How to use Attention / Transformers for Vision?

Last lecture

- Transformers – encoder and decoder blocks
- SA, Multihead SA, Masked SA

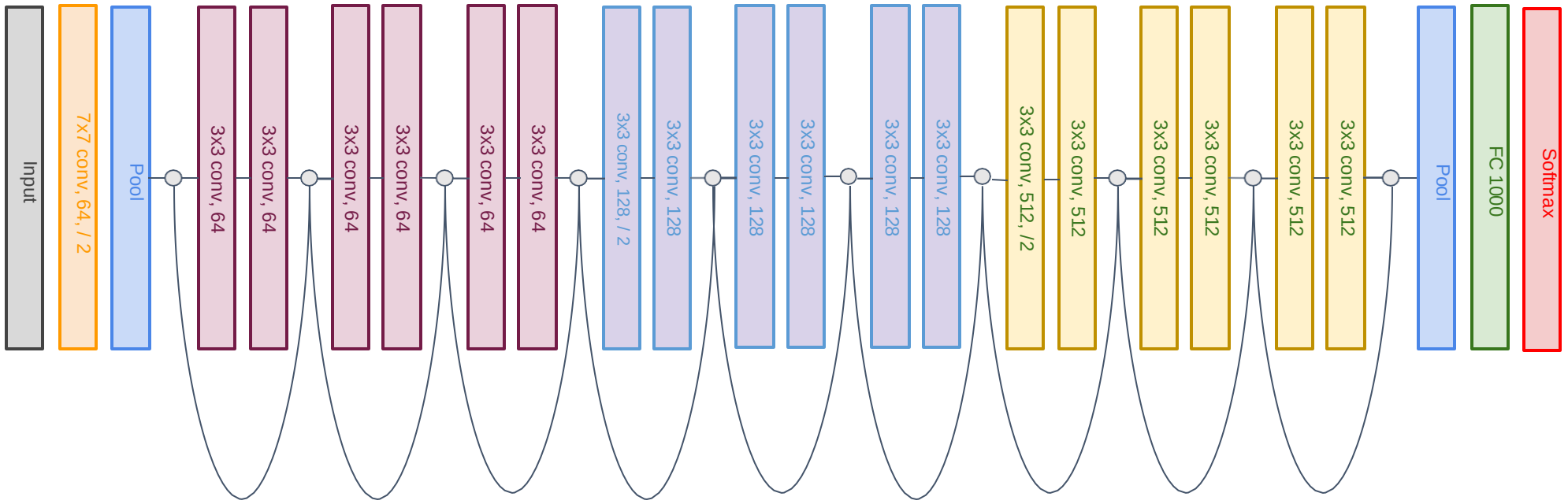
This lecture

- ViT, SwinT
- Improving ViT training using distillation and augmentation/regularization

Idea #1: Add attention to existing CNNs

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

How can you incorporate SA?

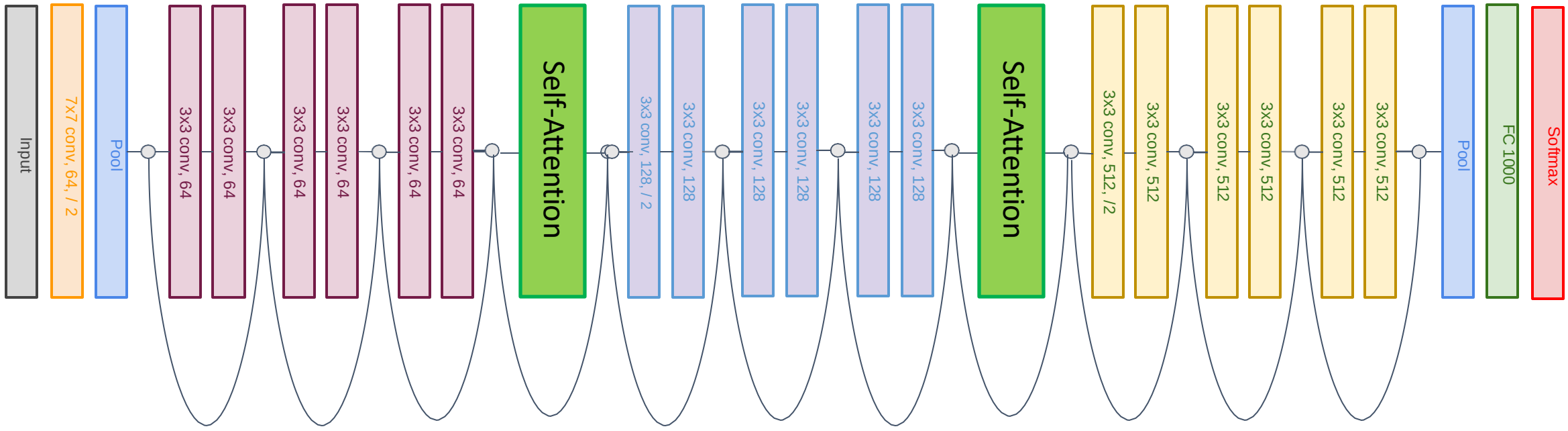


Zhang et al, "Self-Attention Generative Adversarial Networks", ICML 2018
Wang et al, "Non-local Neural Networks", CVPR 2018

Idea #1: Add attention to existing CNNs

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

Add Self-Attention blocks between existing ResNet blocks



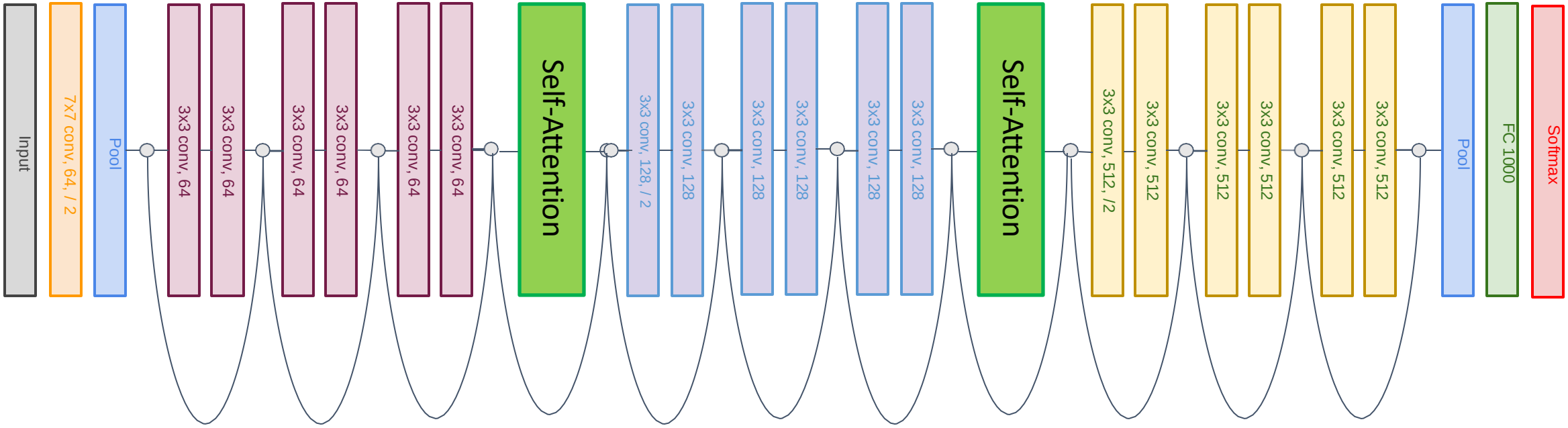
Zhang et al, "Self-Attention Generative Adversarial Networks", ICML 2018
Wang et al, "Non-local Neural Networks", CVPR 2018

Idea #1: Add attention to existing CNNs

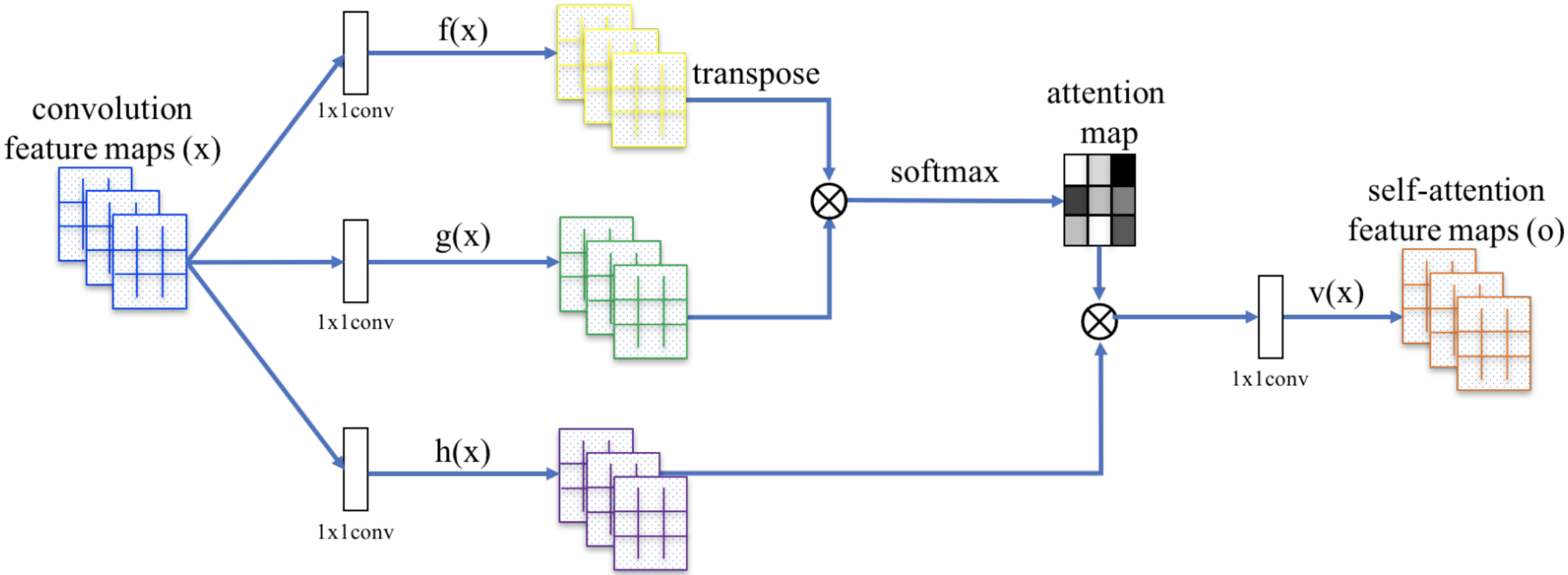
Model is still a CNN!
Can we replace convolution entirely?

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

Add Self-Attention blocks between existing ResNet blocks

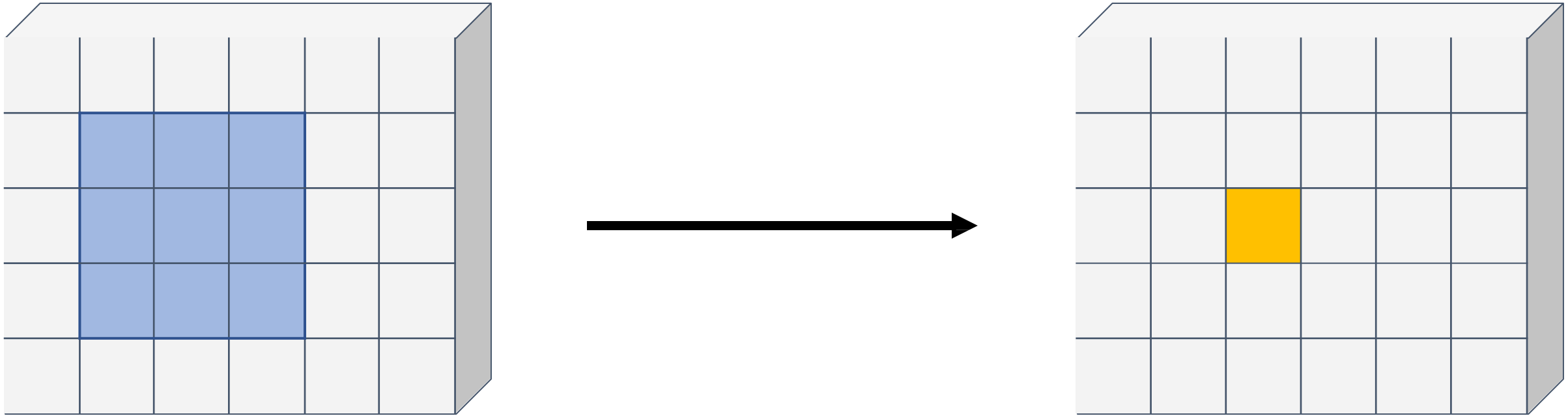


Zhang et al, "Self-Attention Generative Adversarial Networks", ICML 2018
Wang et al, "Non-local Neural Networks", CVPR 2018


convolution
feature maps (x)
1x1conv
f(x)
transpose
1x1conv
g(x)
1x1conv
h(x)
softmax
attention
map
1x1conv
v(x)
self-attention
feature maps (o)

Idea #2: Replace Convolution with “Local Attention”

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



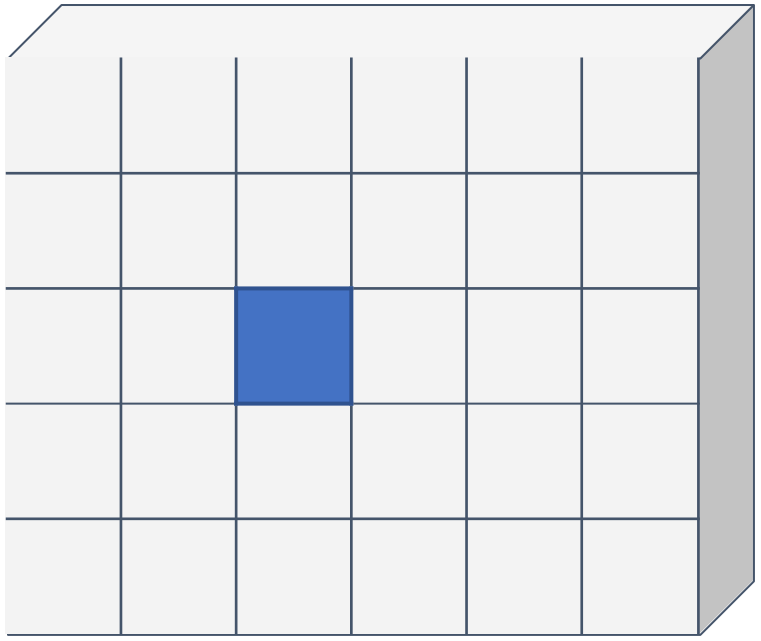
Input: C x H x W

Output: C’ x H x W

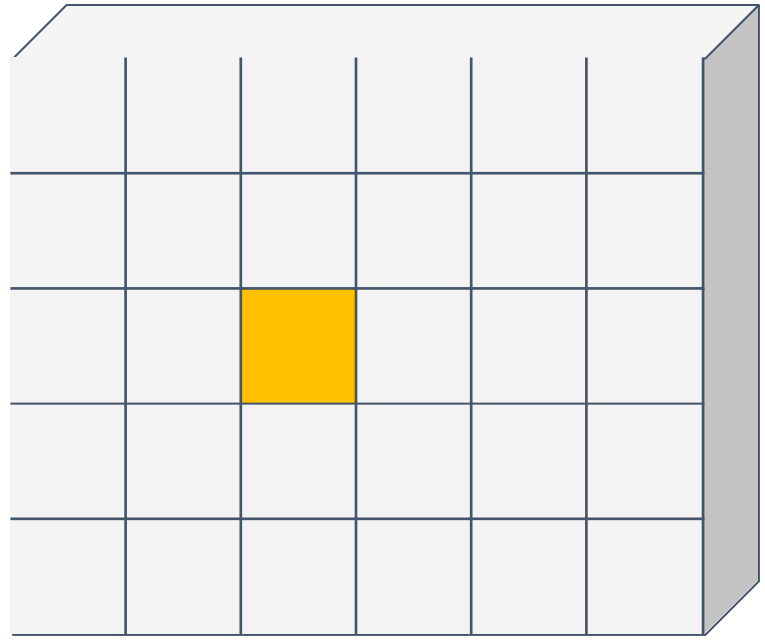
Hu et al, “Local Relation Networks for Image Recognition”, ICCV 2019; Ramachandran et al, “Stand-Alone Self-Attention in Vision Models”, NeurIPS 2019

Idea #2: Replace Convolution with “Local Attention”

Map center of receptive field to **query**



Query: D_Q



Input: C x H x W

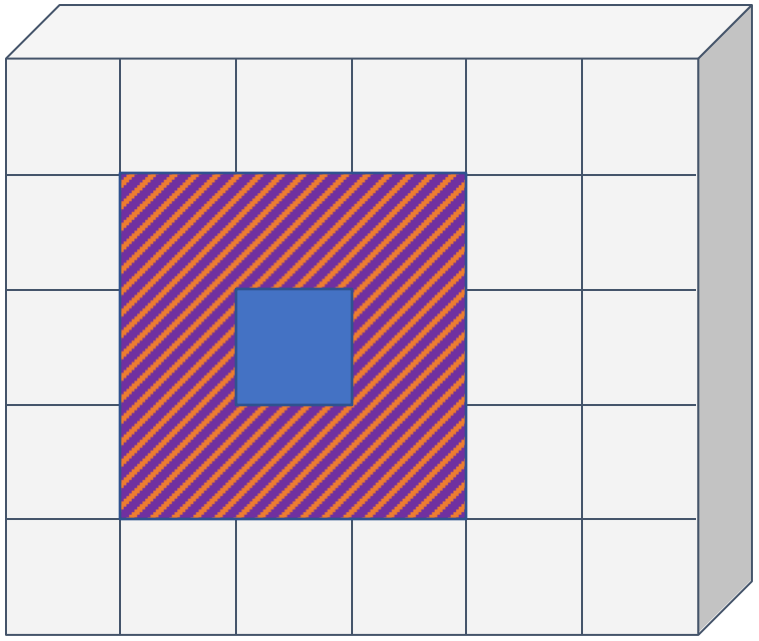
Output: C’ x H x W

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**

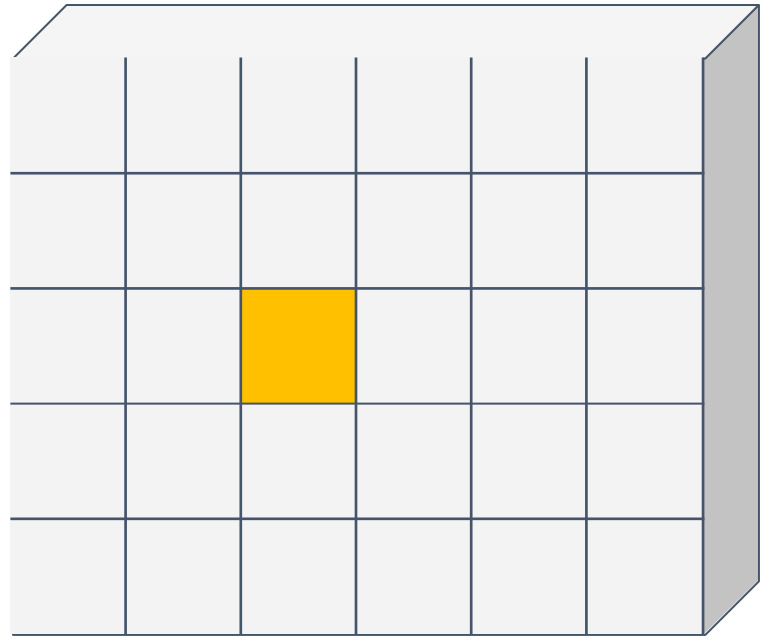


Input: C x H x W

Query: D_Q

Keys: R x R x D_Q

Values: R x R x C'



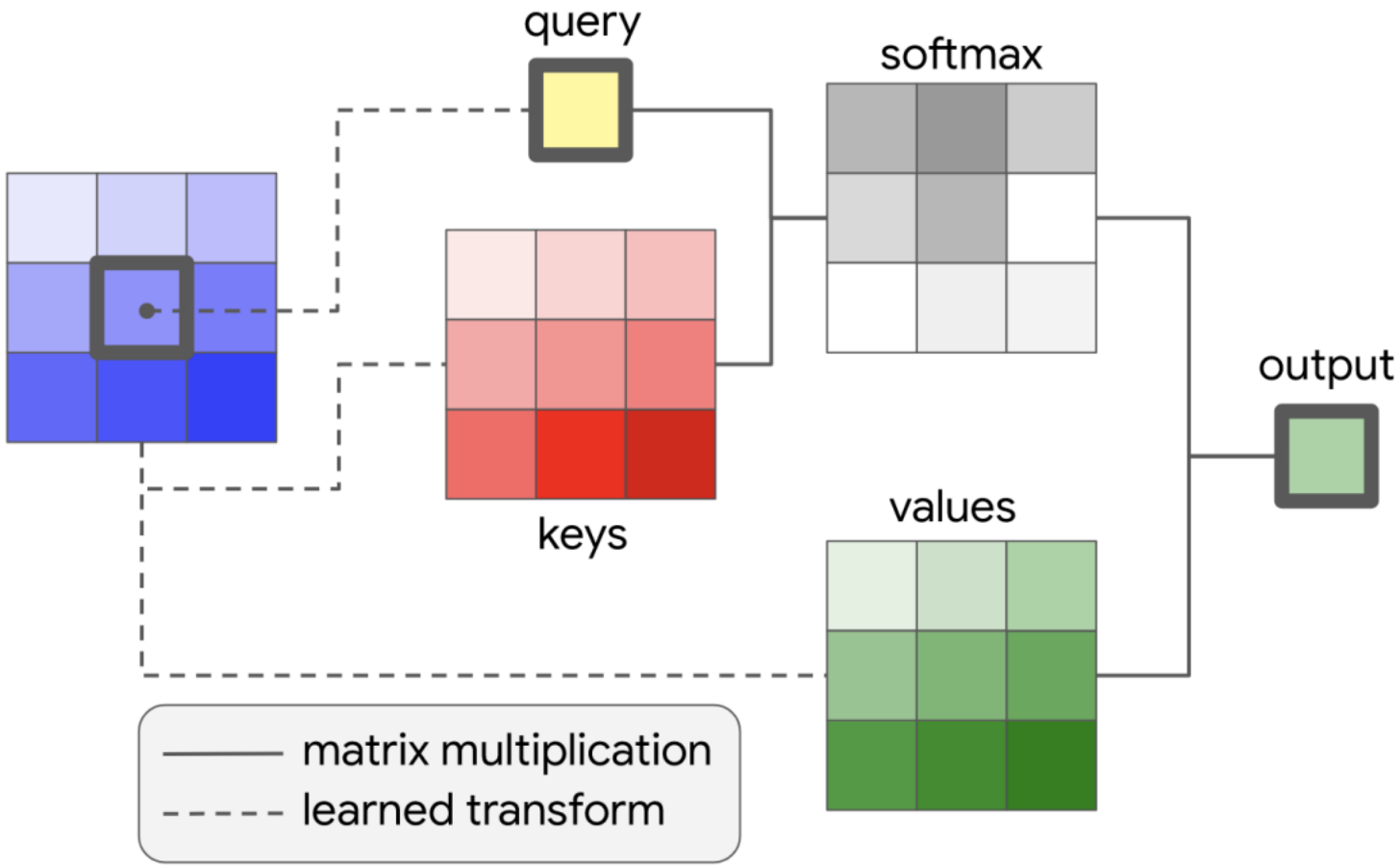
Output: C' x H x W

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**



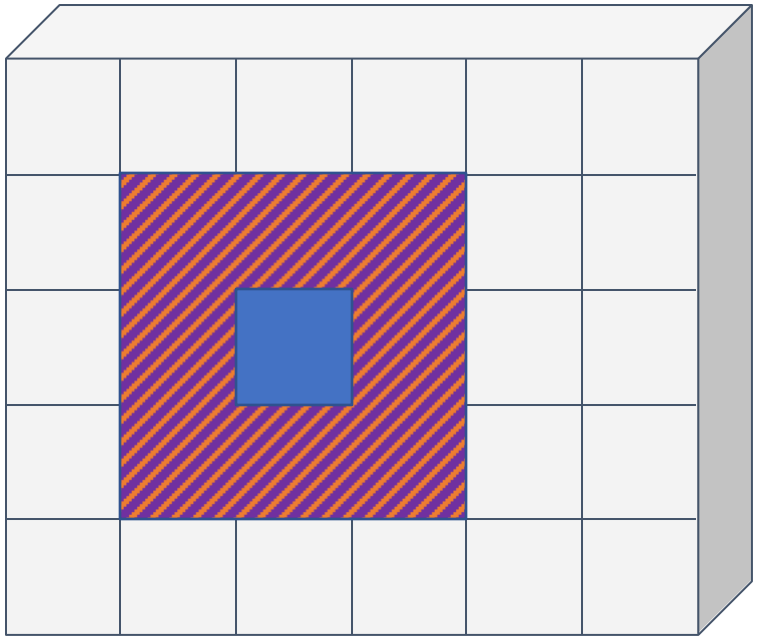
Ramachandran et al, "Stand-Alone Self-Attention in Vision Models", NeurIPS 2019

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 x H x W

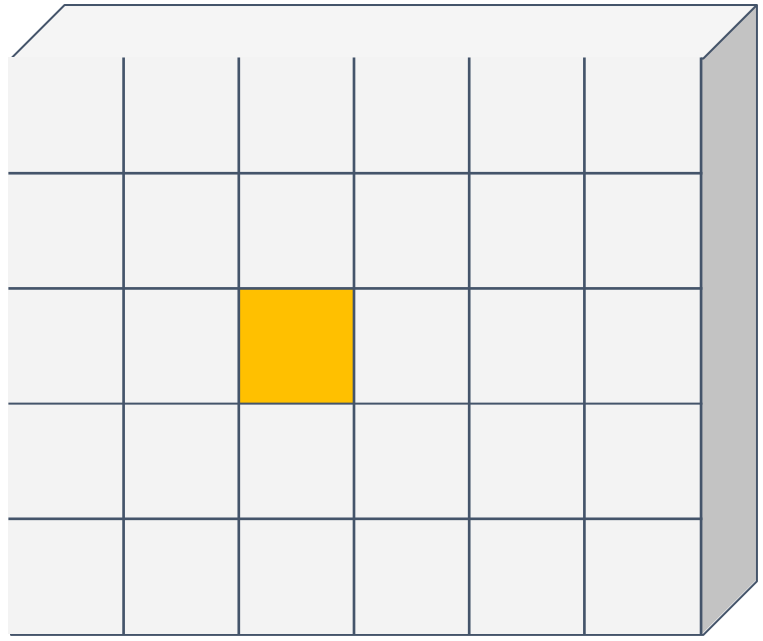
Query: D_Q

Keys: R x R x D_Q

Values: R x R x C'

Output: C

Attention



Output: C' x H x 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 (7×7, m=8)**
res1	112×112	7×7 conv, 64, stride 2	**1×1, 64** **7×7 LR, 64, stride 2**
res2	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		[1×1, 64 3×3 conv, 64 1×1, 256] ×3	[1×1, 100 **7×7 LR, 100** 1×1, 256] ×3
res3	28×28	[1×1, 128 3×3 conv, 128 1×1, 512] ×4	[1×1, 200 **7×7 LR, 200** 1×1, 512] ×4
res4	14×14	[1×1, 256 3×3 conv, 256 1×1, 1024] ×6	[1×1, 400 **7×7 LR, 400** 1×1, 1024] ×6
res5	7×7	[1×1, 512 3×3 conv, 512 1×1, 2048] ×3	[1×1, 800 **7×7 LR, 800** 1×1, 2048] ×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"

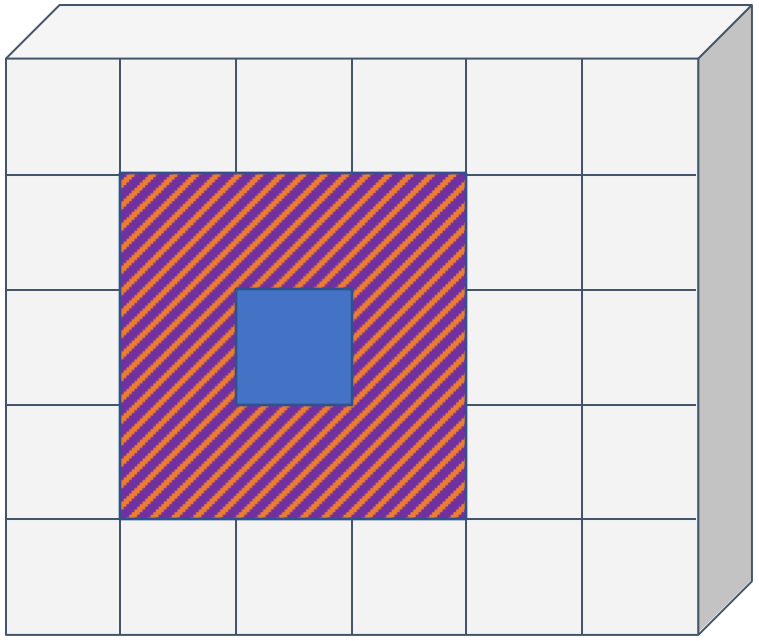
Map center of receptive field to **query**

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 x H x W

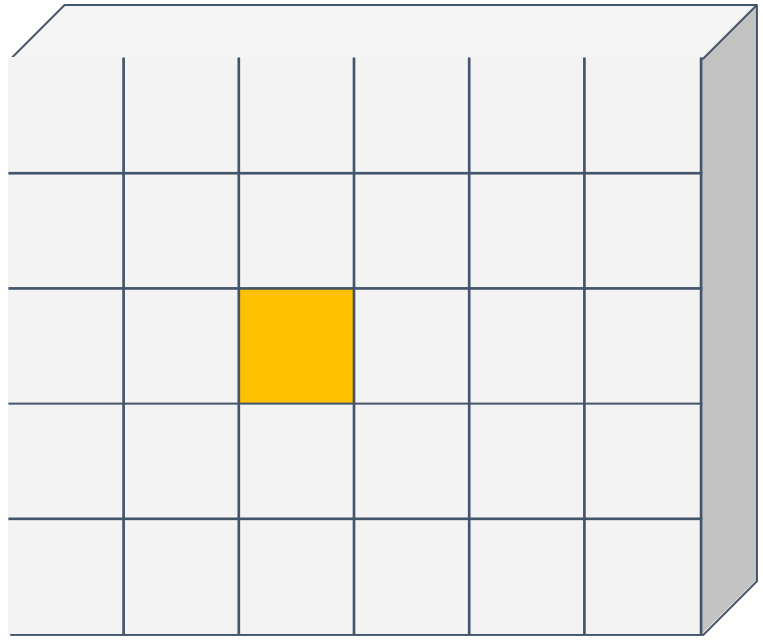
Query: D_Q

Keys: R x R x D_Q

Values: R x R x C'

Output: C

Attention

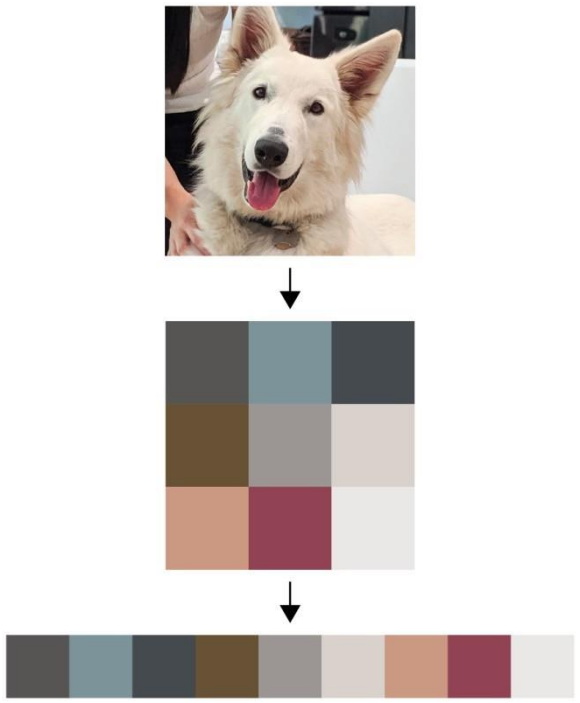


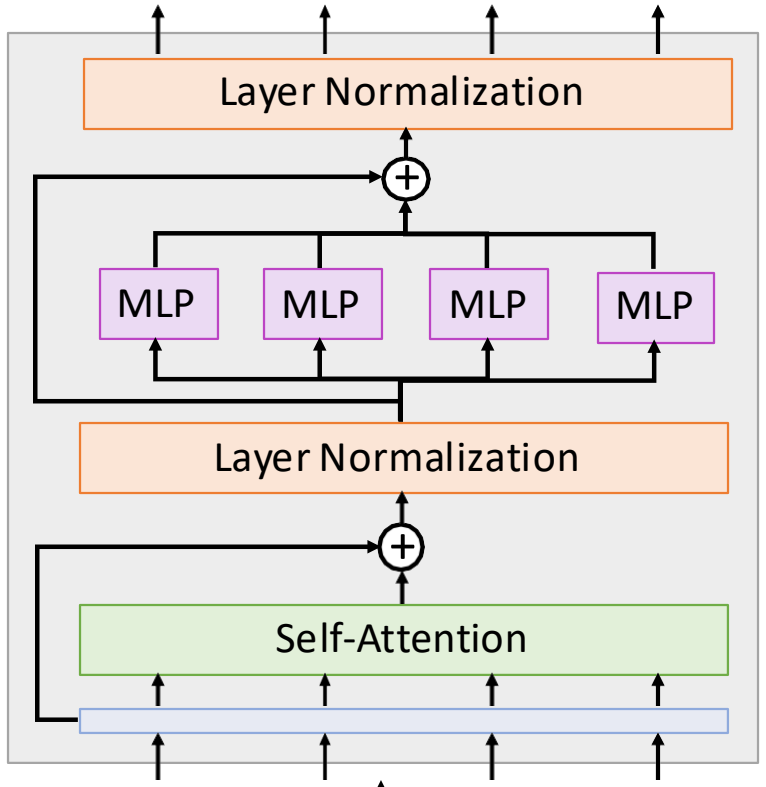
Output: C' x H x W

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

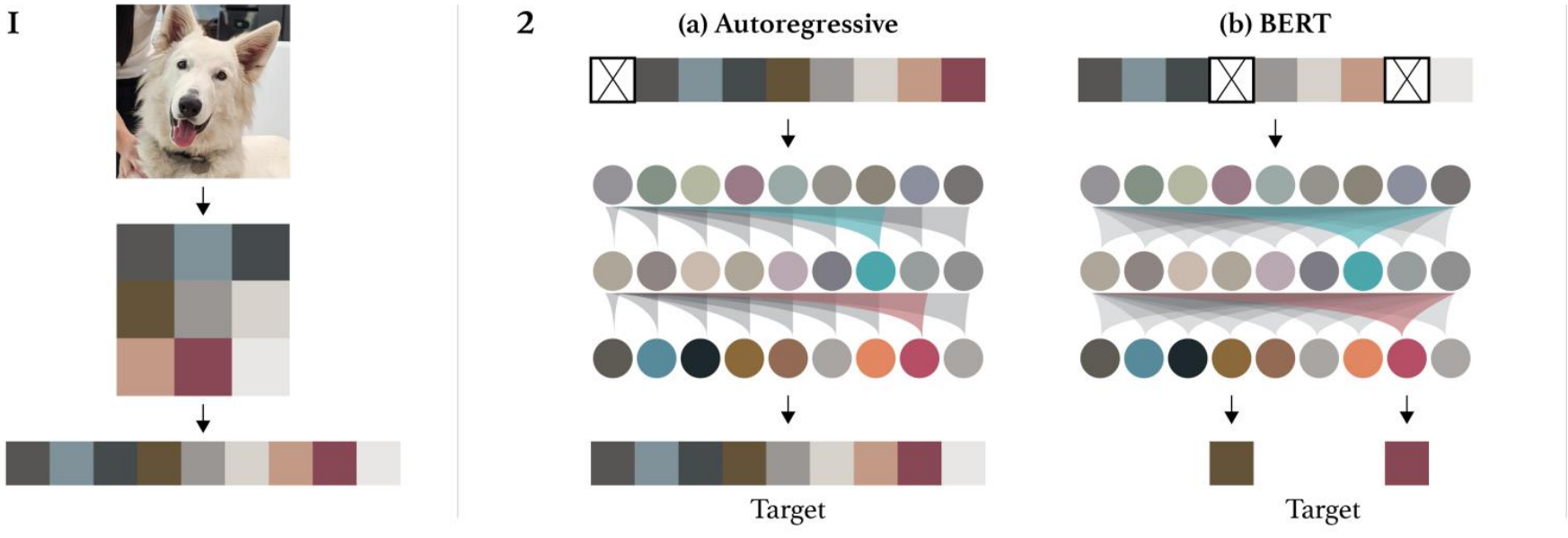
Treat an image as a set of pixel values





Feed as input to standard Transformer

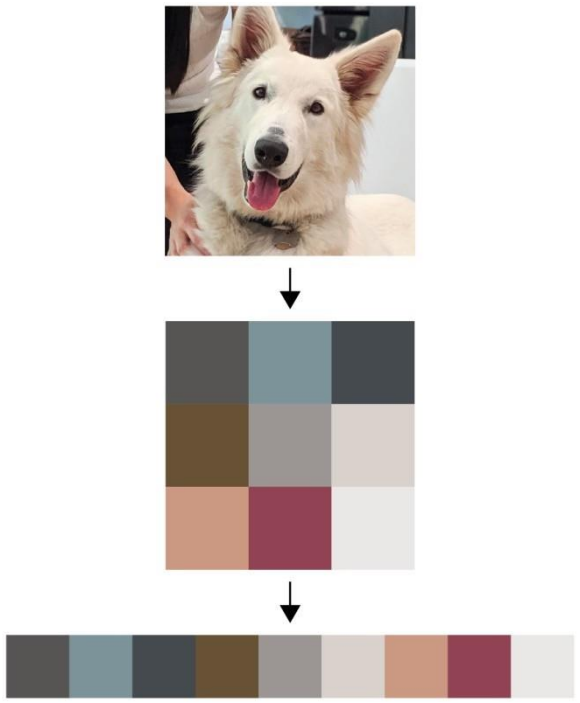
Chen et al, “Generative Pretraining from Pixels”, ICML 2020

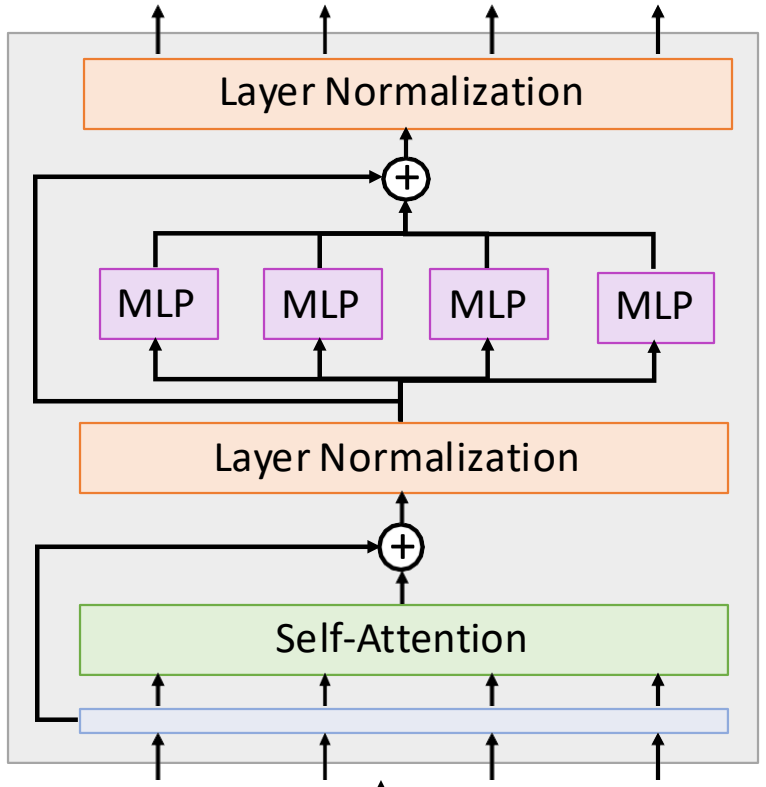


- First, pre-process raw images by resizing to a low resolution and reshaping into a 1D sequer
- pre-training objectives, auto-regressive next pixel prediction or masked pixel prediction.
- Evaluate the representations learned by these objectives with linear probes or fine-tuning.

Idea #3: Standard Transformer on Pixels

Treat an image as a set of pixel values





Feed as input to standard Transformer

Problem: Memory use!

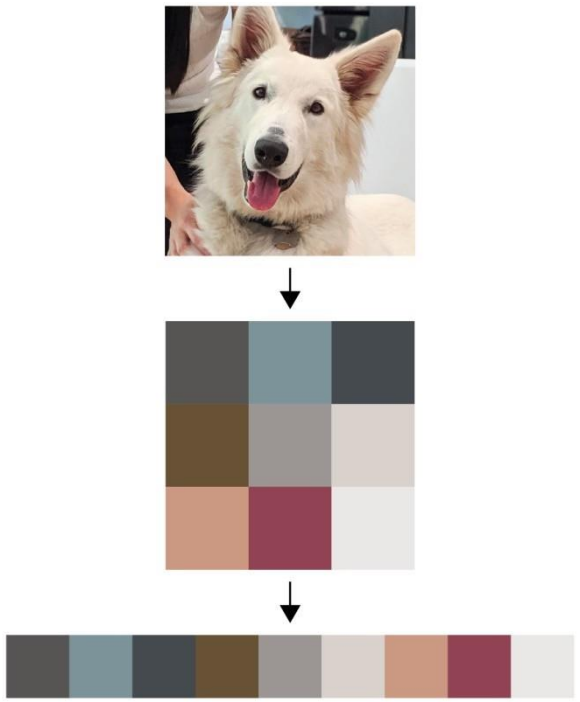
R x R image needs R^4 elements per attention matrix

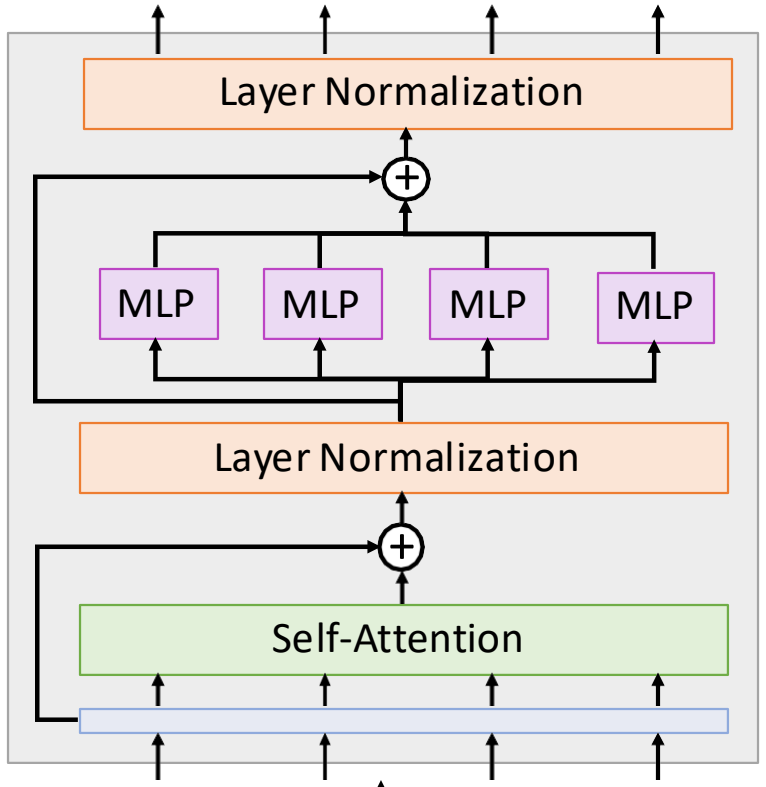
What is the problem?

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

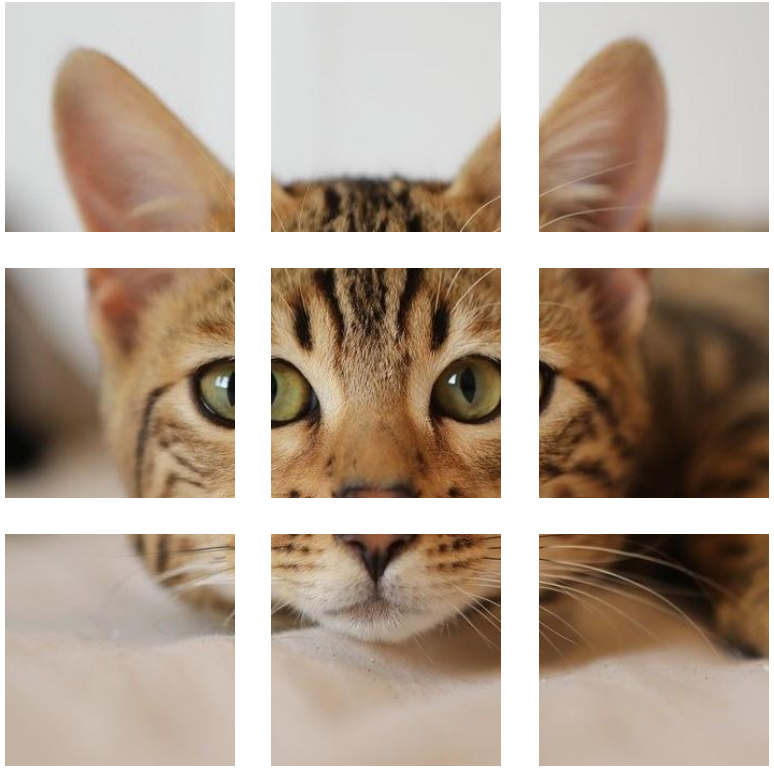
R=128, 48 layers, 16 heads per layer takes 768GB of memory for attention matrices for a single example…

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

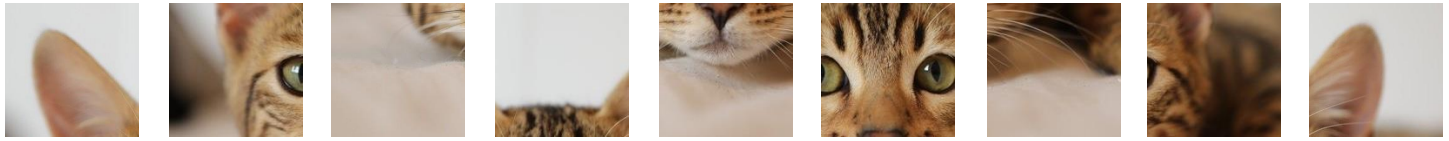
Idea #4: Standard Transformer on Patches



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

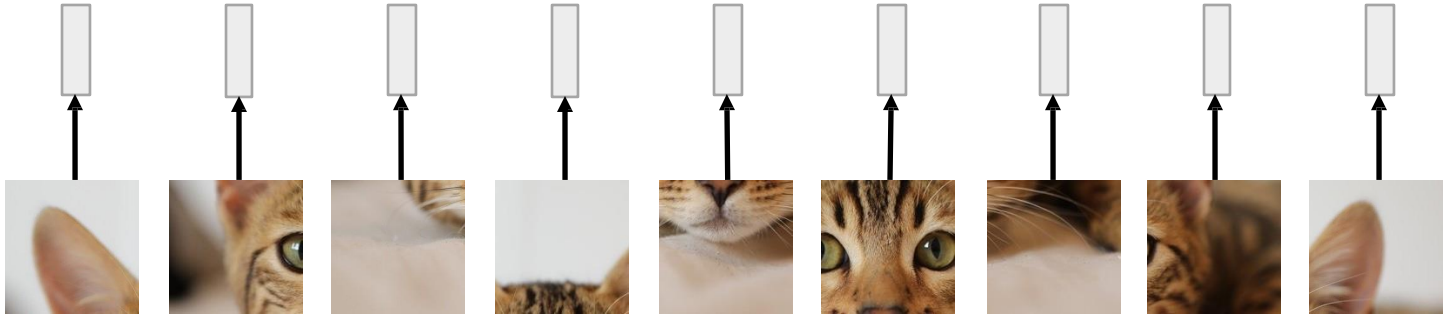


Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Idea #4: Standard Transformer on Patches

Linear projection to D-dimensional vector

N input patches, each of shape 3x16x16



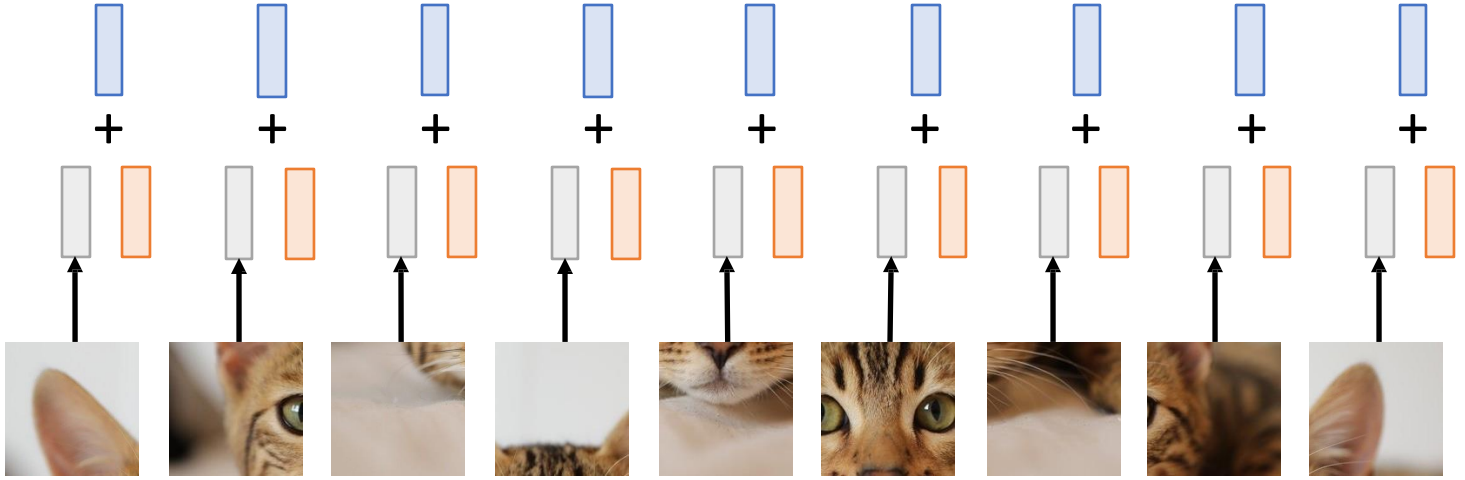
Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Idea #4: Standard Transformer on Patches

Add positional embedding: learned D-dim vector per position

Linear projection to D-dimensional vector

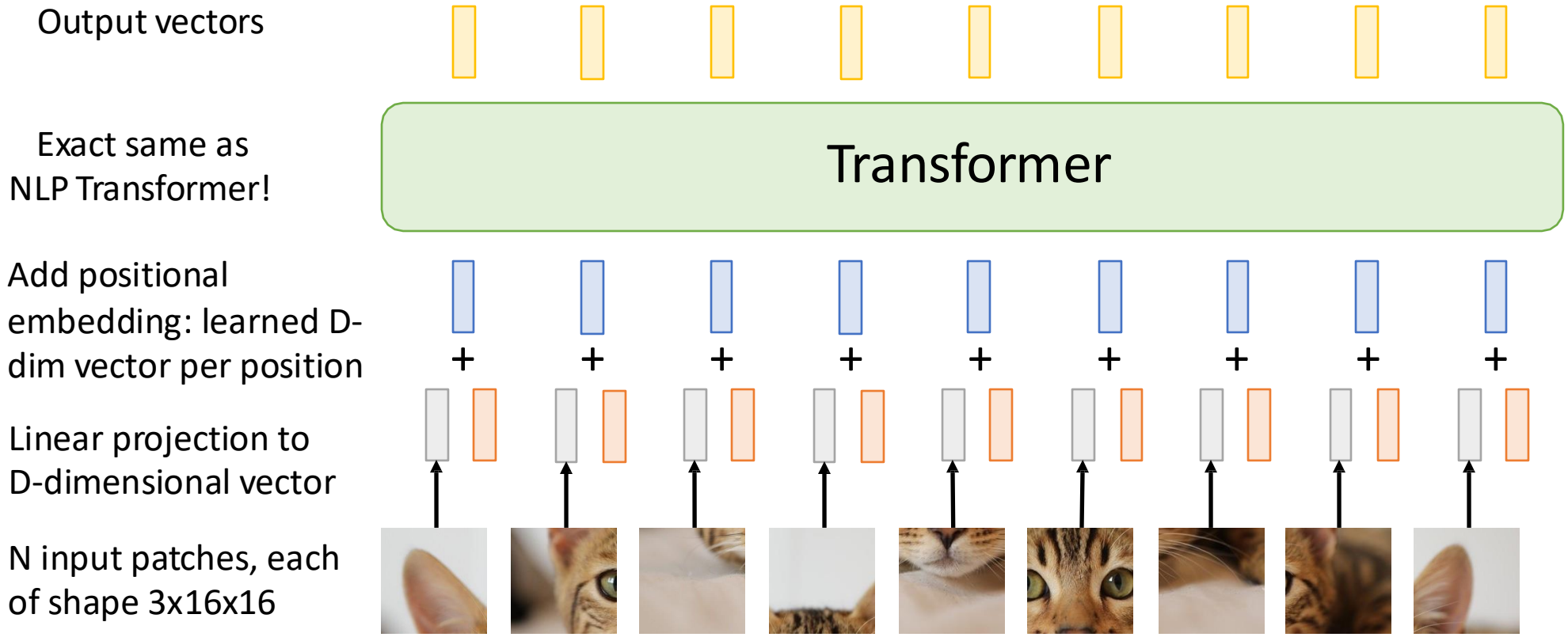
N input patches, each of shape 3x16x16



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

Cat image is free for commercial use under a Pixabay license

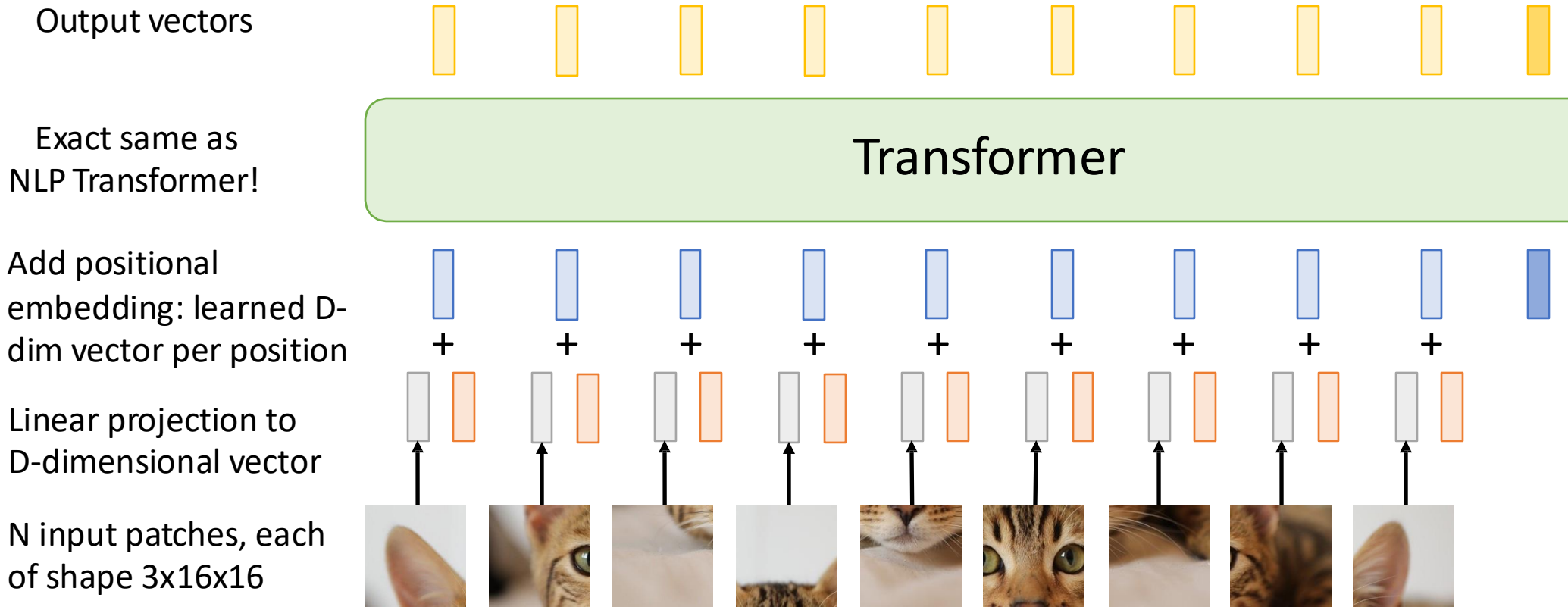
Idea #4: Standard Transformer on Patches



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

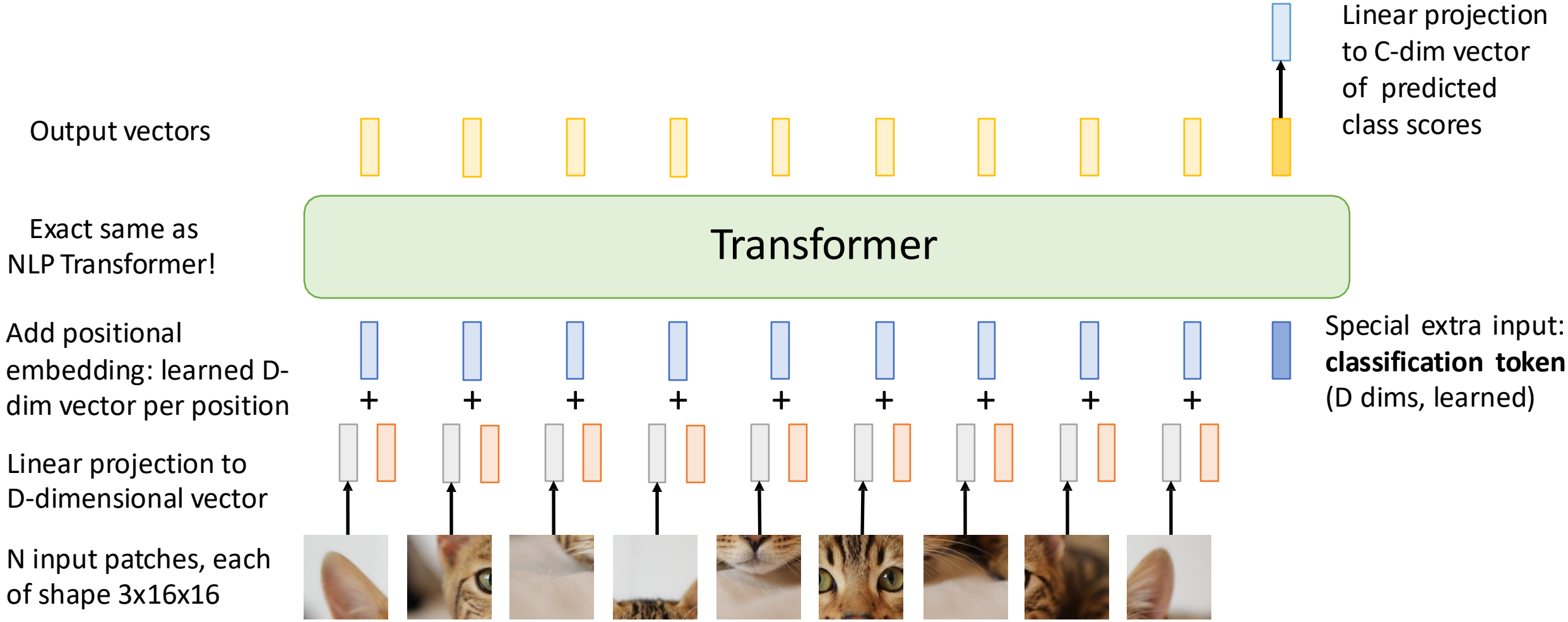
Idea #4: Standard Transformer on Patches



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

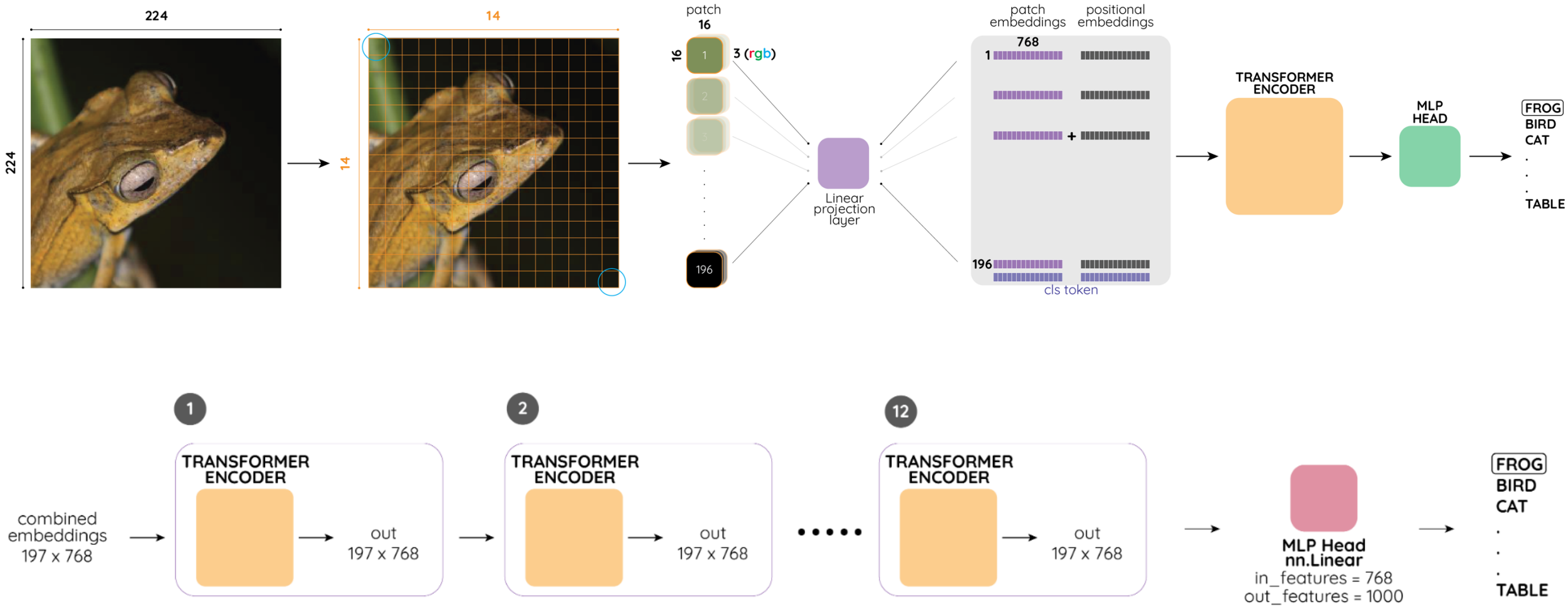
Cat image is free for commercial use under a Pixabay license

Idea #4: Standard Transformer on Patches



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

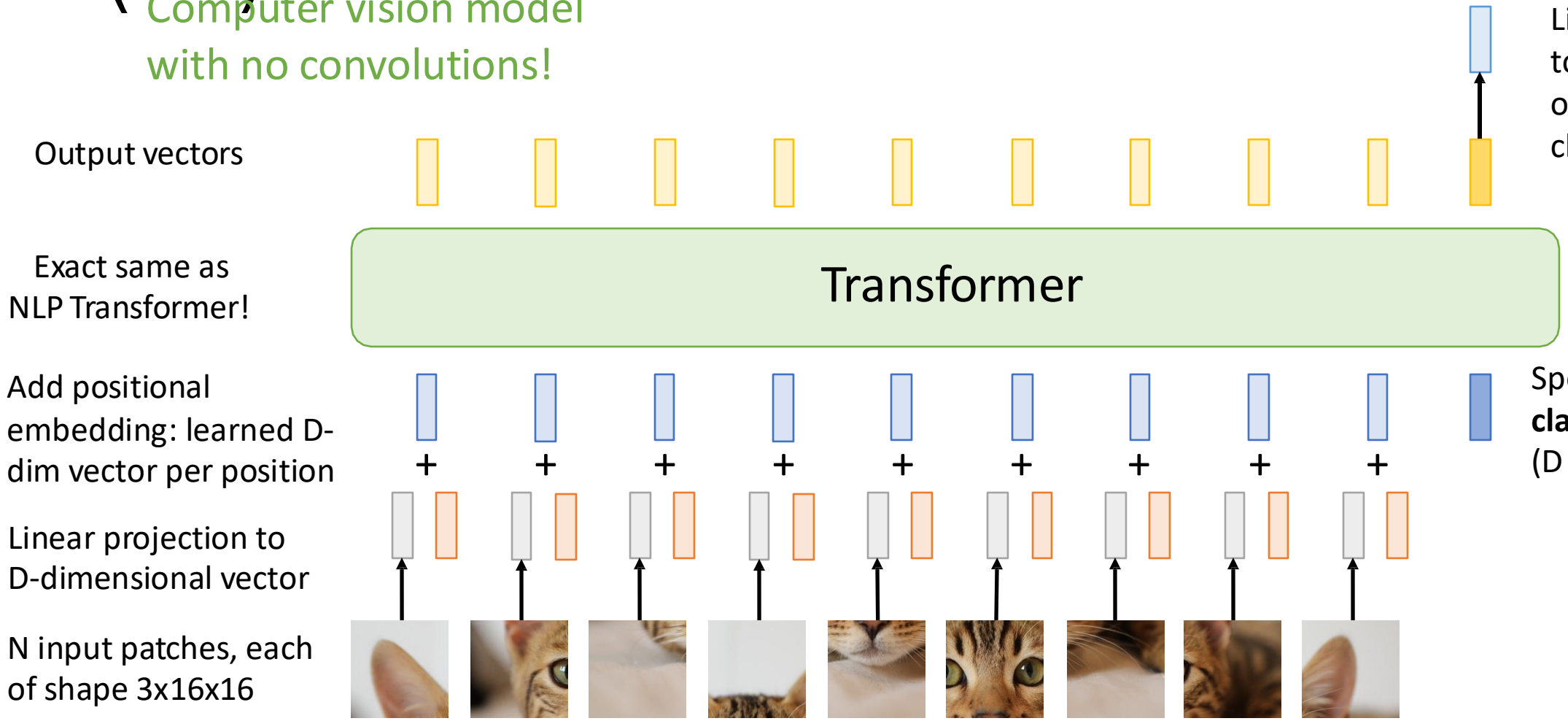
Cat image is free for commercial use under a Pixabay license


224
224
14
14
patch
16
16
1
2
3
196
3 (rgb)
Linear
projection
layer
patch
embeddings
positional
embeddings
768
1
196
cls token
TRANSFORMER
ENCODER
MLP
HEAD
FROG
BIRD
CAT
TABLE
combined
embeddings
197 x 768
1
2
12
TRANSFORMER
ENCODER
out
197 x 768
MLP Head
nn.Linear
in_features = 768
out_features = 1000

What will be the order of attention matrix?

Vision Transformer (ViT)

Computer vision model with no convolutions!



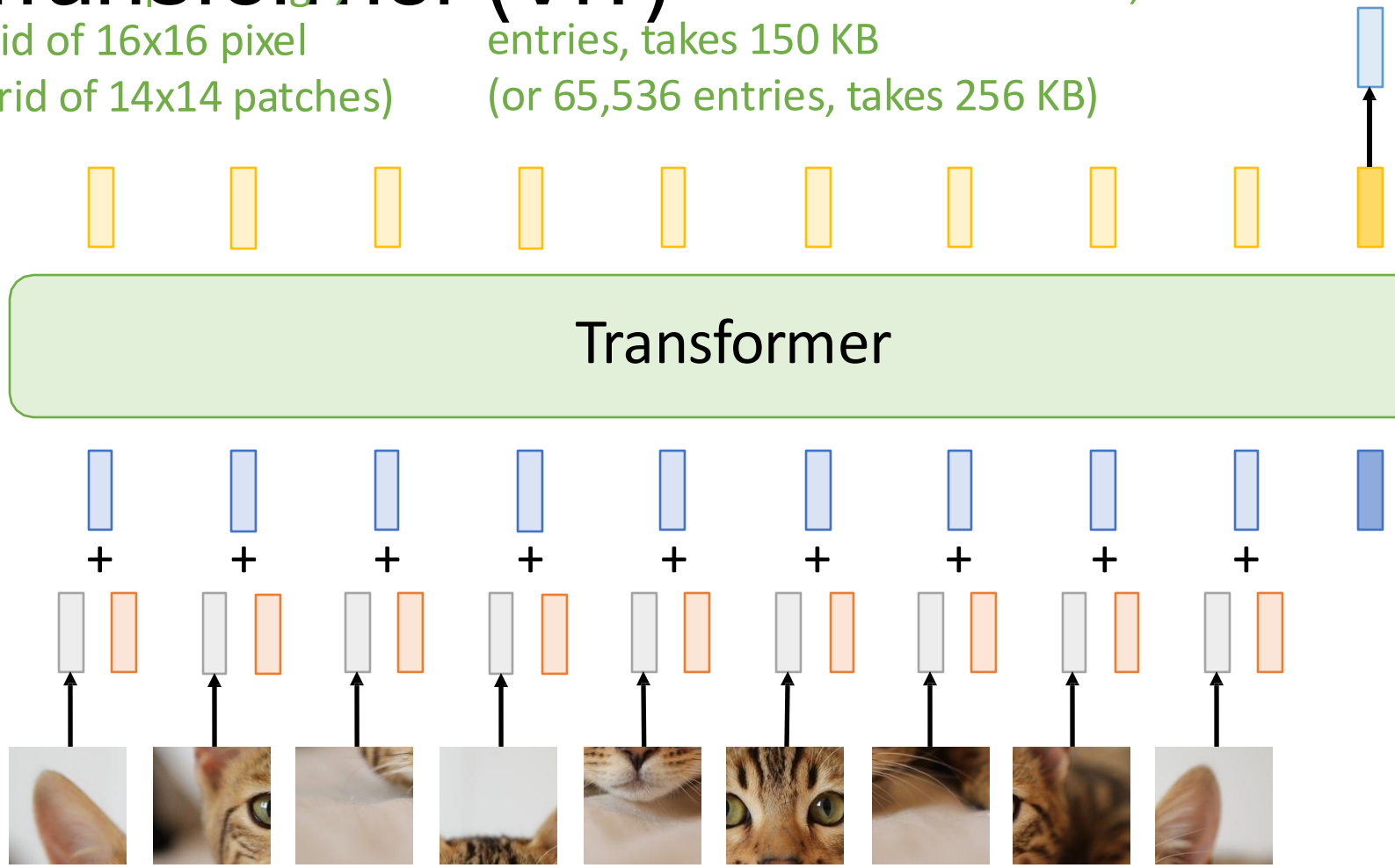
Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Cat image is free for commercial use under a Pixabay license

Vision Transformer (ViT)

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

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



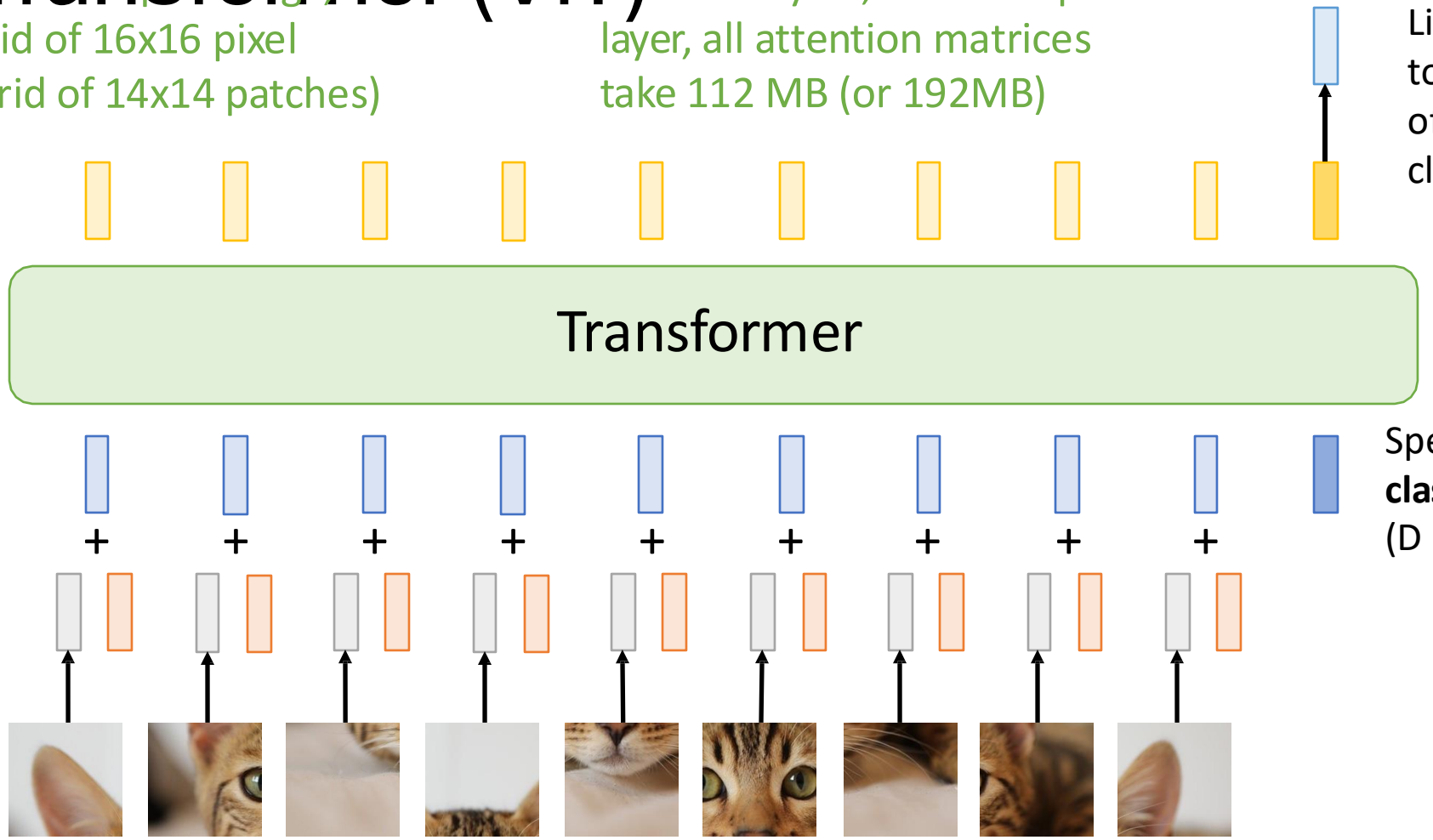
Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Cat image is free for commercial use under a Pixabay license

Vision Transformer (ViT)

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

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



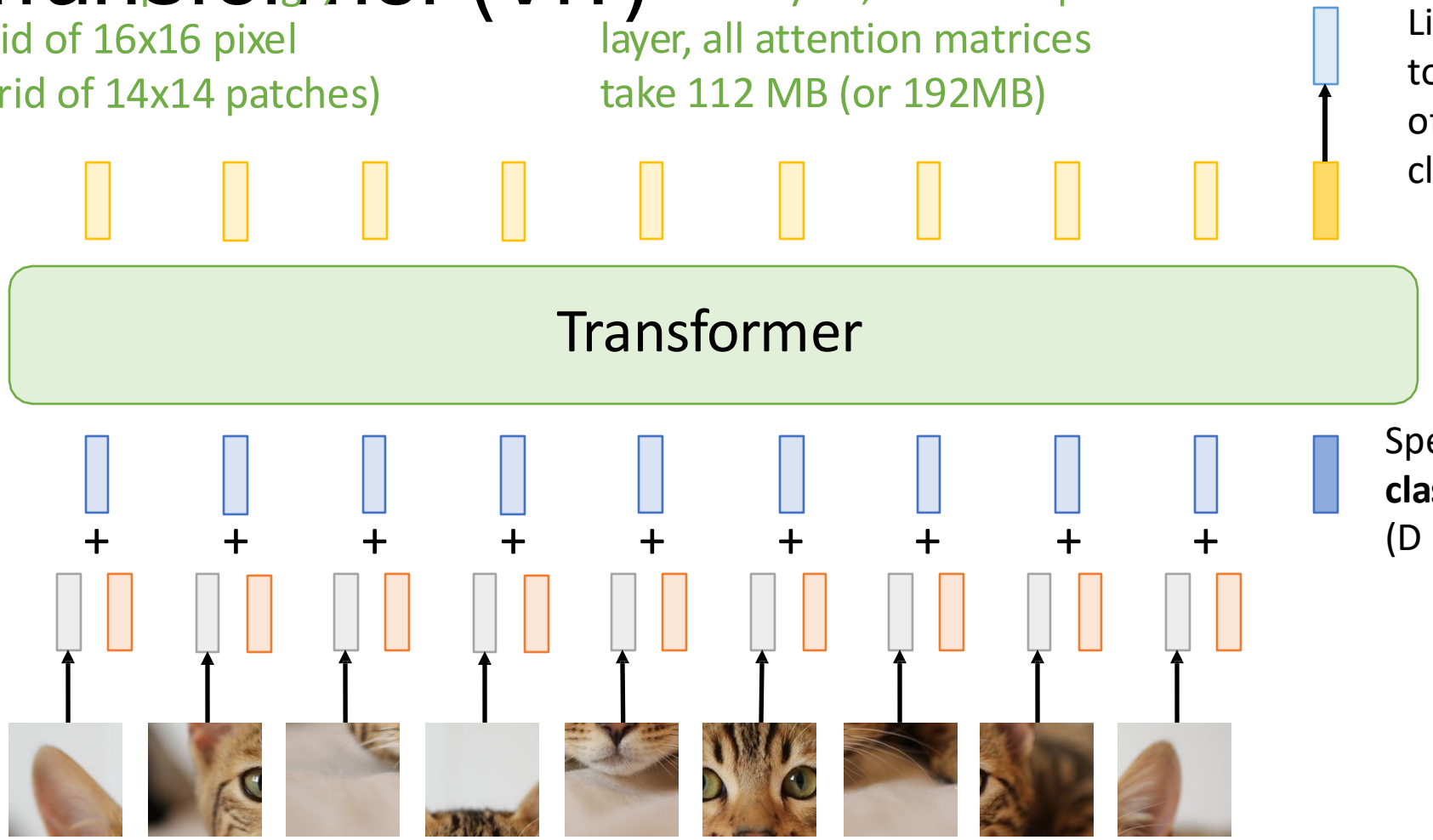
Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Cat image is free for commercial use under a Pixabay license

Vision Transformer (ViT)

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

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

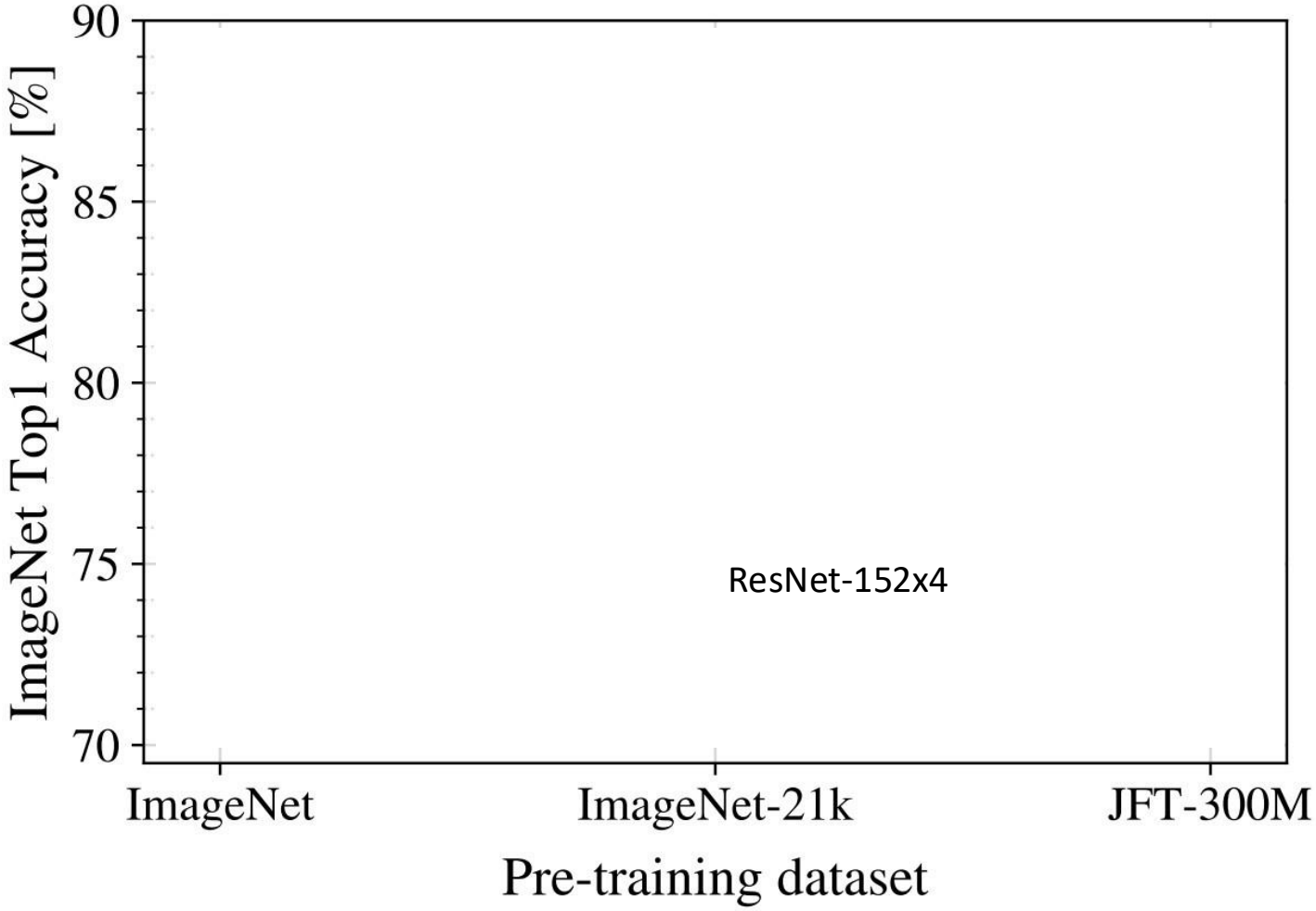


Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Cat image is free for commercial use under a Pixabay license

Vision Transformer (ViT) vs ResNets

Imagenet 21k - 21k classes with 14M images
JFT 300M - 18,291 classes



B = Base
L = Large
H = Huge

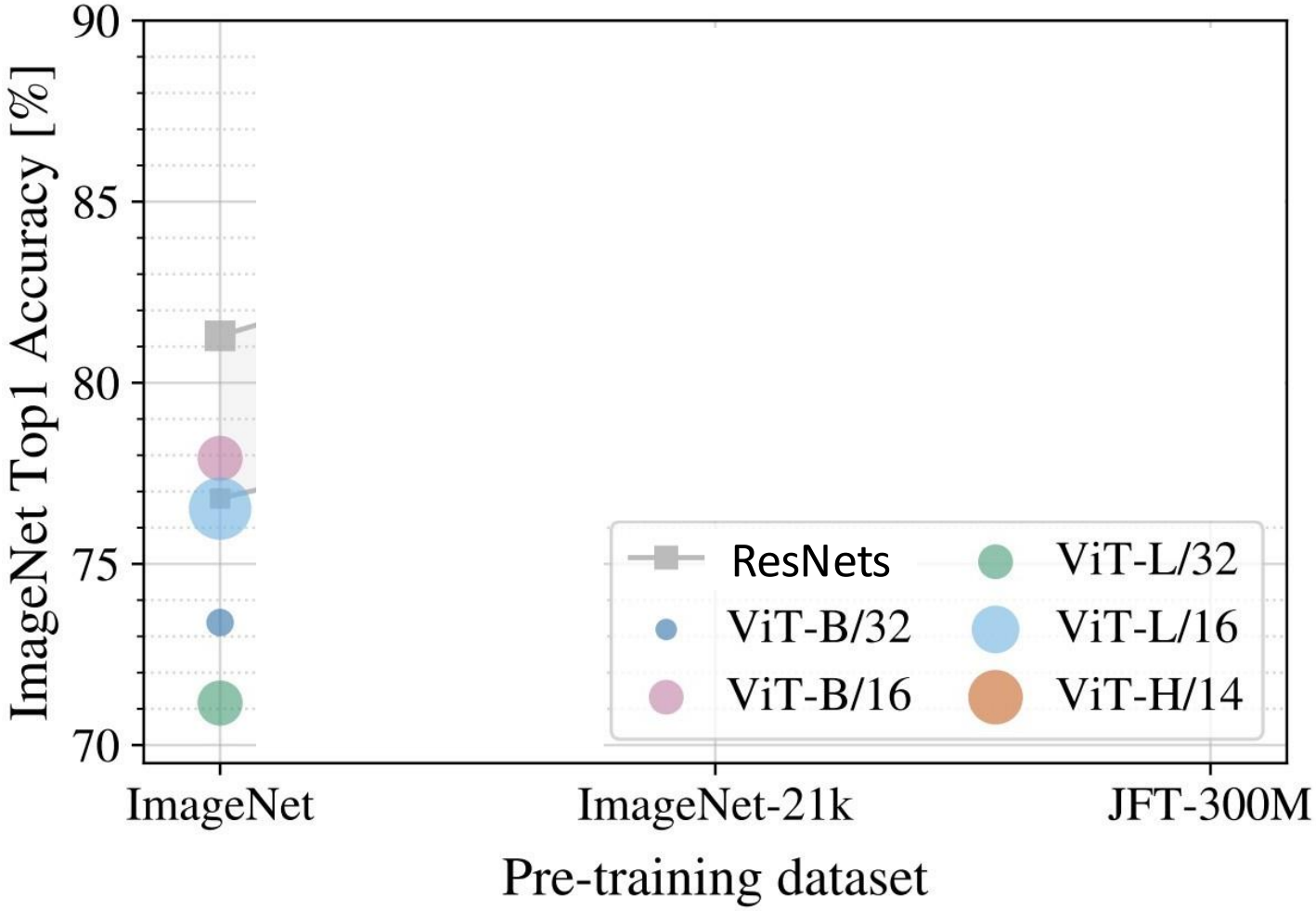
/32, /16, /14 is patch size; smaller patch size is a bigger model (more patches)

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

Vision Transformer (ViT) vs ResNets

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

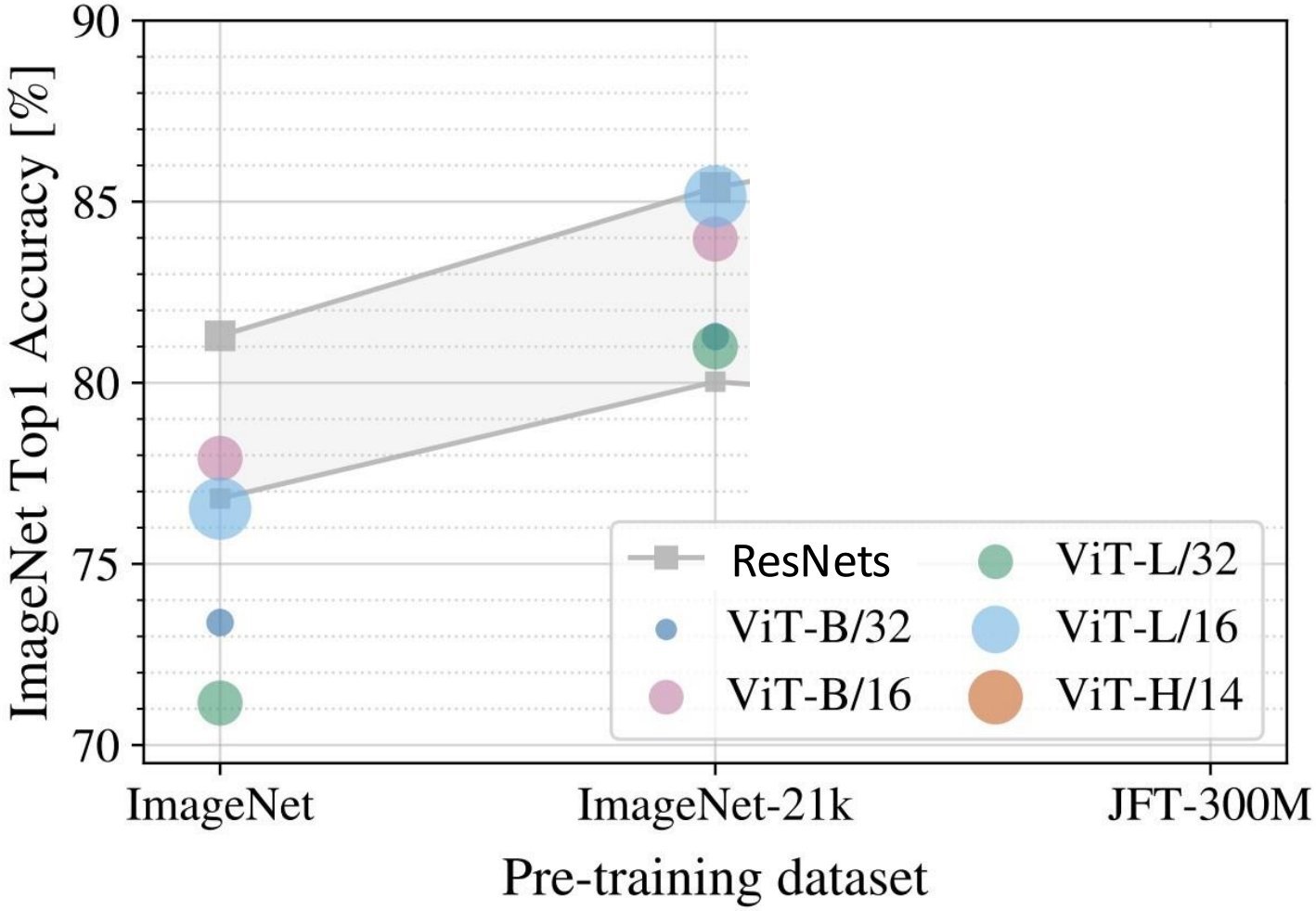
/32, /16, /14 is patch size; smaller patch size is a bigger model (more patches)

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)

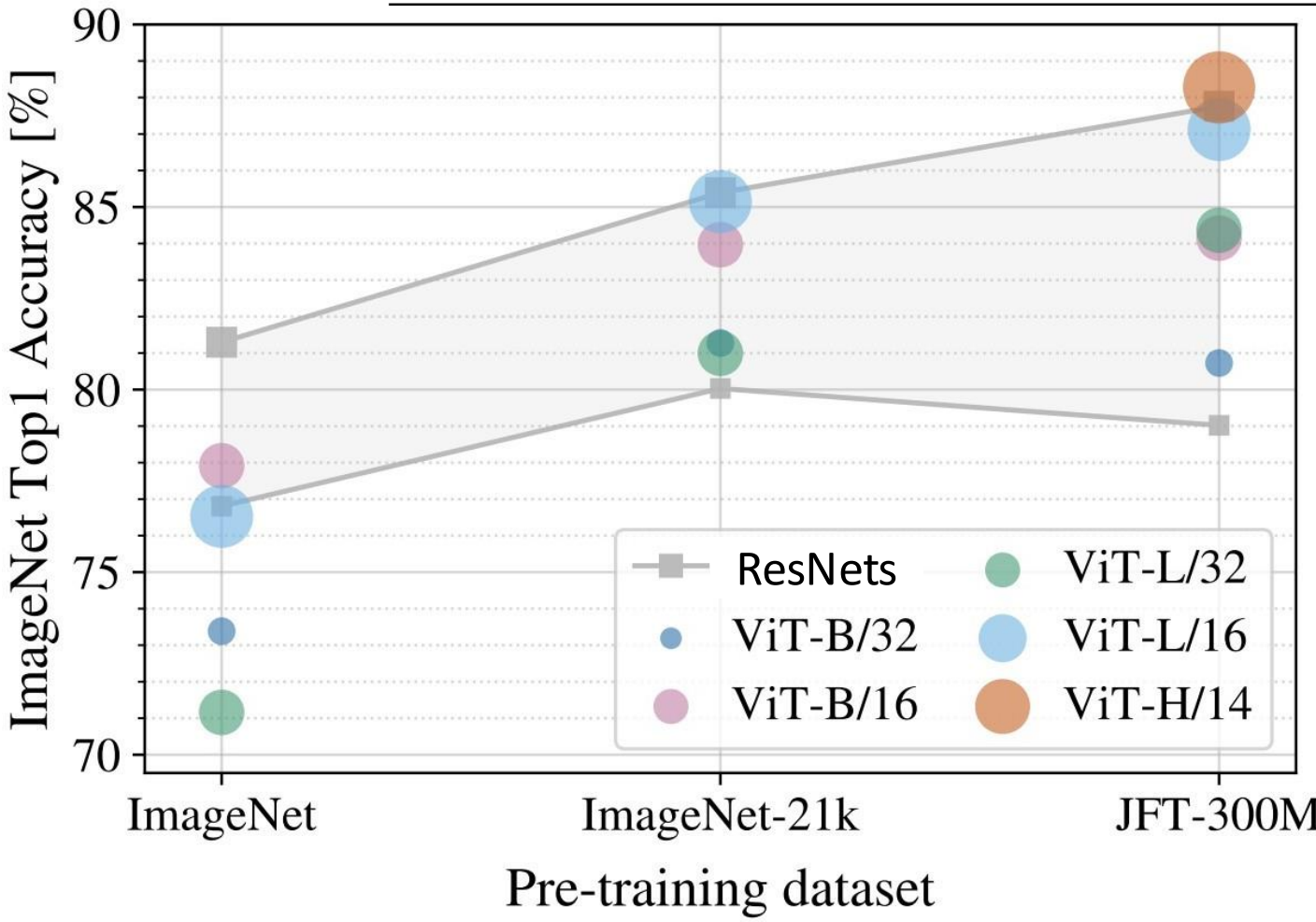
Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Vision Transformer ResNets

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

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



Resnet-152 60M

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

The "ViT-L/16" configuration of the
Patch Size: 16x16 (The input image is divided into 16x16 patches)
•**Embedding Dimension**: 1024
•**Number of Transformer Blocks**: 24
•**Number of Heads in Multi-Head Attention**: 16
•**MLP Hidden Size**: 4096
•**Number of Classes**: Typically 1000 for ImageNet classification

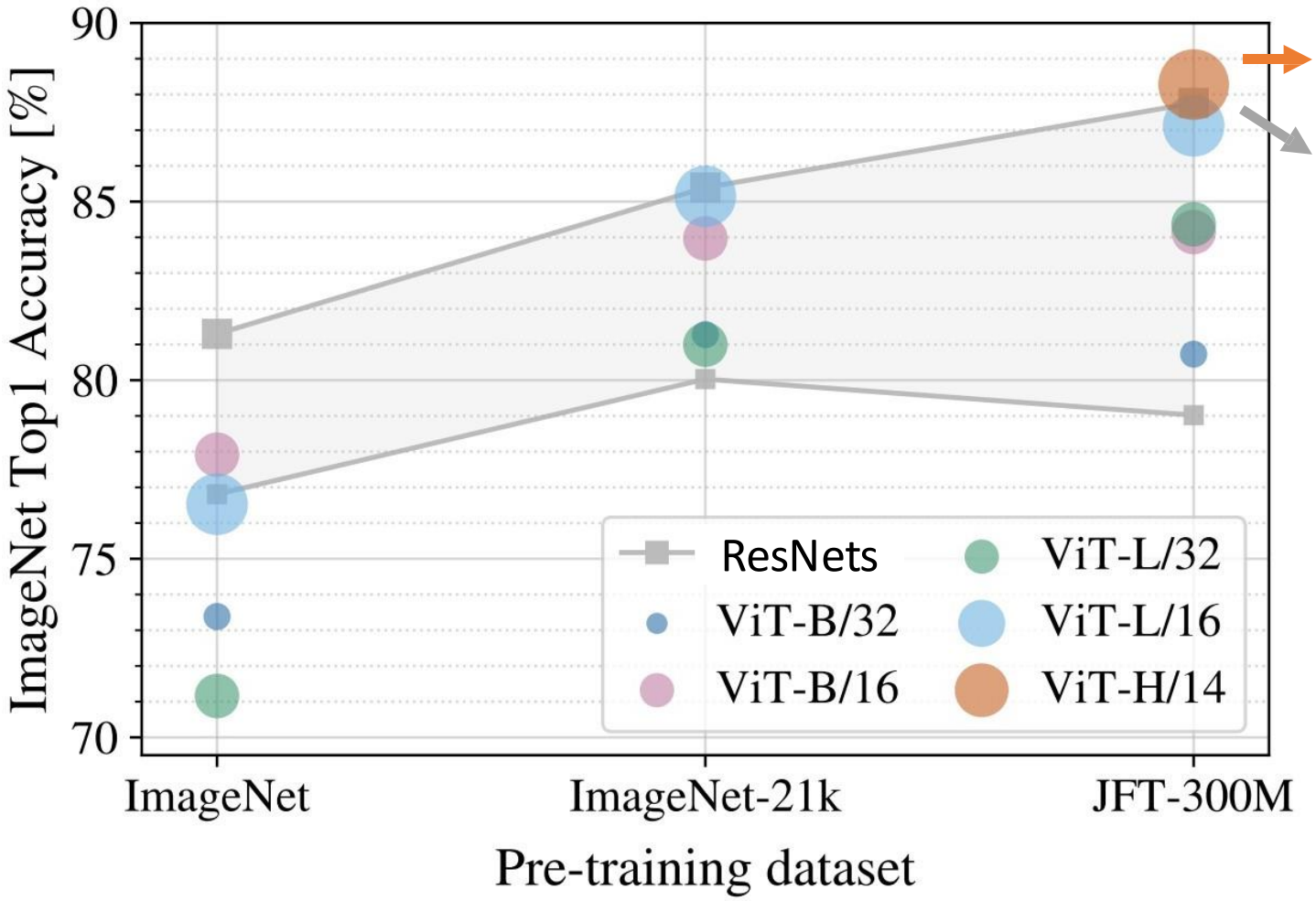
So the "ViT-L/16" model has approximately 339 million parameters.

1.Patch Embedding:
16×16×3×1024=786,43216×16×3×1024=786,432
2.Positional Embedding:
1024×197=201,7281024×197=201,728
3.Transformer Blocks:
24×((1024×1024×3)+(1024×4096×2))≈337,002,49624×((1024×1024×3)+(1024×4096×2))≈337,002,496
4.Classification Head:
1024×1000=1,024,0001024×1000=1,024,000
The total is roughly:
786,432+201,728+337,002,496+1,024,000≈339,014,656786,432+201,728+337,002,496+1,024,000≈339,014,656

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

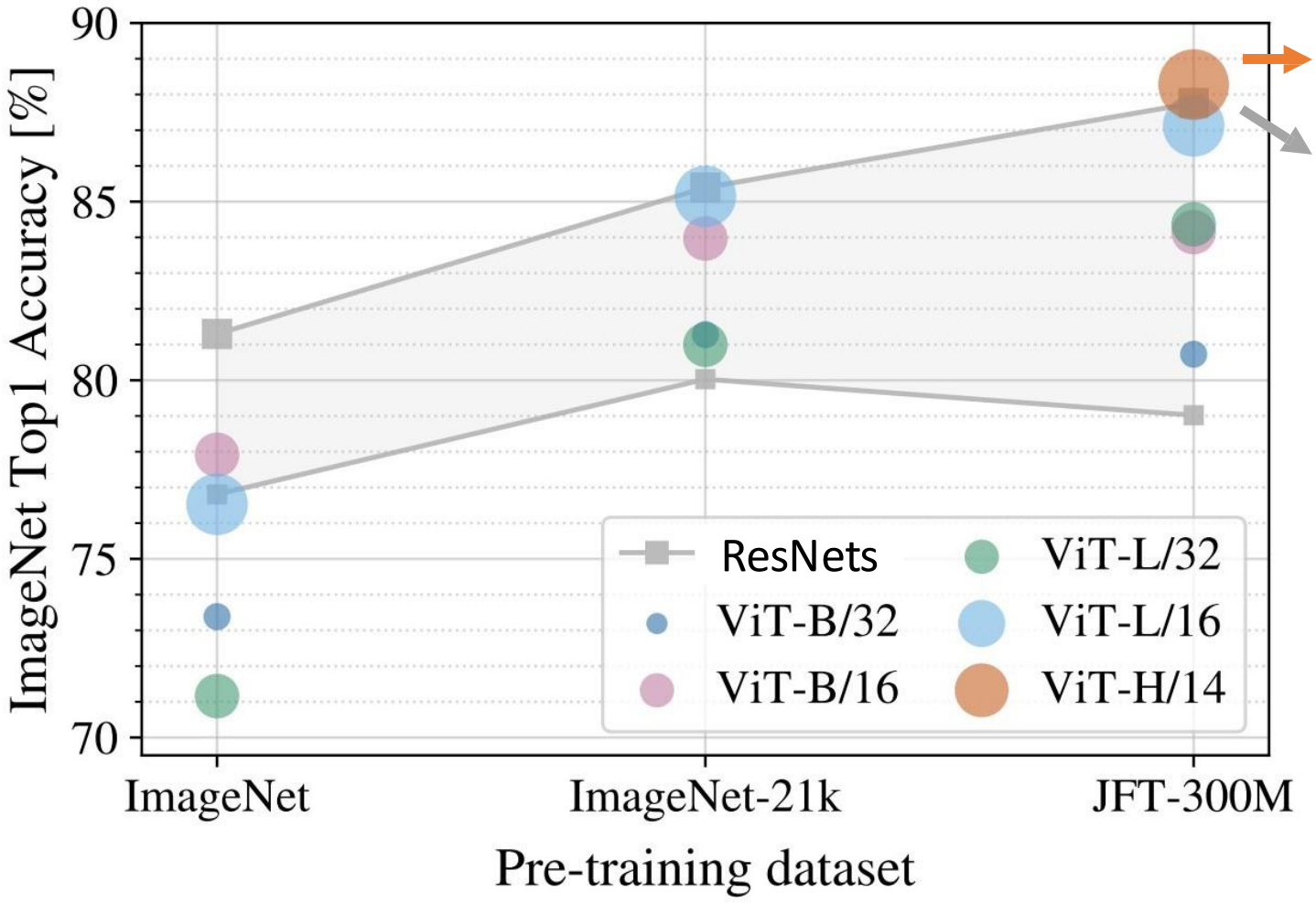
ResNet: 9.9k TPU-v3 core days of training

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

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



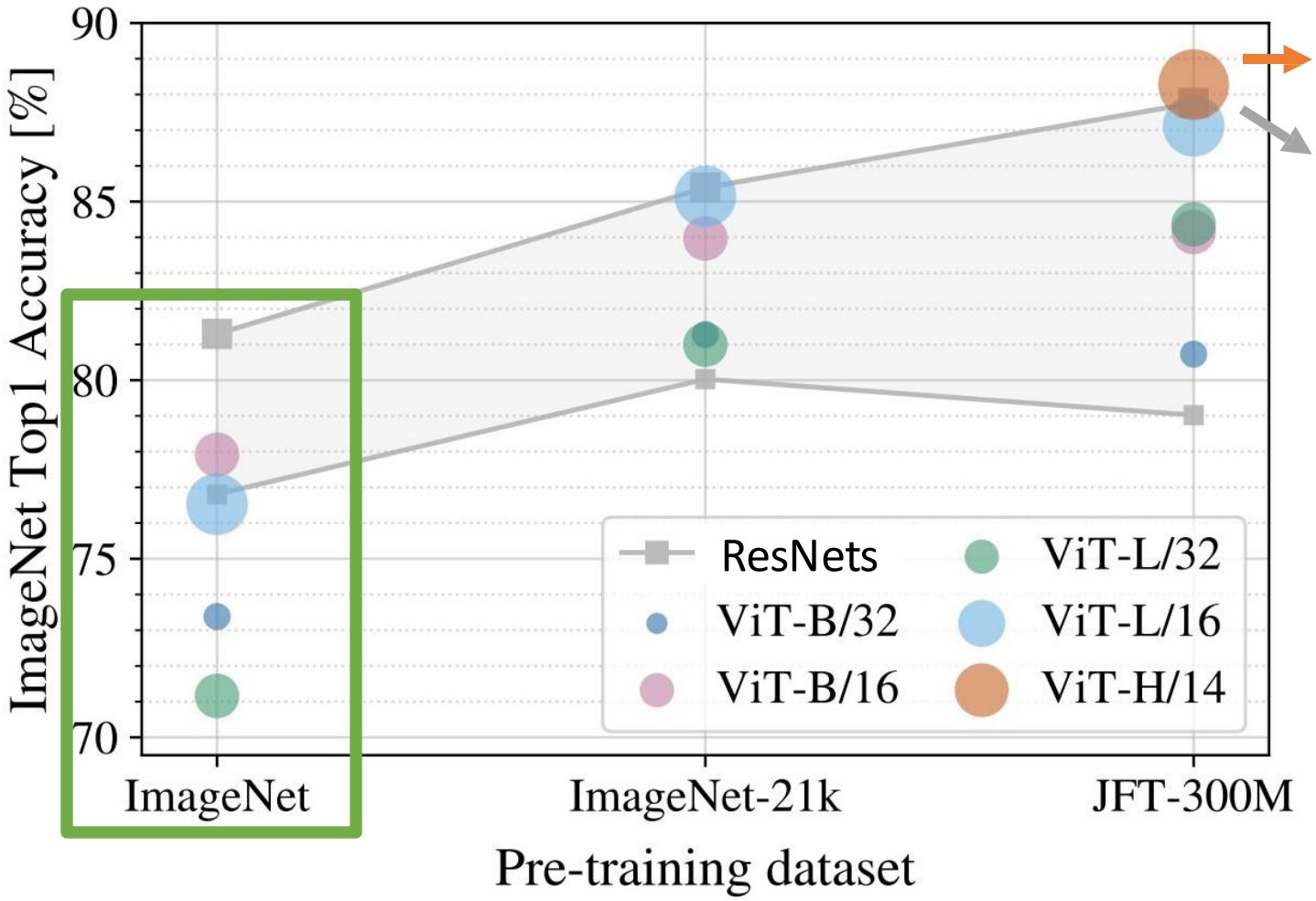
ViT: 2.5k TPU-v3 core days of training

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



ViT: 2.5k TPU-v3 core days of training

ResNet: 9.9k TPU-v3 core days of training

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

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

Stochastic Depth

Layer Dropping: During training, stochastic depth randomly drops certain layers in the neural network. Each mini-batch might use a different network depth due to this random dropping.

Survival Probability: Each layer has a predetermined probability, with which the layer will be active during a particular forward pass in training.

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

Mix up

Select Two Examples: Randomly select two examples from the training data. Let's denote them as (xi,yi) and (xj,yj), where x represents the input features (e.g., images) and y represents the labels.

Mix Inputs and Labels: Form a new example (x',y') by performing a weighted average of the two selected examples:

$x' = \lambda x_i + (1 - \lambda) x_j$

$y' = \lambda y_i + (1 - \lambda) y_j$

RandAugment

Predefined Transformations: RandAugment uses a fixed set of image transformations, which might include operations like rotation, shearing, translation, color adjustment, contrast enhancement, and more.

Random Selection and Application:

•For each image in the training dataset, a fixed number N of transformations are randomly selected from the pool.

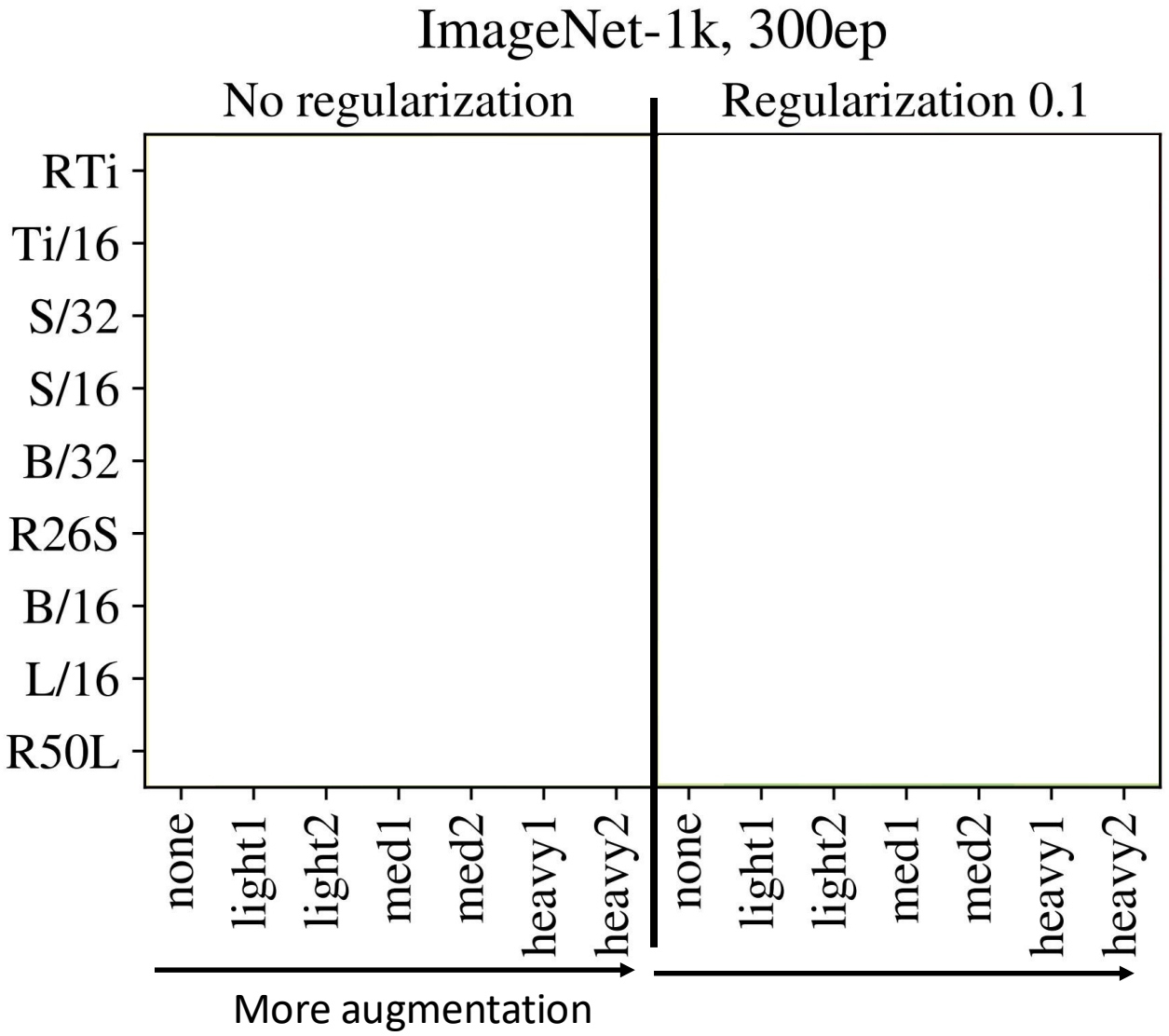
•Each selected transformation is applied with a certain intensity or magnitude M.

•Both N and M are hyperparameters

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

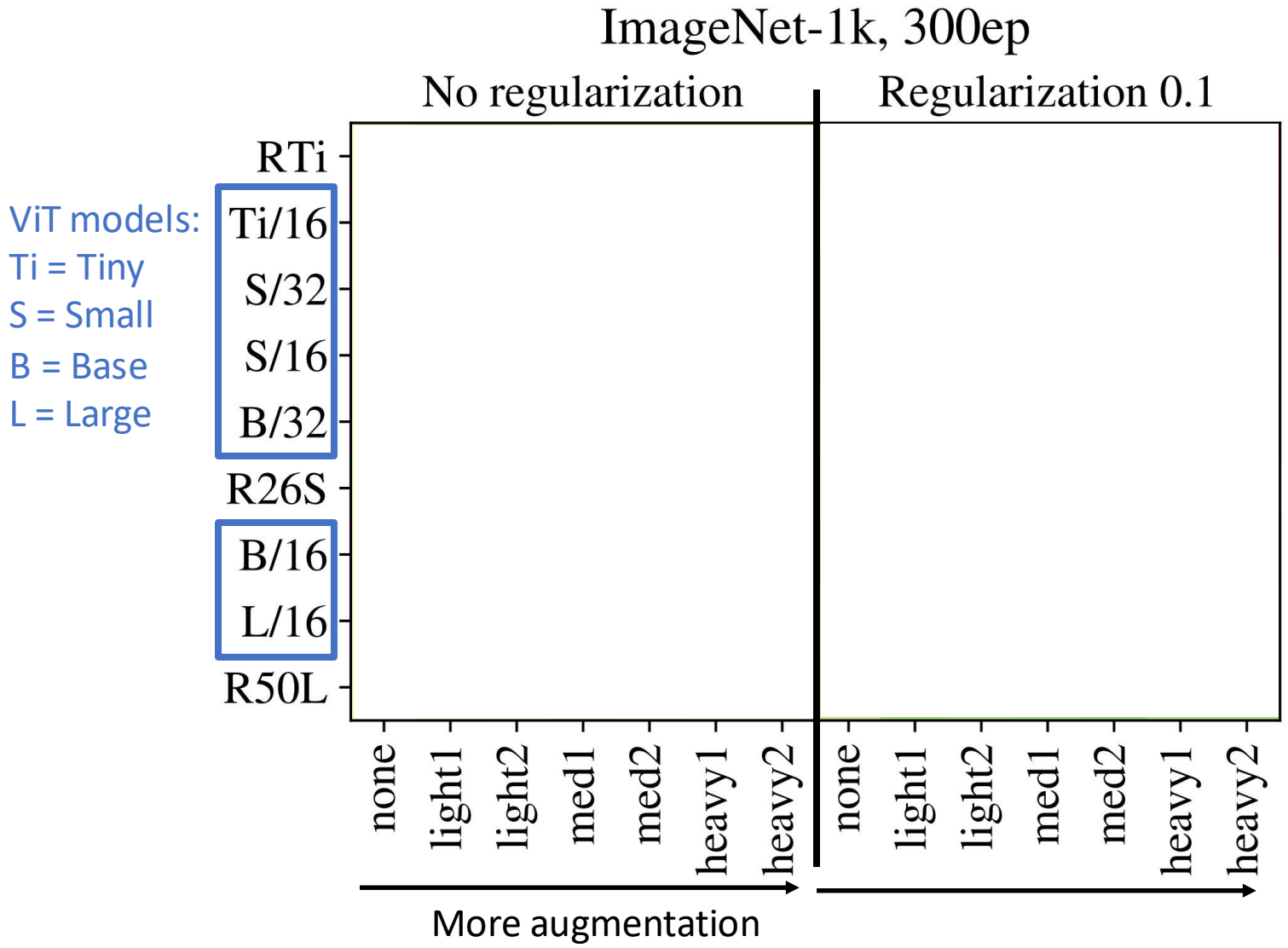


Steiner et al, “How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers”, arXiv 2021

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

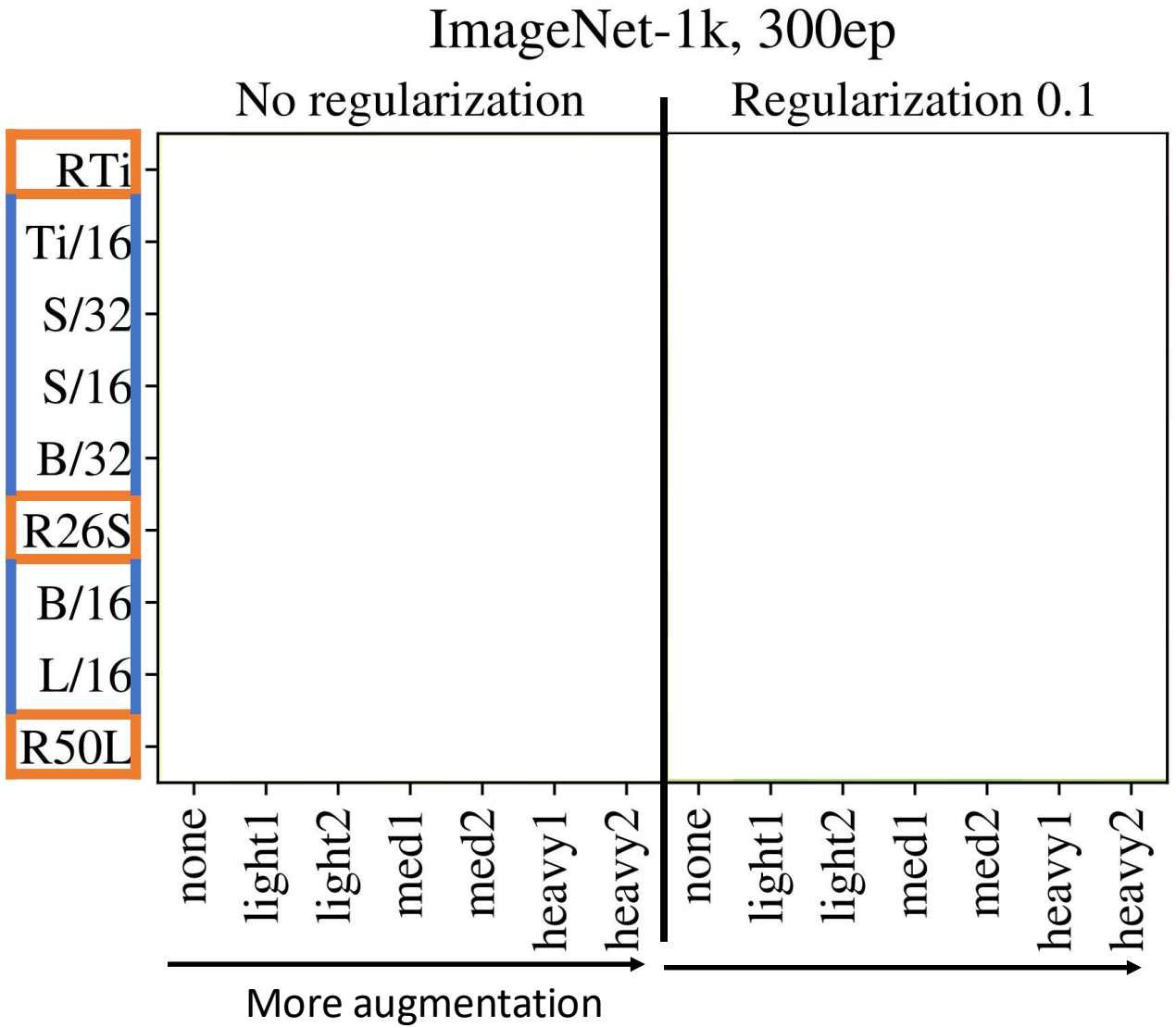
Improving ViT: Augmentation and Regularization

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

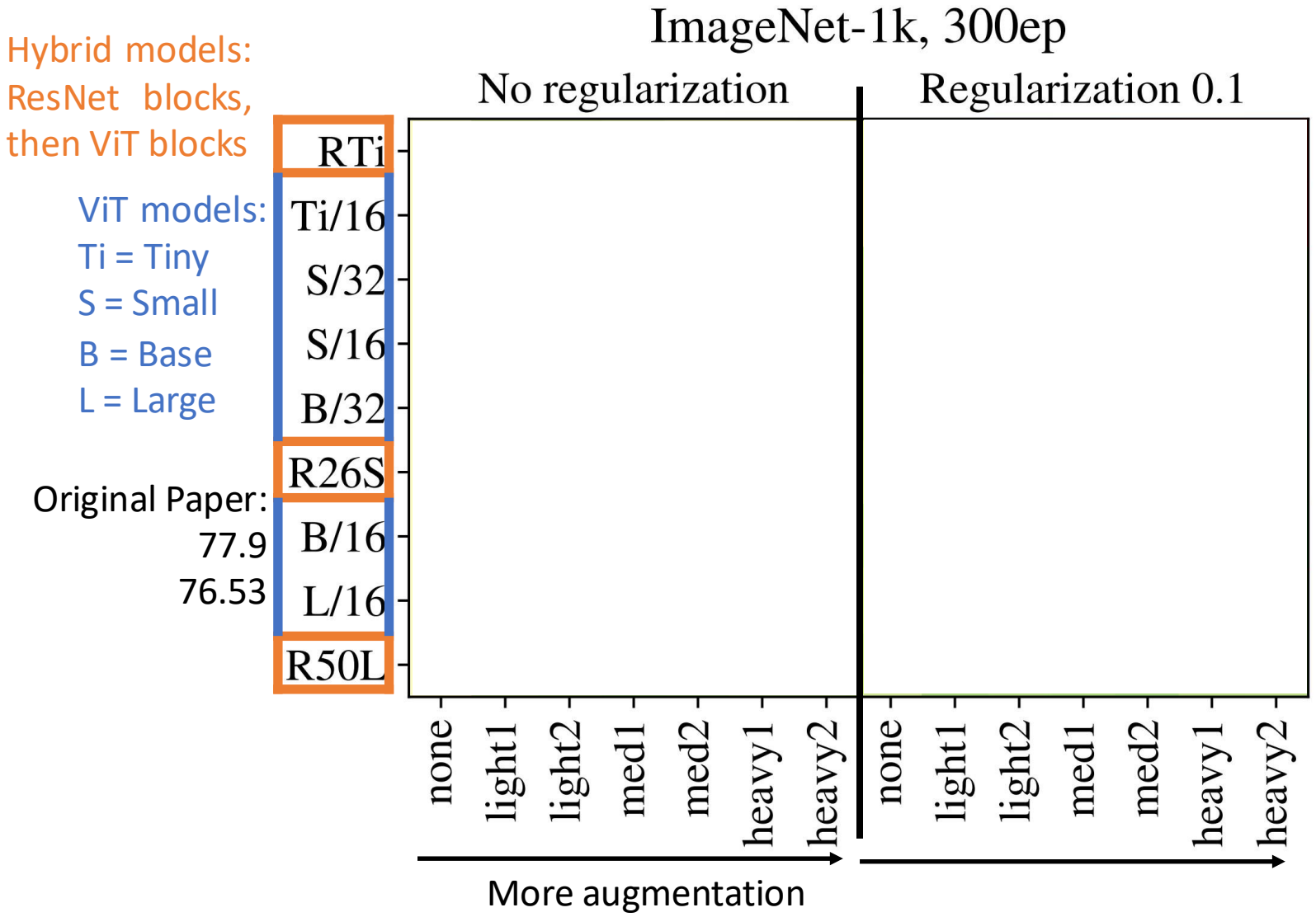


Steiner et al, "How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers", arXiv 2021

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



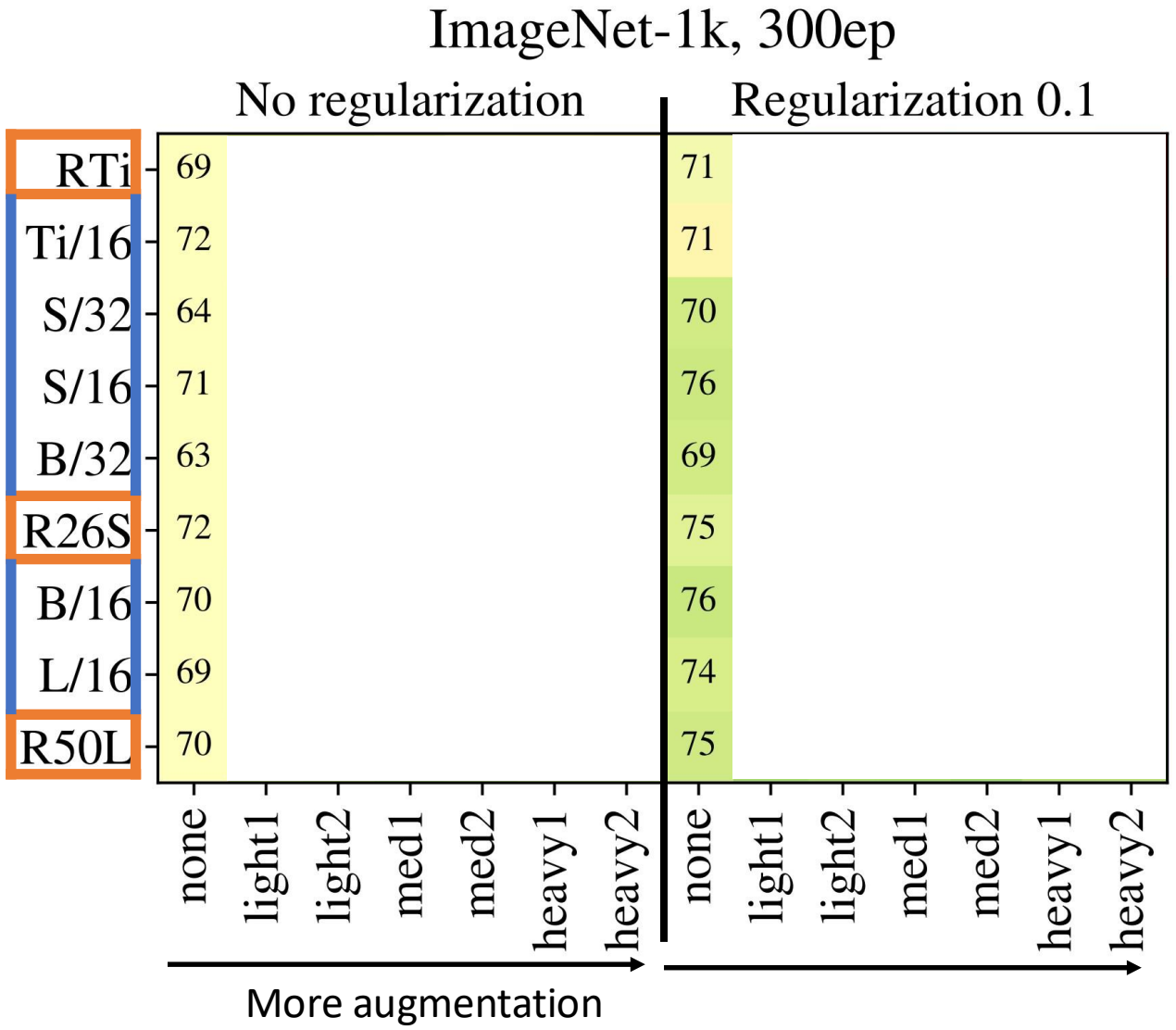
Steiner et al, "How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers", arXiv 2021

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

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)

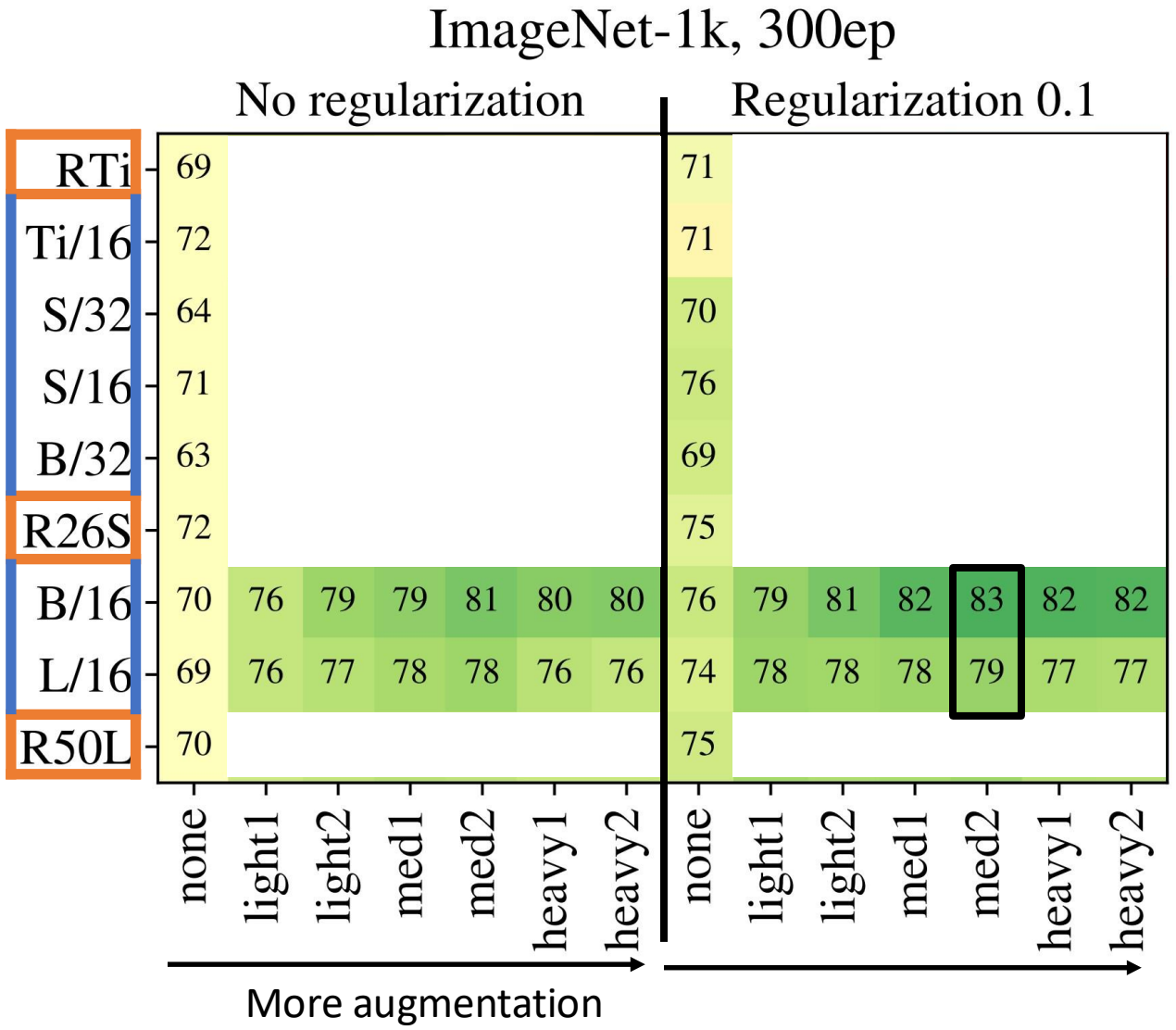
Data Augmentation for ViT models:
- MixUp
- RandAugment

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

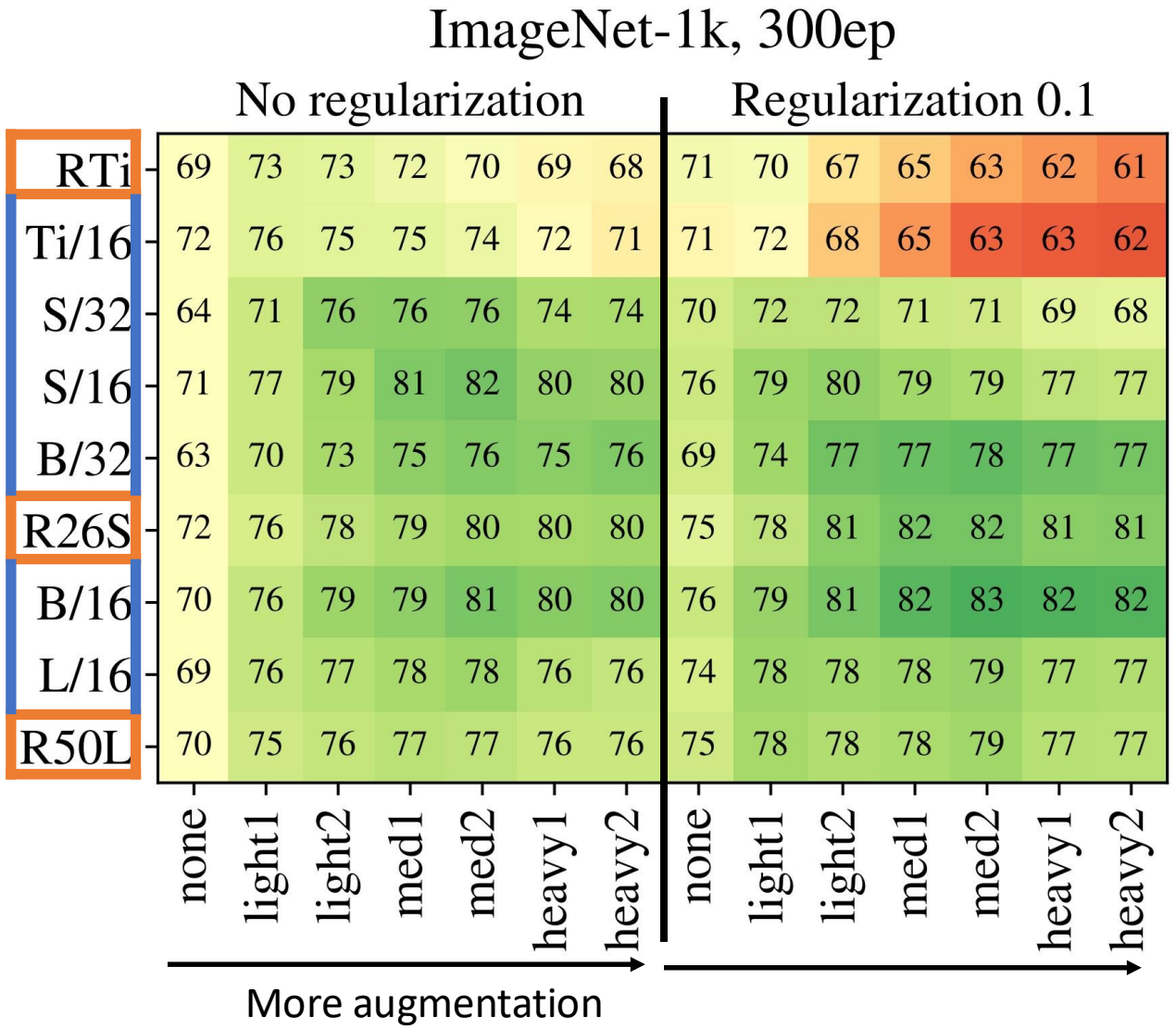
Regularization for ViT models:
- Weight Decay
- Stochastic Depth
- Dropout (in FFN layers of Transformer)

Data Augmentation for ViT models:
- MixUp
- RandAugment

Hybrid models: ResNet blocks, then ViT blocks

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

Original Paper: Lots of other patterns in full results



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

- Why right top is poor performer?

Given an image.

- Can you obtain its caption?

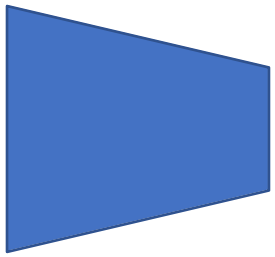
- Can you obtain its embedding?

Can you use text embedding and make image look like same as input but the manipulated image should give a different/irrelevant/harmful caption?

Can you do vice-versa?

Improving ViT: Distillation

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



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

→ Cross Entropy Loss ← GT label: Cat

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

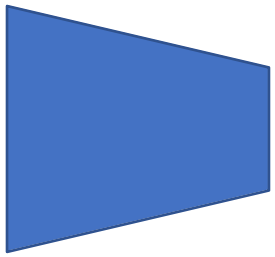
Assumption:

P(x_teacher) != P(x_student), for instance teacher sees day images, student is asked for night images

P(y) is same for both teacher and student, that is, labels fully overlap

Improving ViT: Distillation

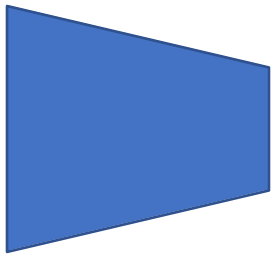
Step 1: Train a **teacher model** on images and ground-truth labels



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

→ Cross Entropy Loss ← GT label: Cat

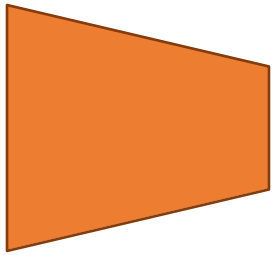
Step 2: Train a **student model** to match predictions from the **teacher**



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

$\lambda\tau^2 KL(\psi(Zs/\tau),\psi(Zt/\tau))$.

→ KL Divergence Loss

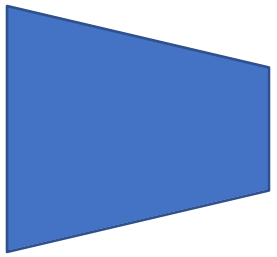


P(cat) = 0.2
P(dog) = 0.8

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

Improving ViT: Distillation

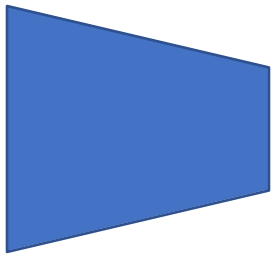
Step 1: Train a **teacher model** on images and ground-truth labels



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

→ Cross Entropy Loss ← GT label: Cat

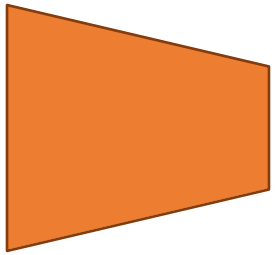
Step 2: Train a **student model** to match predictions from the **teacher** (sometimes also to match GT labels)



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

$\lambda\tau^2 KL(\psi(Zs/\tau),\psi(Zt/\tau))$.

→ KL Divergence Loss



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

→ Cross Entropy Loss ← GT label: Dog

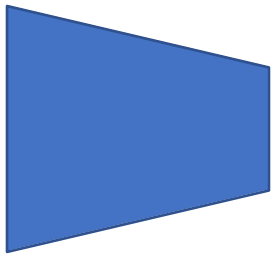
$(1 - \lambda)LCE(\psi(Zs), y)$

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

Improving ViT: Distillation

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

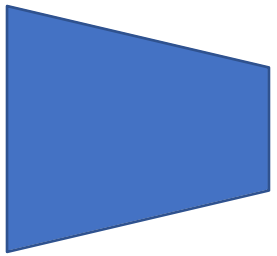
Step 1: Train a **teacher model** on images and ground-truth labels



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

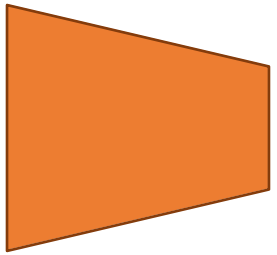
→ Cross Entropy Loss ← GT label: Cat

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



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

→ KL Divergence Loss



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

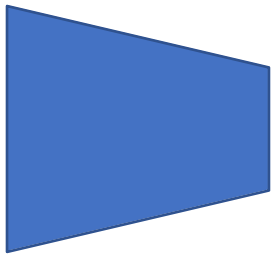
→ Cross Entropy Loss ← GT label: Dog

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

Improving ViT: Distillation

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

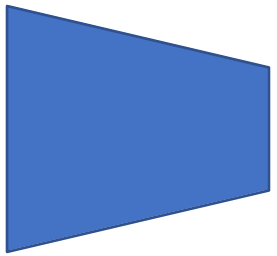
Step 1: Train a **teacher model** on images and ground-truth labels



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

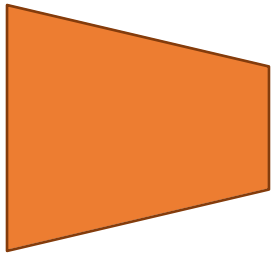
→ Cross Entropy Loss ← GT label: Cat

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



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

→ KL Divergence Loss



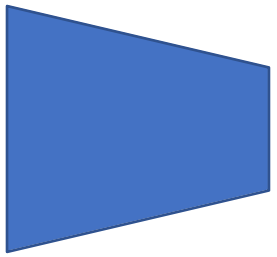
P(cat) = 0.2
P(dog) = 0.8

→ Cross Entropy Loss ← GT label: Dog

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

Improving ViT: Distillation

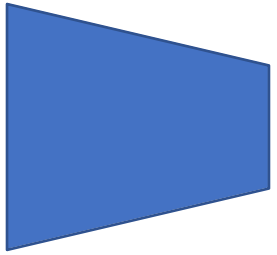
Step 1: Train a teacher CNN on ImageNet



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

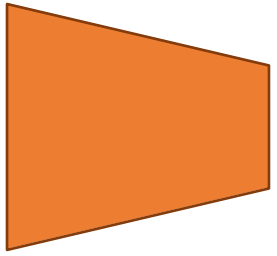
→ Cross Entropy Loss ← GT label: Cat

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



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

→ KL Divergence Loss



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

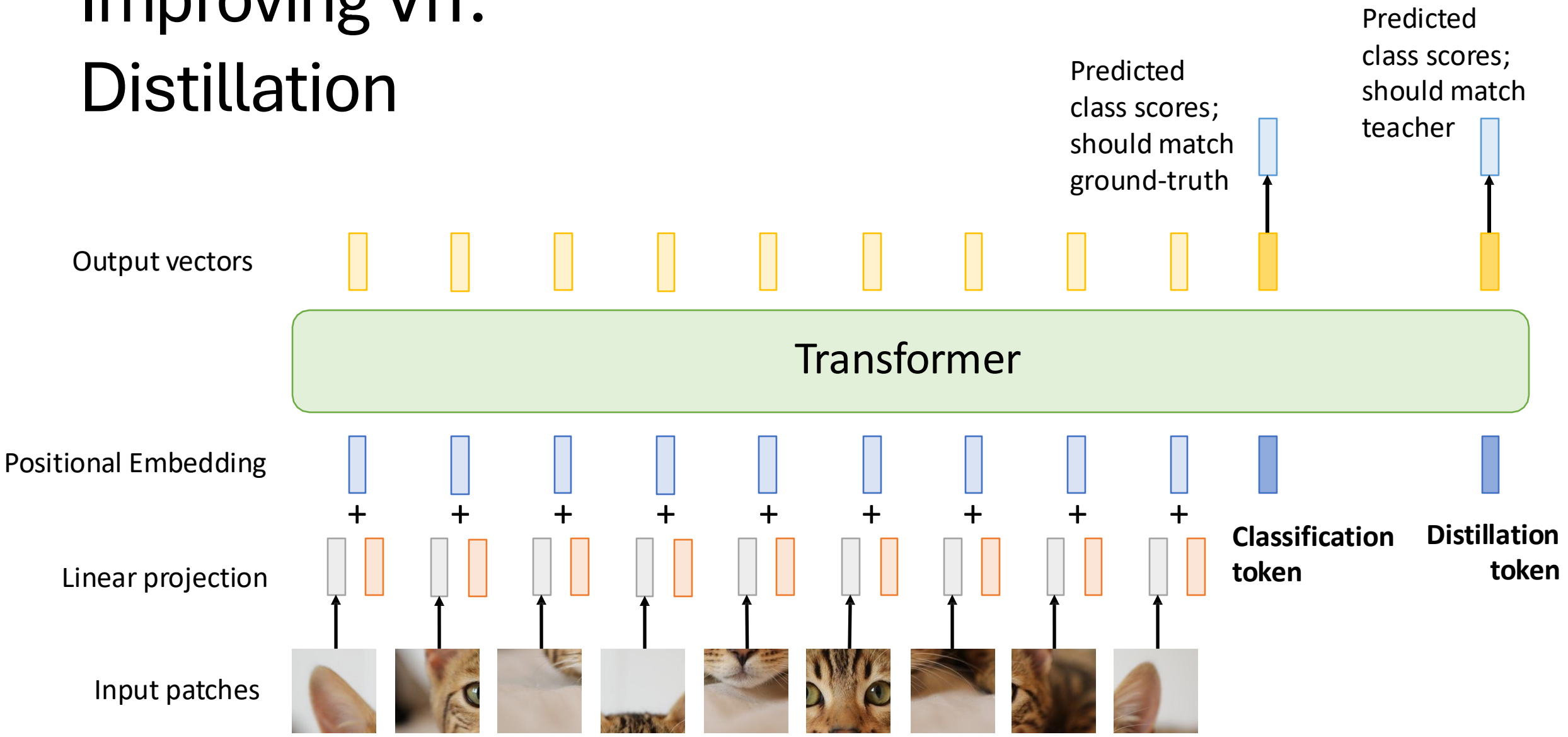
→ Cross Entropy Loss ← GT label: Dog

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

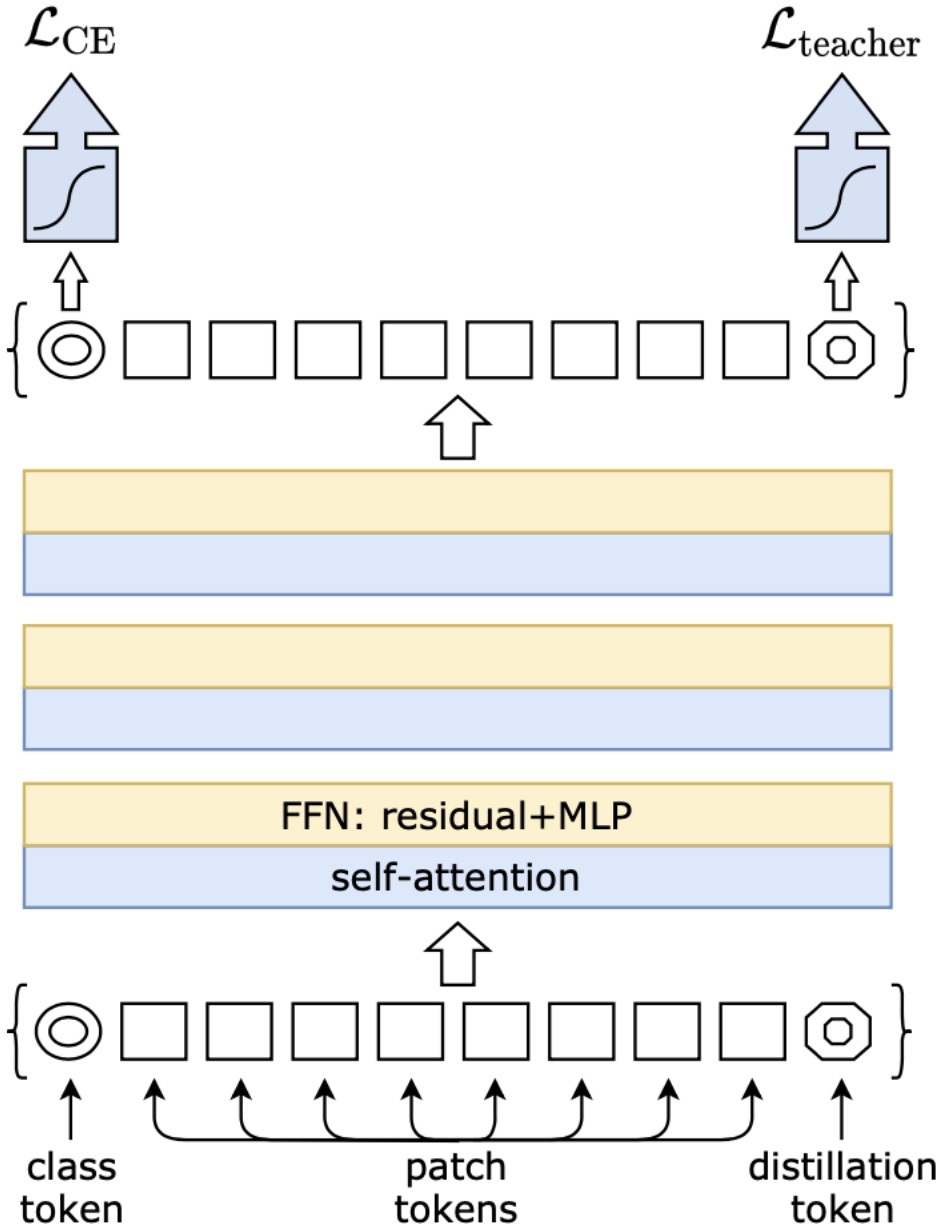
Transfer of Knowledge: In model distillation, the goal is to transfer knowledge from a larger, more complex teacher model to a smaller student model. By using a softer distribution, the teacher model conveys not just information about the correct class but also about the relationships between incorrect classes. This nuanced information helps the student model learn more about the underlying data structure.

Avoiding Overconfidence: Without temperature scaling, the teacher model's predictions might be overly confident, focusing only on the most likely class. By softening the distribution, the teacher model provides more balanced information, allowing the student model to learn from the teacher's uncertainty and make more robust predictions.

Improving ViT: Distillation



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



$L_{global} = 0.5L_{CE}(\psi(Z_s), y) + 0.5L_{CE}(\psi(Z_s), y_t)$

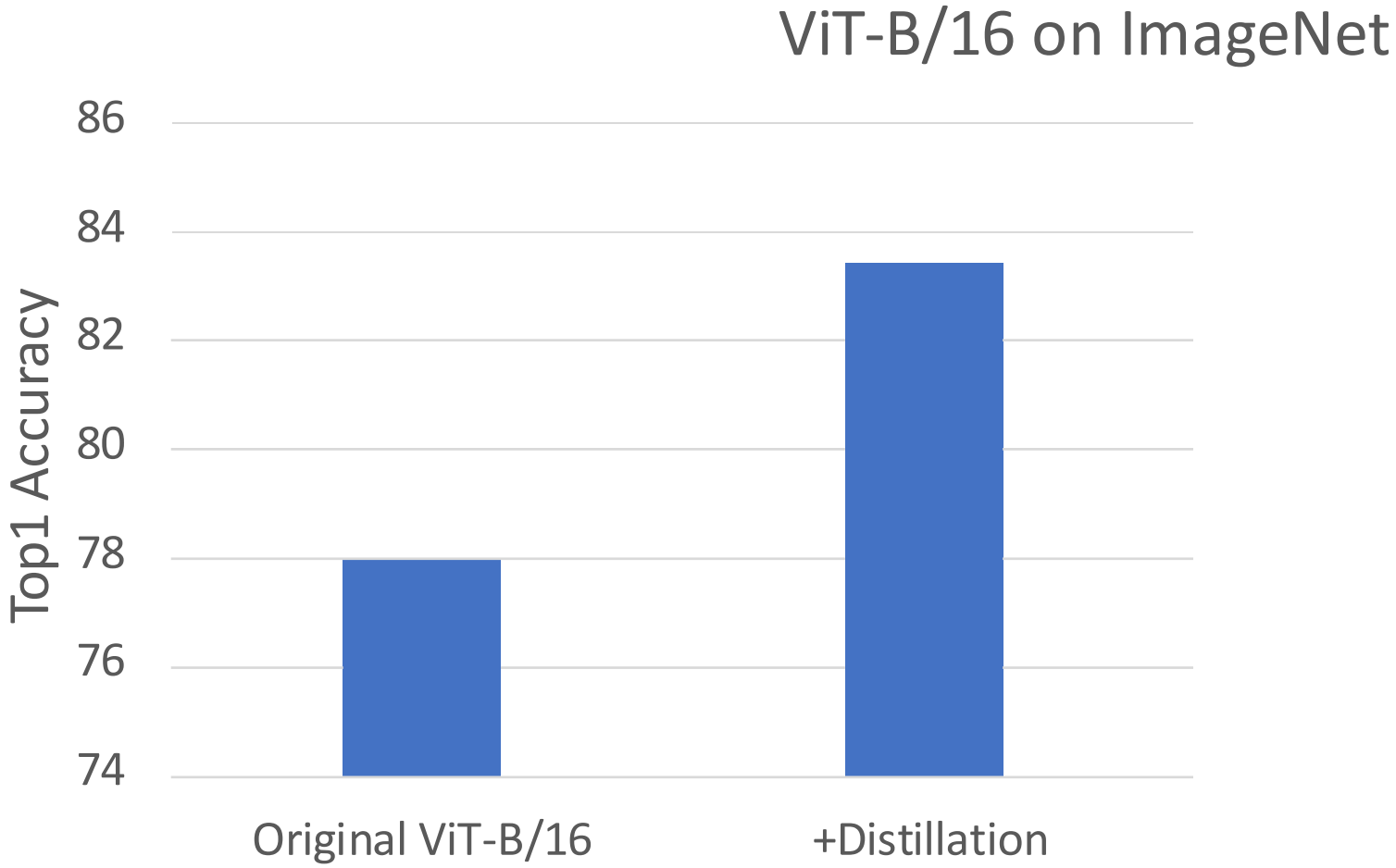
$y_t = \text{argmax}_c Z_t(c)$

~~Seems fishy, directly taken from the pape~~
~~Zt should be softmax probs.~~

- augmentation like mix-up can change the label, two different images are mixed, so are there labels.
- applied label smoothing, prediction gets prob of 1-eps, eps=0.1, applied uniformly to other classes

For prediction, softmax output is added

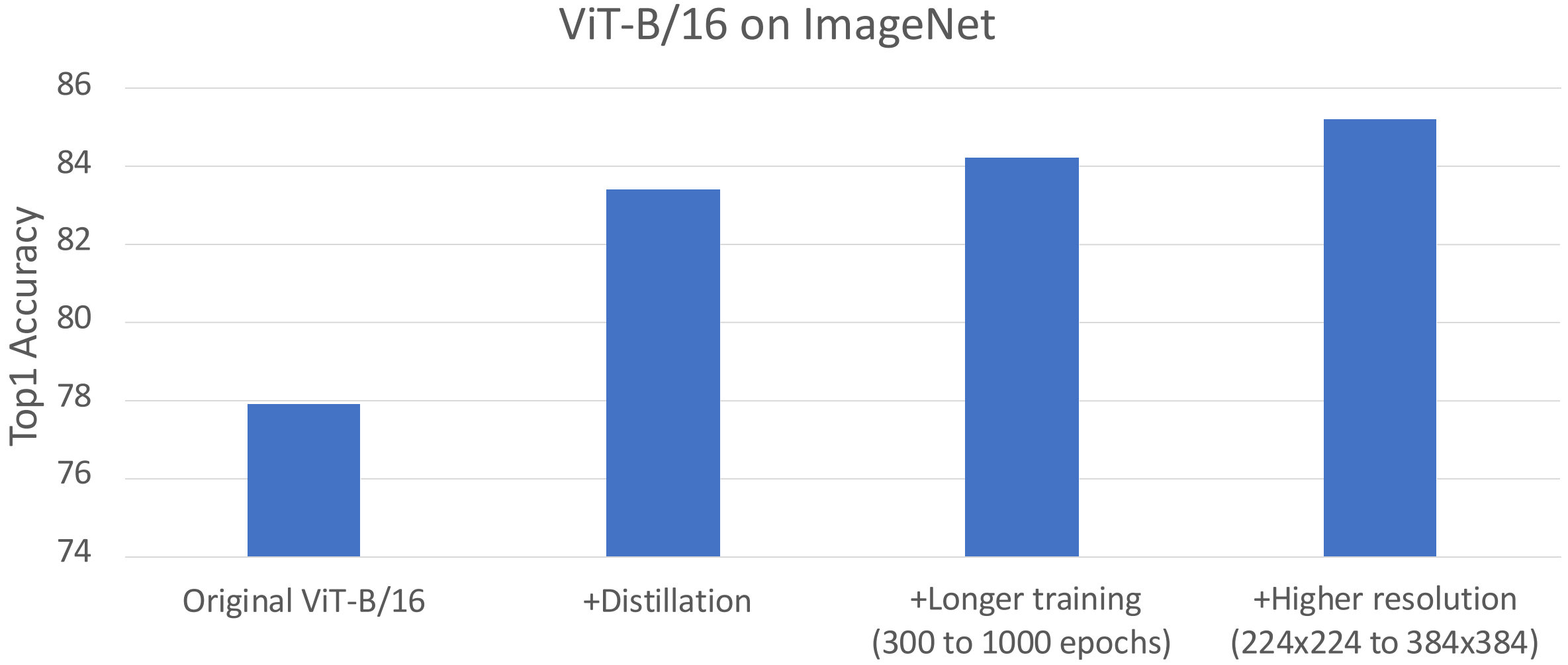
Improving ViT: Distillation



Teacher RegNetY-16GF (Radosavovic et al., 2020) with 84M parameters

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

a. Suppose you want to distill from resnet trained on imagenet to cifar-10. The Setting is few-shot with 10-way-5-shot classification.
b. Further to it, you need to deploy it on edge device.

Note, train data is less so you need to find a niche way to fine-tune resnet.

As edge will expect a small model and you have less train data so you cannot Train it in classical way.

How will you address the two problems?

Can you find some difference between ViT and CNN?

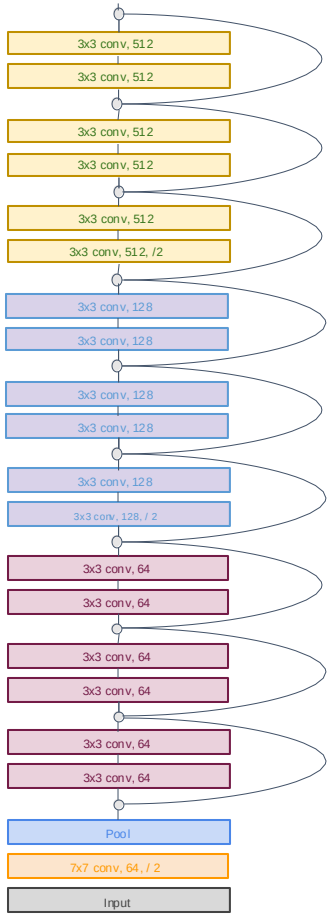
ViT vs CNN

Stage 3:
256 x 14 x 14

Stage 2:
128 x 28 x 28

Stage 1:
64 x 56 x 56

Input:
3 x 224 x 224



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

Useful since objects in images can occur at various scales

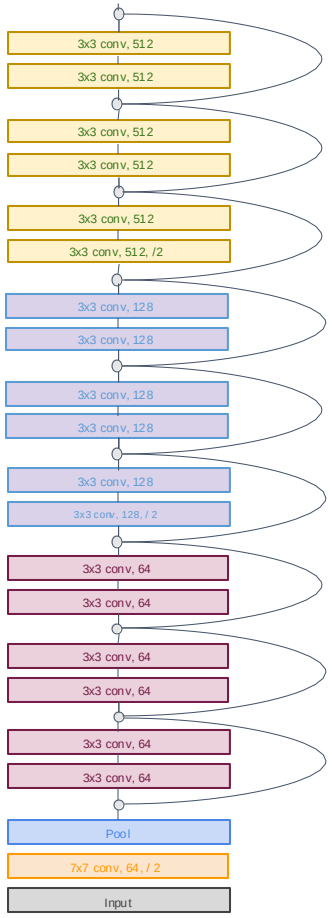
ViT vs CNN

Stage 3:
256 x 14 x 14

Stage 2:
128 x 28 x 28

Stage 1:
64 x 56 x 56

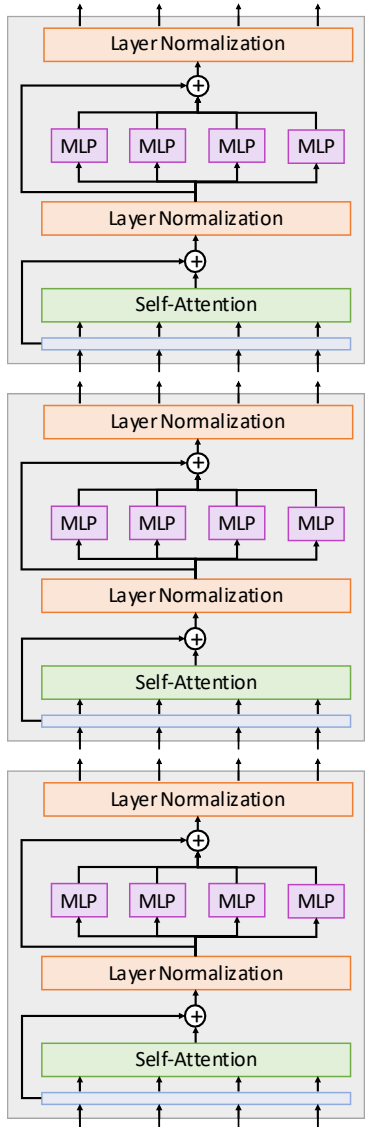
Input:
3 x 224 x 224



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

Useful since objects in images can occur at various scales

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



3rd block:
768 x 14 x 14

2nd block:
768 x 14 x 14

1st block:
768 x 14 x 14

Input:
3 x 224 x 224

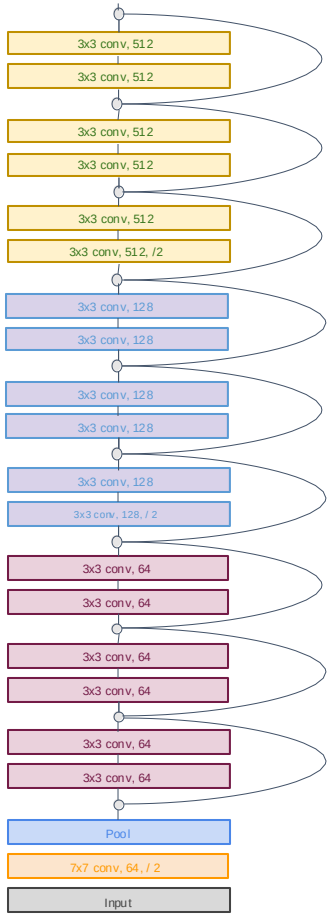
ViT vs CNN

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

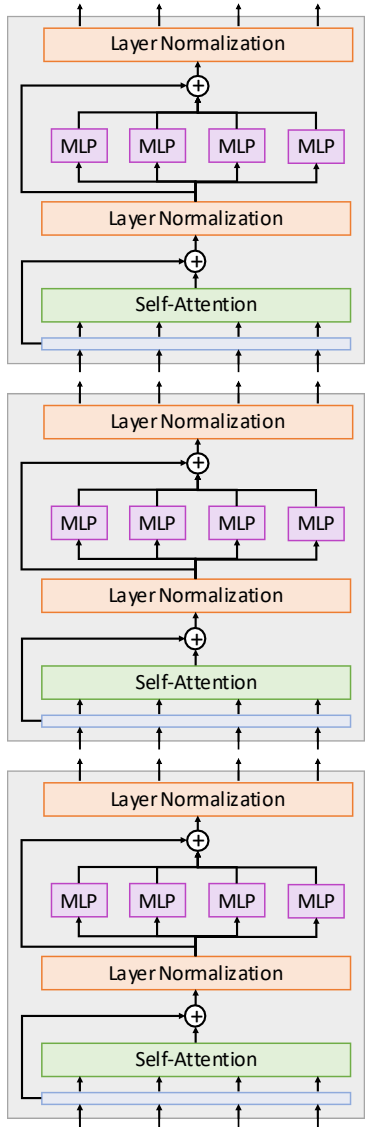


In most CNNs (including ResNets), **decrease** resolution and **increase** channels as you go deeper in the network (Hierarchical architecture)

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