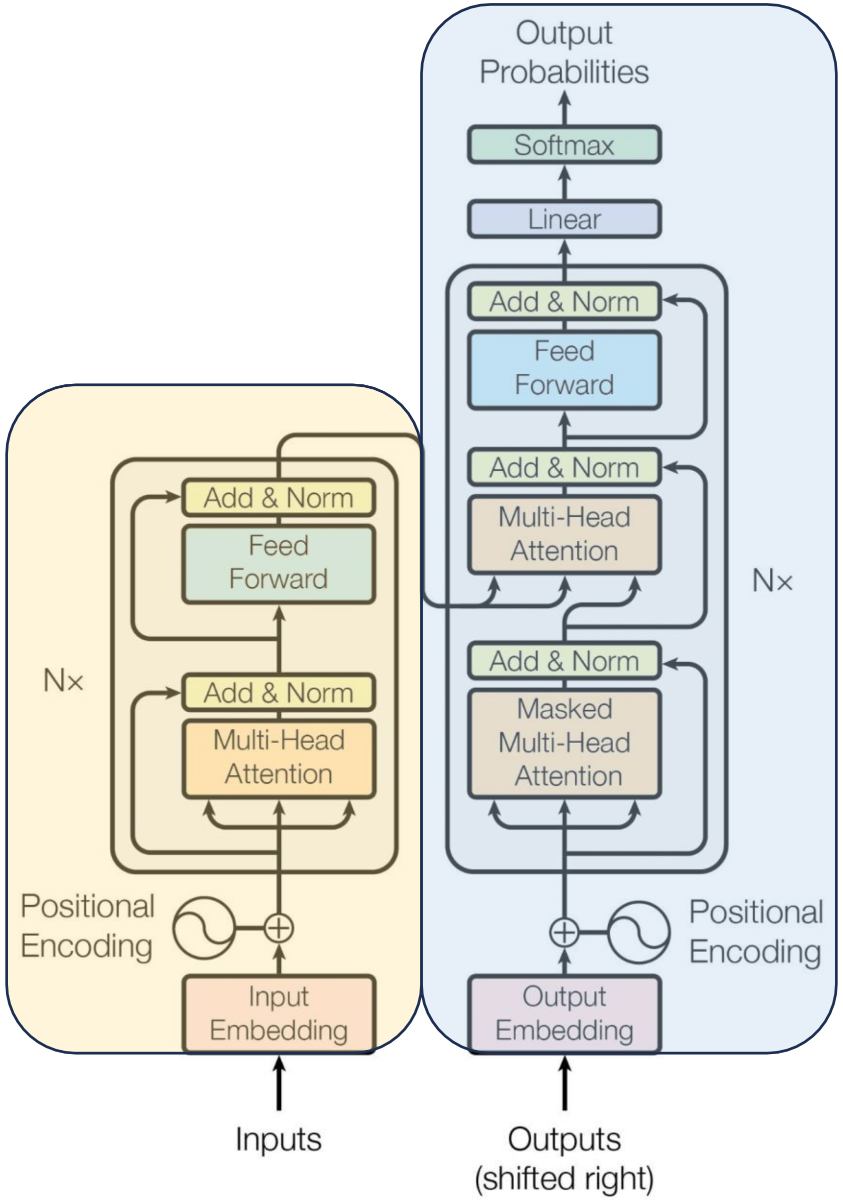
Transformers

Input – input tokens
Output – hidden states

Model can see all timesteps

Does not usually output tokens, so no inherent auto-regressivity

Representation



Input – output tokens and hidden states
Output – output tokens

Model can only see previous timesteps

Model is auto-regressive with previous timesteps' outputs

Generation

Transformers

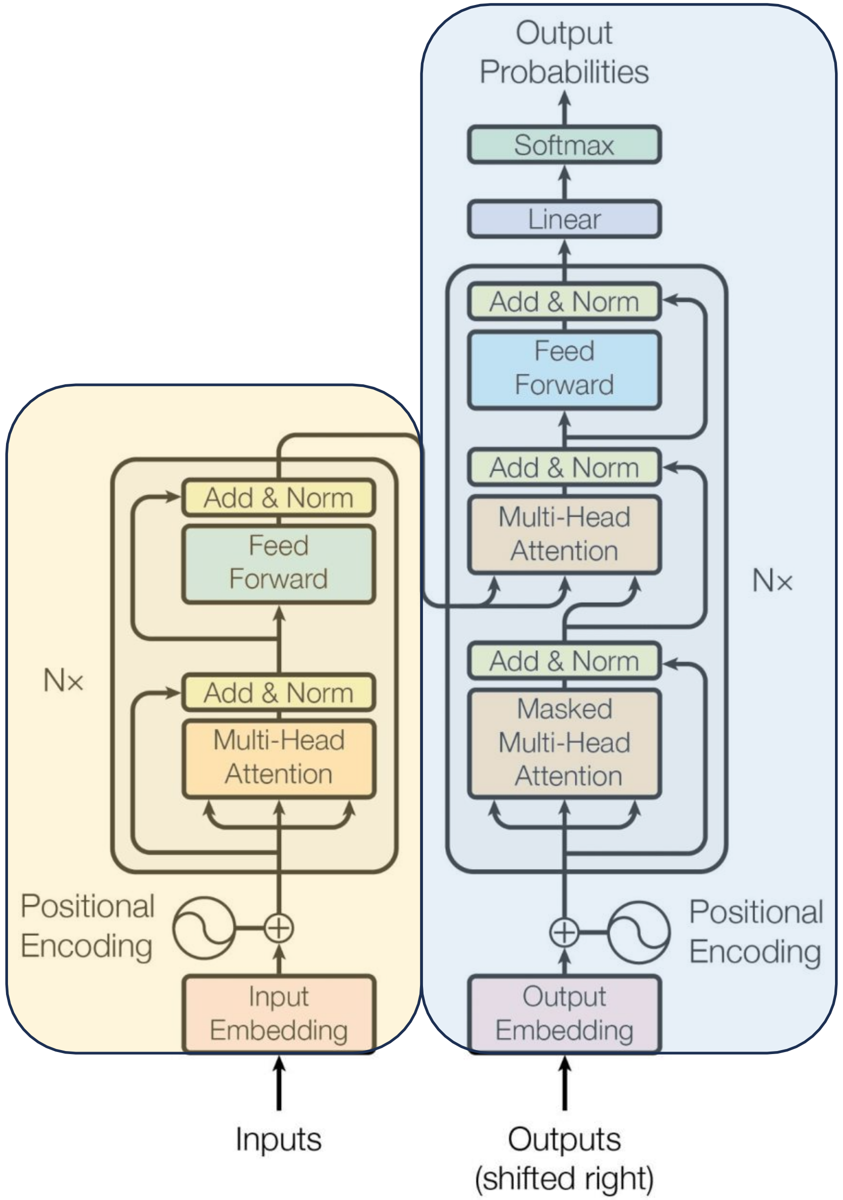
Input – input tokens
Output – hidden states

Model can see all timesteps

Does not usually output tokens, so no inherent auto-regressivity

Can also be adapted to generate tokens *by appending a module that maps hidden state dimensionality to vocab size*

Representation



Input – output tokens and hidden states
Output – output tokens

Model can only see previous timesteps

Model is auto-regressive with previous timesteps' outputs

Can also be adapted to generate hidden states *by looking before token outputs*

Generation

Transformers

- ✓ Tokenization
- ✓ Input Embeddings
- ✓ Position Encodings
- ✓ Query, Key, & Value
- ✓ Attention
- ✓ Self Attention
- ✓ Multi-Head Attention
- ✓ Feed Forward
- ✓ Add & Norm
- ✓ Encoders
- ✓ Masked Attention
- ✓ Encoder Decoder Attention
- ✓ Linear
- ✓ Softmax
- ✓ Decoders
- ✓ Encoder-Decoder Models

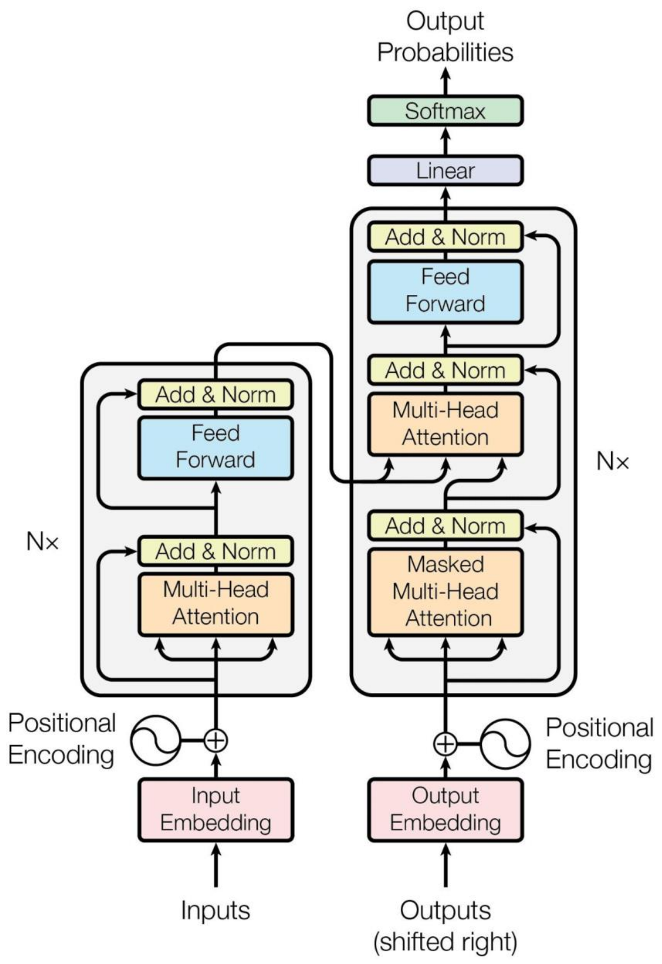


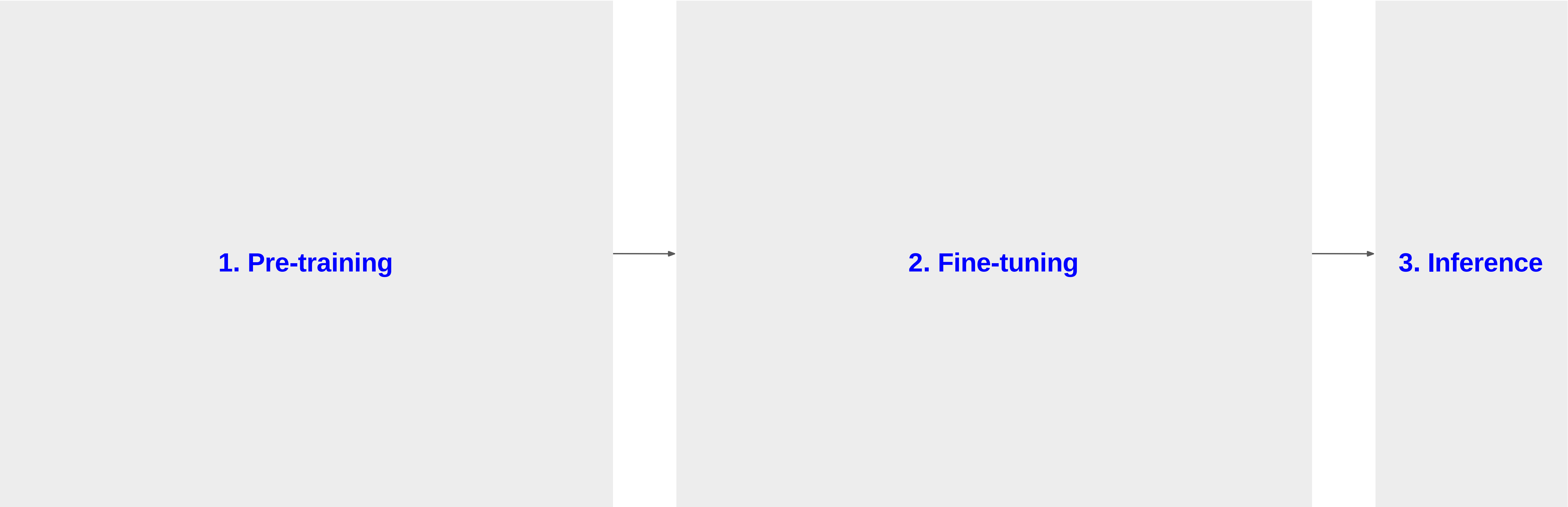
Table of contents

How to train and fine-tune transformers

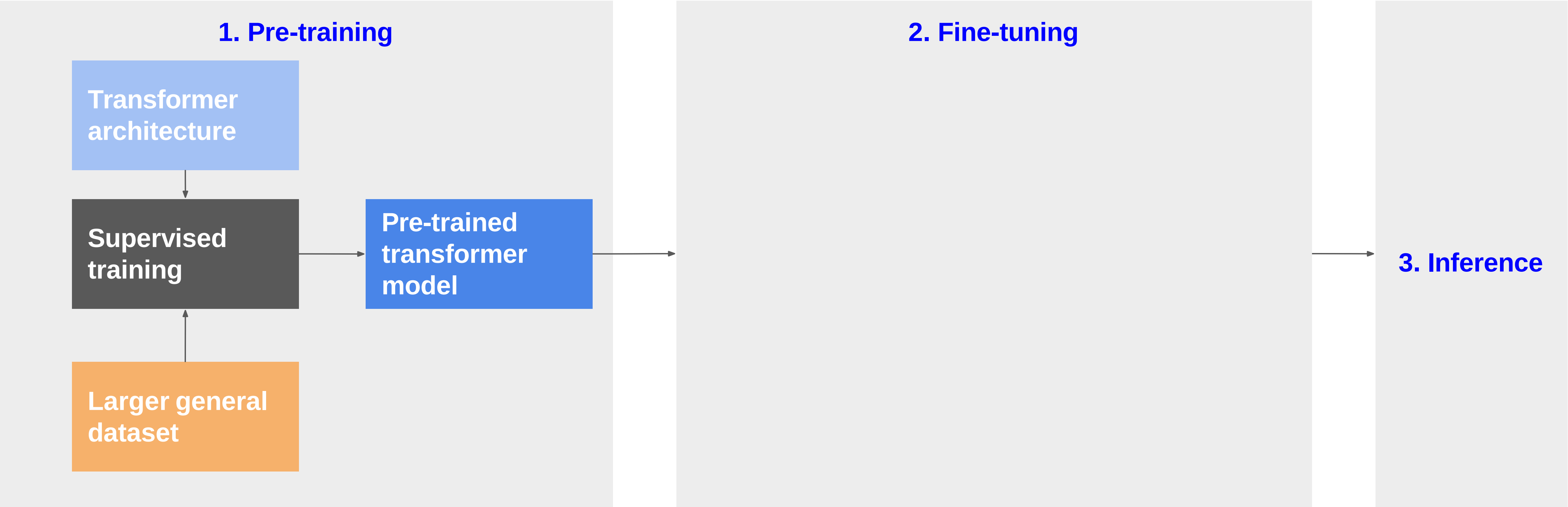
1. Training

2. Inference

How to train and fine-tune transformers



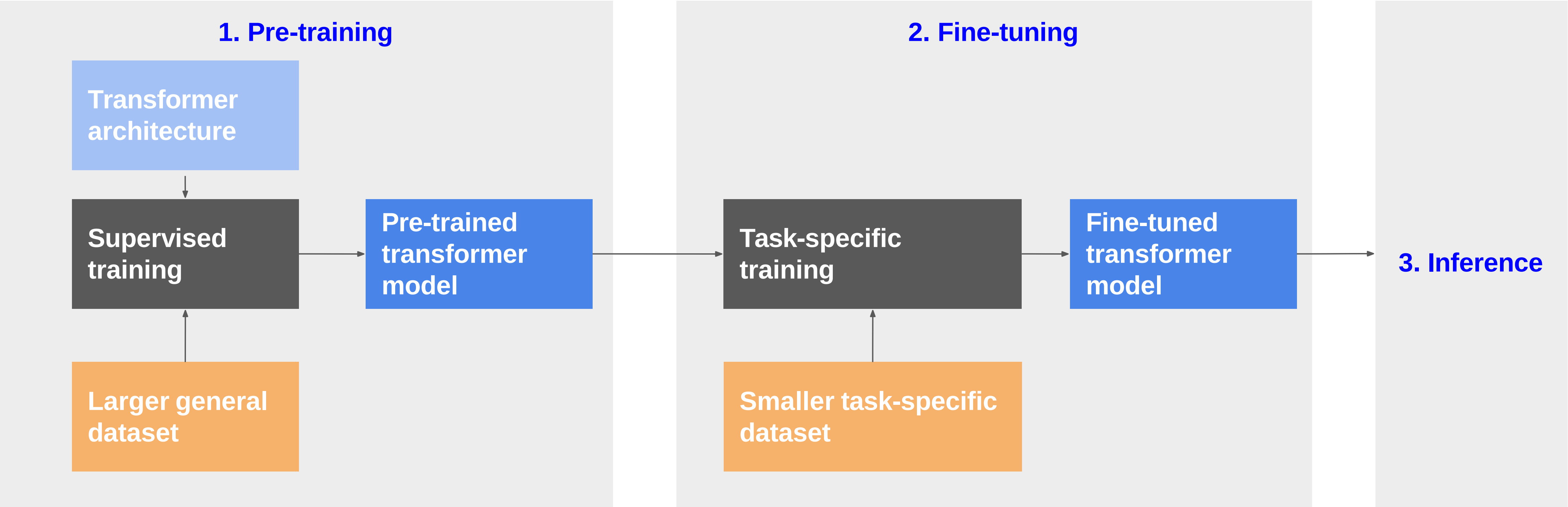
How to train and fine-tune transformers



Lots of data, learn general things. May serve as a parameter initialization.

Usually requires significant computational resources and time.

How to train and fine-tune transformers



Lot’s of data, learn general things. May serve as a parameter initialization.

Usually requires significant computational resources and time.

Adaptation to the specific task.

Potentially less computationally intensive.

Parameter-Efficient Fine-Tuning Techniques

LoRA (Lower-Rank Adaptation)

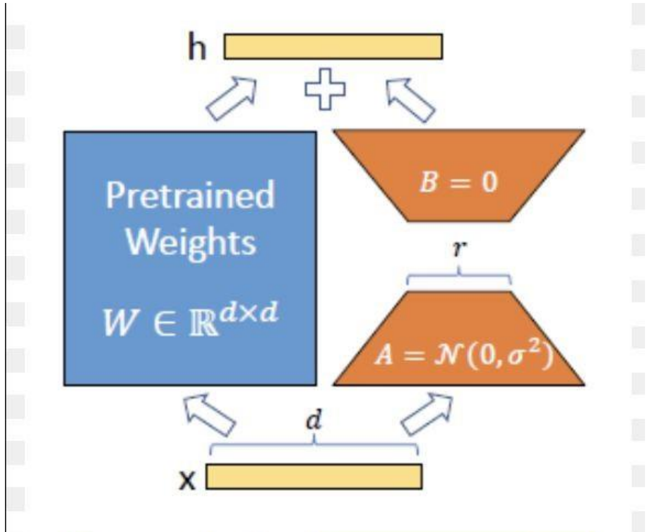
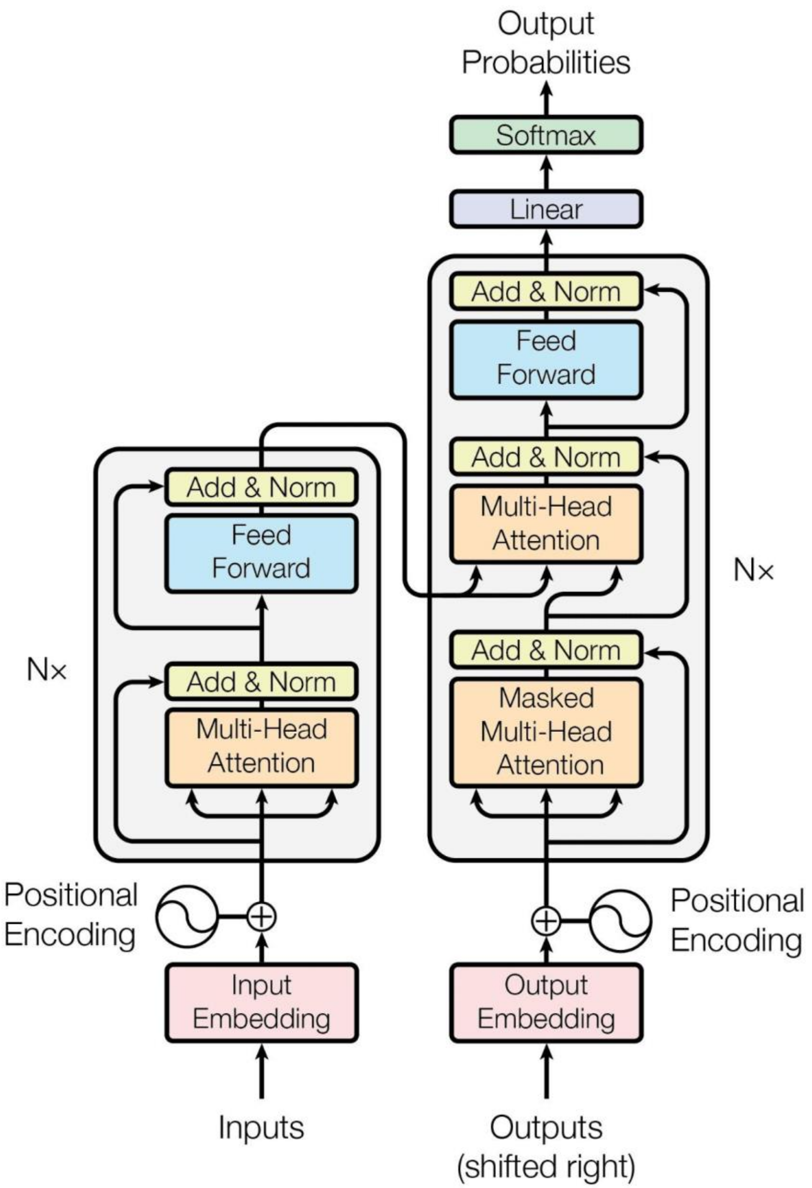


Figure 1: Our reparametrization. We only train A and B.



LoRA: https://arxiv.org/abs/2106.09685
BitFit: https://arxiv.org/abs/2106.10199

Parameter-Efficient Fine-Tuning Techniques

Can be applied to:
1. Attention layers (Q/K/V projections)
2. Feed-forward layers (as shown)
3. Anywhere with large linear projections

LoRA (Lower-Rank Adaptation)

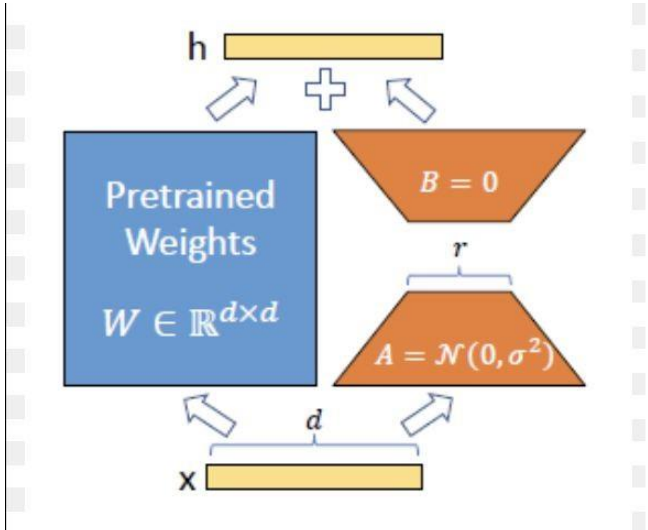
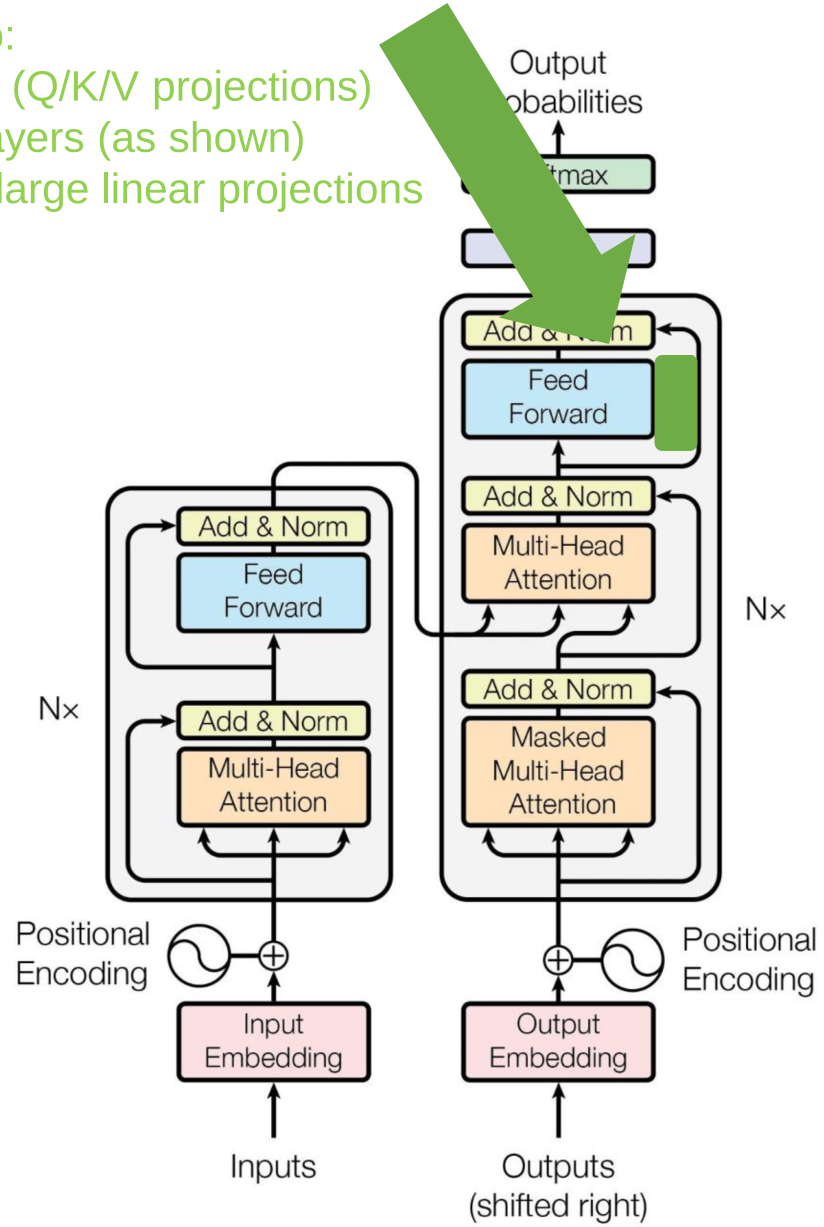


Figure 1: Our reparametrization. We only train A and B.

BitFit

$$\mathbf{Q}^{m,\ell}(\mathbf{x}) = \mathbf{W}_q^{m,\ell}\mathbf{x} + \mathbf{b}_q^{m,\ell}$$
$$\mathbf{K}^{m,\ell}(\mathbf{x}) = \mathbf{W}_k^{m,\ell}\mathbf{x} + \mathbf{b}_k^{m,\ell}$$
$$\mathbf{V}^{m,\ell}(\mathbf{x}) = \mathbf{W}_v^{m,\ell}\mathbf{x} + \mathbf{b}_v^{m,\ell}$$

Finetune only the additive bis terms b

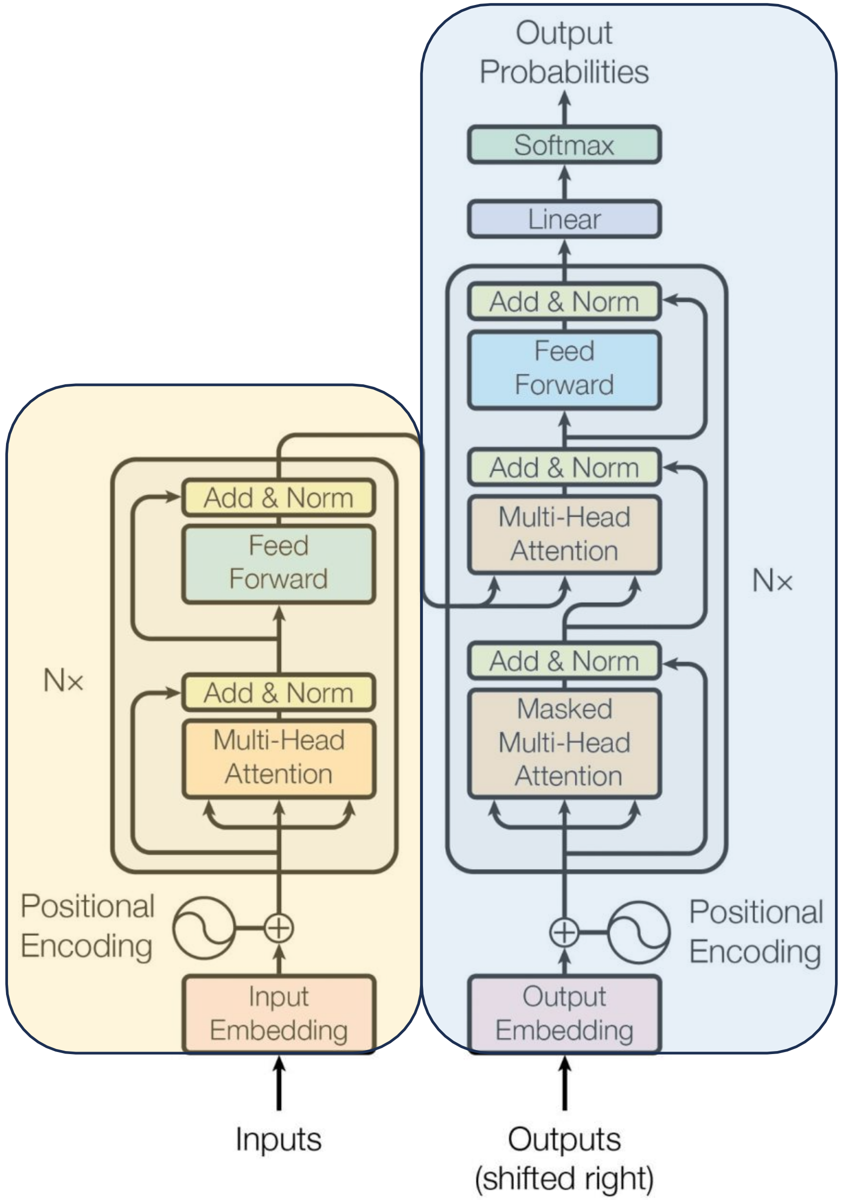


LoRA: https://arxiv.org/abs/2106.09685
BitFit: https://arxiv.org/abs/2106.10199

Table of contents

Transformers

Representation / Encoder



Generation / Decoder

Data Modalities

✓ Language

- **Vision**
- Audio
- … and many other modalities (e.g., biological/physiological signals, etc.)
- Multimodal (>2 data modalities)

Computer Vision

1. In computer vision convolutional architectures remain largely dominant.
2. Inspired by NLP successes, multiple works try introducing combining CNN-like architectures with self-attention or replacing the convolutions entirely.
3. However, they faced challenges with performance and scaling.
4. Key breakthrough - Vision Transformer (ViT) released in 2020

An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale

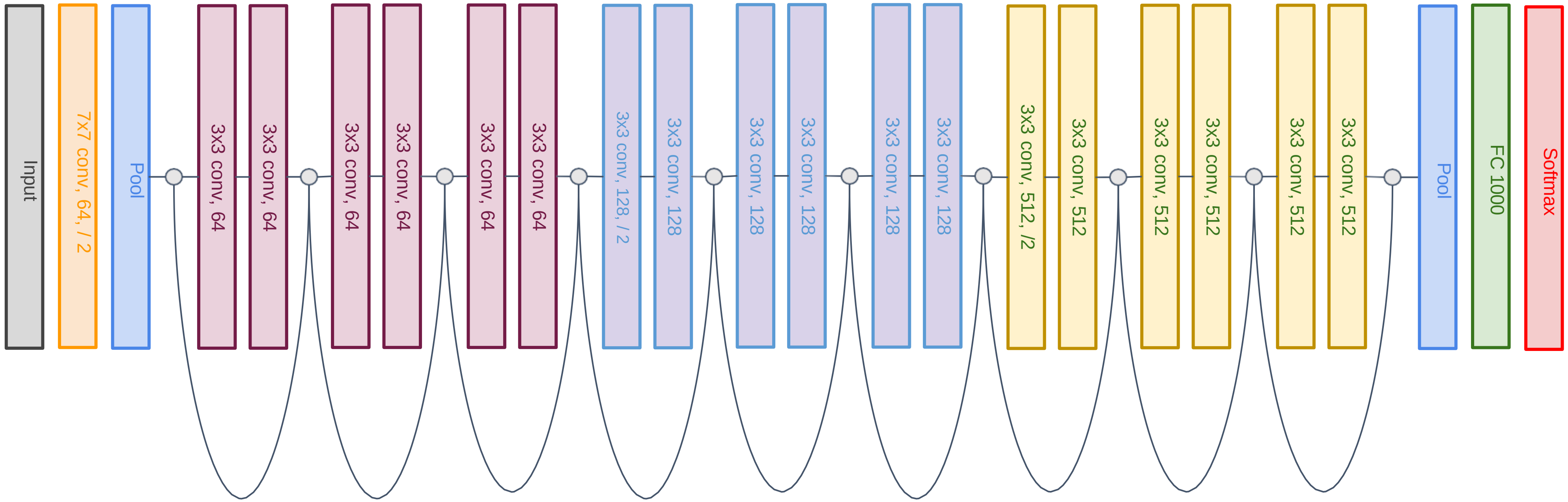
Use Attention / Transformers for Vision

-- Earlier attempts

-- Vision Transformer (ViT)

Idea #1: Add attention to existing CNNs

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

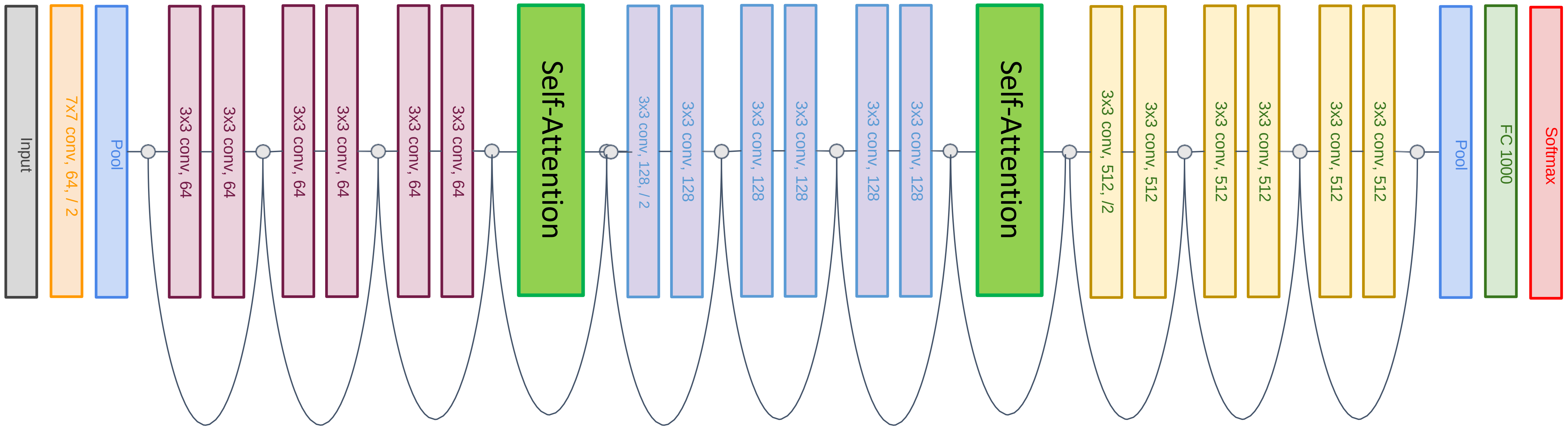


Zhang et al, "Self-Attention Generative Adversarial Networks", ICML 2018
Wang et al, "Non-local Neural Networks", CVPR 2018

Idea #1: Add attention to existing CNNs

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

Add Self-Attention blocks between existing ResNet blocks



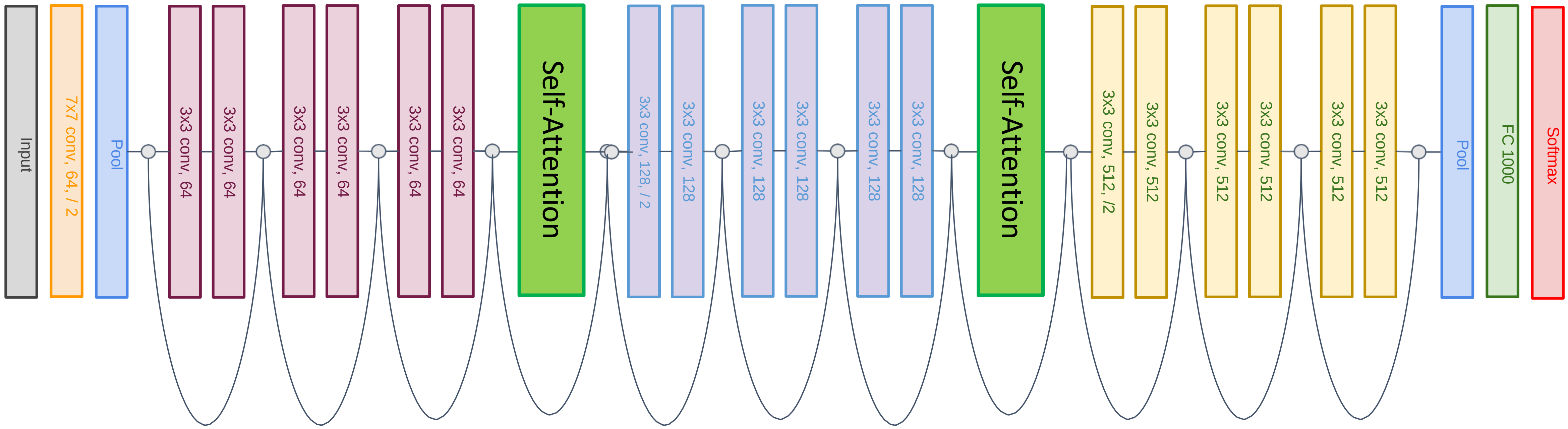
Zhang et al, "Self-Attention Generative Adversarial Networks", ICML 2018
Wang et al, "Non-local Neural Networks", CVPR 2018

Idea #1: Add attention to existing CNNs

Model is still a CNN!
Can we replace convolution entirely?

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

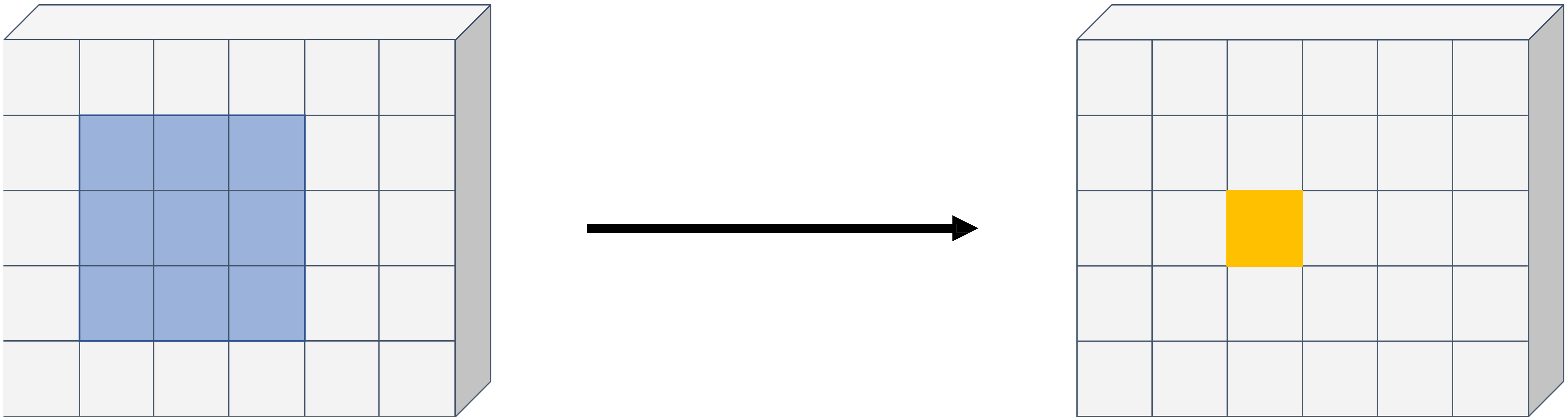
Add Self-Attention blocks between existing ResNet blocks



Zhang et al, "Self-Attention Generative Adversarial Networks", ICML 2018
Wang et al, "Non-local Neural Networks", CVPR 2018

Idea #2: Replace Convolution with "Local Attention"

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

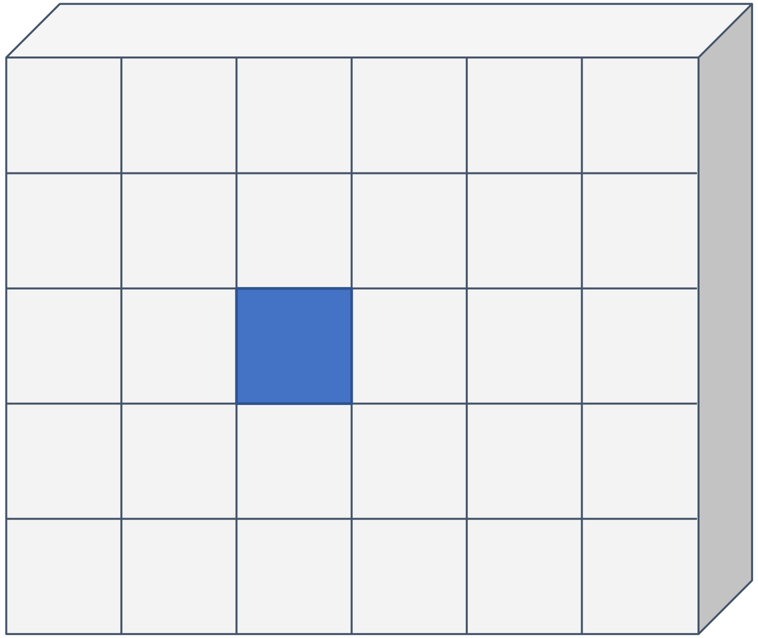


Hu et al, "Local Relation Networks for Image Recognition", ICCV 2019;

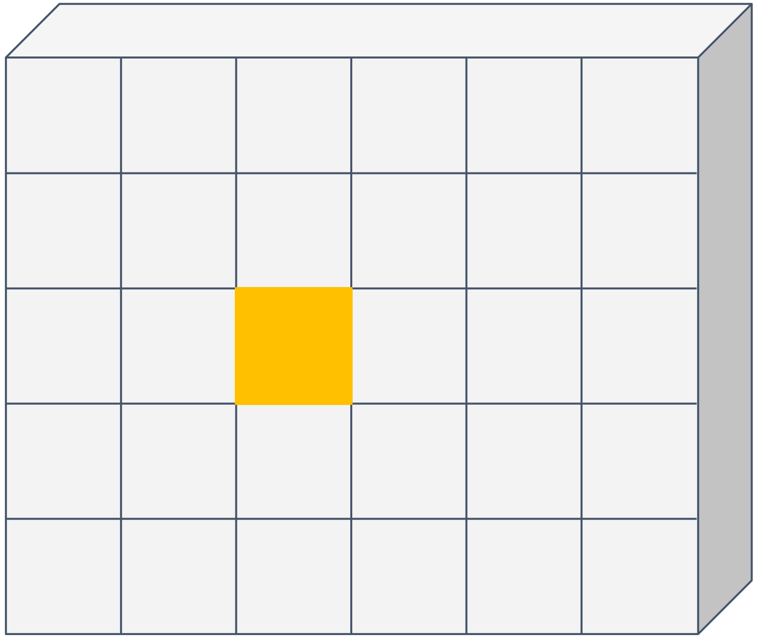
Ramachandran et al, "Stand-Alone Self-Attention in Vision Models", NeurIPS 2019

Idea #2: Replace Convolution with "Local Attention"

Map center of receptive field to query



Query: D_Q

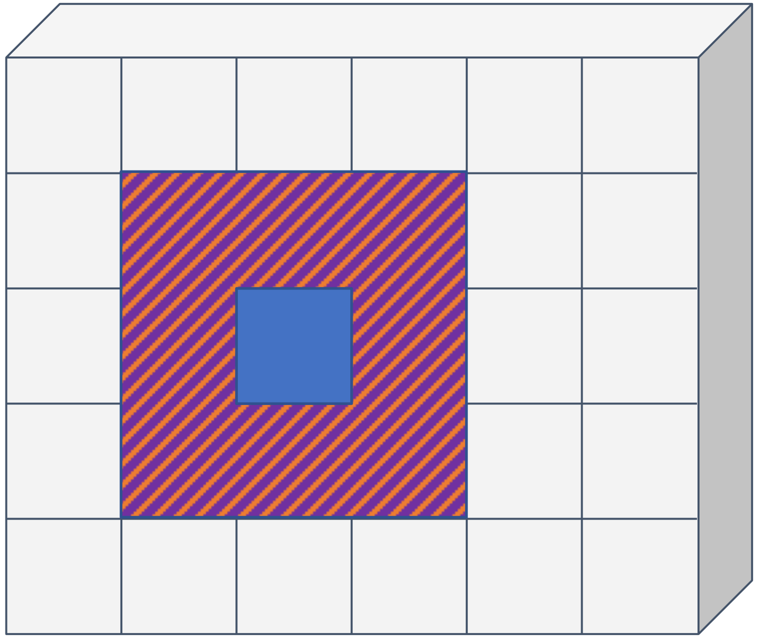


Hu et al, "Local Relation Networks for Image Recognition", ICCV 2019;

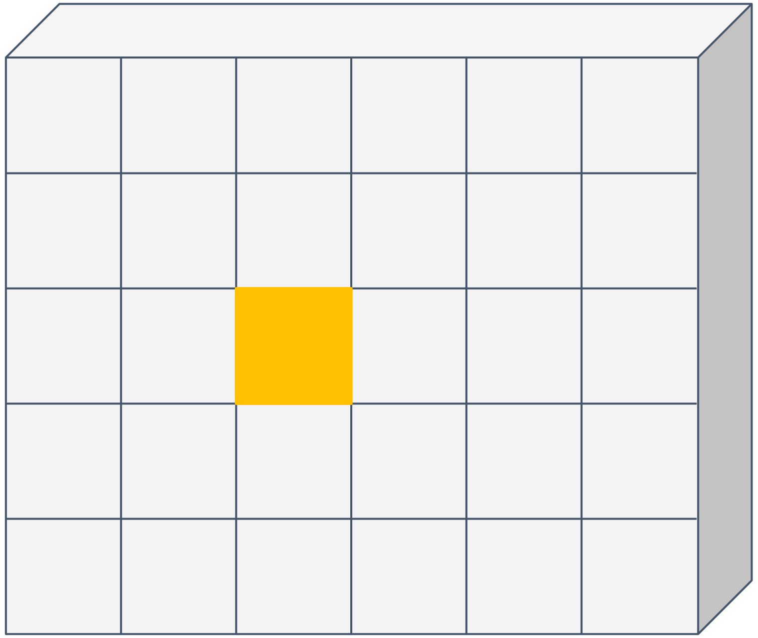
Ramachandran et al, "Stand-Alone Self-Attention in Vision Models", NeurIPS 2019

Idea #2: Replace Convolution with "Local Attention"

Map center of receptive field to **query**
Map each element in receptive field to **key** and **value**



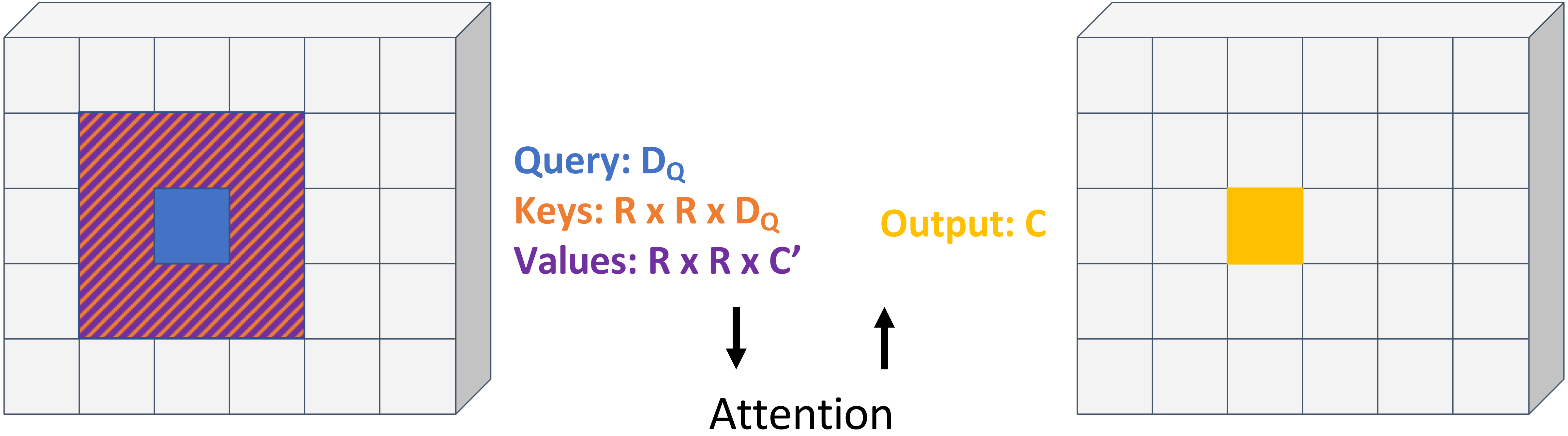
Query: D_Q
Keys: $R \times R \times D_Q$
Values: $R \times R \times C'$



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"

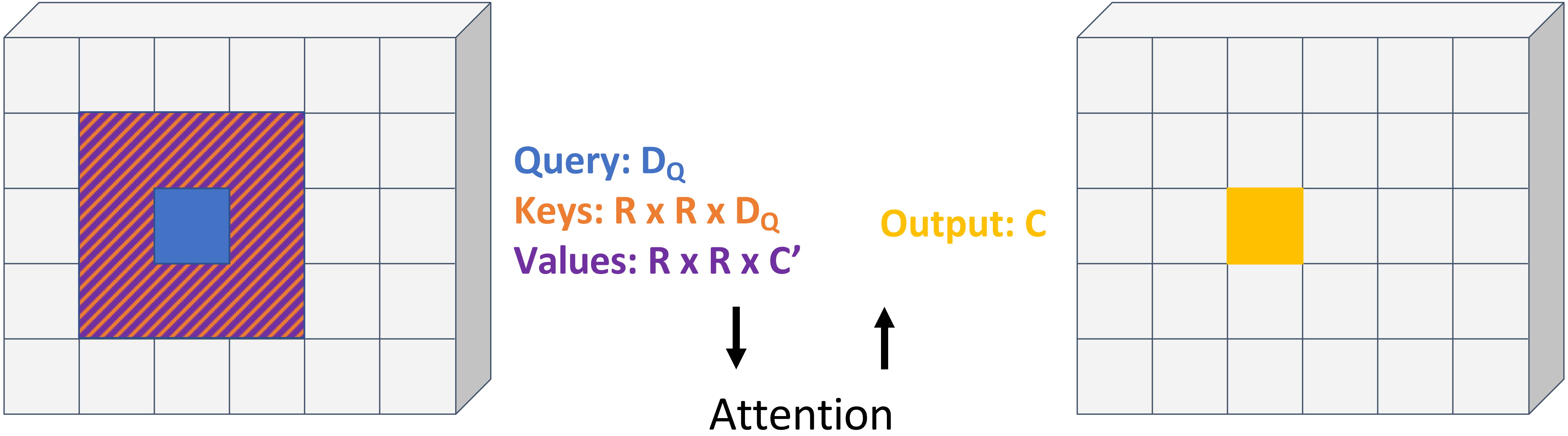
Map center of receptive field to **query**
Map each element in receptive field to **key** and **value**
Compute **output** using attention



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

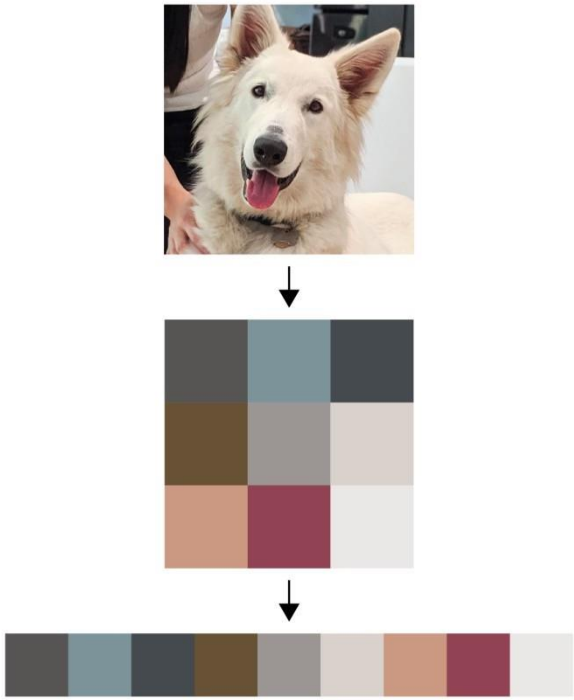


Lots of tricky details, hard to implement, only marginally better than ResNets

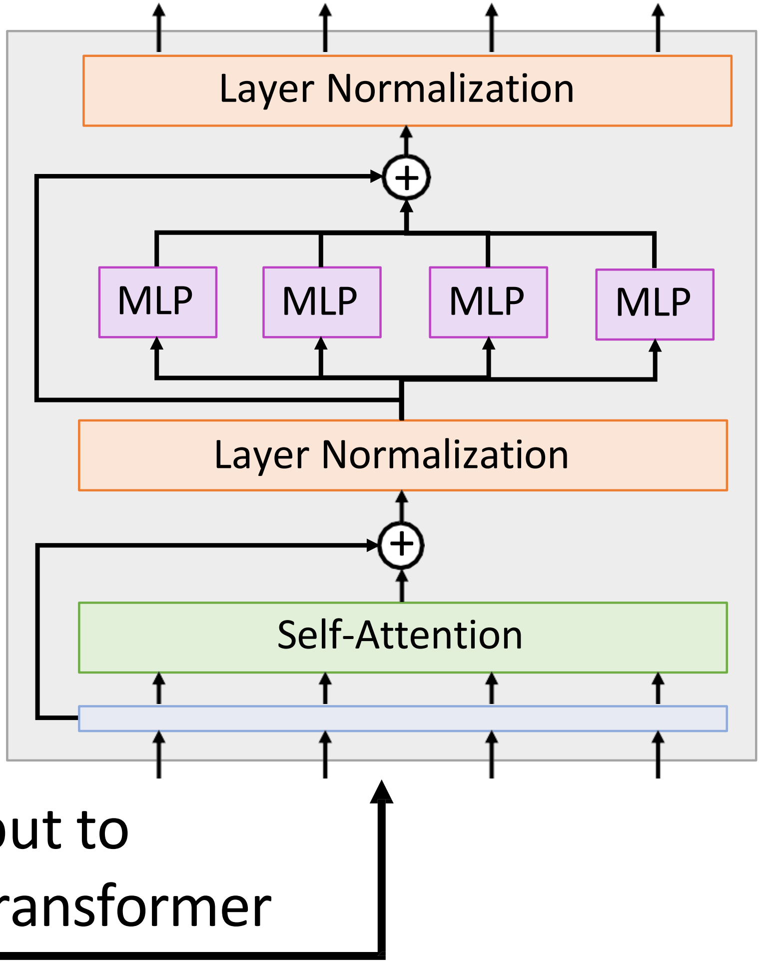
Hu et al, "Local Relation Networks for Image Recognition", ICCV 2019;
Ramachandran et al, "Stand-Alone Self-Attention in Vision Models", NeurIPS 2019

Idea #3: Standard Transformer on Pixels

Treat an image as a set of pixel values



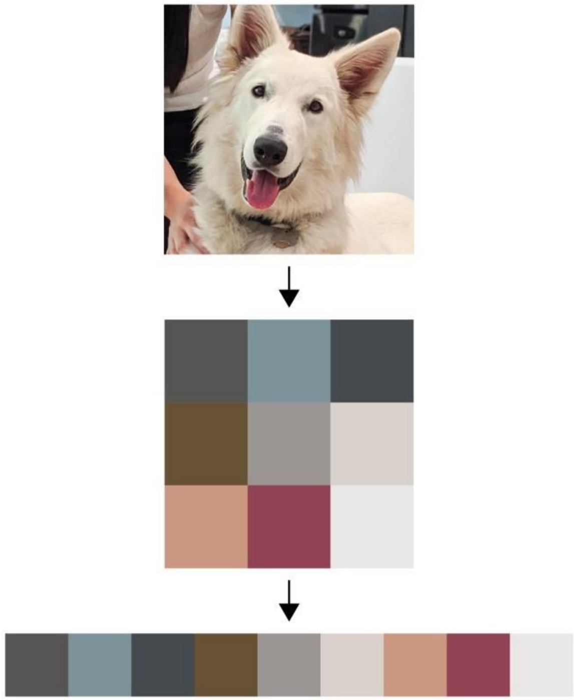
Feed as input to standard Transformer



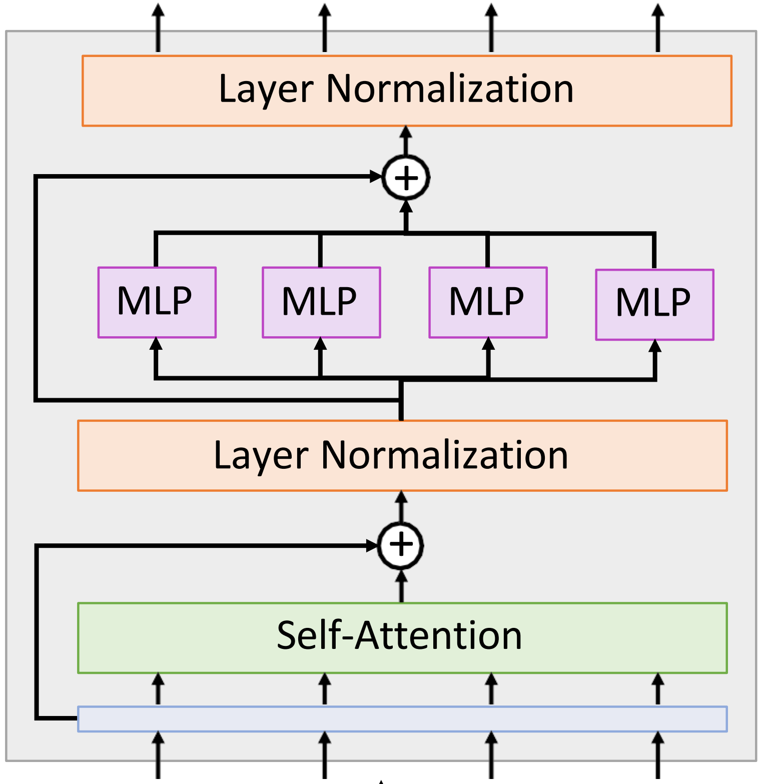
Chen et al, "Generative Pretraining from Pixels", ICML 2020

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

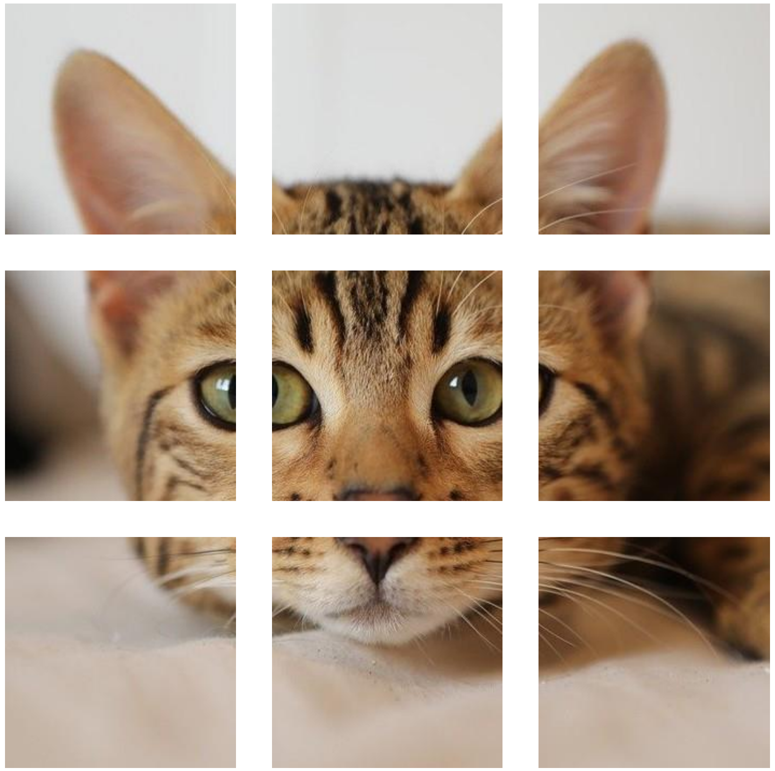
R=128, 48 layers, 16 heads per layer takes 768GB of memory for attention matrices for a single example…

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

Idea #4: Standard Transformer on Patches

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

Idea #4: Standard Transformer on Patches



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

Idea #4: Standard Transformer on Patches

N input patches, each of shape 3x16x16

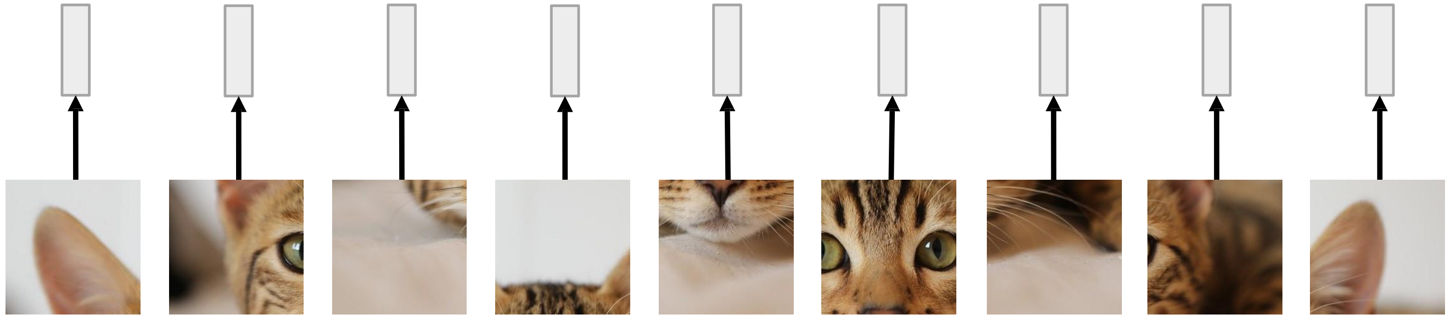
Cat image is free for commercial use under a Pixabay license

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

Idea #4: Standard Transformer on Patches

Linear projection to D-dimensional vector

N input patches, each of shape 3x16x16



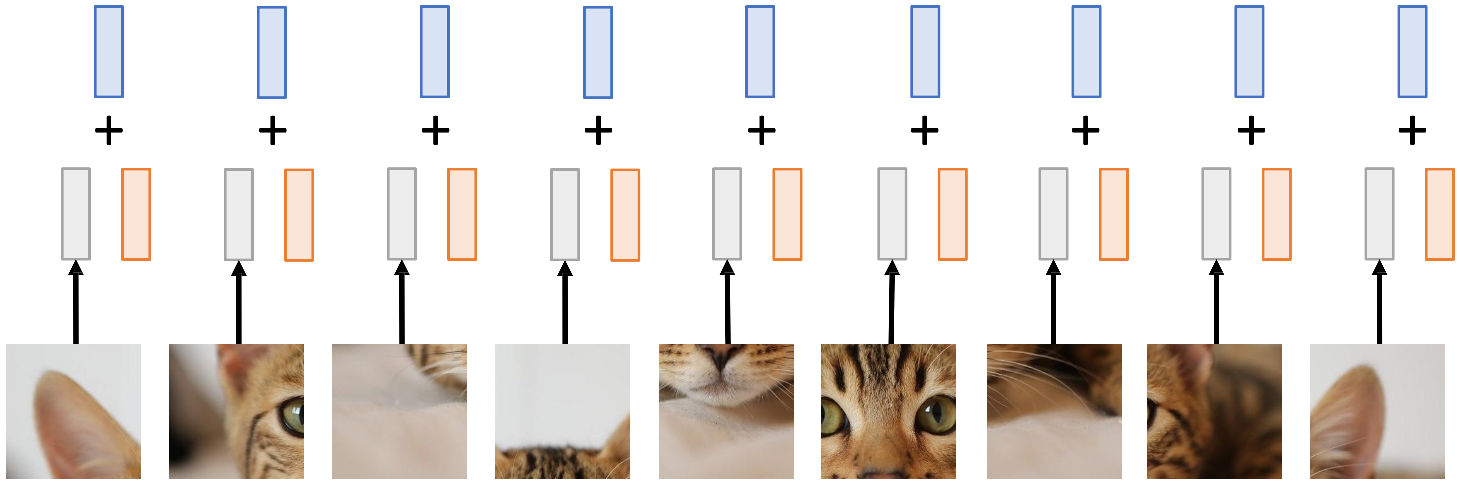
Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

Idea #4: Standard Transformer on Patches

Add positional embedding: learned D-dim vector per position

Linear projection to D-dimensional vector

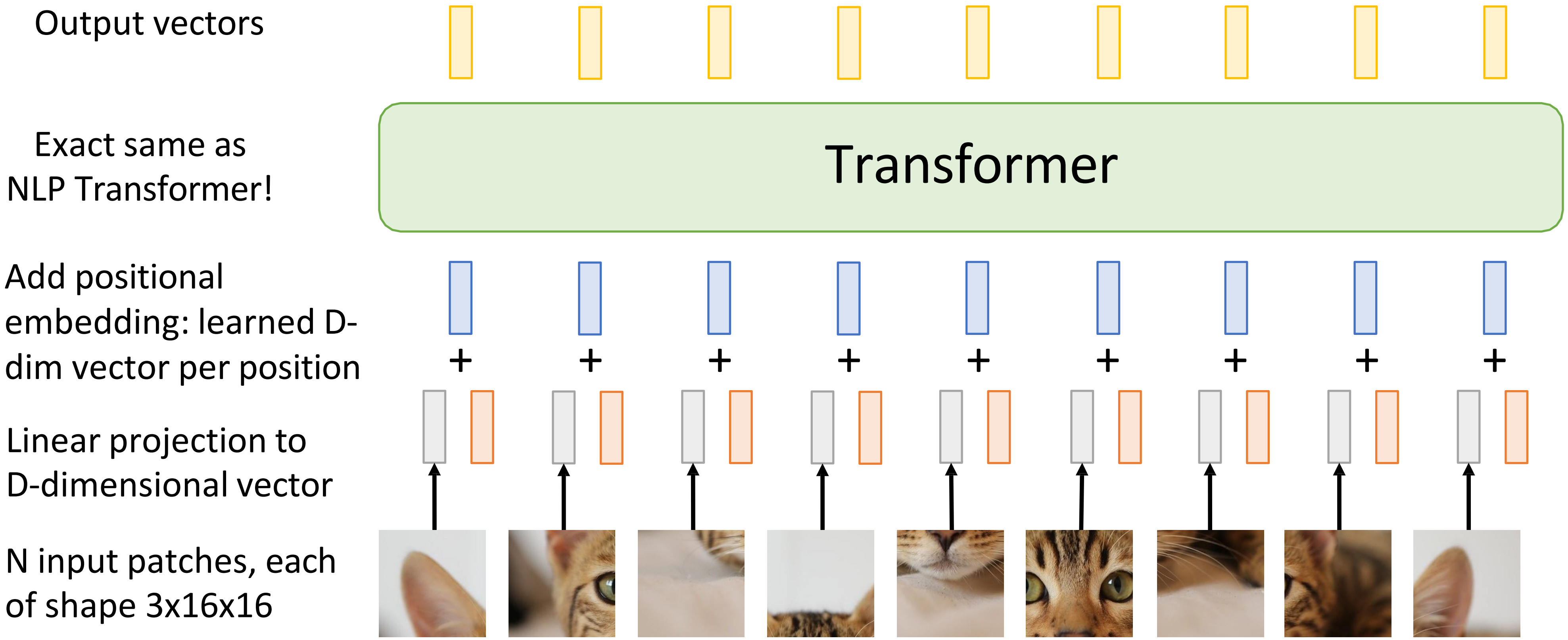
N input patches, each of shape 3x16x16



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

Idea #4: Standard Transformer on Patches



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

Idea #4: Standard Transformer on Patches

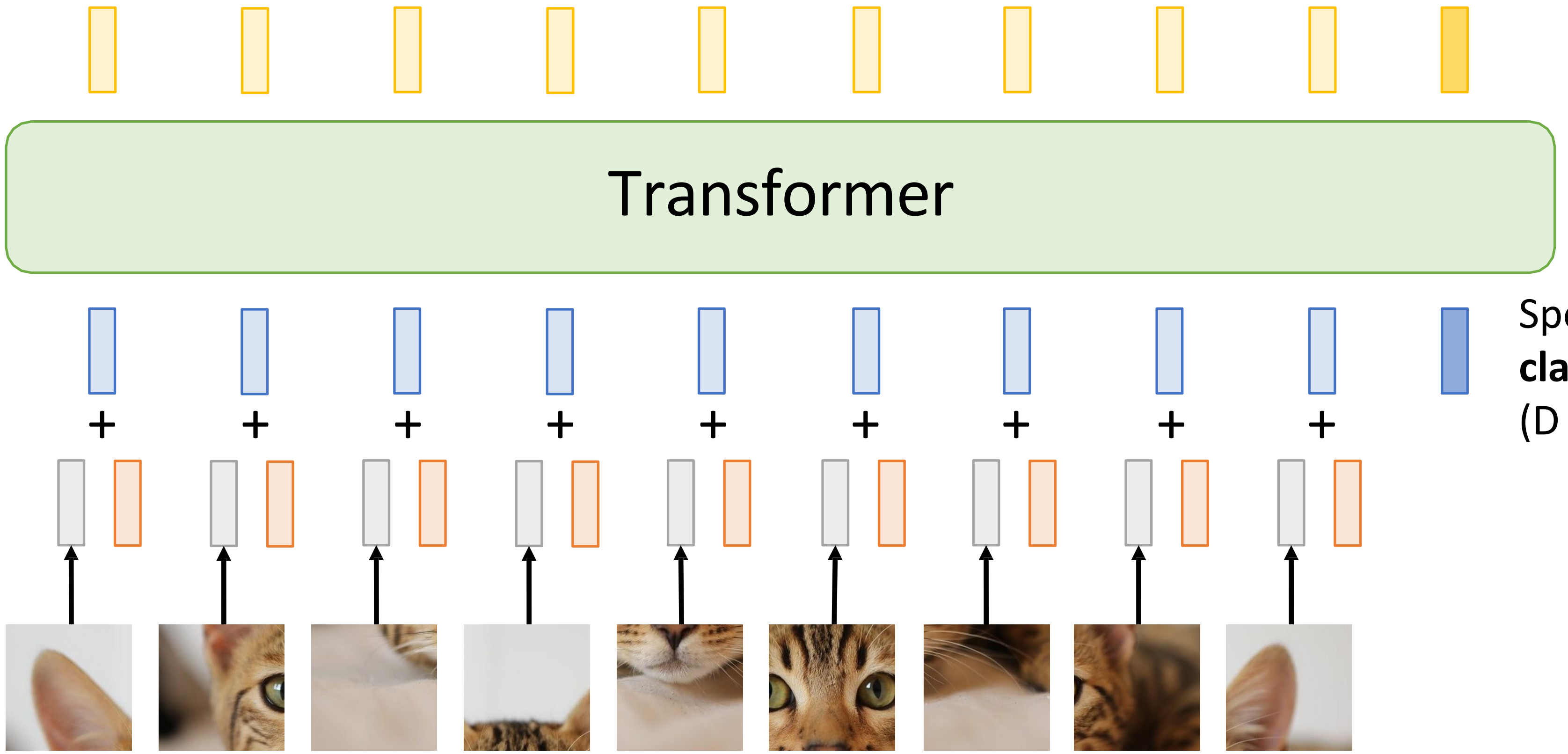
Output vectors

Exact same as
NLP Transformer!

Add positional
embedding: learned D-
dim vector per position

Linear projection to
D-dimensional vector

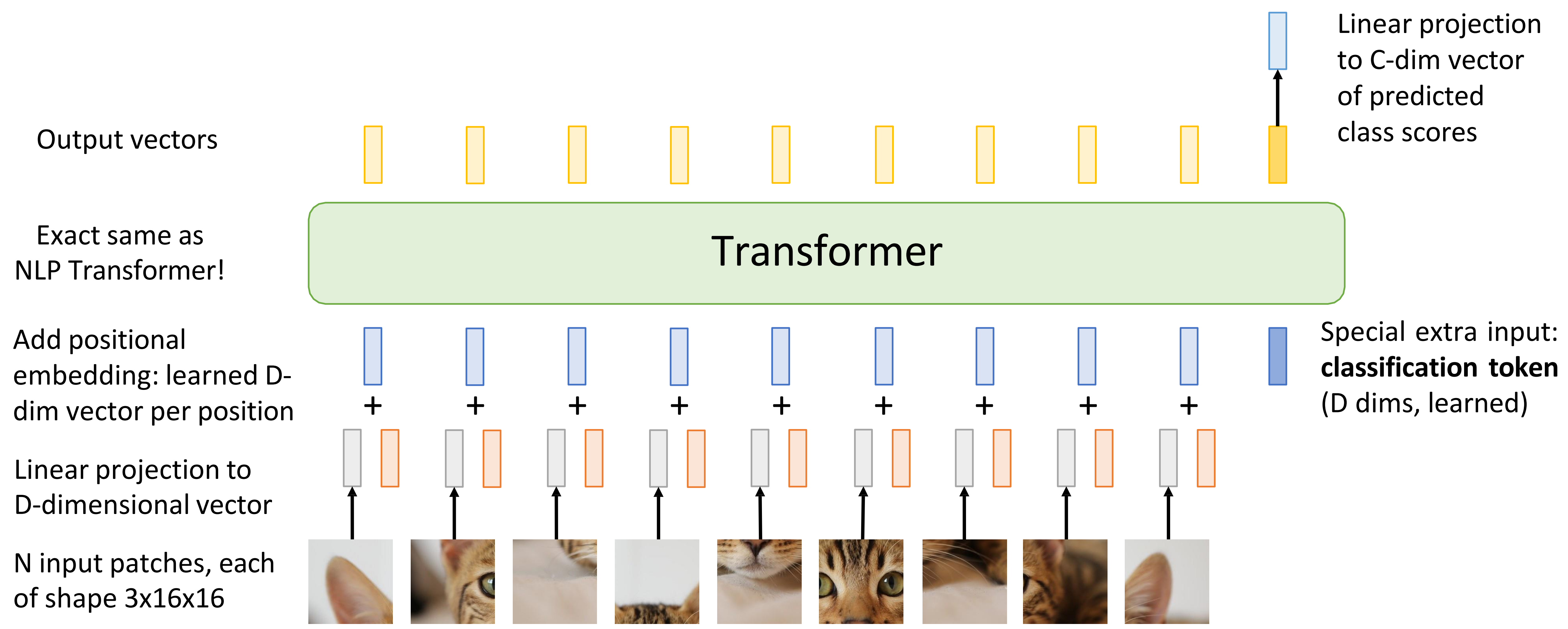
N input patches, each
of shape 3x16x16



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

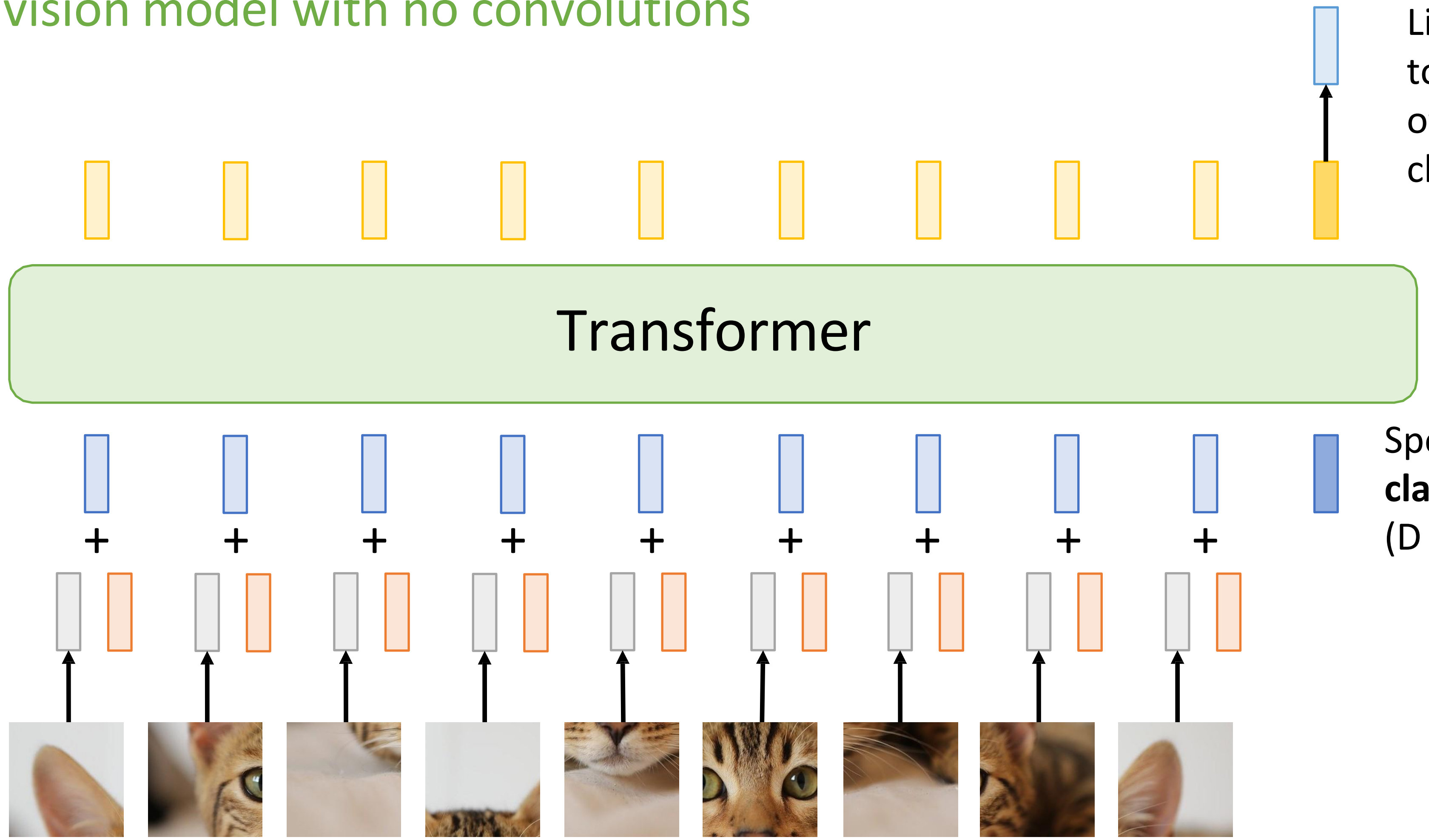
Idea #4: Standard Transformer on Patches



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



Linear projection to C-dim vector of predicted class scores

Output vectors

Exact same as NLP Transformer!

Add positional embedding: learned D-dim vector per position

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

Linear projection to D-dimensional vector

N input patches, each of shape 3x16x16

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

Vision Transformer (ViT)

Computer vision model with no convolutions

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

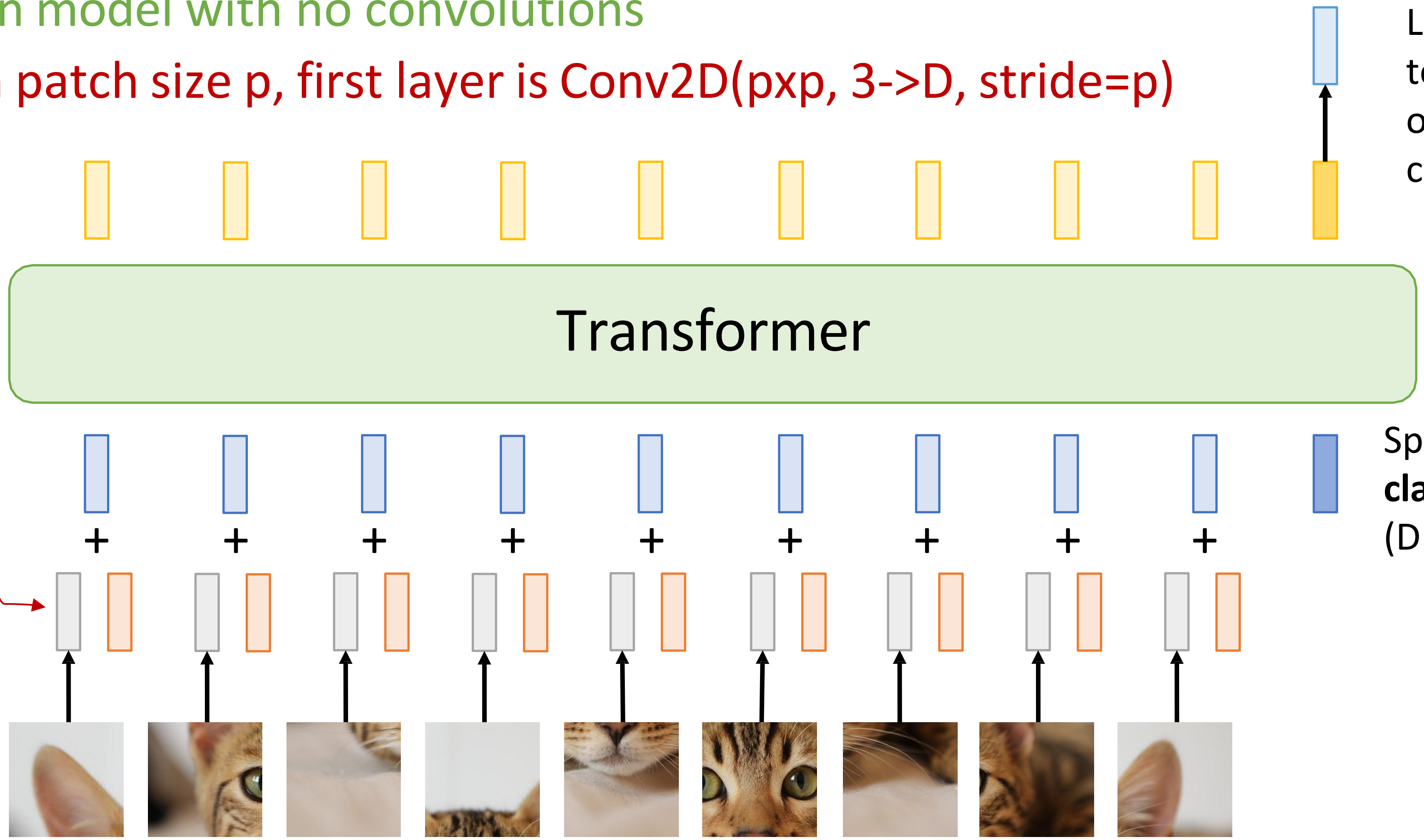
Output vectors

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)

Computer vision model with no convolutions

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

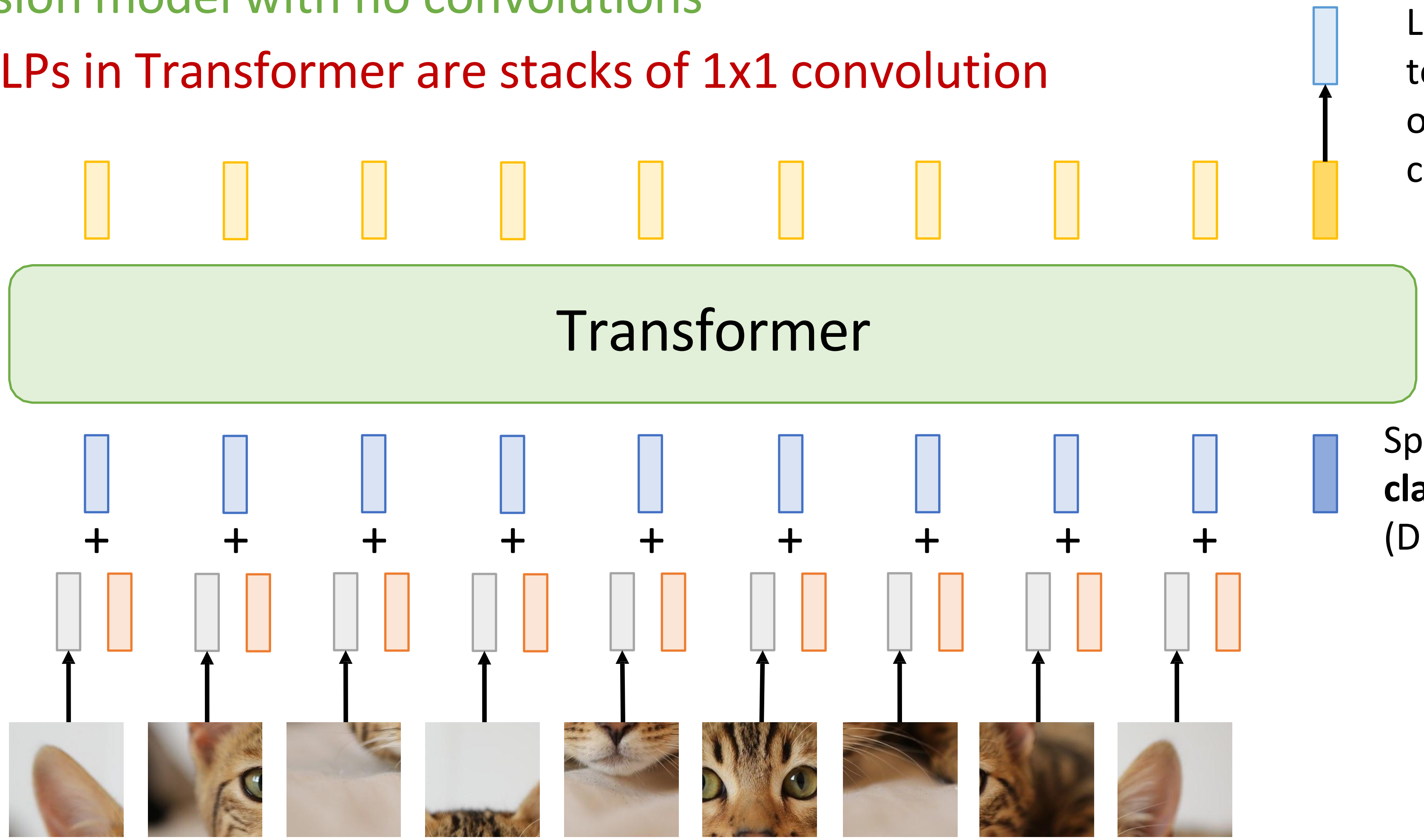
Output vectors

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)

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

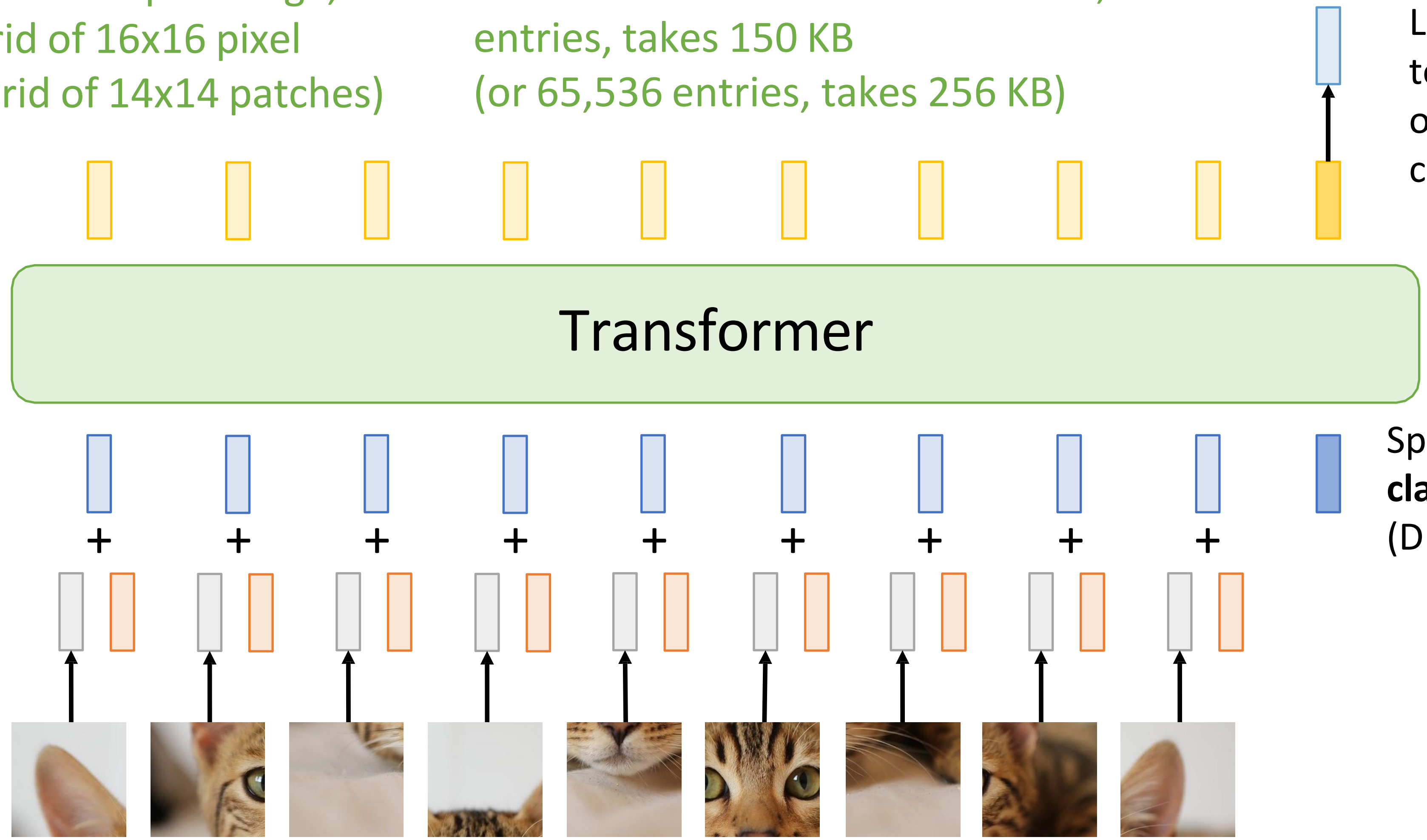
(drop classification token here for simplicity)

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

Output vectors

Exact same as NLP Transformer!



Add positional embedding: learned D-dim vector per position

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

Linear projection to D-dimensional vector

N input patches, each of shape 3x16x16

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

Vision Transformer (ViT)

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

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

Linear projection to C-dim vector of predicted class scores

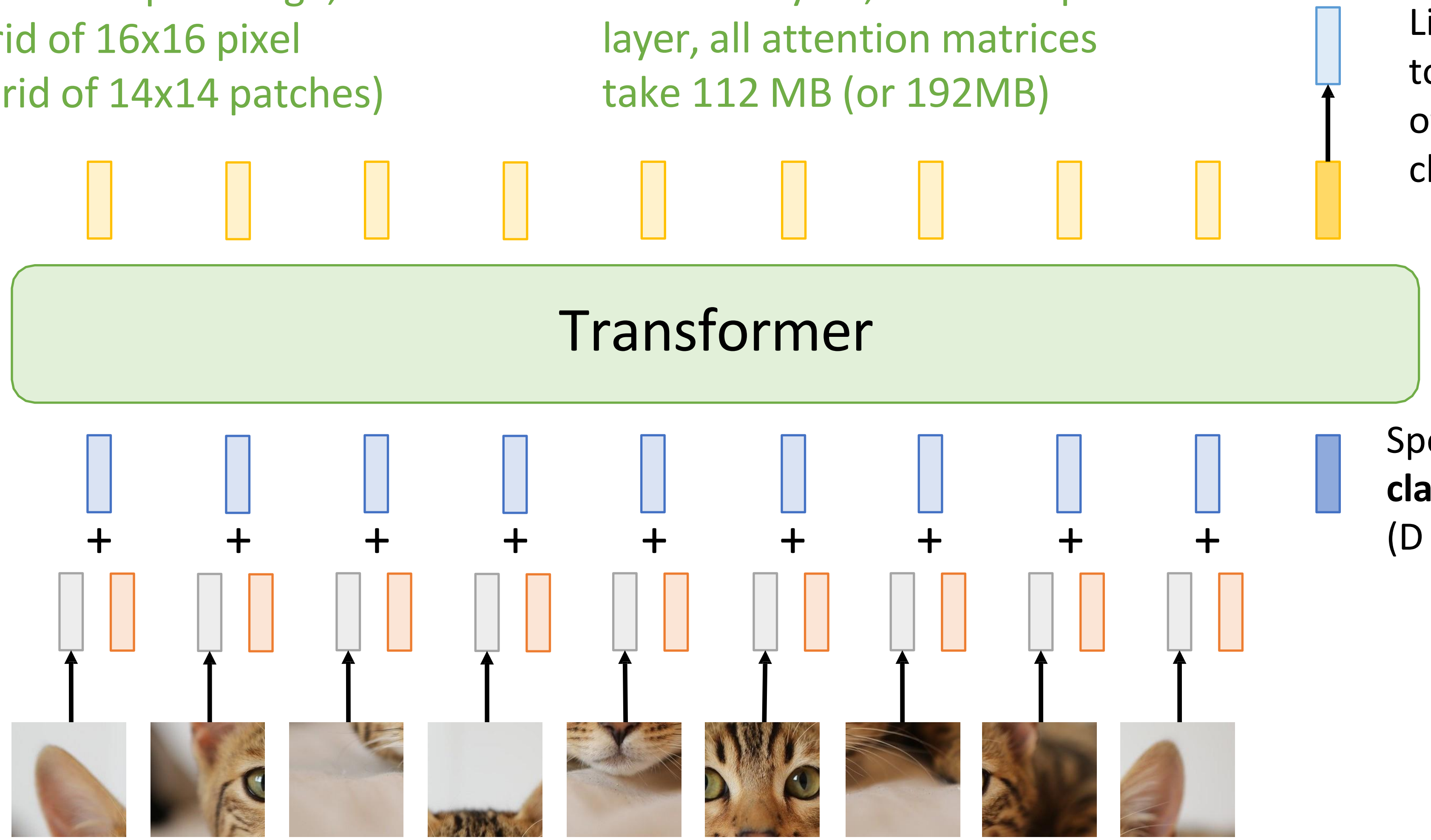
Output vectors

Exact same as NLP Transformer!

Add positional embedding: learned D-dim vector per position

Linear projection to D-dimensional vector

N input patches, each of shape 3x16x16



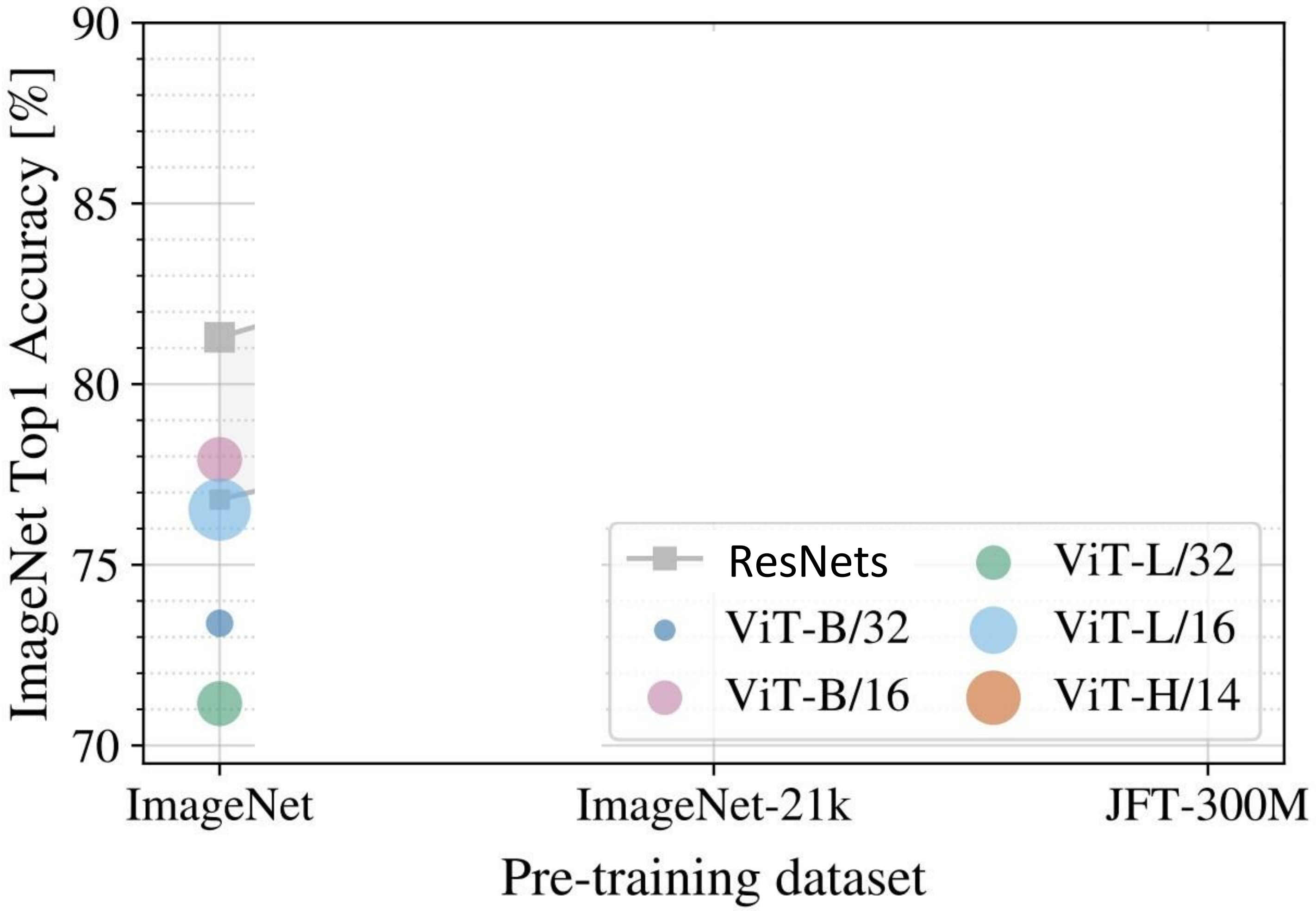
Special extra input: **classification token** (D dims, learned)

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

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

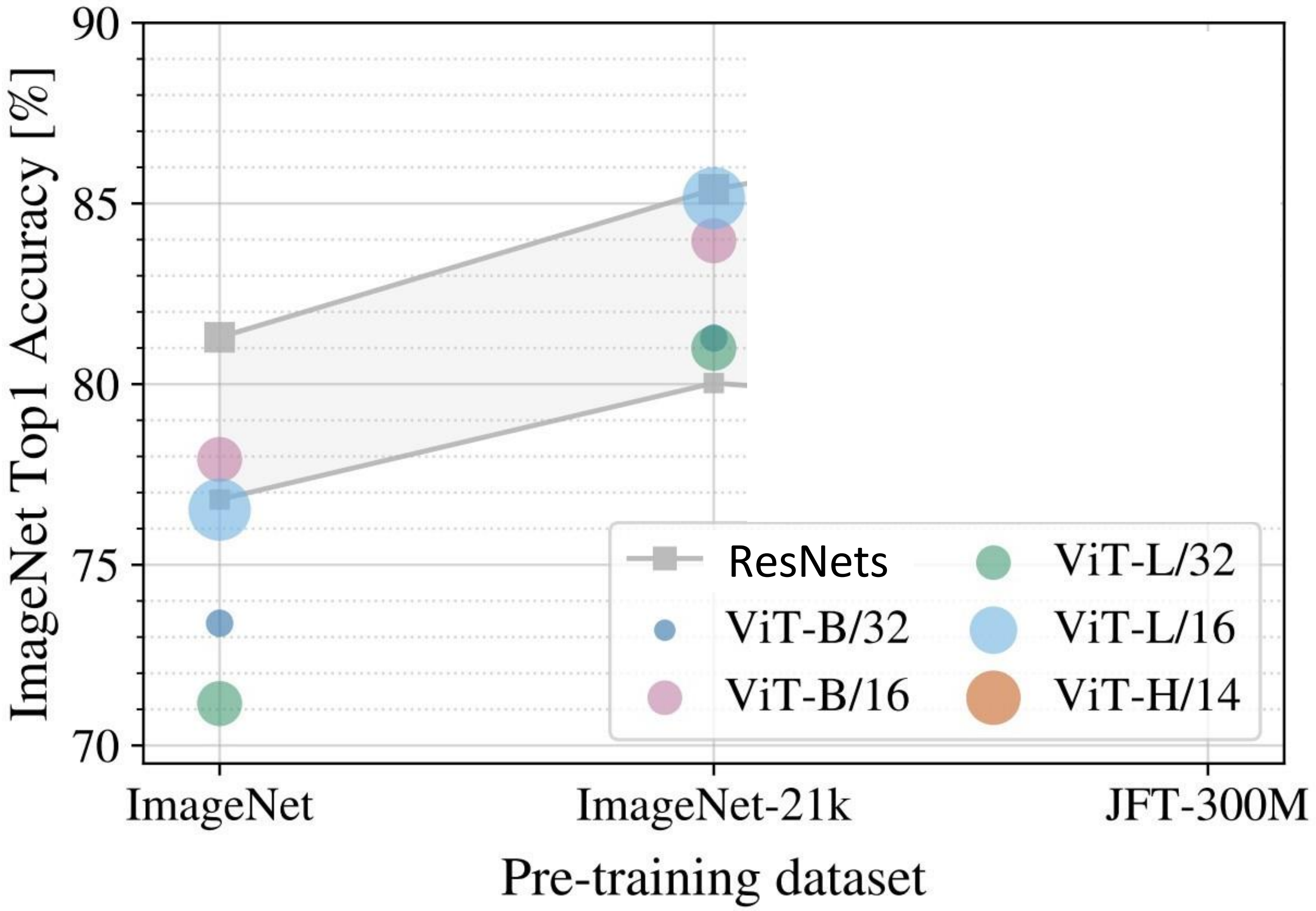
/32, /16, /14 is patch size; smaller patch size is a bigger model (more patches)

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

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

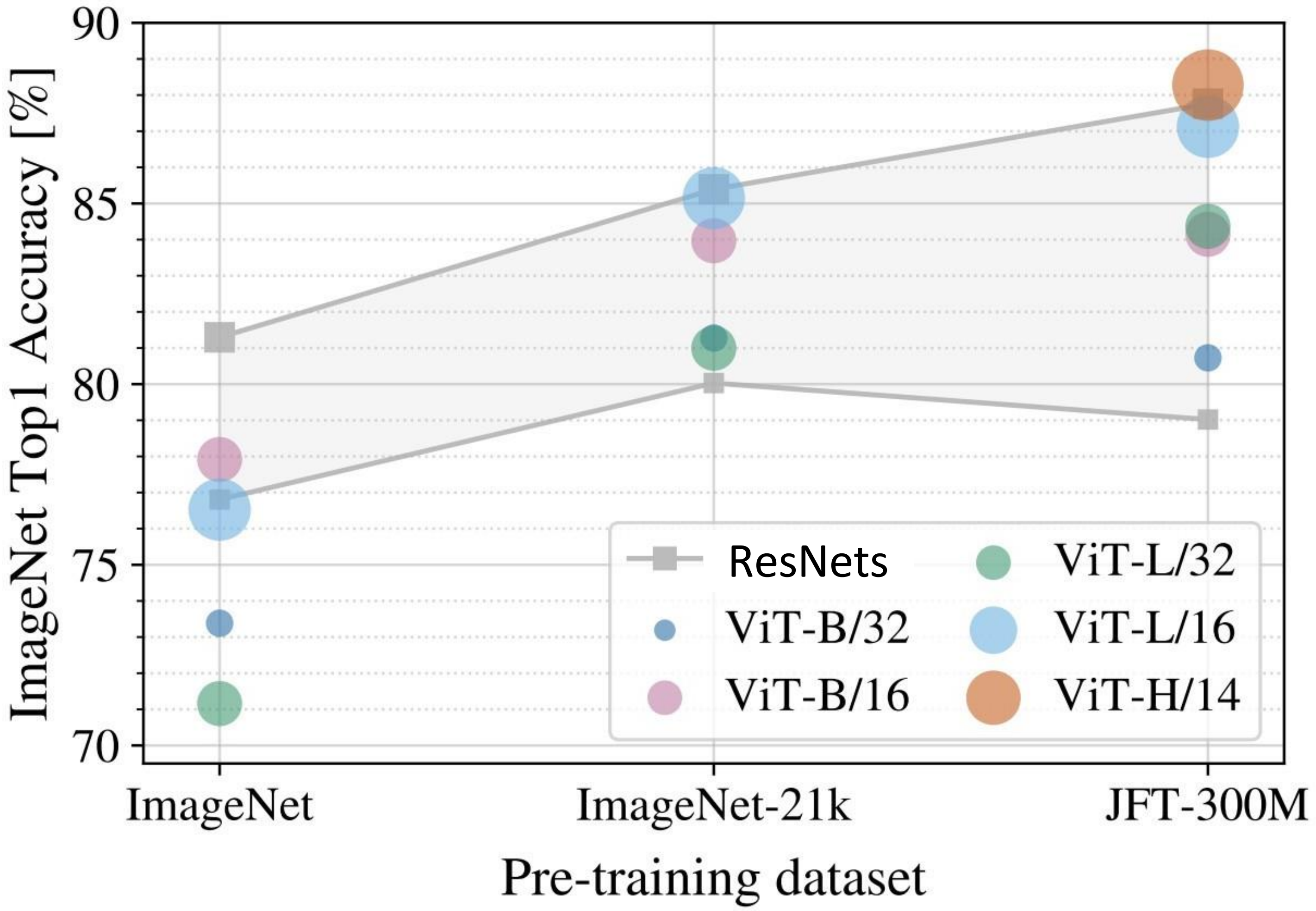
/32, /16, /14 is patch size; smaller patch size is a bigger model (more patches)

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

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

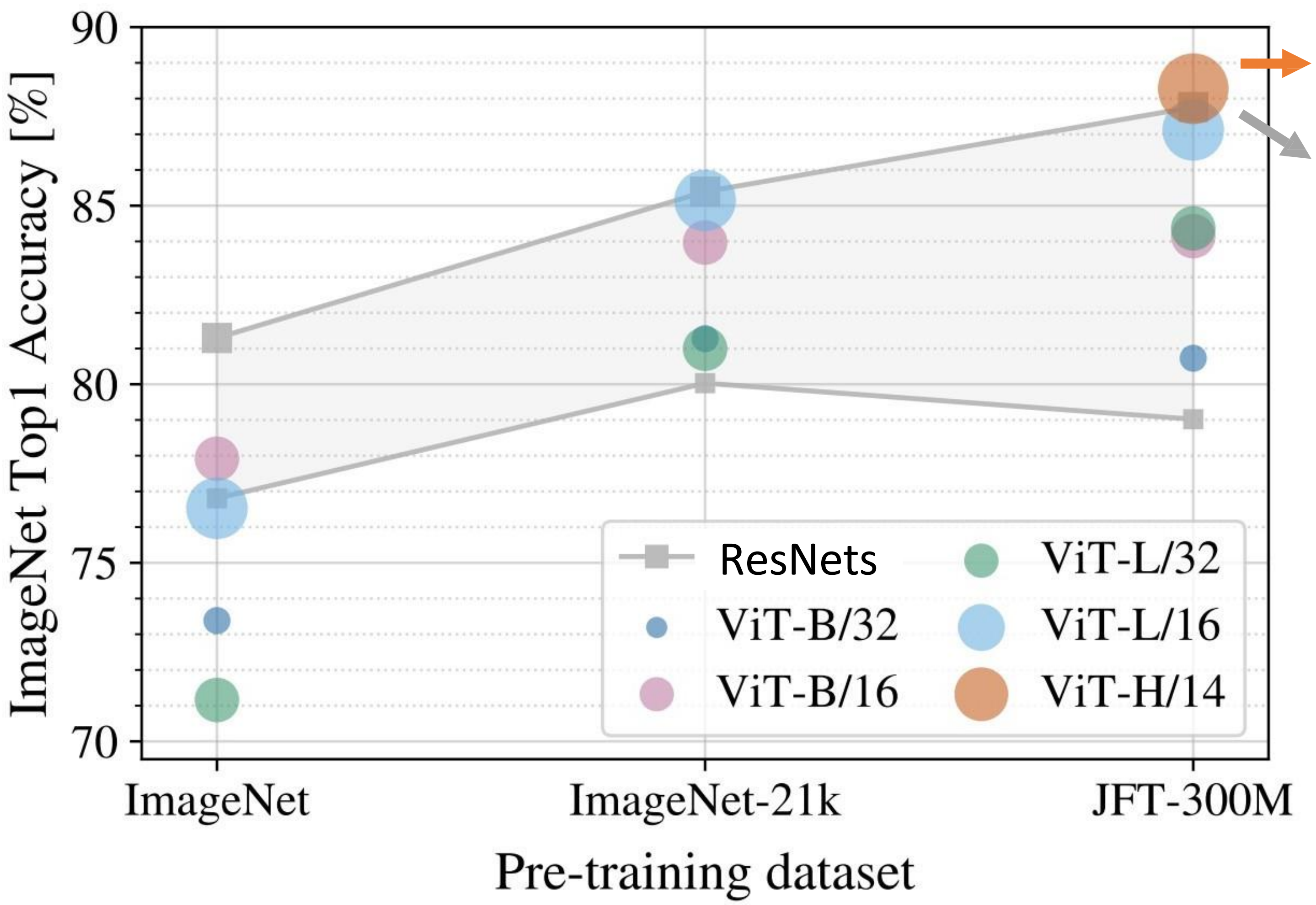
/32, /16, /14 is patch size; smaller patch size is a bigger model (more patches)

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

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



ViT: 2.5k TPU-v3 core days of training

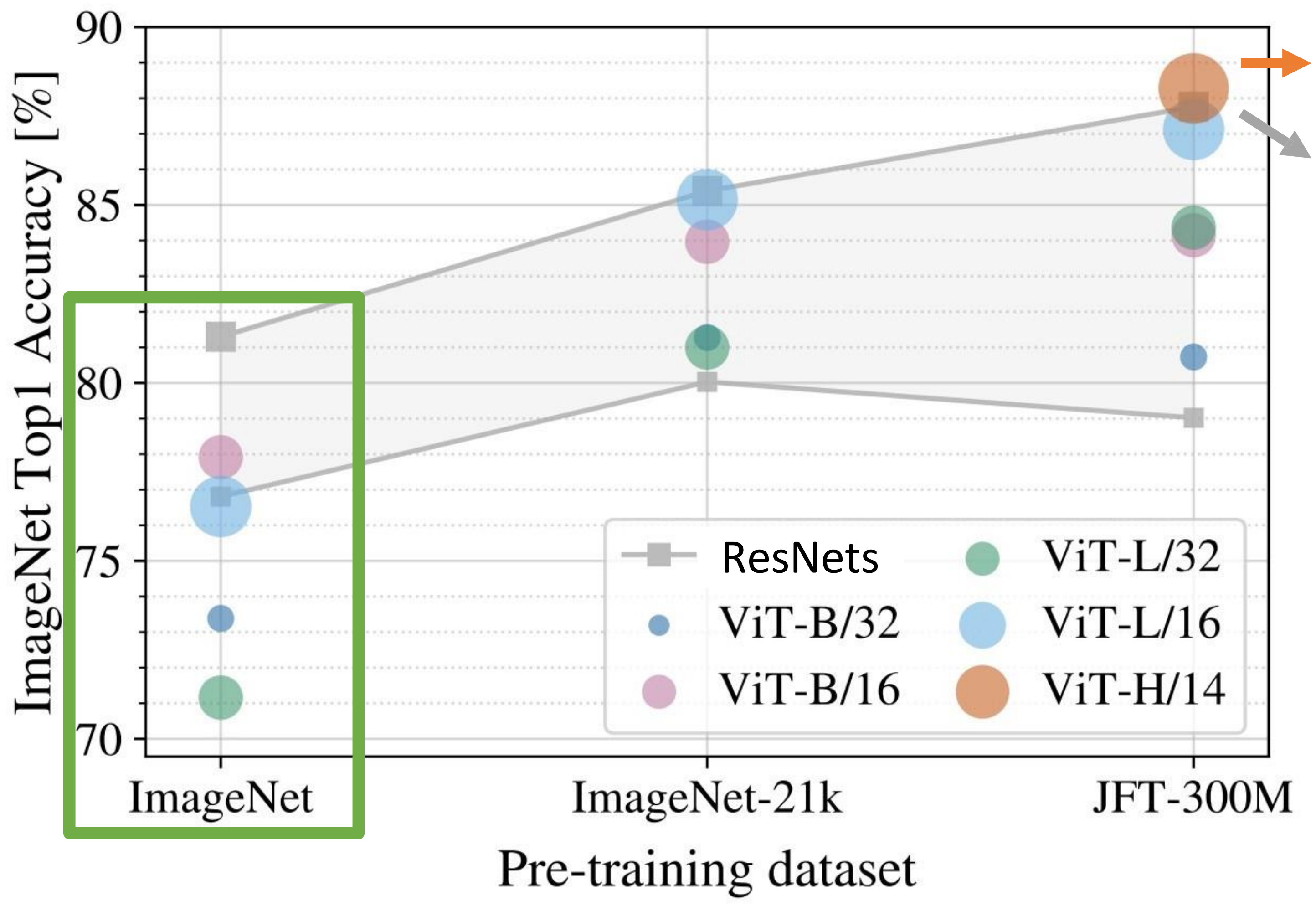
ResNet: 9.9k TPU-v3 core days of training

ViTs make more efficient use of GPU / TPU hardware (matrix multiply is more hardware-friendly than conv)

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?



ViT: 2.5k TPU-v3 core days of training

ResNet: 9.9k TPU-v3 core days of training

ViTs make more efficient use of GPU / TPU hardware (matrix multiply is more hardware-friendly than conv)

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

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

Improving ViT: Augmentation and Regularization

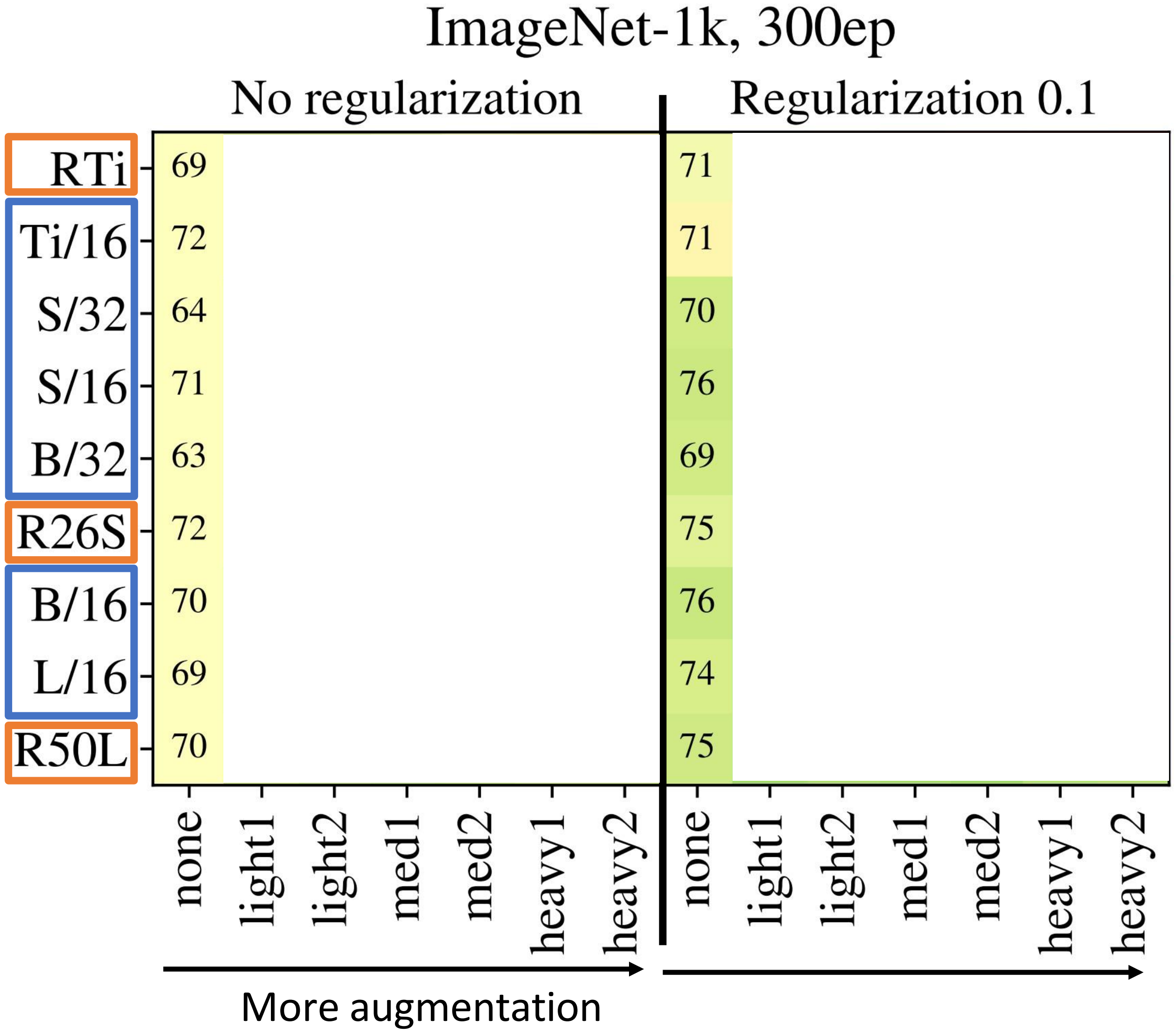
Regularization for ViT models:
- Weight Decay
- Stochastic Depth
- Dropout (in FFN layers of Transformer)

Data Augmentation for ViT models:
- MixUp
- RandAugment

Hybrid models: ResNet blocks, then ViT blocks

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

Original Paper:
77.9
76.53



Adding regularization is (almost) always helpful

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)

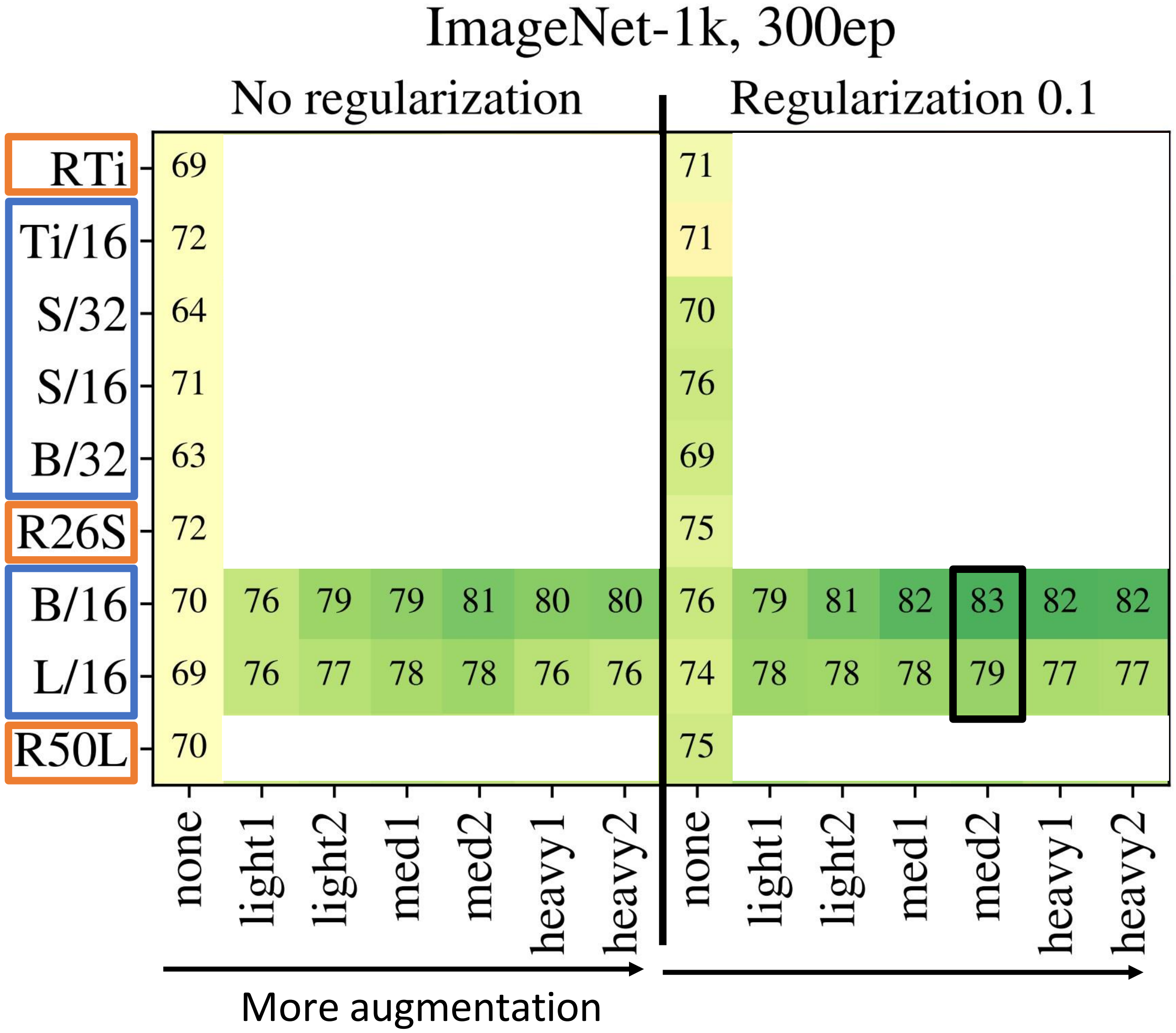
Data Augmentation for ViT models:
- MixUp
- RandAugment

Regularization + Augmentation gives big improvements over original results

Hybrid models: ResNet blocks, then ViT blocks

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

Original Paper:
77.9
76.53



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

Lots of other patterns in full results

Hybrid models: ResNet blocks, then ViT blocks

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

Original Paper:
77.9
76.53

ImageNet-1k, 300ep

	No regularization							Regularization 0.1						
	none	light1	light2	med1	med2	heavy1	heavy2	none	light1	light2	med1	med2	heavy1	heavy2
RTi	69	73	73	72	70	69	68	71	70	67	65	63	62	61
Ti/16	72	76	75	75	74	72	71	71	72	68	65	63	63	62
S/32	64	71	76	76	76	74	74	70	72	72	71	71	69	68
S/16	71	77	79	81	82	80	80	76	79	80	79	79	77	77
B/32	63	70	73	75	76	75	76	69	74	77	77	78	77	77
R26S	72	76	78	79	80	80	80	75	78	81	82	82	81	81
B/16	70	76	79	79	81	80	80	76	79	81	82	83	82	82
L/16	69	76	77	78	78	76	76	74	78	78	78	79	77	77
R50L	70	75	76	77	77	76	76	75	78	78	78	79	77	77

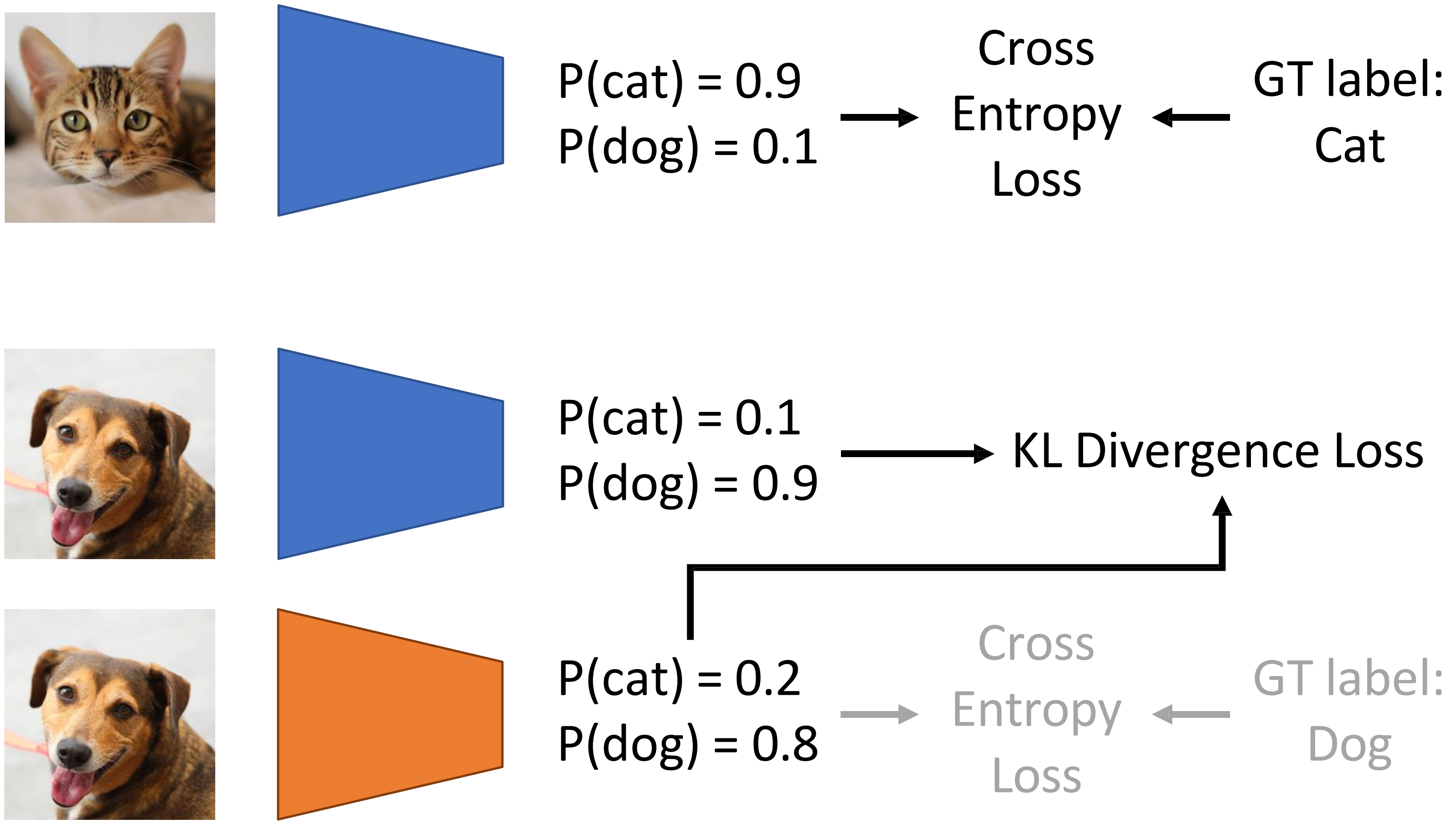
More augmentation

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

Improving ViT: Distillation

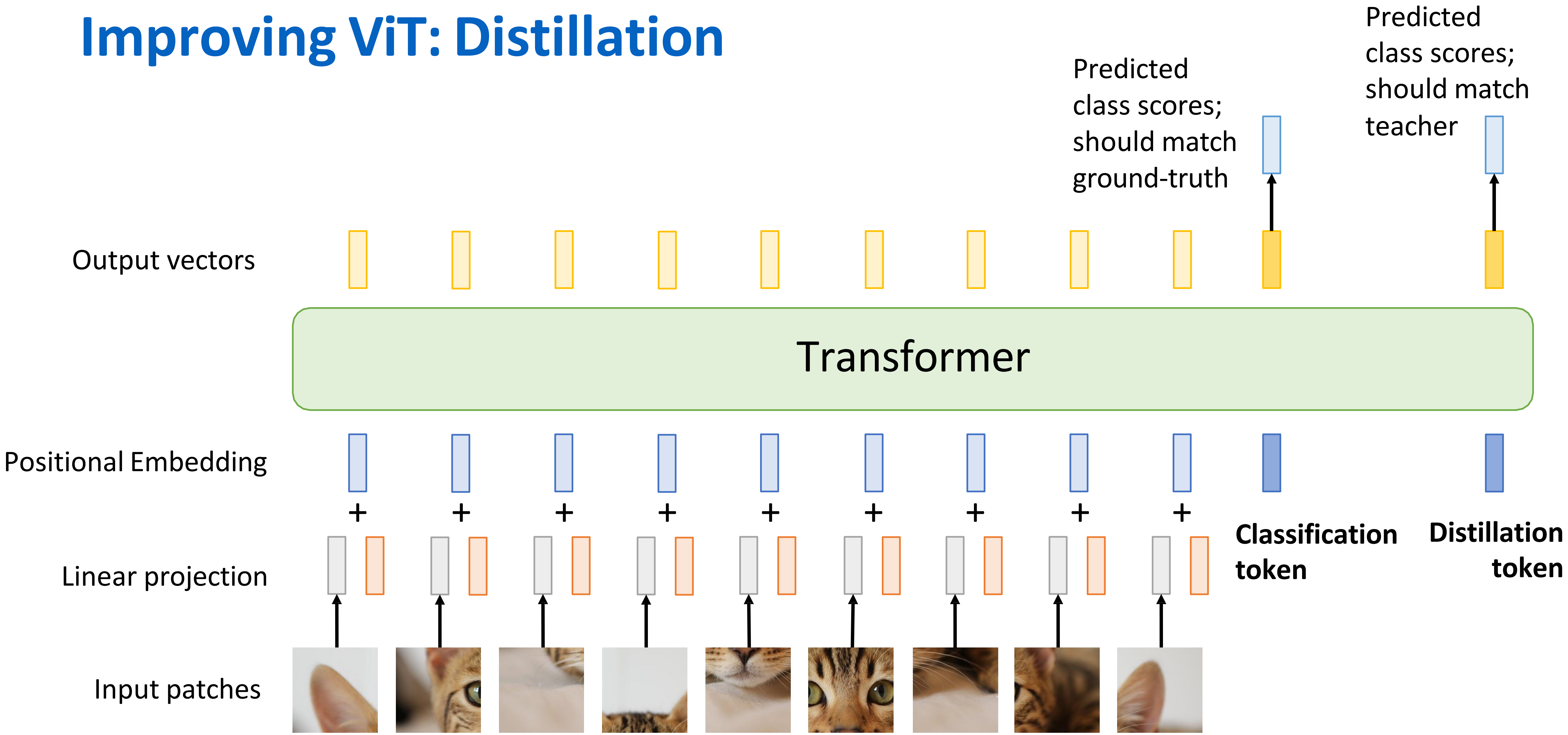
Step 1: Train a teacher CNN on ImageNet

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



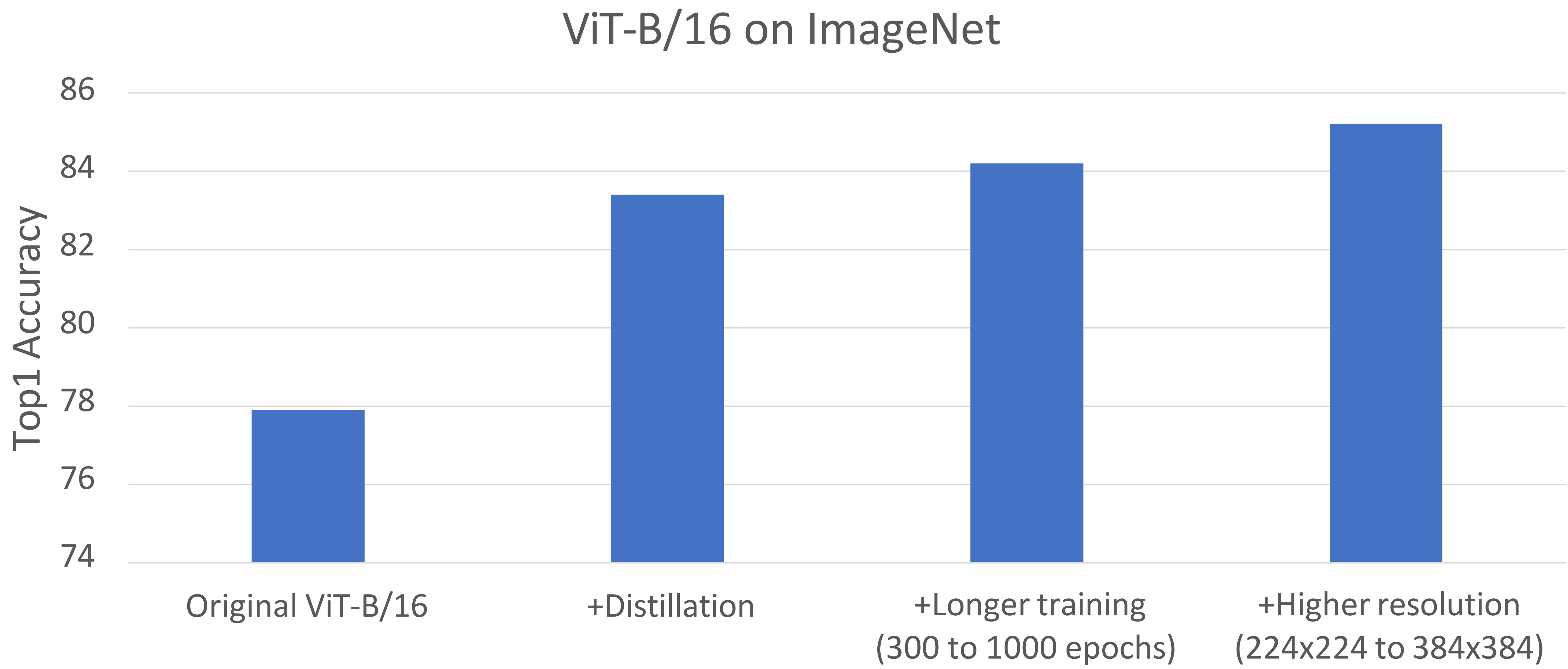
Touvrom et al, “Training data-efficient image transformers & distillation through attention”, ICML 2021

Improving ViT: Distillation



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

Improving ViT: Distillation



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

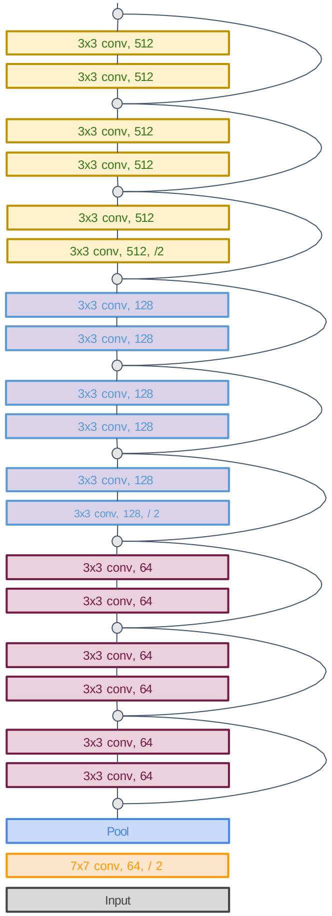
ViT vs CNN

Stage 3:
256 x 14 x 14

Stage 2:
128 x 28 x 28

Stage 1:
64 x 56 x 56

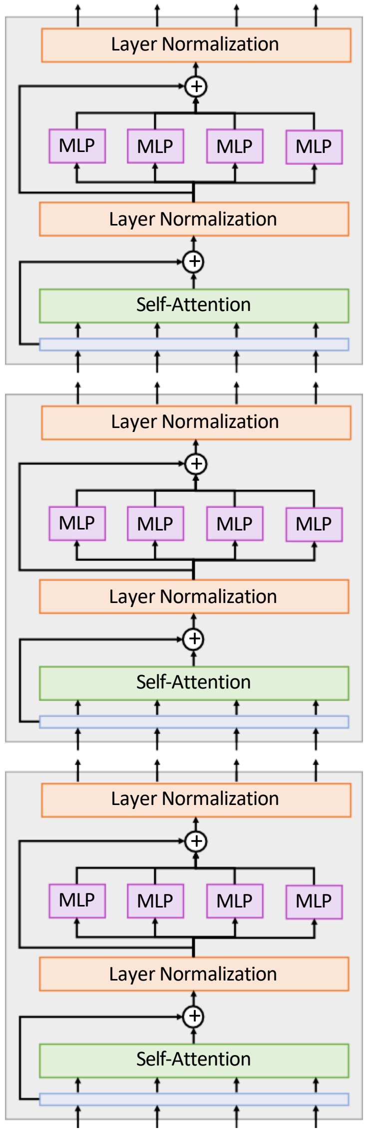
Input:
3 x 224 x 224



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

Useful since objects in images can occur at various scales

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



3rd block:
768 x 14 x 14

2nd block:
768 x 14 x 14

1st block:
768 x 14 x 14

Input:
3 x 224 x 224

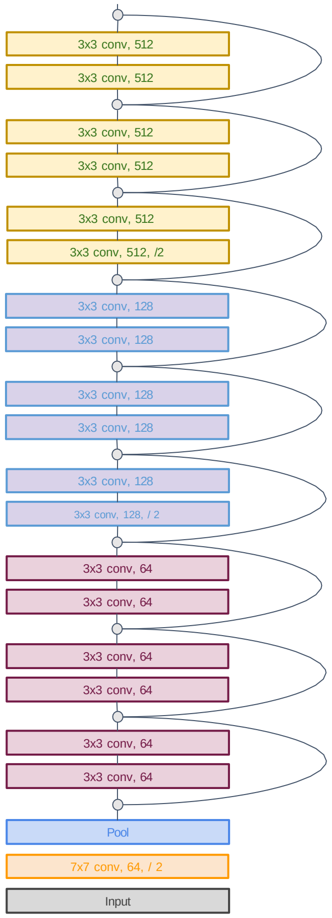
ViT vs CNN

Stage 3:
256 x 14 x 14

Stage 2:
128 x 28 x 28

Stage 1:
64 x 56 x 56

Input:
3 x 224 x 224

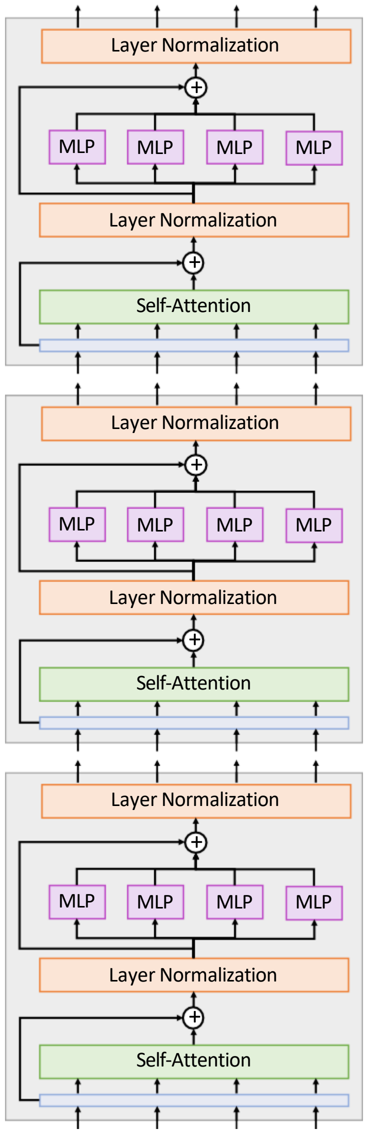


In most CNNs (including ResNets), **decrease** resolution and **increase** channels as you go deeper in the network (Hierarchical architecture)

Useful since objects in images can occur at various scales

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

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



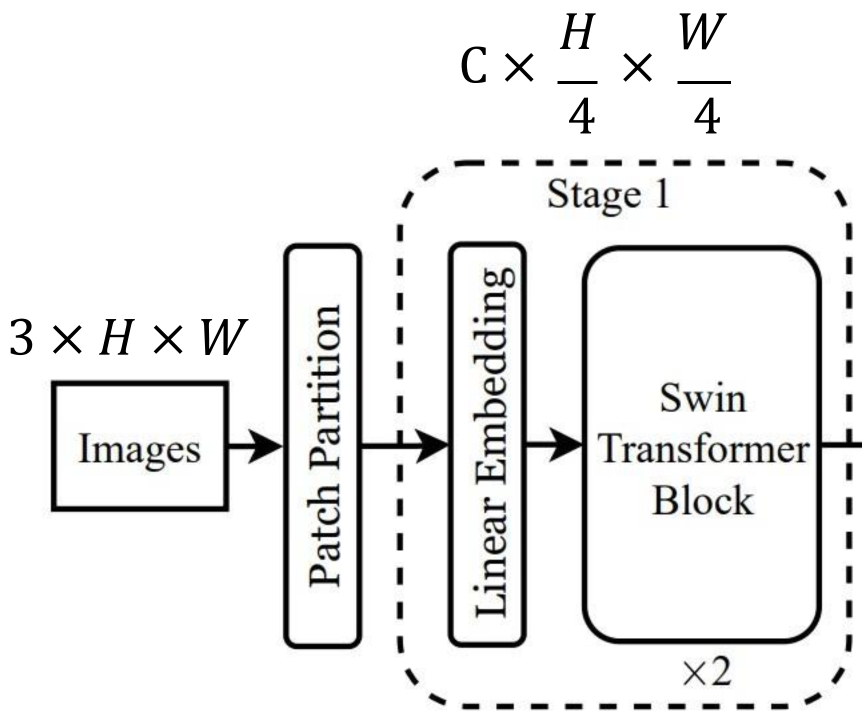
3rd block:
768 x 14 x 14

2nd block:
768 x 14 x 14

1st block:
768 x 14 x 14

Input:
3 x 224 x 224

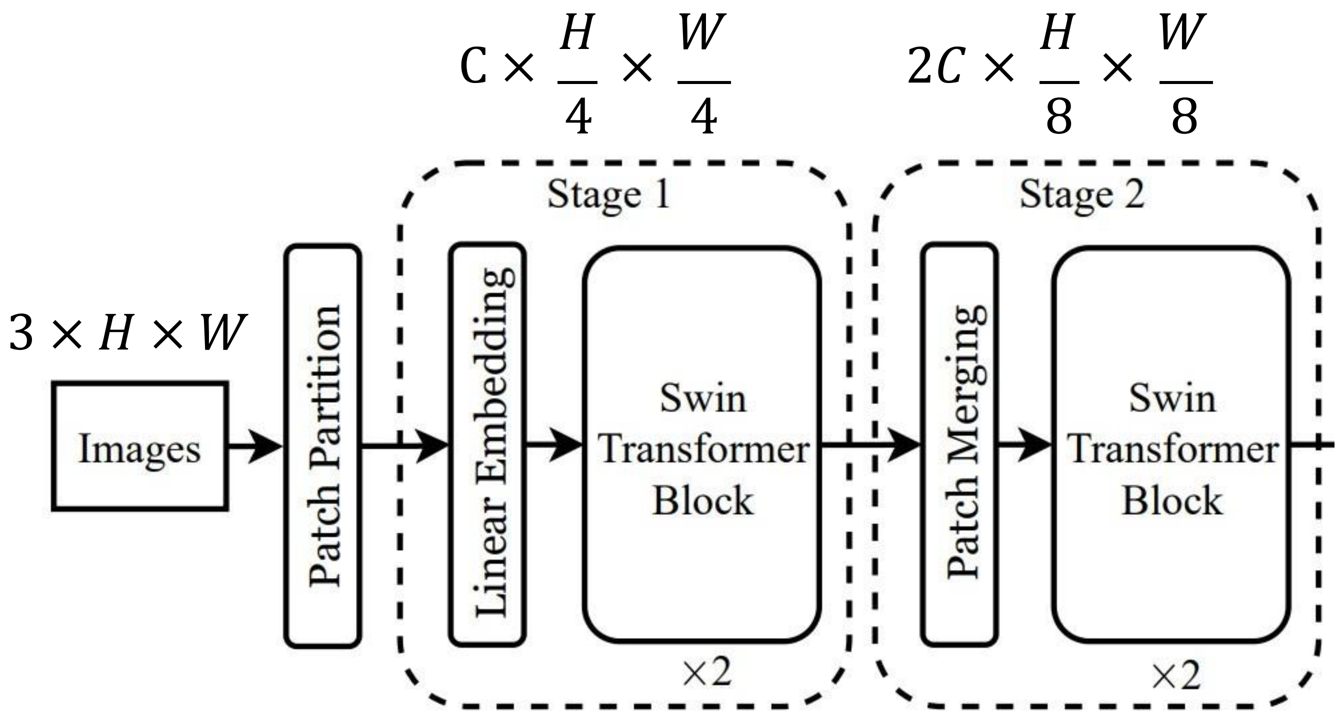
Hierarchical ViT: Swin Transformer



Divide image into 4x4 patches and project to C dimensions

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

Hierarchical ViT: Swin Transformer

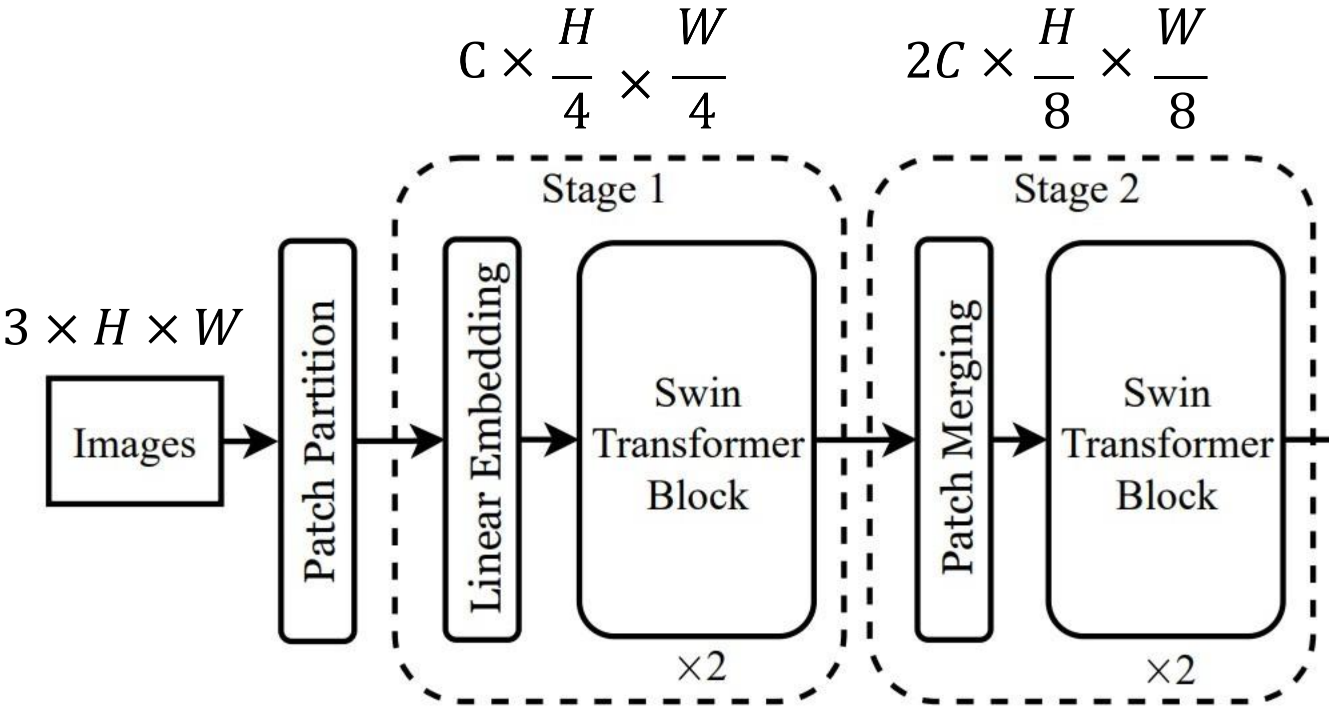


Divide image into 4x4 patches and project to C dimensions

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

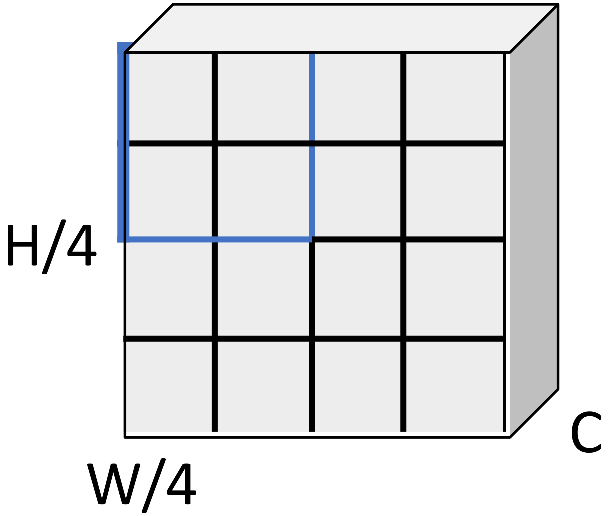
Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Hierarchical ViT: Swin Transformer



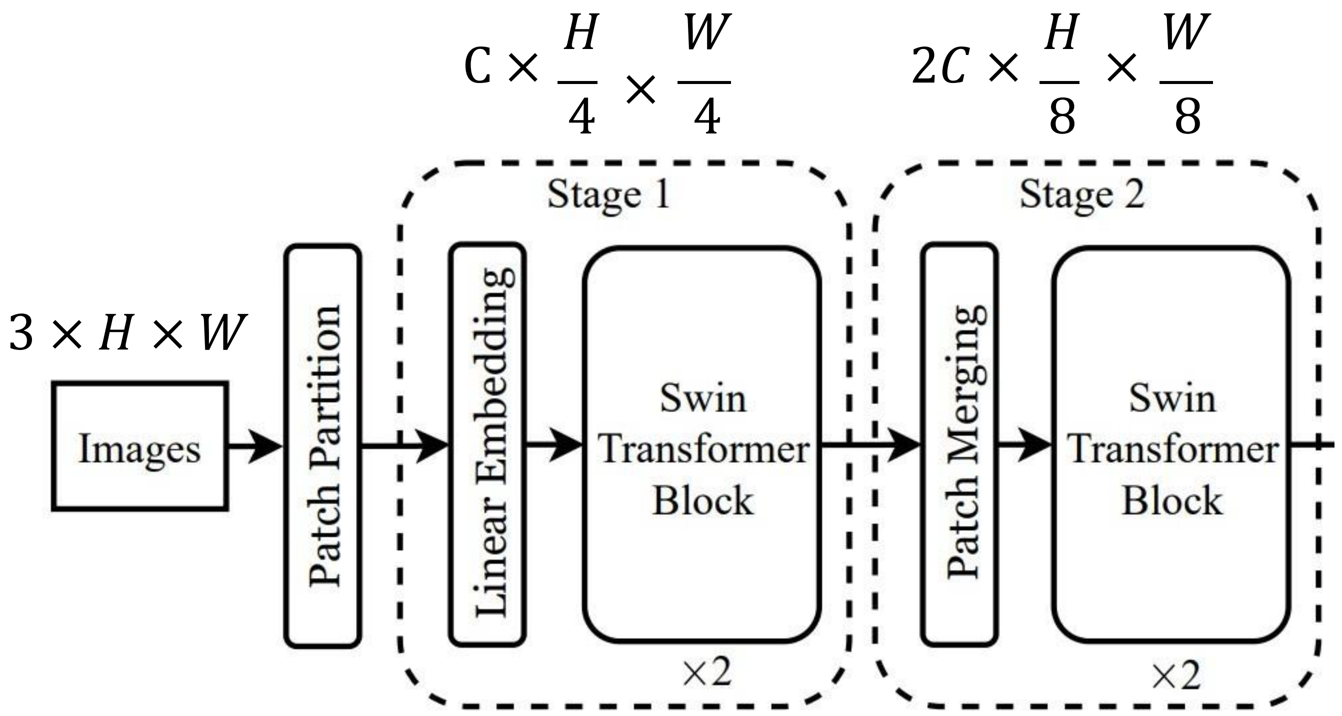
Divide image into 4x4 patches and project to C dimensions

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



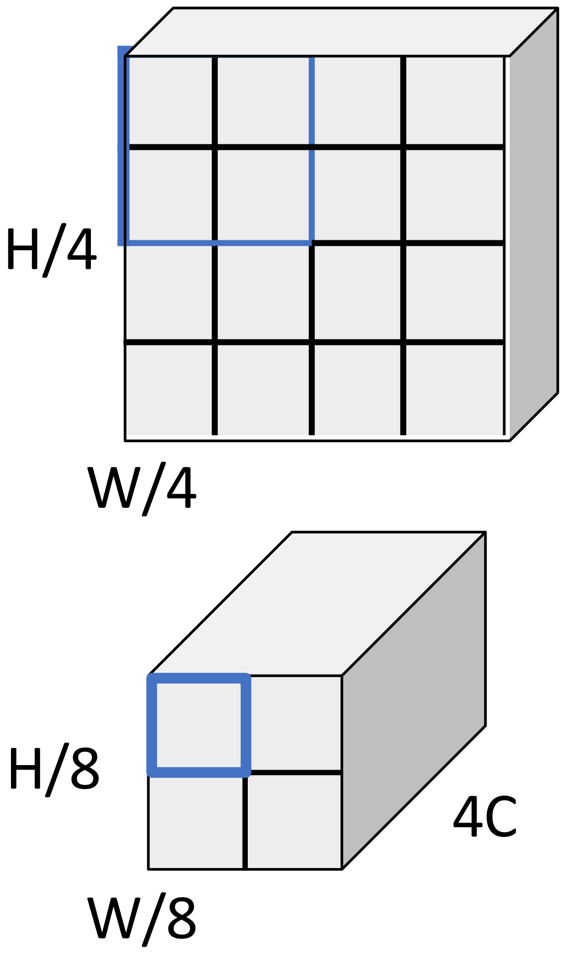
Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021

Hierarchical ViT: Swin Transformer



Divide image into 4x4 patches and project to C dimensions

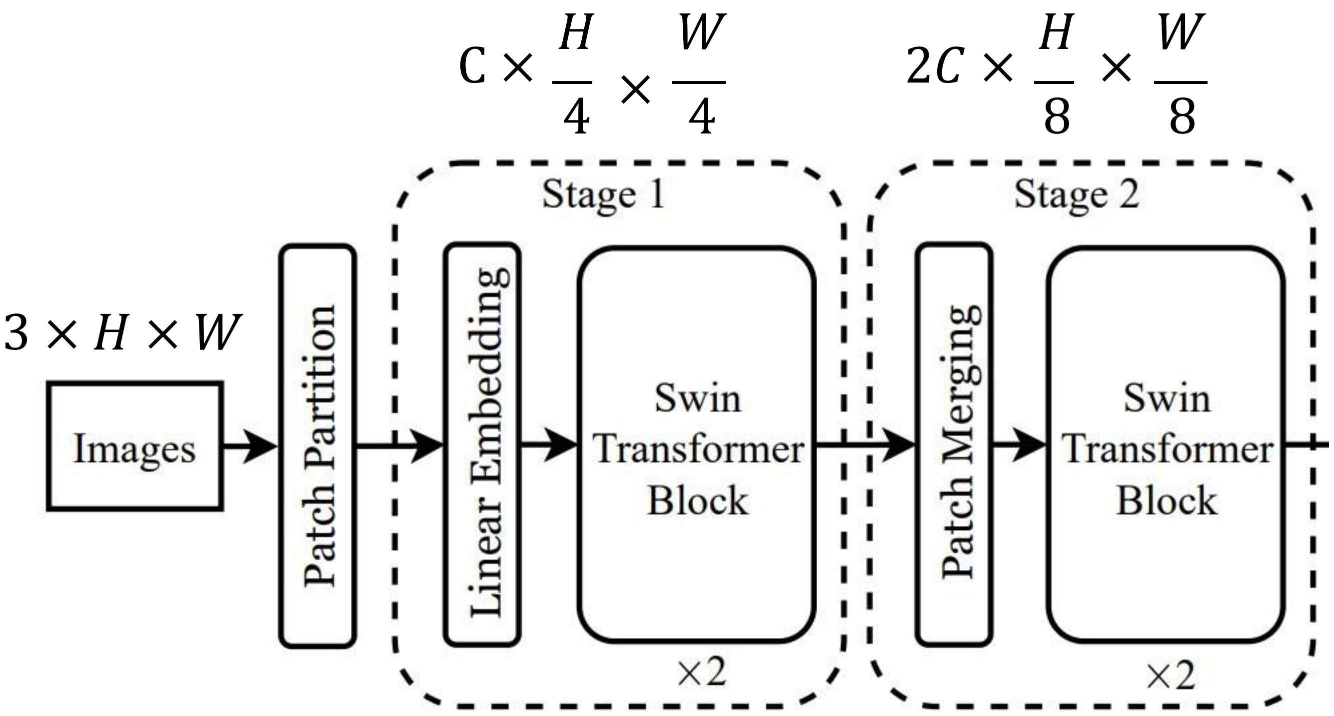
Merge 2x2 neighborhoods; now patches are (effectively) 8x8



Concatenate groups of 2x2 features

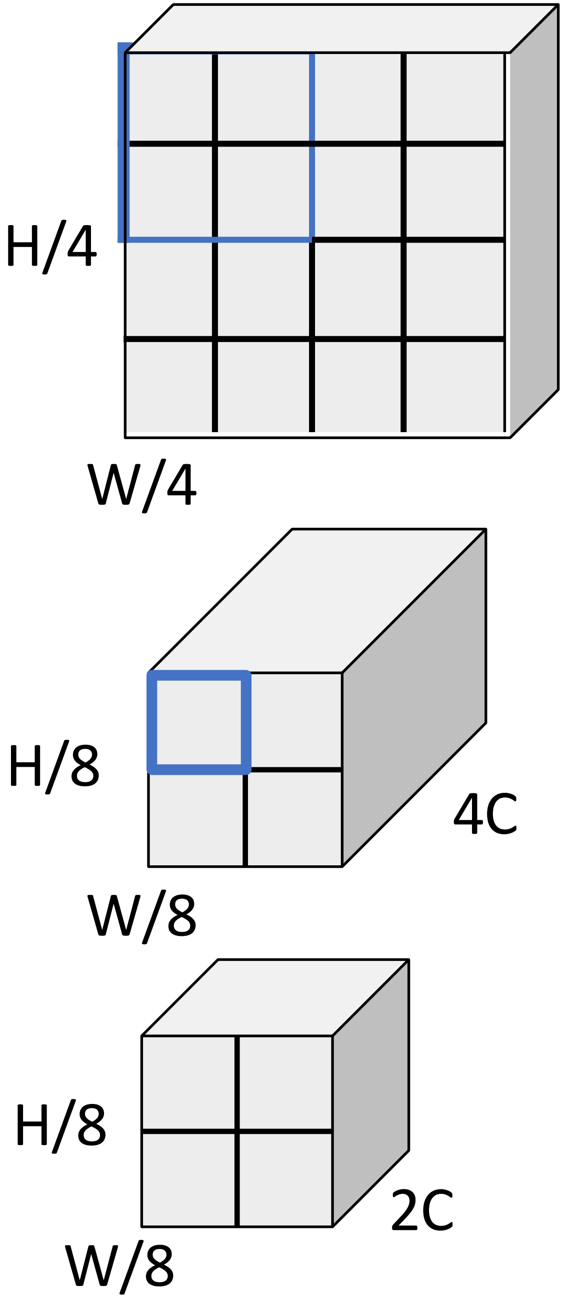
Liu et al, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”, CVPR 2021

Hierarchical ViT: Swin Transformer



Divide image into 4x4 patches and project to C dimensions

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

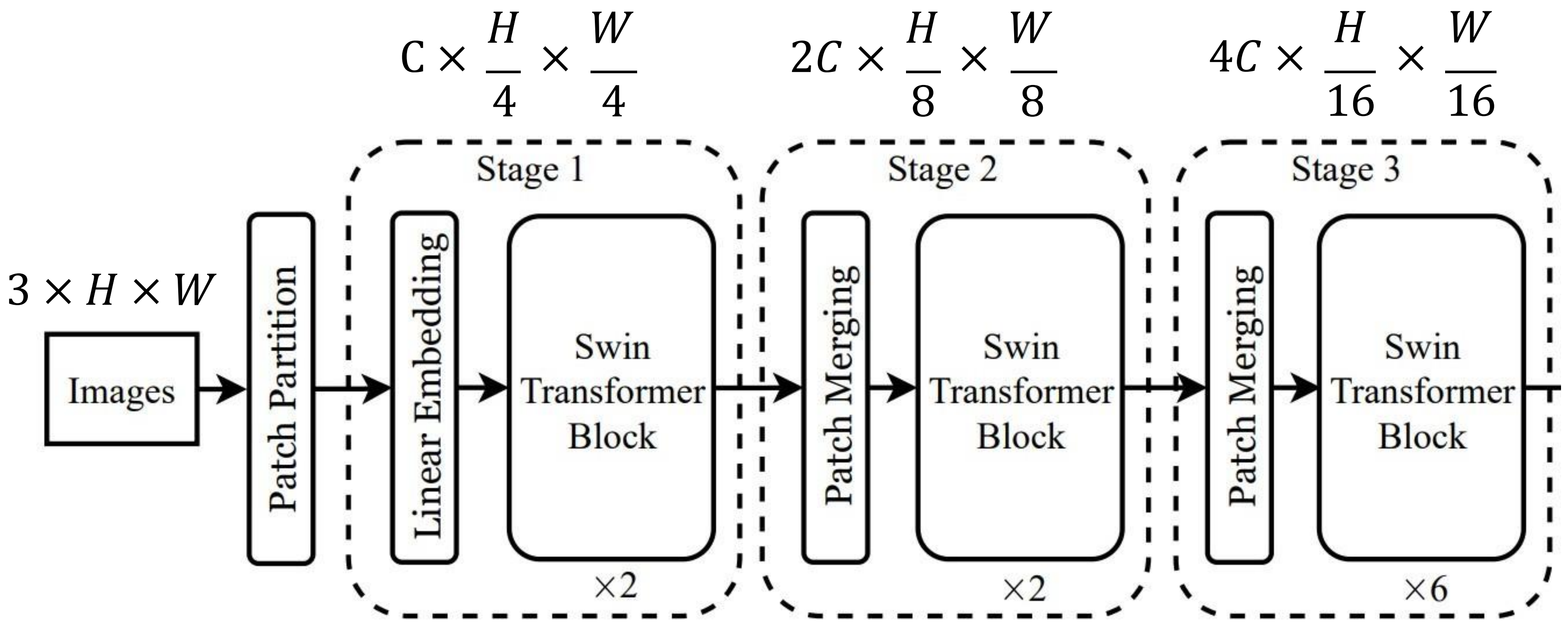


Concatenate groups of 2x2 features

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

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

Hierarchical ViT: Swin Transformer



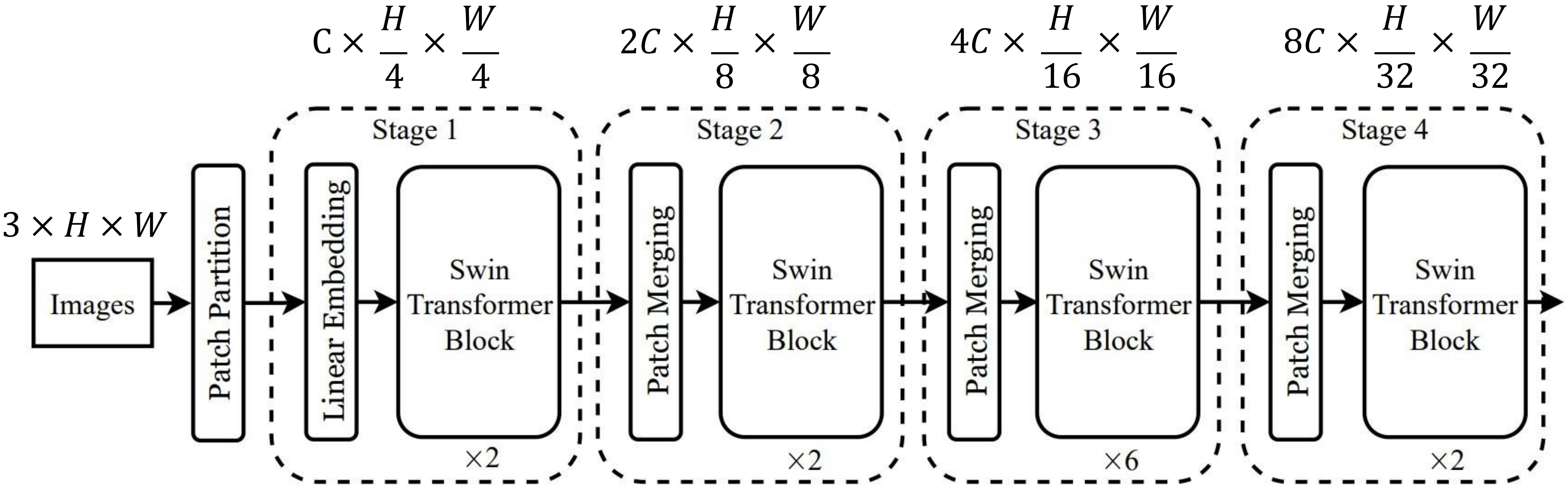
Divide image into 4x4 patches and project to C dimensions

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

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

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

Hierarchical ViT: Swin Transformer



Divide image into 4x4 patches and project to C dimensions

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

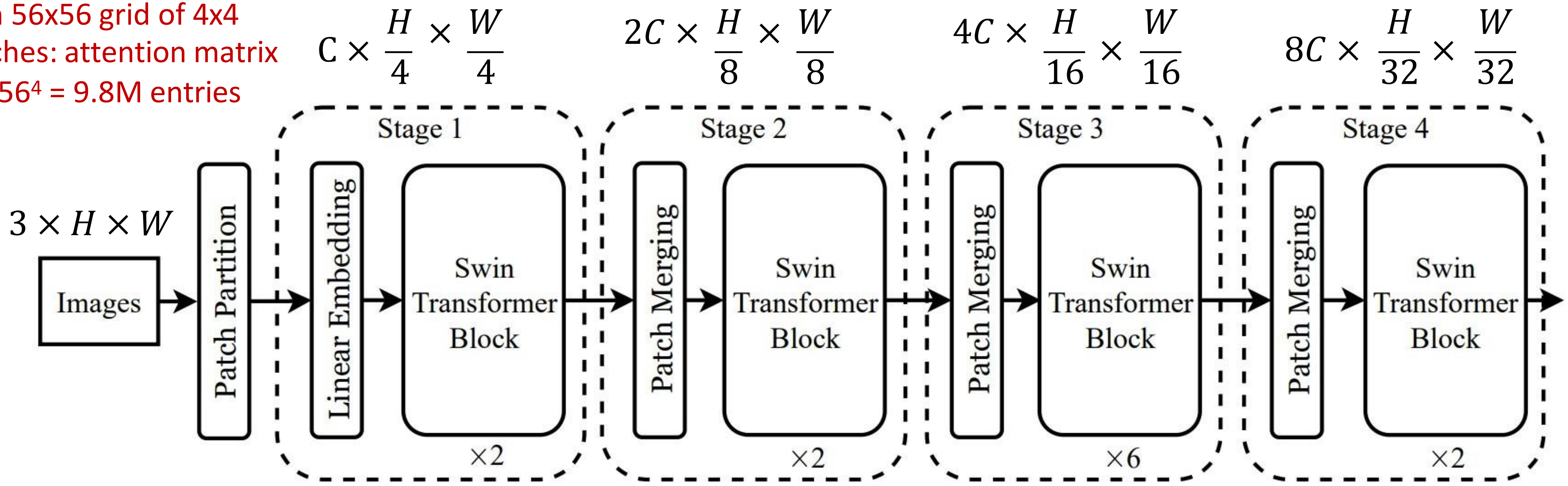
Merge 2x2 neighborhoods; now patches are (effectively) 16x16

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

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

Hierarchical ViT: Swin Transformer

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



Divide image into 4x4 patches and project to C dimensions

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

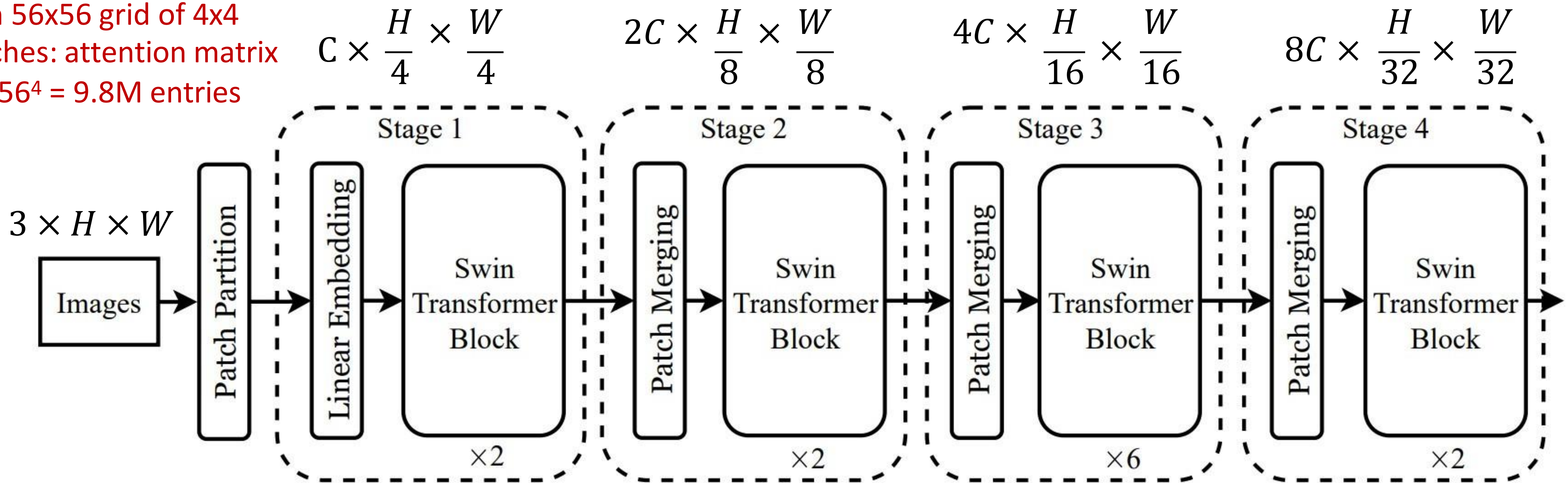
Merge 2x2 neighborhoods; now patches are (effectively) 16x16

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

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

Hierarchical ViT: Swin Transformer

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



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

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

Swin Transformer: Window Attention

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

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

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

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

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

Swin Transformer: Window Attention

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

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

Swin Transformer: Shifted Window Attention

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

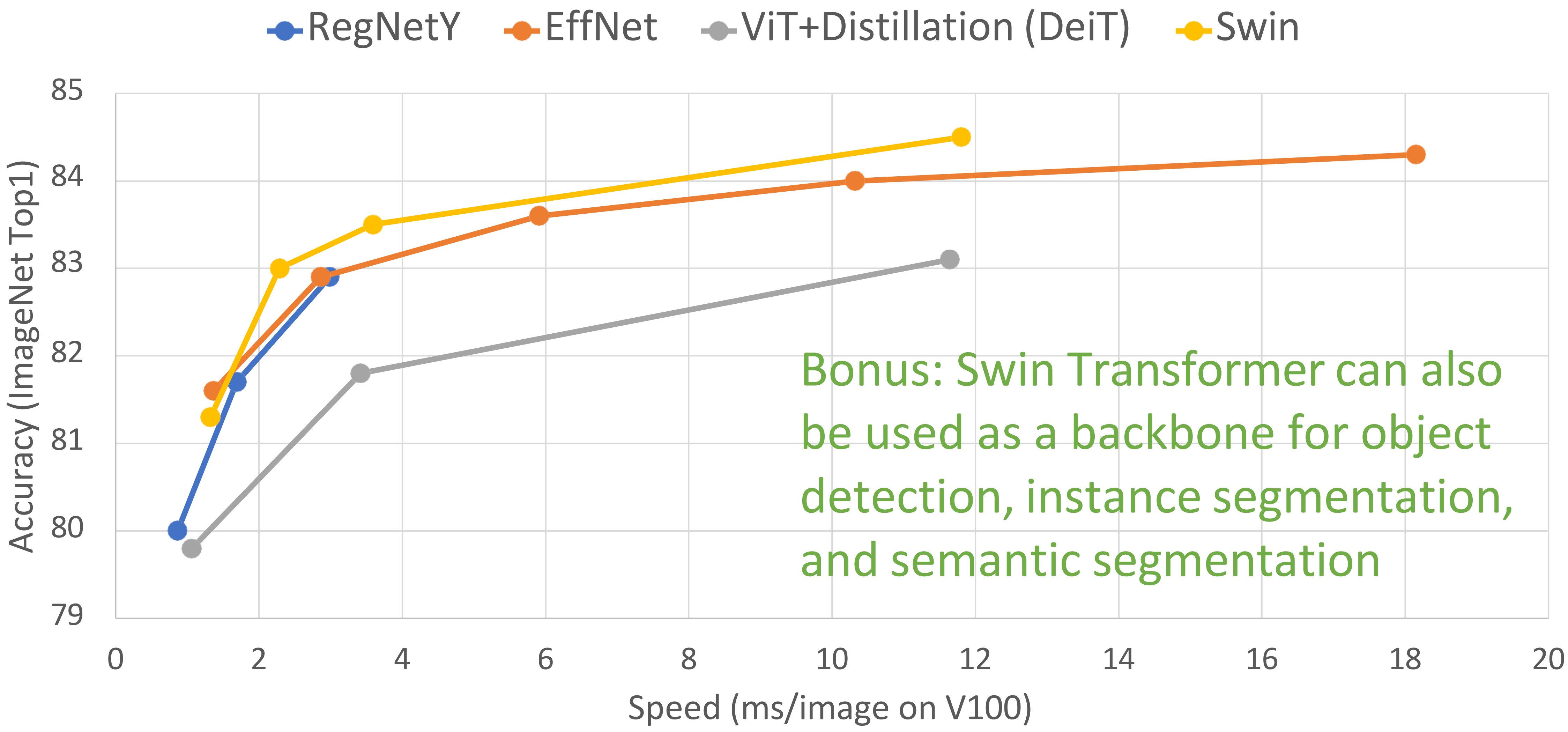
Block L: Normal windows

Block L+1: Shifted Windows

Ugly detail: Non-square windows at edges and corners

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

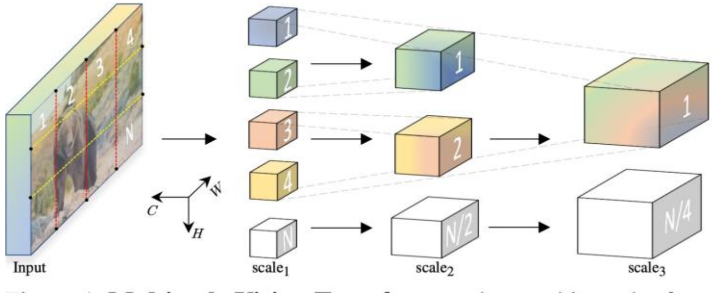
Swin Transformer: Speed vs Accuracy



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

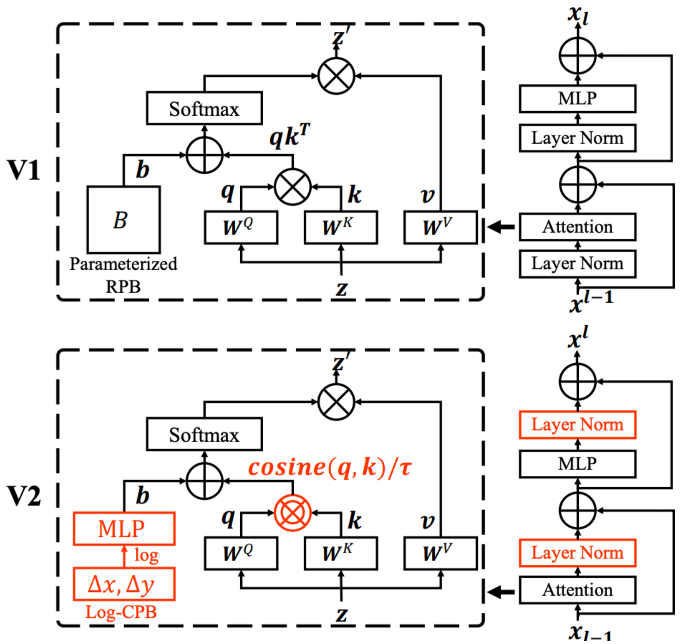
Other Hierarchical Vision Transformers

MViT



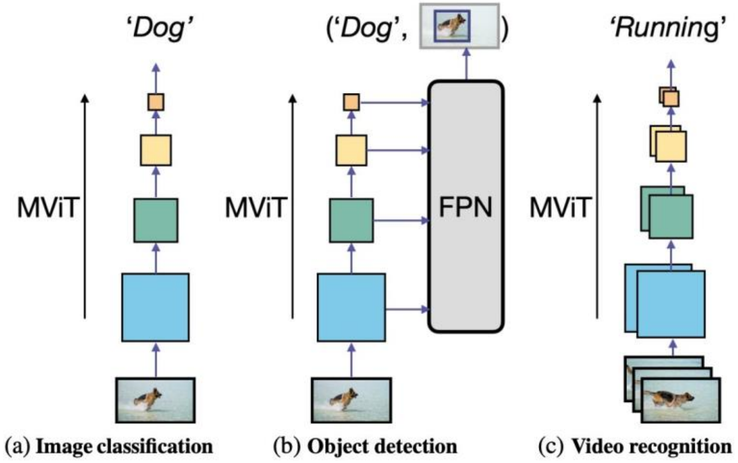
Fan et al, "Multiscale Vision Transformers", ICCV 2021

Swin-V2



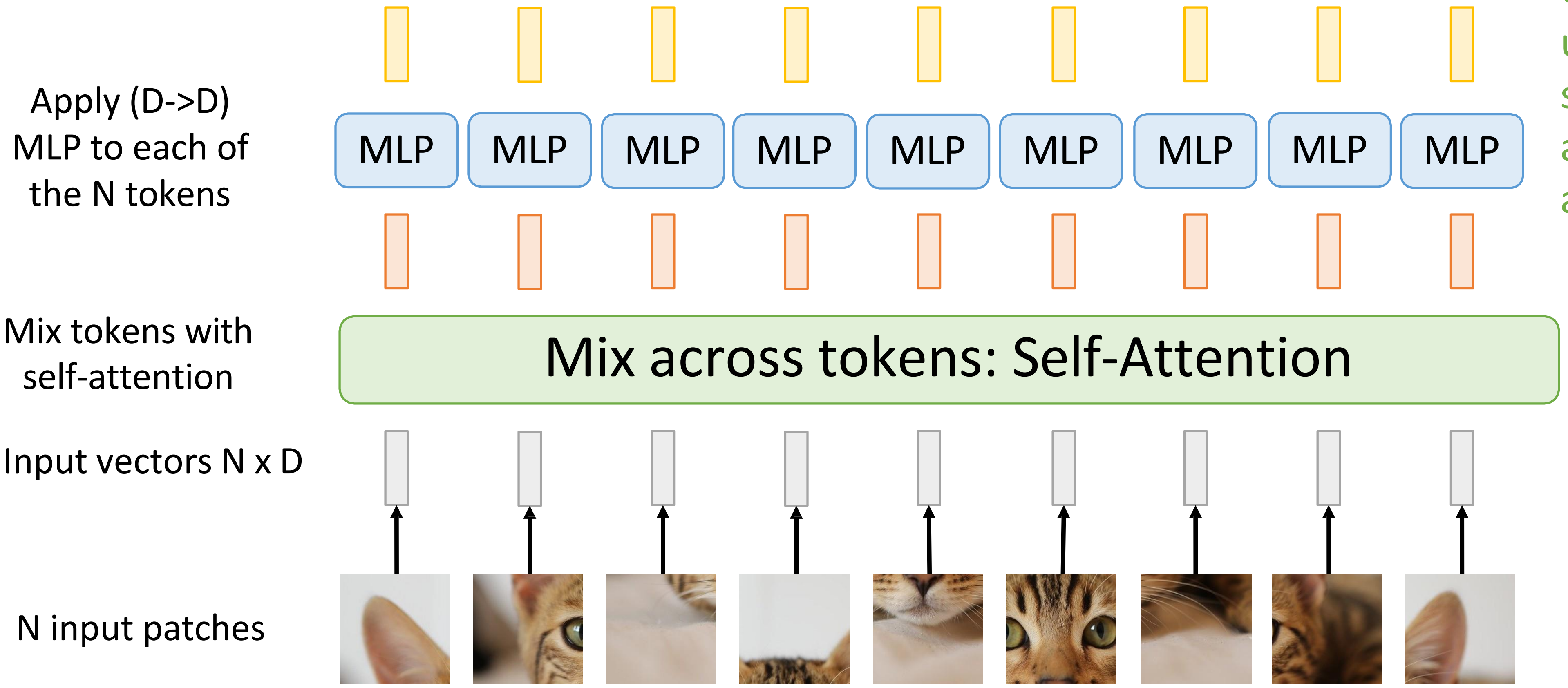
Liu et al, "Swin Transformer V2: Scaling up Capacity and Resolution", CVPR 2022

Improved MViT



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

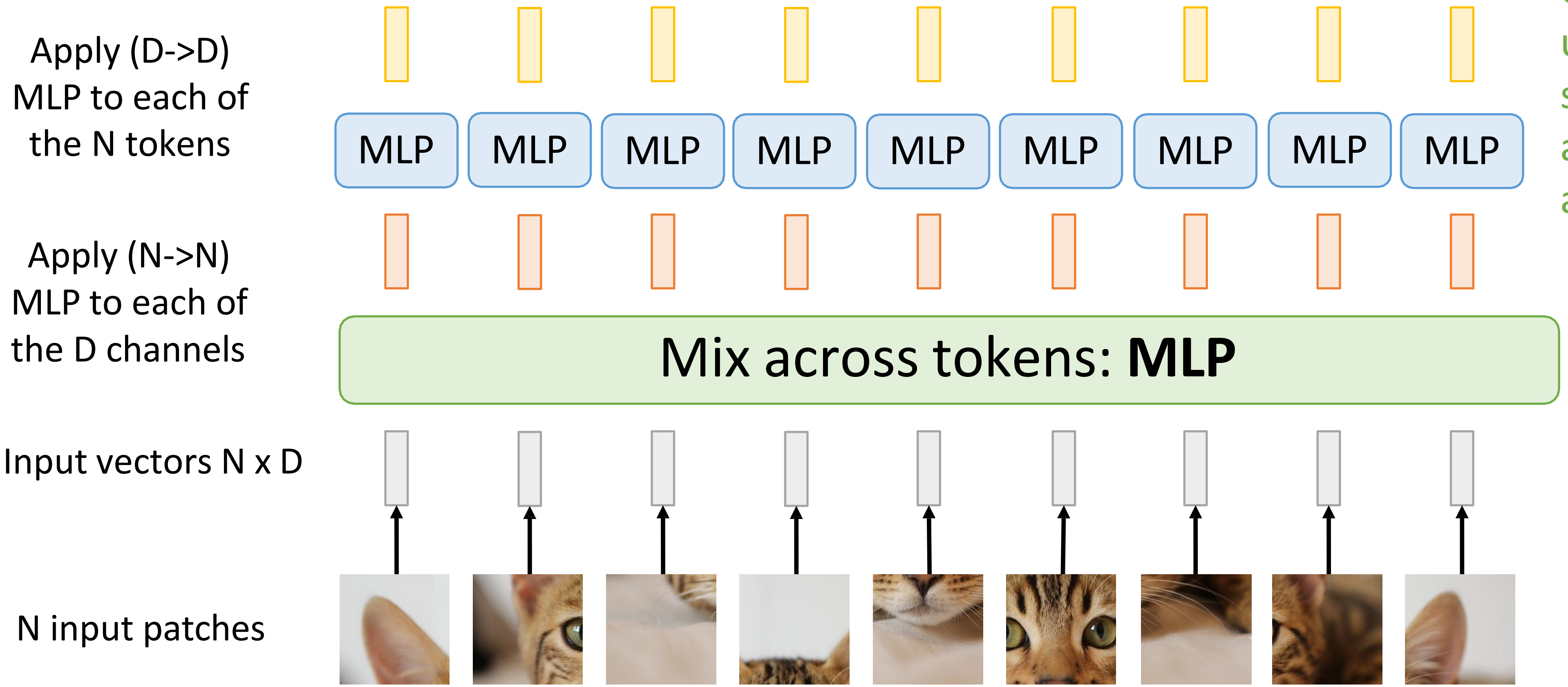
Vision Transformer: Another Look



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

MLP-Mixer: An All-MLP Architecture



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

Tolstikhin et al, "MLP-Mixer: An all-MLP architecture for vision", NeurIPS 2021

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

Data Modalities

- ✓ Language
- ✓ Vision
- Audio
- … and many other modalities (e.g., biological/physiological signals, etc.)
- Multimodal (>2 data modalities)

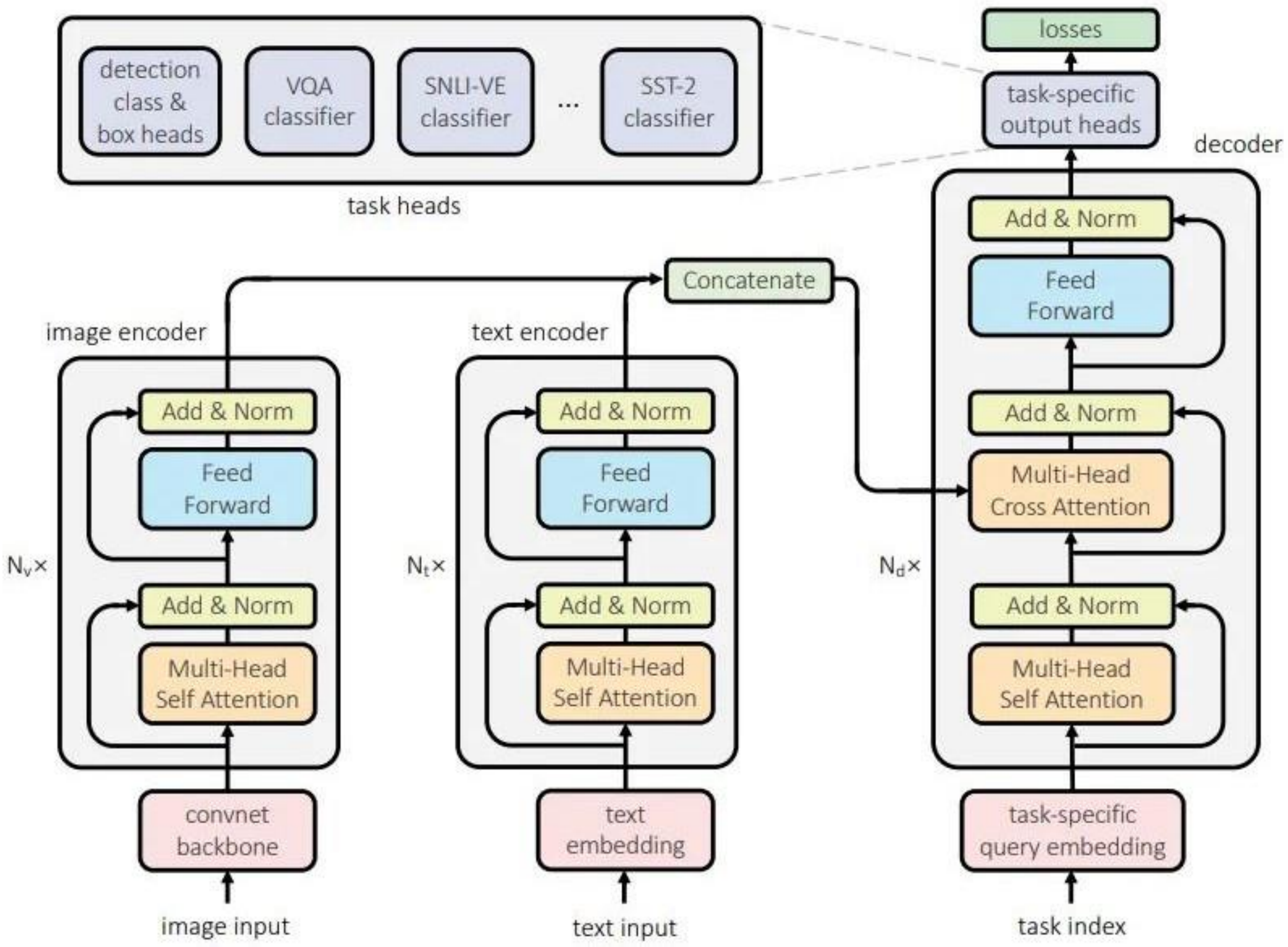
Audio

- Similar to the computer vision but with spectrograms instead of images.
- Exists as encoder-decoder variants or as an encoder-only variant with CTC loss.
- Could be augmented with the CNN.

Conformer: Convolution-augmented Transformer for Speech Recognition

AST: Audio Spectrogram Transformer

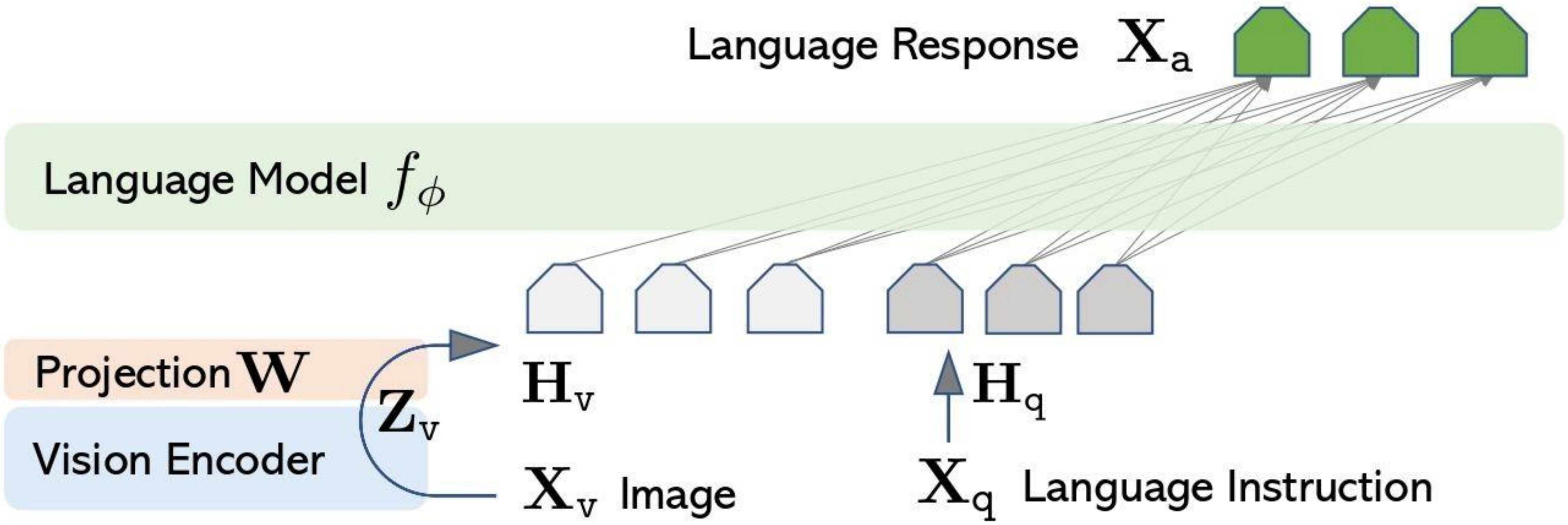
Multimodal Transformer - UniT



1. UniT handles 7 tasks ranging from object detection to vision-and language reasoning and natural language understanding.
2. Components:
   - An image encoder to encode the visual inputs.
   - A text encoder to encode the language inputs.
   - A joint decoder with per-task query embedding.
   - Task-specific heads to make the final outputs for each task.

[UniT: Multimodal Multitask Learning with a Unified Transformer]()

Multimodal Transformer - LLaVA



[Visual Instruction Tuning (LLaVA - Large Language and Vision Assistant)]

Multimodal Transformer - LLaVA

Start a new conversation, and the history is cleared.

User

Do you know who drew this painting?

LLaVA

The painting depicts a dog in a humorous situation, where it appears to be wearing a woman's dress and standing in front of a landscape. Although the artist of the painting is not explicitly identified, it could be a reference to the famous artist Da Vinci, known for his artistic works, including the Mona Lisa. The painting is likely a creative and comical take on the traditional portrait style, replacing a human figure with a dog dressed in a woman's clothing, making it an interesting and amusing visual piece.

[Visual Instruction Tuning (LLaVA - Large Language and Vision Assistant)]