The University of Texas at Austin
Electrical and Computer Engineering
*Cockrell School of Engineering*

**Fall 2021**

# ADVANCED TOPICS IN COMPUTER VISION

**Atlas Wang**
Assistant Professor, The University of Texas at Austin

**Visual Informatics Group@UT Austin**
https://vita-group.github.io/

# Why Transformer for Vision?

- Towards a **general, conceptual simple**, and **sufficiently versatile** architecture yet still achieving competitive performance for vision?
- The inductive bias of CNNs, e.g., spatially invariant and locality-based, also may not be sufficient …


# Basics: Transformer in NLP

- Standard model in NLP tasks
- Only consists of self-attention modules, instead of RNN
- Encoder-decoder
- Requires large dataset and high computational cost
- Pre-training and fine-tuning approaches : BERT & GPT

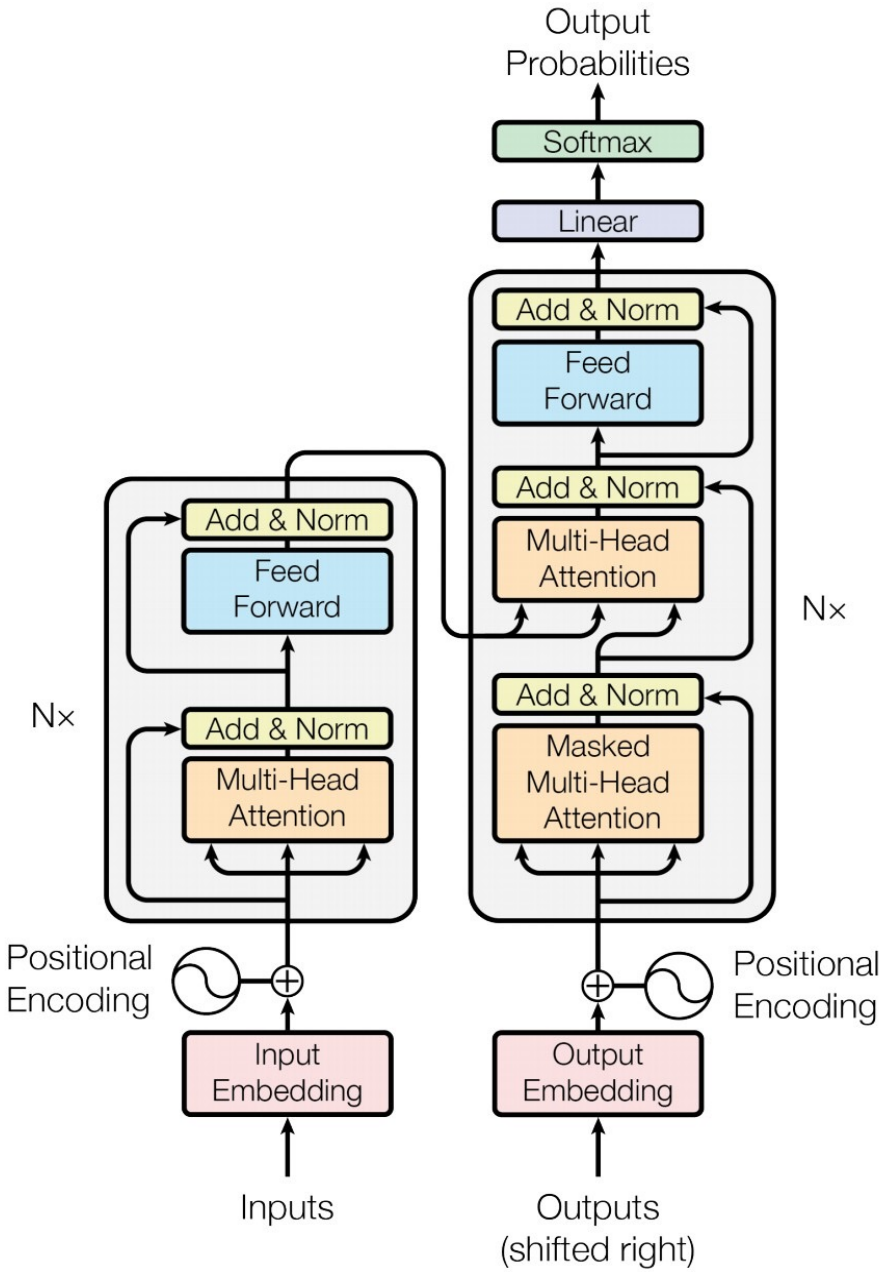
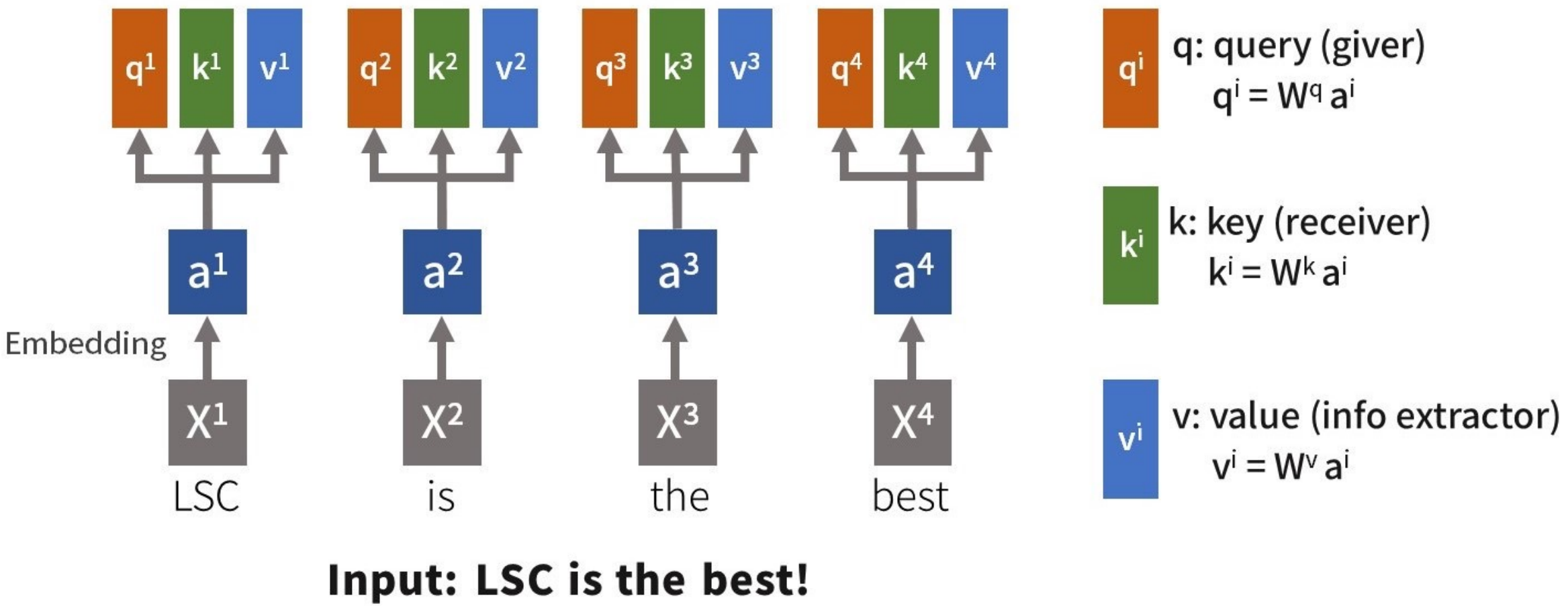
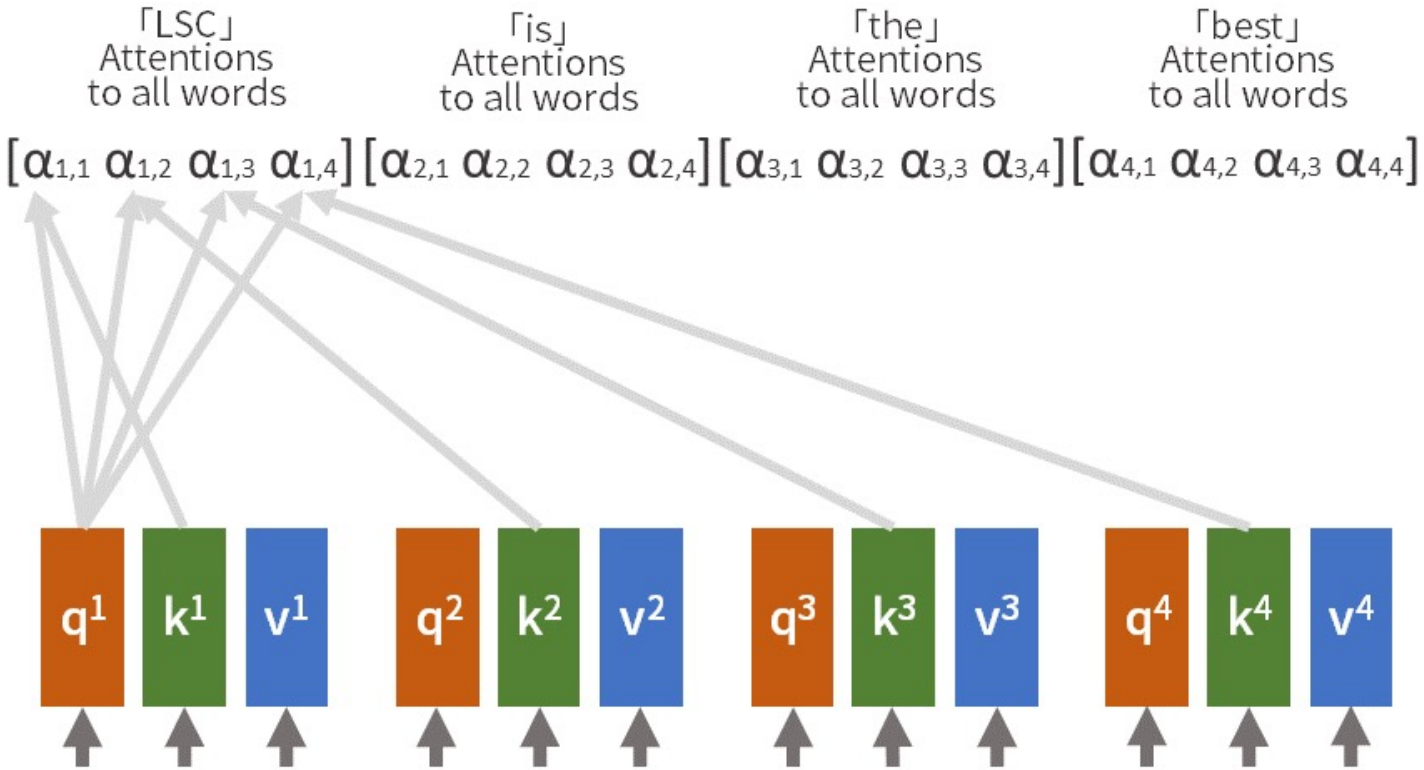



Figure 1: The Transformer - model architecture.

# Basics: Self-Attention



# Basics: Self-Attention



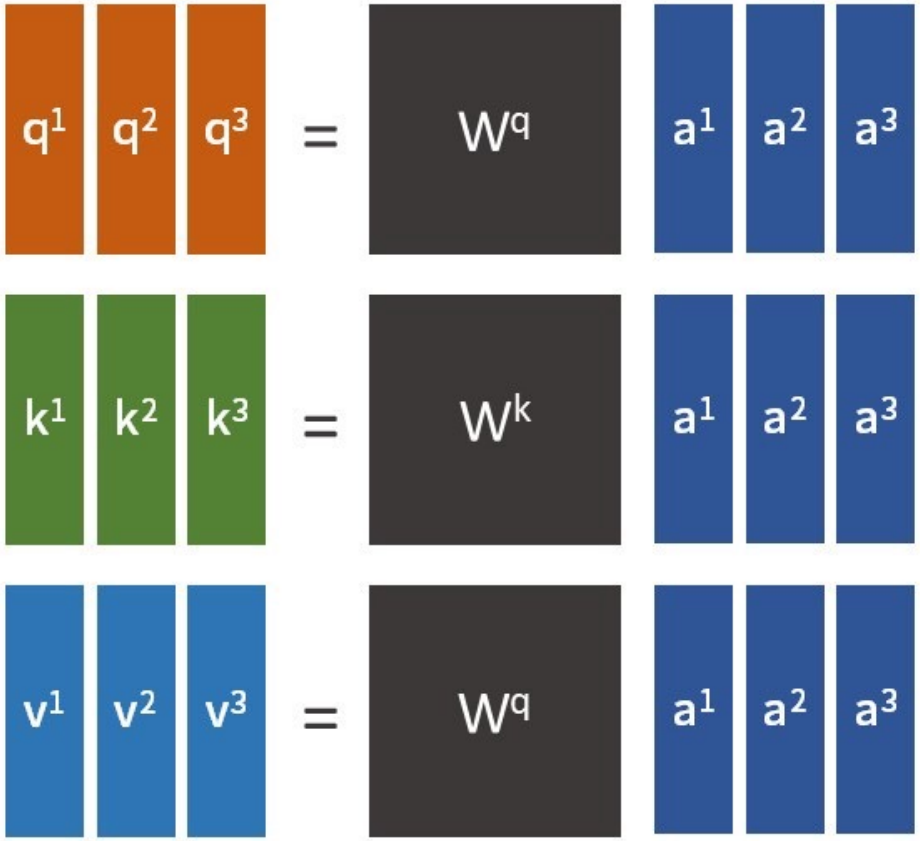

$$\alpha_{i,j} = \frac{q^i \cdot k^j}{\sqrt{d}}$$

d: dimension of q, k

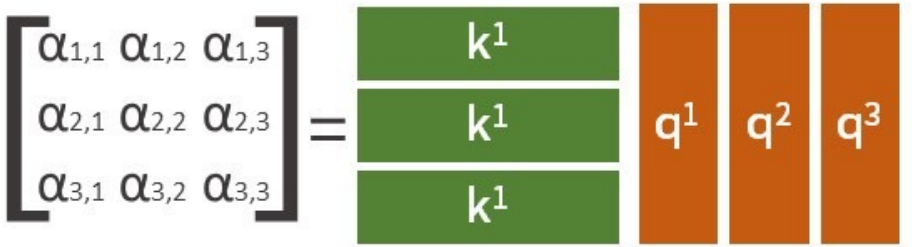
$$A = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} \\ \alpha_{4,1} & \alpha_{4,2} & \alpha_{4,3} & \alpha_{4,4} \end{bmatrix}$$

Attention Matrix

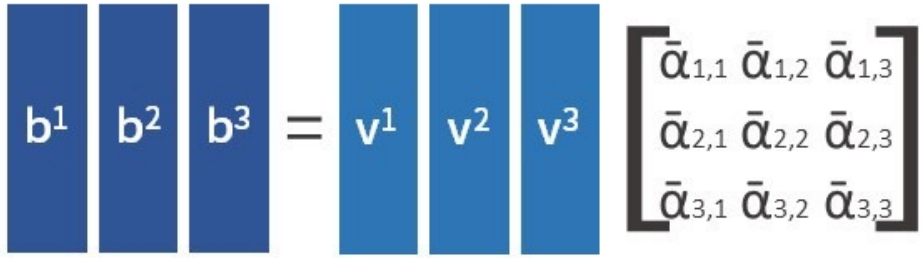
# Basics: Self-Attention




Attention A:




Output:



# Bringing Transformers into Computer Vision

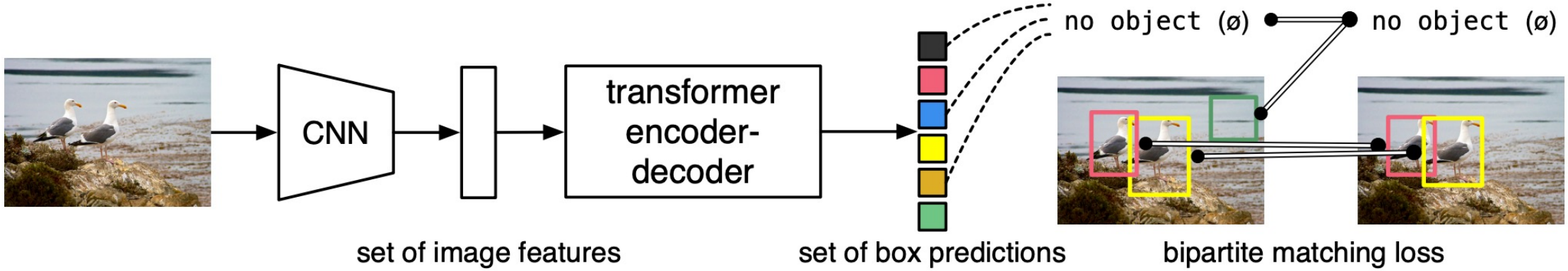
- Only in local neighborhoods (1)
  - Image Transformer, ICML 2018
  - Stand-alone self-attention in vision models, NeurIPS 2019
  - On the relationship between self-attention and convolutional layers, ICLR 2020
  - Exploring self-attention for image recognition, CVPR 2020
- Scalable approximations to global self-attention (2)
  - Generating long sequences with sparse transformers, arXiv 2019
- Blocks of varying sizes (3)
  - Scaling autoregressive video models, ICLR 2019
- Only along individual axes (4)
  - Axial attention in multidimensional transformers, arXiv 2019
  - Axial-deeplab: Stand-alone axial-attention for panoptic segmentation, ECCV 2020

# Bringing Transformers into Computer Vision

- Combining CNN with self-attention (5)
  - Attention augmented convolutional networks, ICCV 2019, image classification
  - End-to-end object detection with transformers, ECCV 2020, object detection
  - Videobert: A joint model for video and language representation learning, ICCV 2019, video processing
  - Visual transformers, arxiv 2020, image classification
- Unified text-vision tasks (6)
  - VQA
  - Image Retrieval
  - OCR (Document Layout Analysis)
- Most Related Works (7)
  - Generative pretraining from pixels (iGPT), ICML 2020
  - Big Transfer (BiT): General Visual Representation Learning, ECCV 2020

# DETR: End-to-End Object Detection with Transformers (ECCV'20)

- DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. It does **NOT** rely on the many hand-designed components like in FasterRCNN.




- **The takeaway from DETR is bi-folds:**
  - DETR achieved comparable performance to Faster R-CNN, but not on par with more recent detectors (especially on small objects), also requiring extra-long training schedule and auxiliary decoding losses
  - DETR showed significant promise of generalizability, e.g., the same model easily applied to panoptic segmentation in a unified manner

# “Pure Transformer”: Visual Transformer (ViT, ICLR’21)



GIF from https://github.com/lucidrains/vit-pytorch

# Implementation

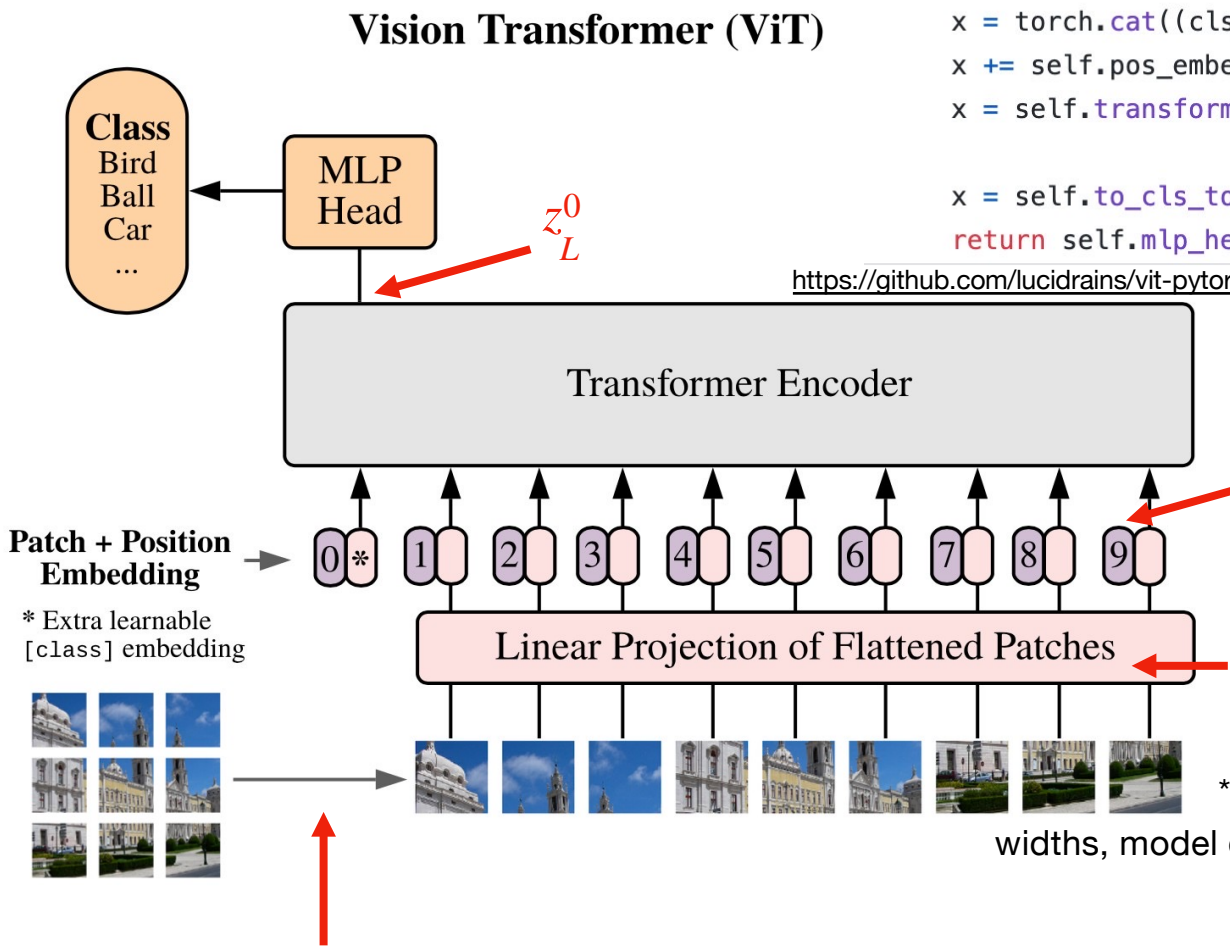
```
def forward(self, img, mask = None):
    p = self.patch_size

    x = rearrange(img, 'b c (h p1) (w p2) -> b (h w) (p1 p2
    x = self.patch_to_embedding(x)

    cls_tokens = self.cls_token.expand(img.shape[0], -1, -1)
    x = torch.cat((cls_tokens, x), dim=1)
    x += self.pos_embedding
    x = self.transformer(x, mask)

    x = self.to_cls_token(x[:, 0])
    return self.mlp_head(x)
```

https://github.com/lucidrains/vit-pytorch/blob/main/vit_pytorch/vit_pytorch.py#L99-L111




Learnable Position Embedding
$E_{pos} \in R^{(N+1)\times D}$

* to retain positional information

Trainable linear projection maps
$x_p \in R^{N\times(P^2\cdot C)} \rightarrow x_p E \in R^{N\times D}$

* Because Transformer uses constant widths, model dimension , through all of its layers

Image $x \in R^{H\times W\times C}$ → A sequence of flattened 2D patches $x_p \in R^{N\times(P^2\cdot C)}$

# Implementation

```python
class Transformer(nn.Module):
    def __init__(self, dim, depth, heads, mlp_dim):
        super().__init__()
        self.layers = nn.ModuleList([])
        for _ in range(depth):
            self.layers.append(nn.ModuleList([
                Residual(PreNorm(dim, Attention(dim, heads = heads))),
                Residual(PreNorm(dim, FeedForward(dim, mlp_dim)))
            ]))
    def forward(self, x, mask = None):
        for attn, ff in self.layers:
            x = attn(x, mask = mask)
            x = ff(x)
        return x
```

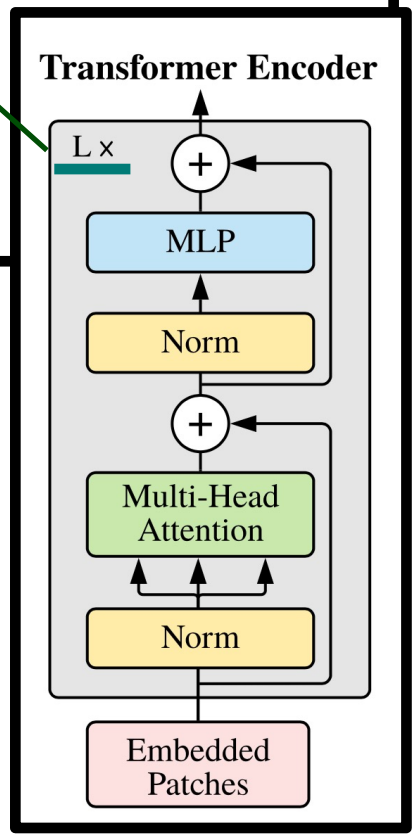
```python
def forward(self, img, mask = None):
    p = self.patch_size

    x = rearrange(img, 'b c (h p1) (w p2) -> b (h w) (p1 p2 c)', p1 = p, p2 = p)
    x = self.patch_to_embedding(x)

    cls_tokens = self.cls_token.expand(img.shape[0], -1, -1)
    x = torch.cat((cls_tokens, x), dim=1)
    x += self.pos_embedding
    x = self.transformer(x, mask)

    x = self.to_cls_token(x[:, 0])
    return self.mlp_head(x)
```

https://github.com/lucidrains/vit-pytorch/blob/main/vit_pytorch/vit_pytorch.py



# Implementation

$z \in R^{N \times D}$ : input sequence

$$[\mathbf{q}, \mathbf{k}, \mathbf{v}] = \mathbf{z}\mathbf{U}_{qkv} \qquad \mathbf{U}_{qkv} \in \mathbb{R}^{D \times 3D_h},$$

$$A = \mathrm{softmax}\left(\mathbf{q}\mathbf{k}^\top / \sqrt{D_h}\right) \qquad A \in \mathbb{R}^{N \times N},$$

$$\mathrm{SA}(\mathbf{z}) = A\mathbf{v}\,.$$

Attention weight $A_{ij}$ : similarity btw $q^i, k^j$

$$\mathrm{MSA}(\mathbf{z}) = [\mathrm{SA}_1(z); \mathrm{SA}_2(z); \cdots; \mathrm{SA}_k(z)]\,\mathbf{U}_{msa} \qquad \mathbf{U}_{msa} \in \mathbb{R}^{k \cdot D_h \times D}$$

```python
class Attention(nn.Module):
    def __init__(self, dim, heads = 8):
        super().__init__()
        self.heads = heads
        self.scale = dim ** -0.5

        self.to_qkv = nn.Linear(dim, dim * 3, bias = False)
        self.to_out = nn.Linear(dim, dim)
    def forward(self, x, mask = None):
        b, n, _, h = *x.shape, self.heads
        qkv = self.to_qkv(x)
        q, k, v = rearrange(qkv, 'b n (qkv h d) -> qkv b h n d', qkv = 3, h = h)

        dots = torch.einsum('bhid,bhjd->bhij', q, k) * self.scale

        if mask is not None:
            mask = F.pad(mask.flatten(1), (1, 0), value = True)
            assert mask.shape[-1] == dots.shape[-1], 'mask has incorrect dimensions'
            mask = mask[:, None, :] * mask[:, :, None]
            dots.masked_fill_(~mask, float('-inf'))
            del mask

        attn = dots.softmax(dim=-1)

        out = torch.einsum('bhij,bhjd->bhid', attn, v)
        out = rearrange(out, 'b h n d -> b n (h d)')
        out =  self.to_out(out)
        return out
```

https://github.com/lucidrains/vit-pytorch/blob/main/vit_pytorch/vit_pytorch.py

# Experiments

| | Ours (ViT-H/14) | Ours (ViT-L/16) | BiT-L (ResNet152x4) | Noisy Student (EfficientNet-L2) |
|---|---|---|---|---|
| ImageNet | 88.36 | $87.61 \pm 0.03$ | $87.54 \pm 0.02$ | $88.4/\mathbf{88.5}^{*}$ |
| ImageNet ReaL | **90.77** | $90.24 \pm 0.03$ | 90.54 | 90.55 |
| CIFAR-10 | **99.50** $\pm 0.06$ | $99.42 \pm 0.03$ | $99.37 \pm 0.06$ | – |
| CIFAR-100 | **94.55** $\pm 0.04$ | $93.90 \pm 0.05$ | $93.51 \pm 0.08$ | – |
| Oxford-IIIT Pets | **97.56** $\pm 0.03$ | $97.32 \pm 0.11$ | $96.62 \pm 0.23$ | – |
| Oxford Flowers-102 | $99.68 \pm 0.02$ | **99.74** $\pm 0.00$ | $99.63 \pm 0.03$ | – |
| VTAB (19 tasks) | **77.16** $\pm 0.29$ | $75.91 \pm 0.18$ | $76.29 \pm 1.70$ | – |
| TPUv3-days | 2.5k | 0.68k | 9.9k | 12.3k |

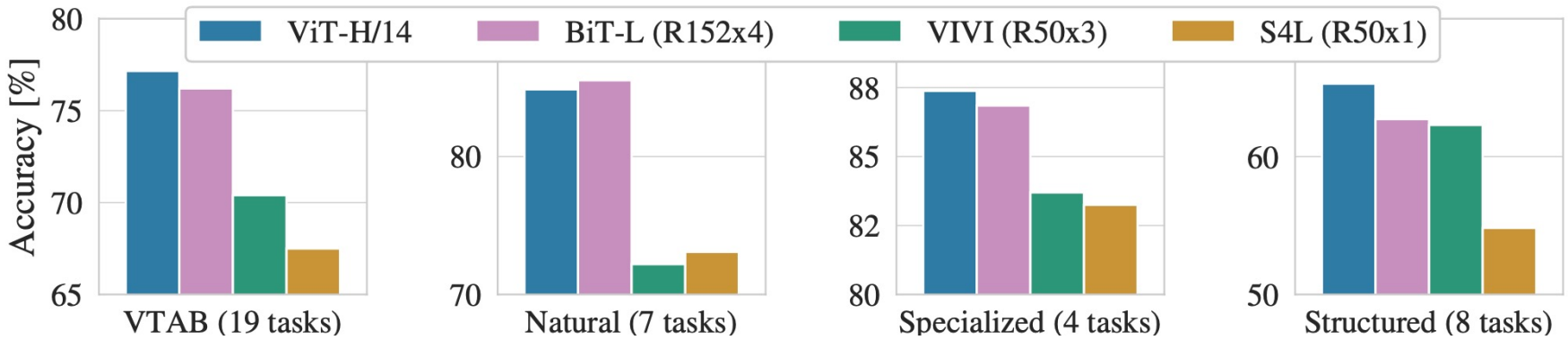



Figure 2: Breakdown of VTAB performance in *Natural*, *Specialized*, and *Structured* task groups.

# Experiments

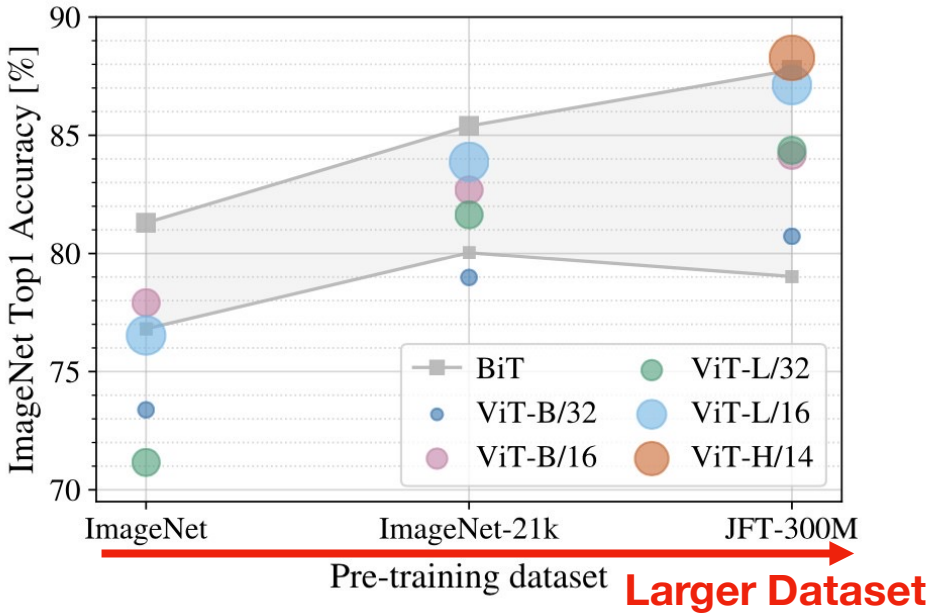



Figure 3: Transfer to ImageNet. While large ViT models perform worse than BiT ResNets (shaded area) when pre-trained on small datasets, they shine when pre-trained on larger datasets. Similarly, larger ViT variants overtake smaller ones as the dataset grows.

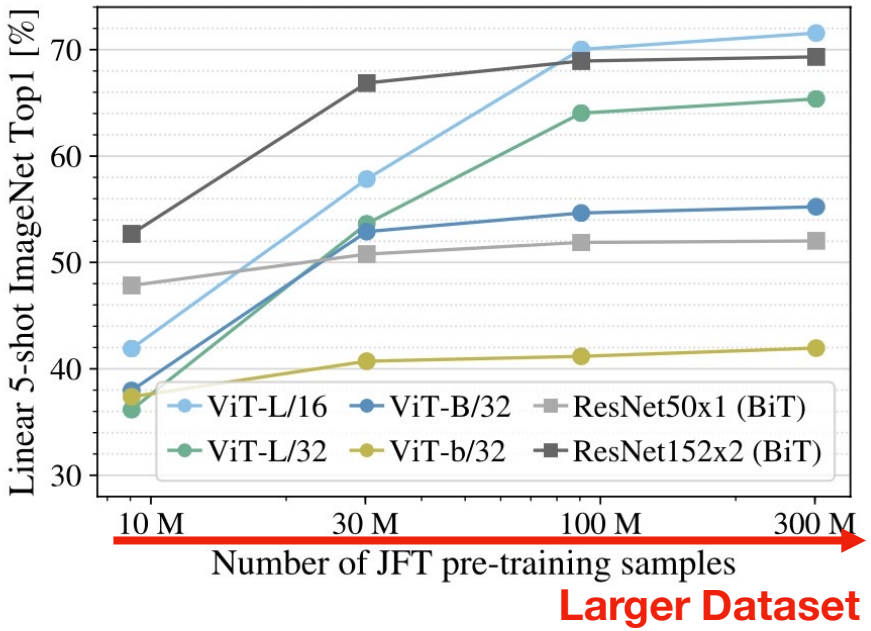
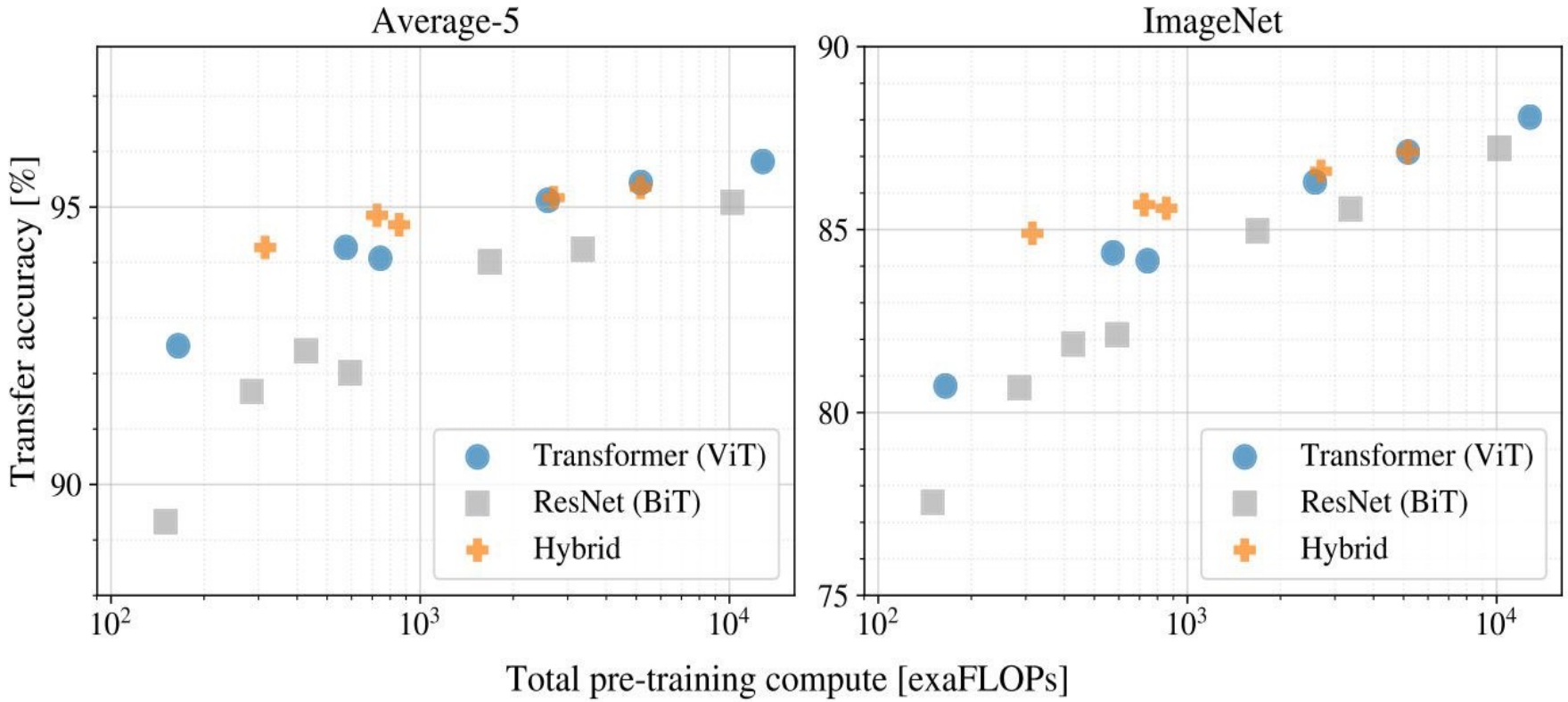



Figure 4: Linear few-shot evaluation on ImageNet versus pre-training size. ResNets perform better with smaller pre-training datasets but plateau sooner than ViT which performs better with larger pre-training. ViT-b is ViT-B with all hidden dimensions halved.

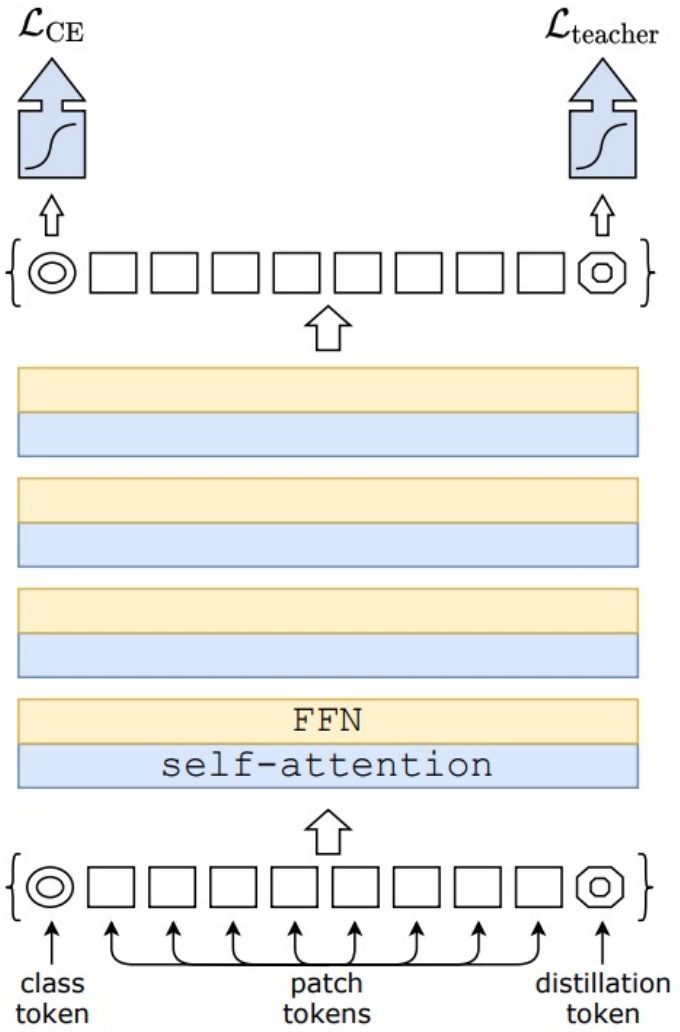
# Experiments



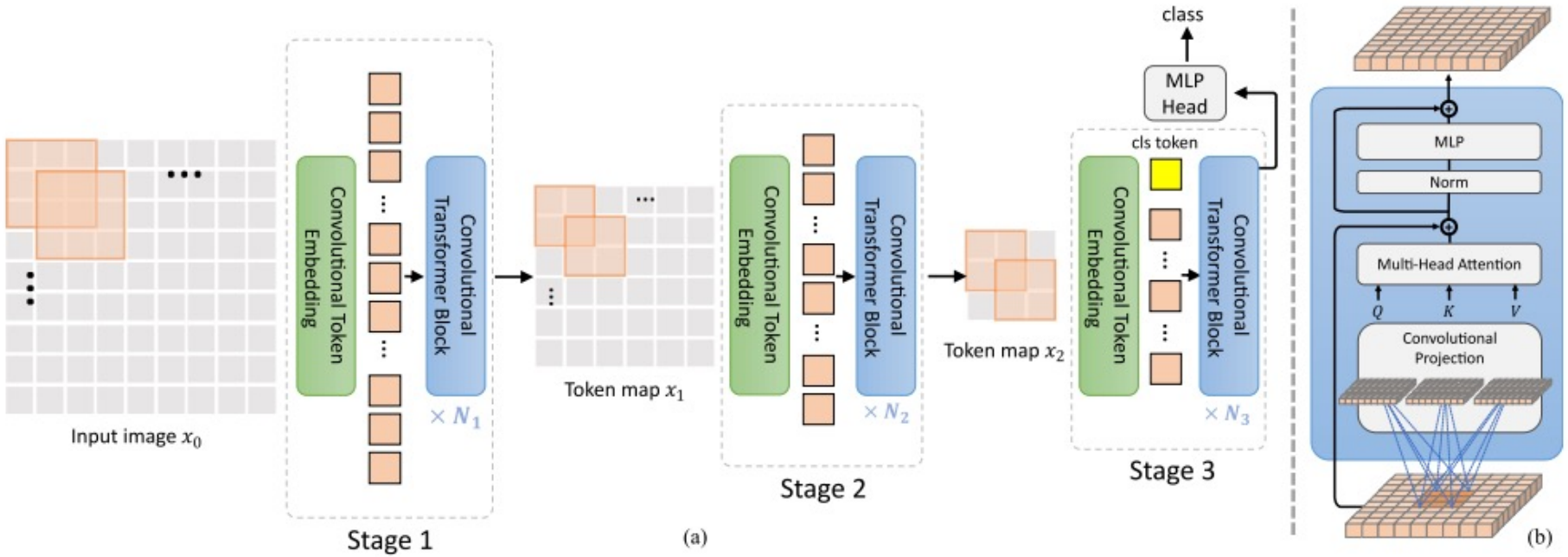

Performance versus cost for different architectures: Vision Transformers, ResNets, and hybrids. Vision Transformers generally outperform ResNets with the same computational budget. Hybrids improve upon pure Transformers for smaller model sizes, but the gap vanishes for larger models.

# DeIT: Data-efficient Image Transformers

- The first competitive convolution-free transformer by training on Imagenet only
- Trained using a teacher-student strategy specific to transformers
  - It relies on a distillation token ensuring that the student learns from the teacher through attention.
- When using CNN as teacher, the distilled model outperforms its teacher in terms of the trade-off between accuracy and throughput

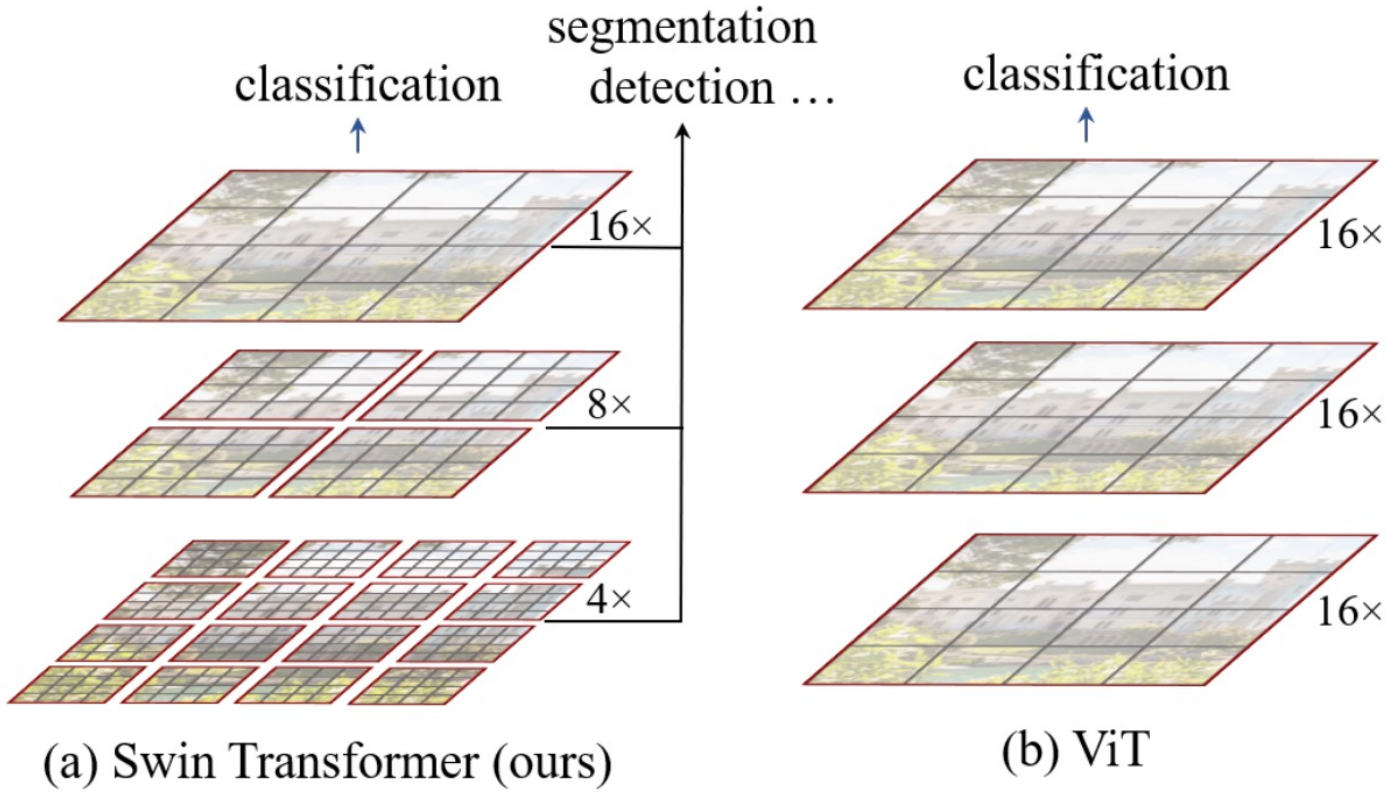


# CvT: Convolutions into Vision Transformers




- Each stage starts with a convolutional token embedding that performs an overlapping convolution operation on a 2D-reshaped token map
- The linear projection prior to every self-attention block is replaced with a depth-wise separable convolution as the projection

# Swin Transformer (ICCV'21 best paper)




- Swin: hierarchical feature maps by merging image patches
  - linear computation complexity to input image size due to computation of self-attention only within each local window (using **S**hifted **win**dows)

# Swin Transformer: Pipeline Overview

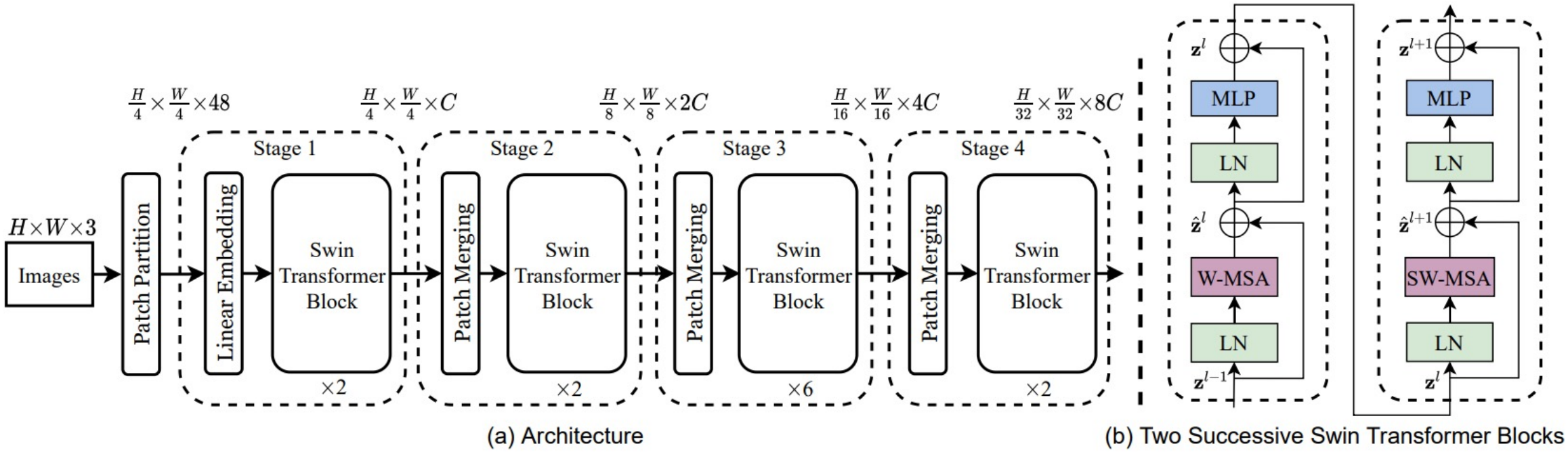
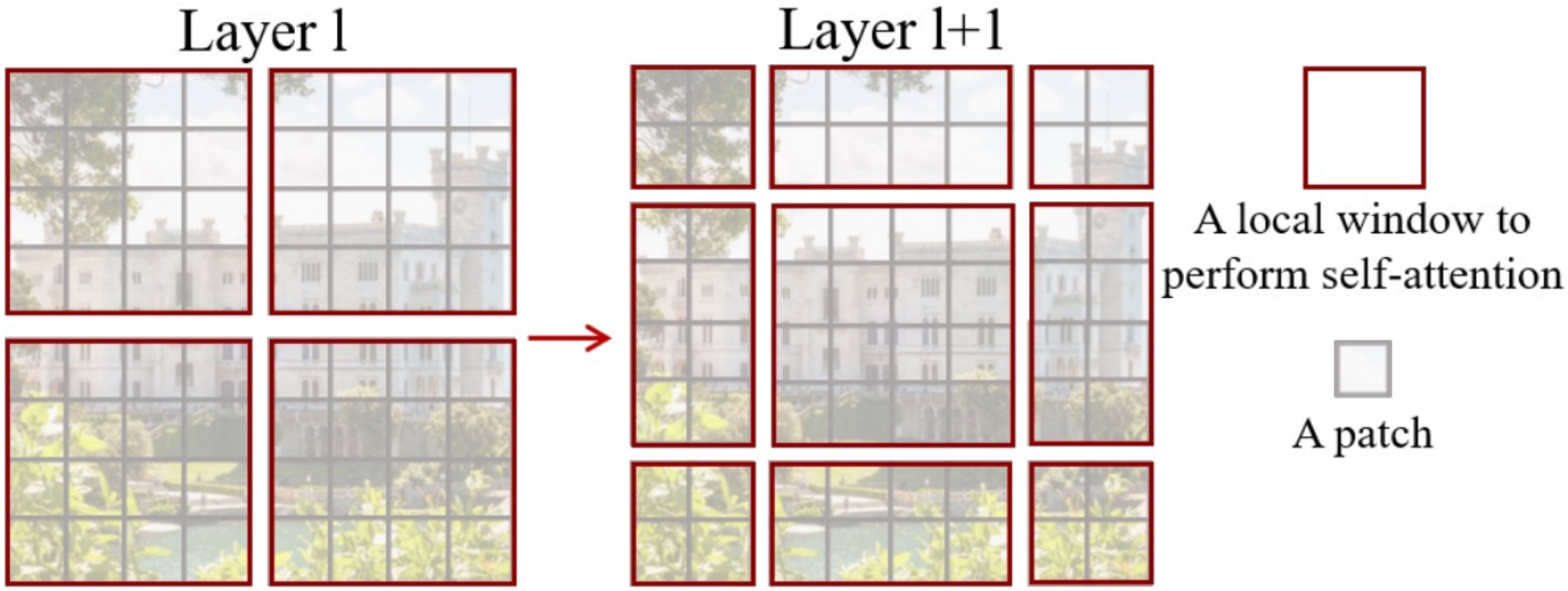
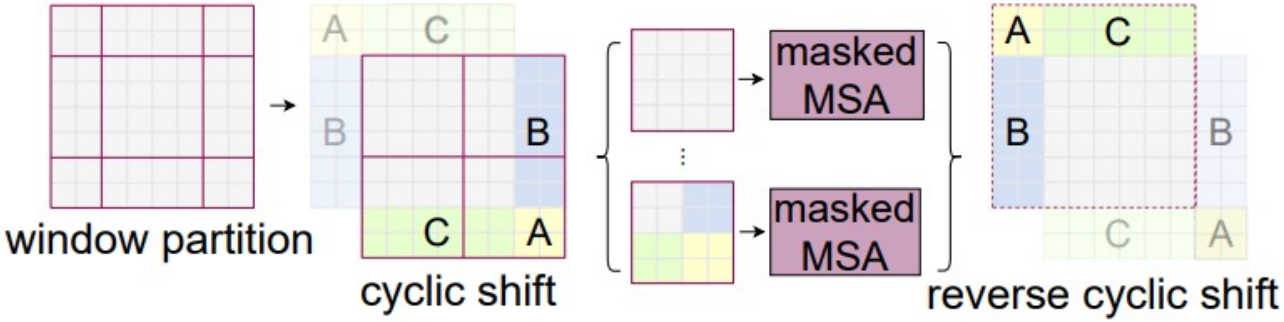



Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

# Swin Transformer: Shifted Window



# TimeSformer: ViT for Video

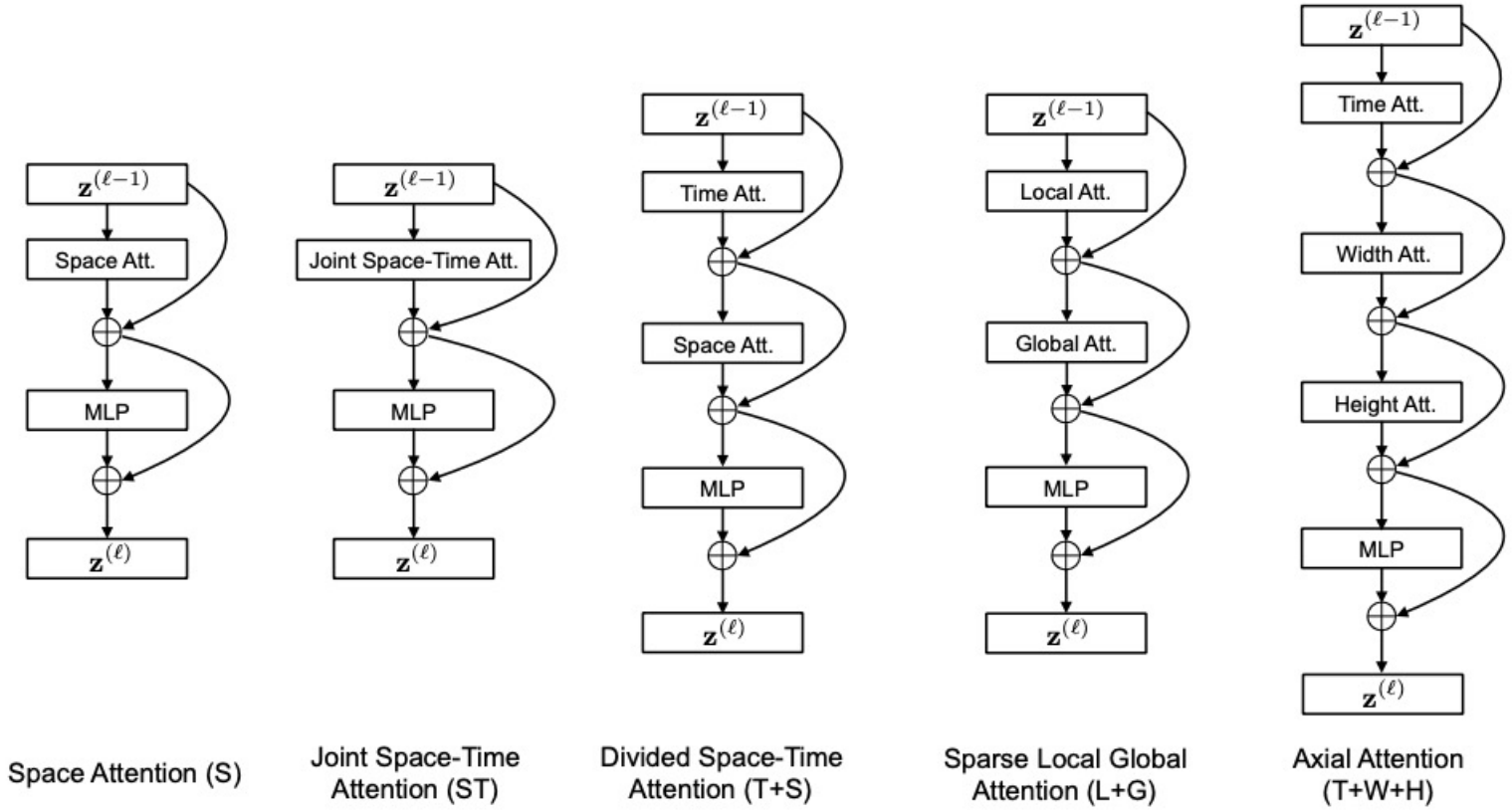



*Figure 1.* The video self-attention blocks that we investigate in this work. Each attention layer implements self-attention (Vaswani et al., 2017b) on a specified spatiotemporal neighborhood of frame-level patches (see Figure 2 for a visualization of the neighborhoods). We use residual connections to aggregate information from different attention layers within each block. A 1-hidden-layer MLP is applied at the end of each block. The final model is constructed by repeatedly stacking these blocks on top of each other.

# TimeSformer: ViT for Video

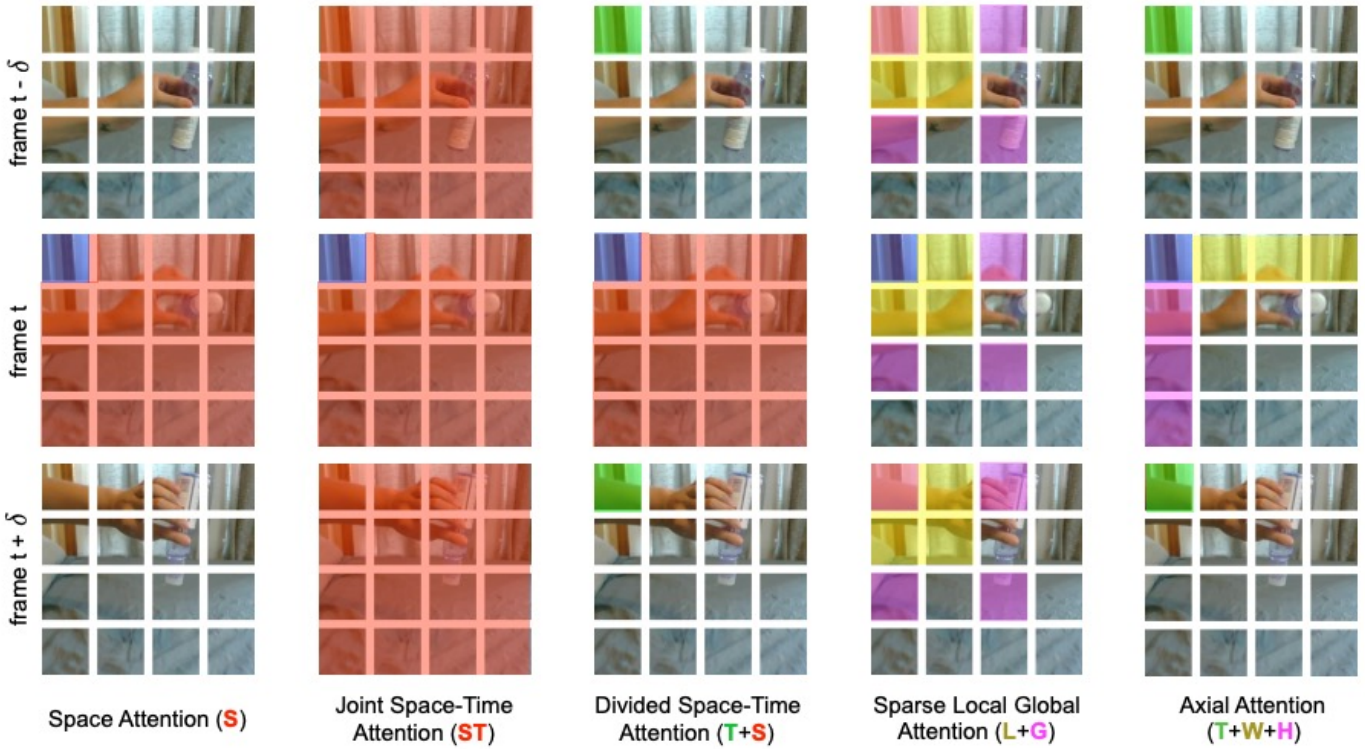



*Figure 2.* Visualization of the five space-time self-attention schemes studied in this work. Each video clip is viewed as a sequence of frame-level patches with a size of $16 \times 16$ pixels. For illustration, we denote in blue the query patch and show in non-blue colors its self-attention space-time neighborhood under each scheme. Patches without color are not used for the self-attention computation of the blue patch. Multiple colors within a scheme denote attentions separately applied along different dimensions (e.g., space and time for (T+S)) or over different neighborhoods (e.g., for (L+G)). Note that self-attention is computed for every single patch in the video clip, i.e., every patch serves as a query. We also note that although the attention pattern is shown for only two adjacent frames, it extends in the same fashion to all frames of the clip.

# TimeSformer: ViT for Video

| Attention | Params | K400 | SSv2 |
|---|---|---|---|
| Space | 85.9M | 76.9 | 36.6 |
| Joint Space-Time | 85.9M | 77.4 | 58.5 |
| Divided Space-Time | 121.4M | **78.0** | **59.5** |
| Sparse Local Global | 121.4M | 75.9 | 56.3 |
| Axial | 156.8M | 73.5 | 56.2 |

*Table 1.* Video-level accuracy for different space-time attention schemes in TimeSformer. We evaluate the models on the validation sets of Kinetics-400 (K400), and Something-Something-V2 (SSv2). We observe that divided space-time attention achieves the best results on both datasets.

| Model | Pretrain | K400 Training Time (hours) | K400 Acc. | Inference TFLOPs | Params |
|---|---|---|---|---|---|
| I3D 8x8 R50 | ImageNet-1K | 444 | 71.0 | 1.11 | 28.0M |
| I3D 8x8 R50 | ImageNet-1K | 1440 | 73.4 | 1.11 | 28.0M |
| SlowFast R50 | ImageNet-1K | 448 | 70.0 | 1.97 | 34.6M |
| SlowFast R50 | ImageNet-1K | 3840 | 75.6 | 1.97 | 34.6M |
| SlowFast R50 | N/A | 6336 | 76.4 | 1.97 | 34.6M |
| TimeSformer | ImageNet-1K | **416** | 75.8 | **0.59** | 121.4M |
| TimeSformer | ImageNet-21K | **416** | **78.0** | **0.59** | 121.4M |

*Table 2.* Comparing TimeSformer to SlowFast and I3D. We observe that TimeSformer has lower inference cost despite having a larger number of parameters. Furthermore, the cost of training TimeSformer on video data is much lower compared to SlowFast and I3D, even when all models are pretrained on ImageNet-1K.

| Method | Top-1 | Top-5 | TFLOPs |
|---|---|---|---|
| R(2+1)D (Tran et al., 2018) | 72.0 | 90.0 | 17.5 |
| bLVNet (Fan et al., 2019) | 73.5 | 91.2 | 0.84 |
| TSM (Lin et al., 2019) | 74.7 | N/A | N/A |
| S3D-G (Xie et al., 2018) | 74.7 | 93.4 | N/A |
| Oct-I3D+NL (Chen et al., 2019) | 75.7 | N/A | 0.84 |
| D3D (Stroud et al., 2020) | 75.9 | N/A | N/A |
| I3D+NL (Wang et al., 2018b) | 77.7 | 93.3 | 10.8 |
| ip-CSN-152 (Tran et al., 2019) | 77.8 | 92.8 | 3.2 |
| CorrNet (Wang et al., 2020a) | 79.2 | N/A | 6.7 |
| LGD-3D-101 (Qiu et al., 2019) | 79.4 | 94.4 | N/A |
| SlowFast (Feichtenhofer et al., 2019b) | 79.8 | 93.9 | 7.0 |
| X3D-XXL (Feichtenhofer, 2020) | 80.4 | 94.6 | 5.8 |
| TimeSformer | 78.0 | 93.7 | **0.59** |
| TimeSformer-HR | 79.7 | 94.4 | 5.11 |
| TimeSformer-L | **80.7** | **94.7** | 7.14 |

*Table 5.* Video-level accuracy on Kinetics-400.

# DINO: Self-Supervised Learning with ViTs

Source: https://ai.facebook.com/blog/dino-paws-computer-vision-with-self-supervised-transformers-and-10x-more-efficient-training/

# DINO: Self-Supervised Learning with ViTs

# DINO: Self-Supervised Learning with ViTs

*Supervised*

*DINO*

# A Debate: ViTs Should Go More Complicated or Less?

- Adding “convolution-like” inductive bias and structures
  - Injecting convolution layers, pyramid structure, dense connections, sliding windows, multi-sized views or attention windows …
- … Or just, keep it simple and “universal”?
  - Always my personal preference
  - **Someone pushed it even further…**

# MLP-Mixer: Is there any “indispensable”?

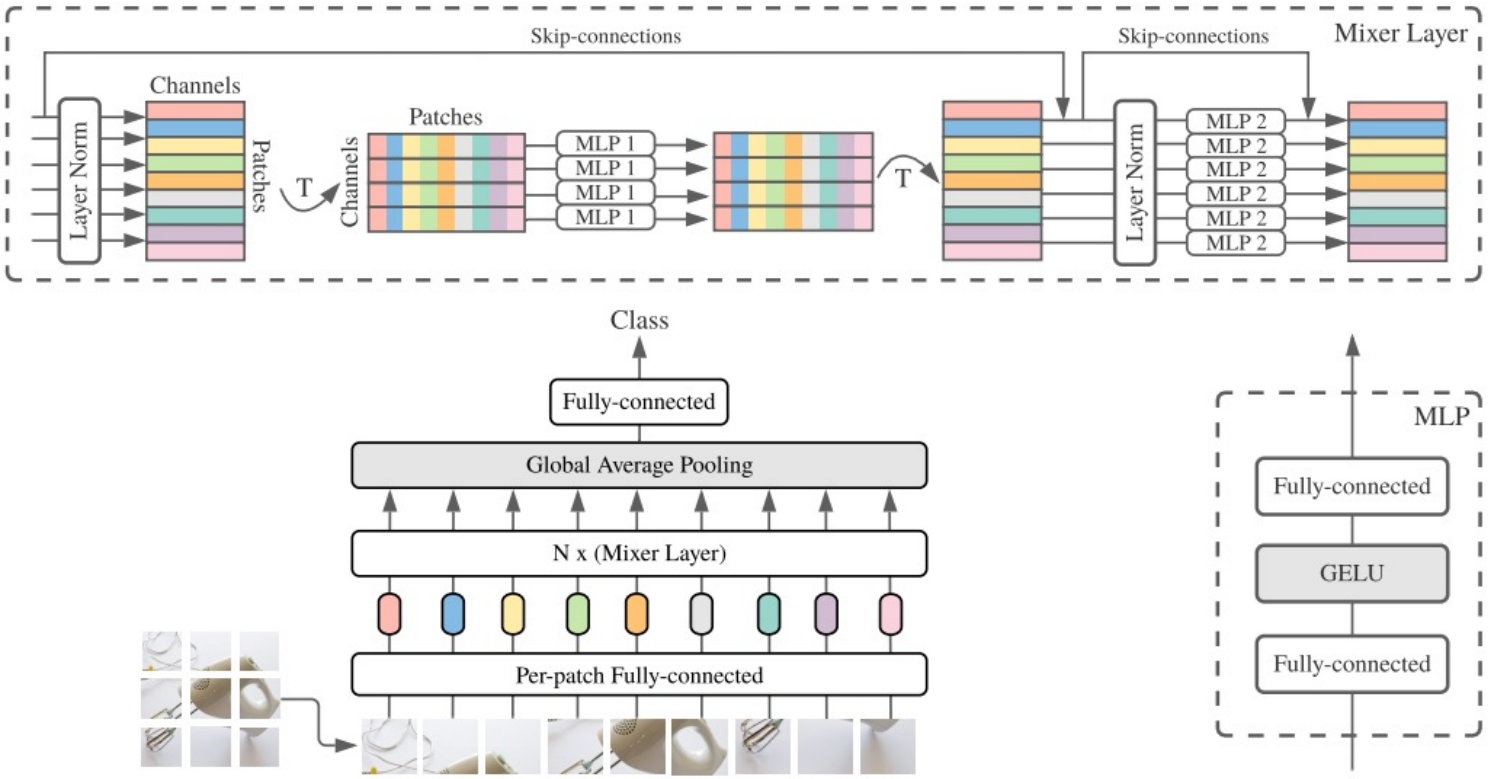
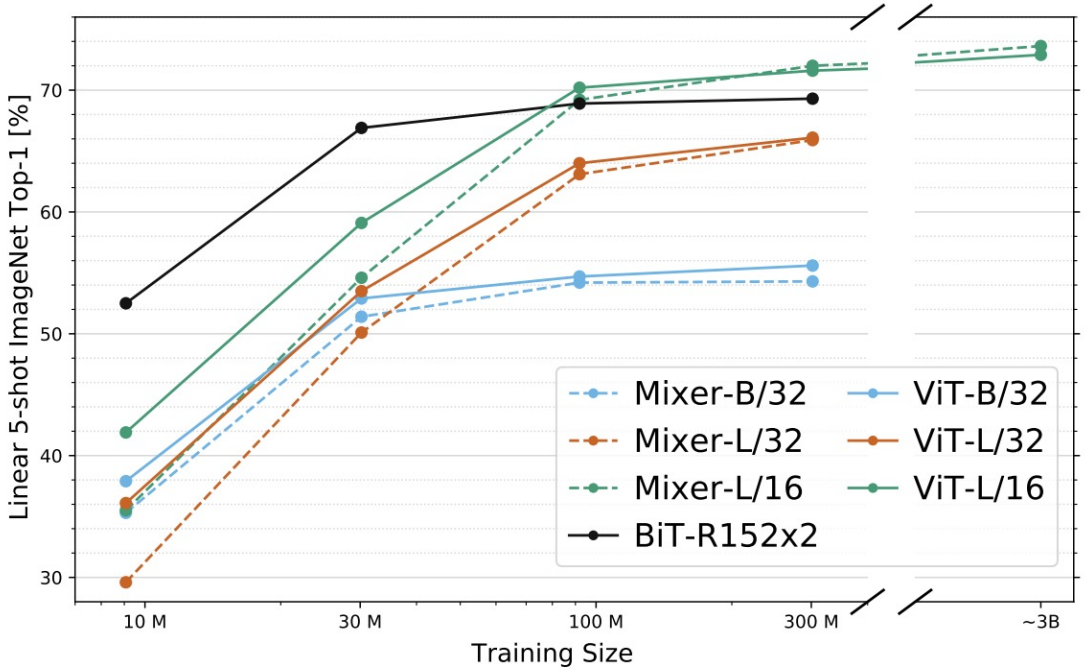



Figure 1: MLP-Mixer consists of per-patch linear embeddings, Mixer layers, and a classifier head. Mixer layers contain one token-mixing MLP and one channel-mixing MLP, each consisting of two fully-connected layers and a GELU nonlinearity. Other components include: skip-connections, dropout, and layer norm on the channels.

# MLP-Mixer: Is there any “indispensable”?

| | ImNet top-1 | ReaL top-1 | Avg 5 top-1 | VTAB-1k 19 tasks | Throughput img/sec/core | TPUv3 core-days |
|---|---|---|---|---|---|---|
| Pre-trained on ImageNet-21k (public) | | | | | | |
| HaloNet [51] | 85.8 | — | — | — | 120 | 0.10k |
| Mixer-L/16 | 84.15 | 87.86 | 93.91 | 74.95 | 105 | 0.41k |
| ViT-L/16 [14] | 85.30 | 88.62 | 94.39 | 72.72 | 32 | 0.18k |
| BiT-R152x4 [22] | 85.39 | — | 94.04 | 70.64 | 26 | 0.94k |
| Pre-trained on JFT-300M (proprietary) | | | | | | |
| NFNet-F4+ [7] | 89.2 | — | — | — | 46 | 1.86k |
| Mixer-H/14 | 87.94 | 90.18 | 95.71 | 75.33 | 40 | 1.01k |
| BiT-R152x4 [22] | 87.54 | 90.54 | 95.33 | 76.29 | 26 | 9.90k |
| ViT-H/14 [14] | 88.55 | 90.72 | 95.97 | 77.63 | 15 | 2.30k |
| Pre-trained on unlabelled or weakly labelled data (proprietary) | | | | | | |
| MPL [34] | 90.0 | 91.12 | — | — | — | 20.48k |
| ALIGN [21] | 88.64 | — | — | 79.99 | 15 | 14.82k |



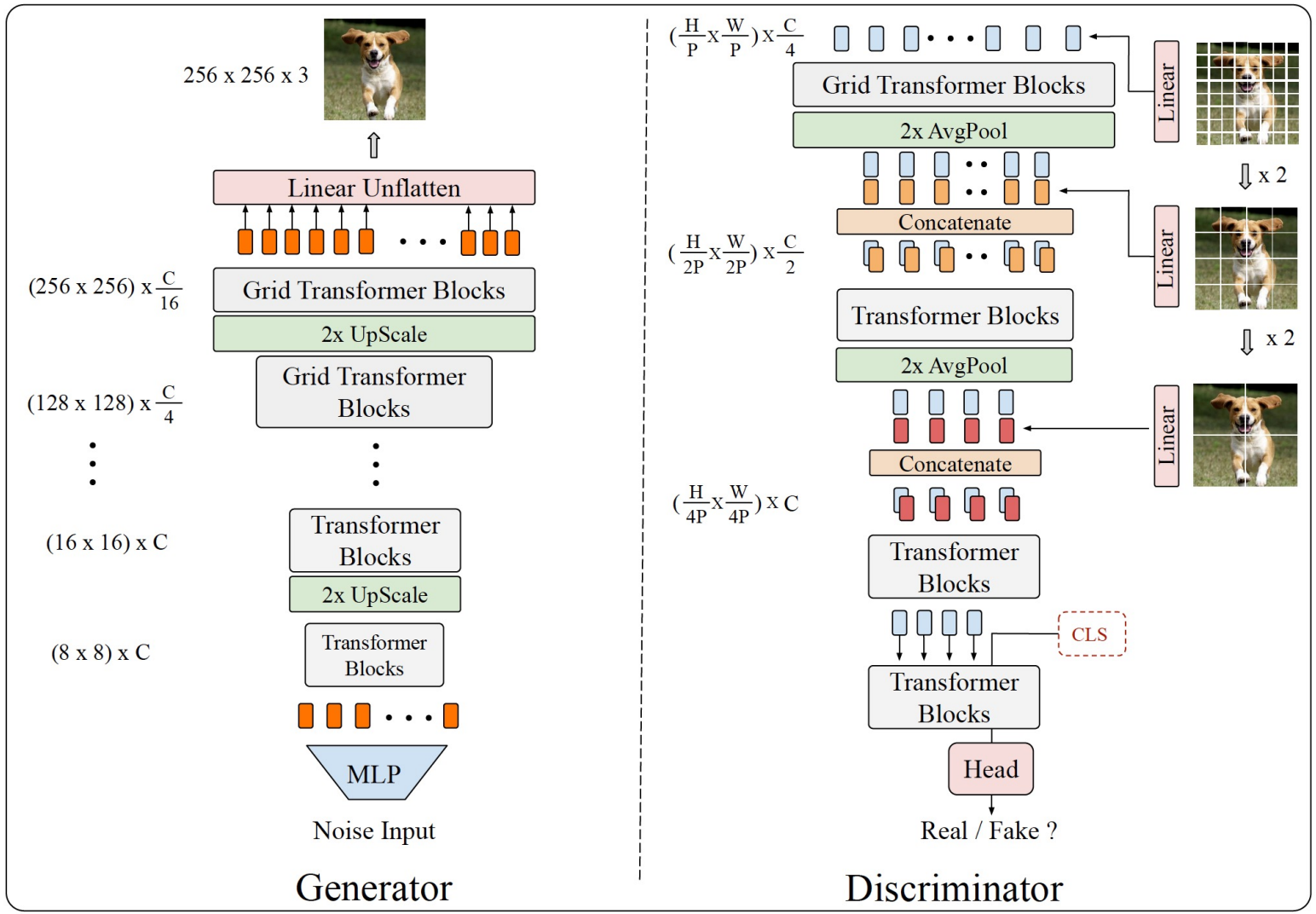
# MLP-Mixer: Is there any “indispensable”?

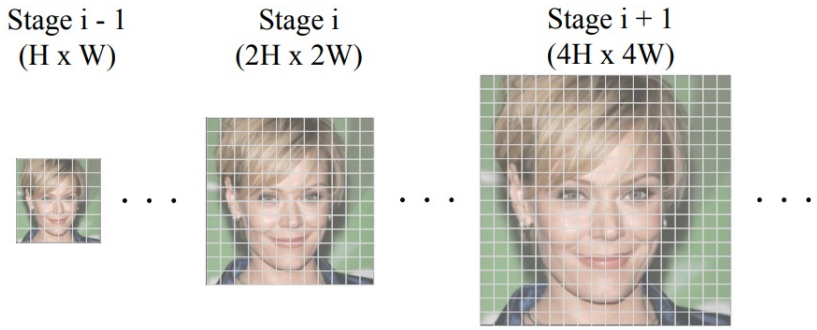
- **Surprise:** while convolutions and attention are both sufficient for good performance, neither of them are necessary!

- This architecture can be seen as a unique CNN, which uses (1×1) convolutions for channel mixing, and single-channel depth-wise convolutions for token mixing
  - However, the converse is not true as CNNs are not special cases of Mixer

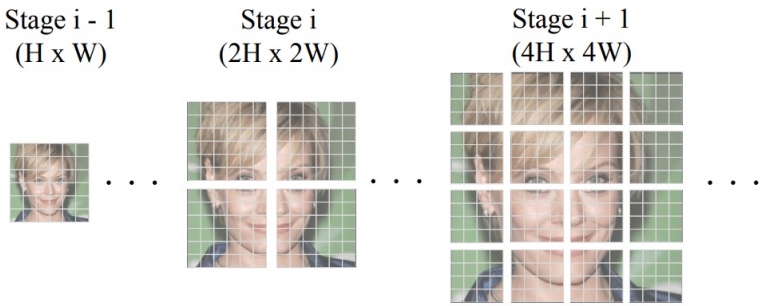
- It has inspired crazy thoughts: *maybe “sufficient information mixing” is just all you need, regardless how you mix it (using whatever architecture…)*

# TransGAN: Two Transformers Make One Strong GAN



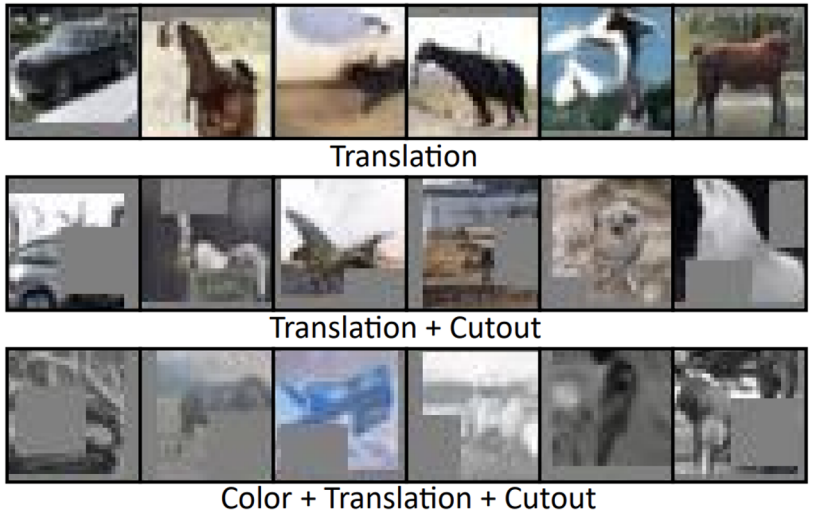



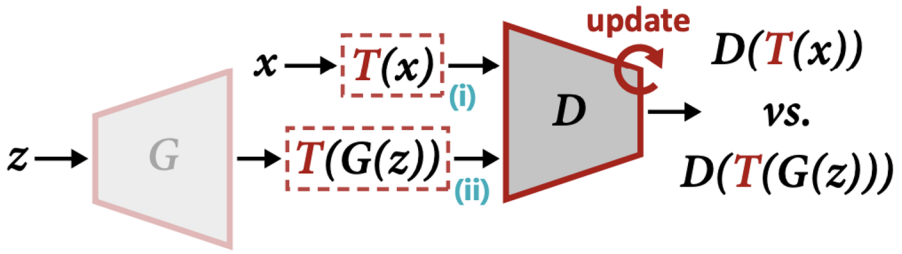

(a) Standard Self-Attention

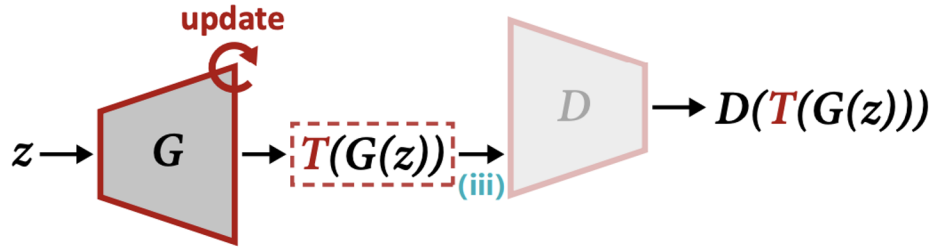



(b) Grid Self-Attention

# Data Augmentation Matters A LOT




| METHODS | DA | IS ↑ | FID ↓ |
|---|---|---|---|
| WGAN-GP | × | **6.49** ± 0.09 | 39.68 |
| (GULRAJANI ET AL., 2017) | √ | 6.29 ± 0.10 | **37.14** |
| AUTOGAN | × | 8.55 ± 0.12 | **12.42** |
| (GONG ET AL., 2019) | √ | **8.60** ± **0.10** | 12.72 |
| STYLEGAN V2 | × | 9.18 | 11.07 |
| (ZHAO ET AL., 2020B) | √ | **9.40** | **9.89** |
| TRANSGAN | × | 6.95 ± 0.13 | 41.41 |
| | √ | **8.15** ± **0.14** | **19.85** |

***Other Useful Techniques:***

- Relative position encoding
- A Modified Form of Normalization

...

# Comparing with SOTA ConvNet-based GANs

Table 1: Unconditional image generation results on CIFAR-10, STl-10, and CelebA (128 × 128) dataset. We train the models with their official code if the results are unavailable, denoted as "*", others are all reported from references.

| Methods | CIFAR-10 | | STL-10 | | CelebA |
|---|---|---|---|---|---|
| | IS↑ | FID↓ | IS↑ | FID↓ | FID↓ |
| WGAN-GP [1] | 6.49 ± 0.09 | 39.68 | - | - | - |
| SN-GAN [46] | 8.22 ± 0.05 | - | 9.16 ± 0.12 | 40.1 | - |
| AutoGAN [18] | 8.55 ± 0.10 | 12.42 | 9.16 ± 0.12 | 31.01 | - |
| AdversarialNAS-GAN [18] | 8.74 ± 0.07 | 10.87 | 9.63 ± 0.19 | 26.98 | - |
| Progressive-GAN [16] | 8.80 ± 0.05 | 15.52 | - | - | 7.30 |
| COCO-GAN [66] | - | - | - | - | 5.74 |
| StyleGAN-V2 [68] | 9.18 | 11.07 | 10.21* ± 0.14 | 20.84* | 5.59* |
| StyleGAN-V2 + DiffAug. [68] | **9.40** | 9.89 | 10.31* ± 0.12 | 19.15* | 5.40* |
| **TransGAN** | 9.02 ± 0.12 | **9.26** | **10.43** ± 0.16 | **18.28** | **5.28** |

*Transformer Eats Data !*

# Visual Examples of TransGAN

(a) Synthesized Image (b) Interpolation on Latent Space

Figure 1: Representative visual examples synthesized by TransGAN, **without using a single convolution**. (a) The synthesized visual examples on CelebA-HQ (256 × 256) dataset. (b) The linear interpolation results between two latent vectors, on CelebA-HQ (256 × 256) dataset.

# Multi-Modality: Video-Audio-Text Transformer (VATT)

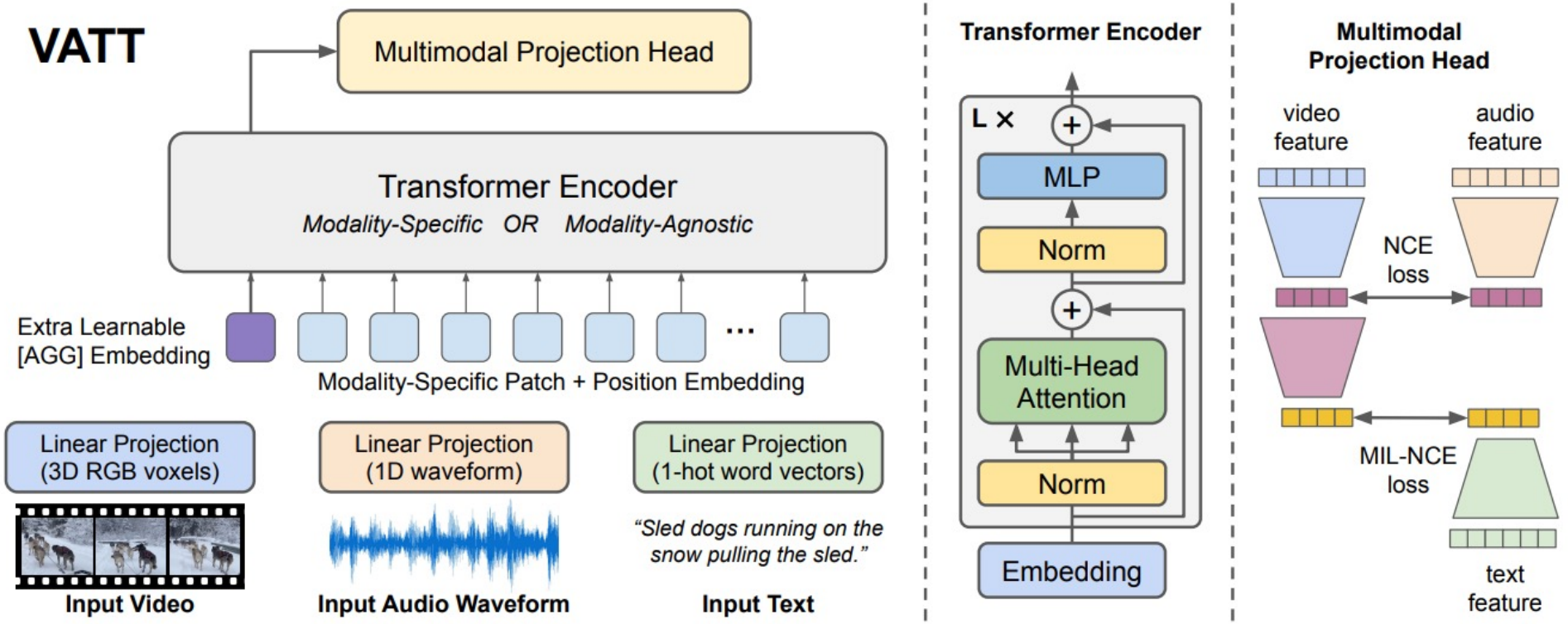



Figure 1: **Overview of the VATT architecture and the self-supervised, multimodal learning strategy**. VATT linearly projects each modality into a feature vector and feeds it into a Transformer encoder. We define a semantically hierarchical common space to account for the granularity of different modalities and employ the Noise Contrastive Estimation (NCE) to train the model.

# Other Ongoing ViT Studies …

Model Compression

Efficient Training & Transfer

Robustness (adversarial attacks, domain shifts…)

Better Optimization Algorithms (overcoming over smoothening, etc.)

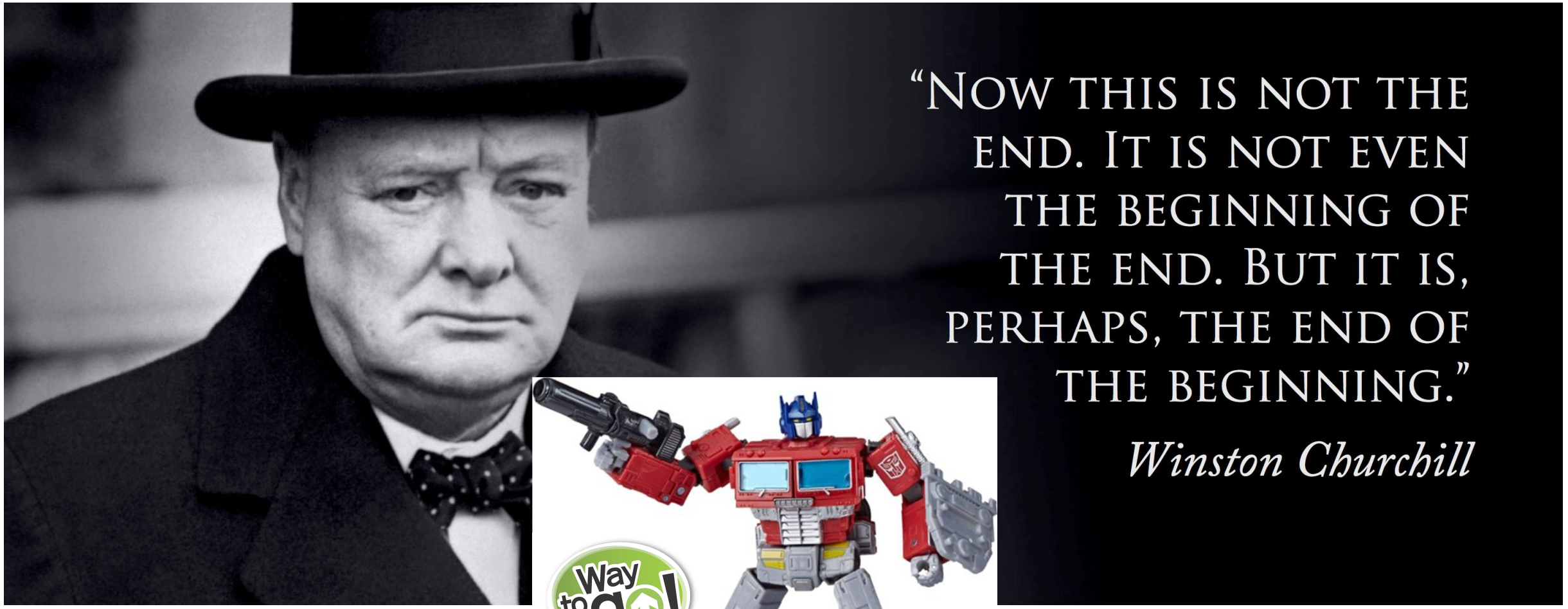
Better Backbone: Neural Architecture Search…

Scaling Up:

Larger Data, Multi-Modality ...

Interpretability

……


"Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning."
Winston Churchill


Way to go!

The University of Texas at Austin
Electrical and Computer Engineering
*Cockrell School of Engineering*