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

**Spring 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/

# A brief history in literature (that keeps growing fast EVERYDAY now…)

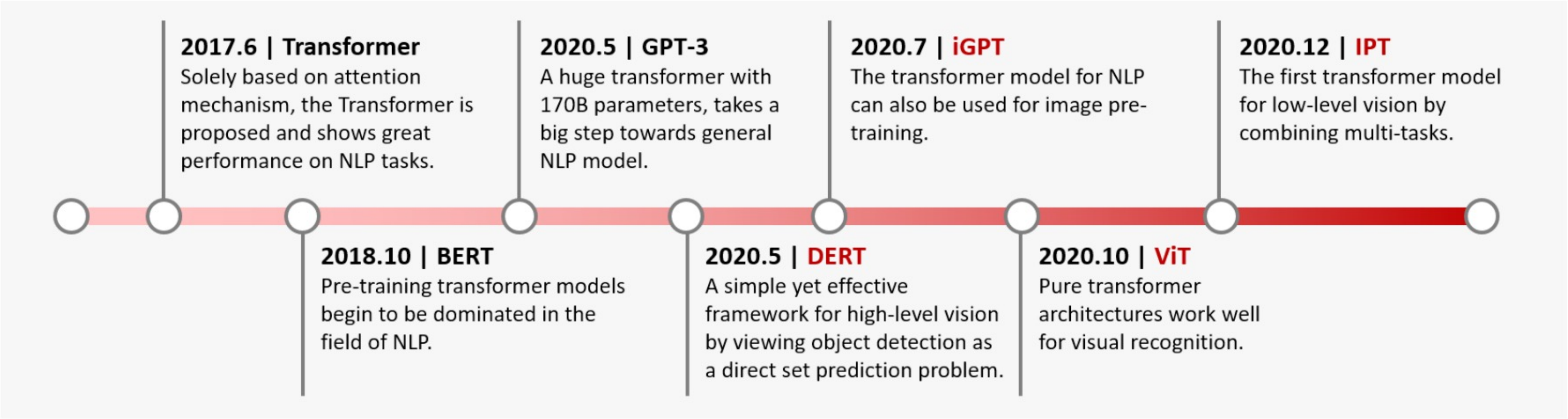



Figure 1: Key milestones in the development of transformer. The visual transformer models are marked in red.

# 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

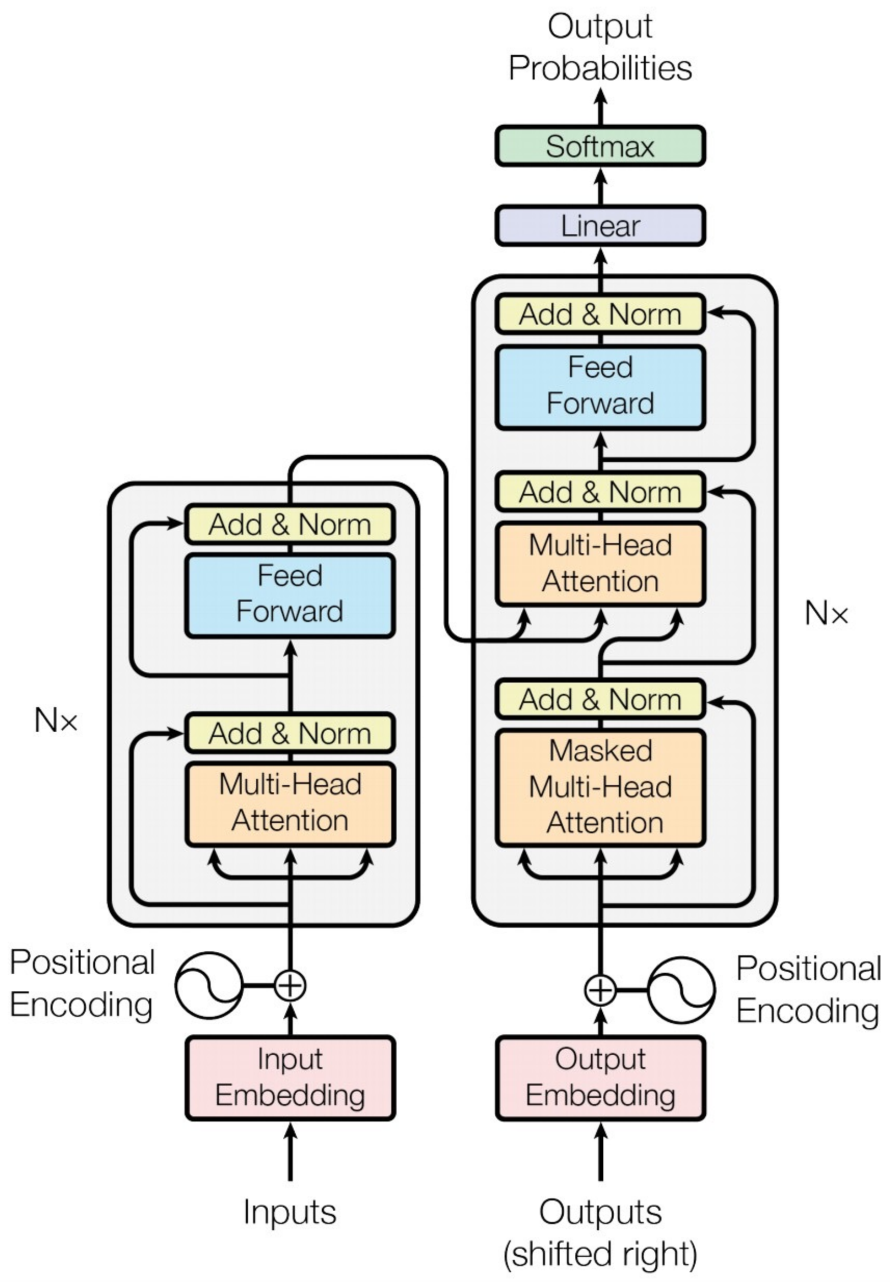
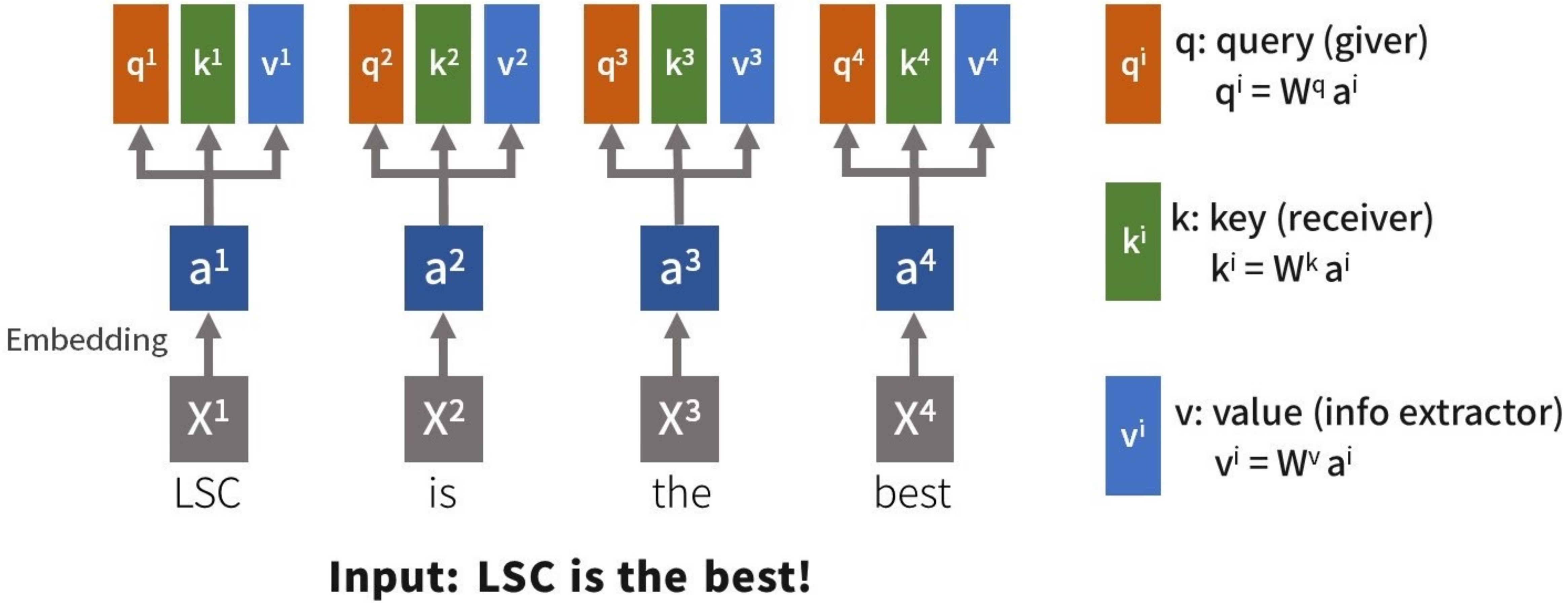
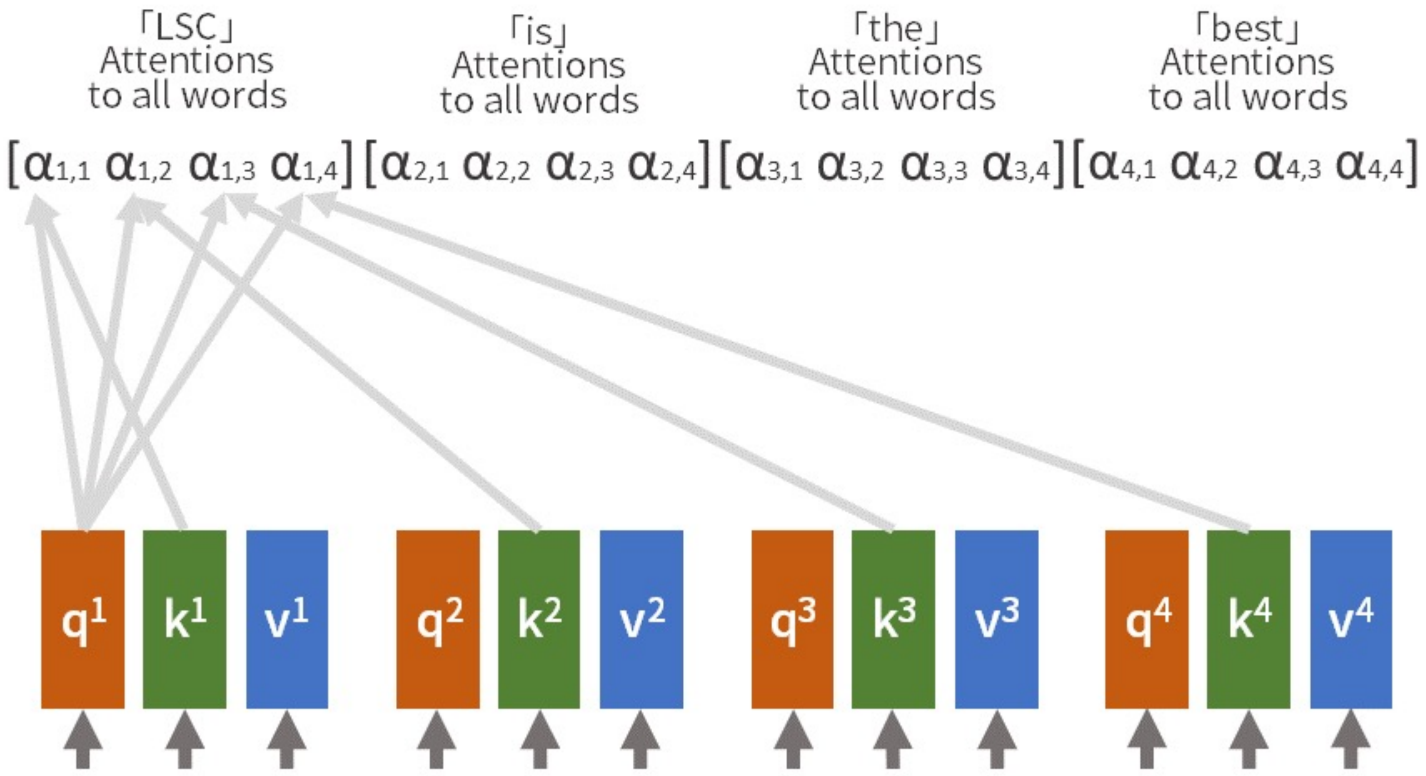



Figure 1: The Transformer - model architecture.

# Basics: Self-Attention



# Basics: Self-Attention

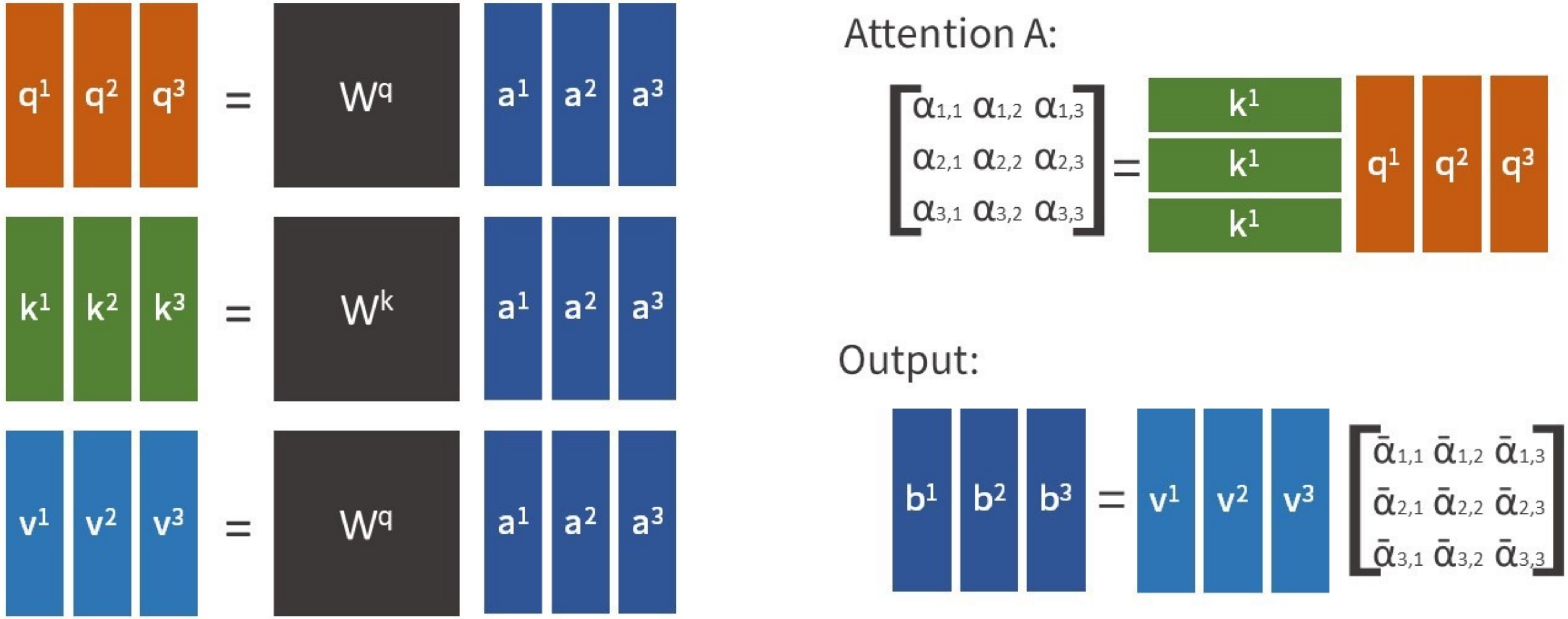



$$\alpha_{i,j} = \frac{q^i \cdot k^j}{\sqrt{d}}$$

d: dimension of q, k

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

# Basics: Self-Attention



# Bringing Transformers into Computer Vision

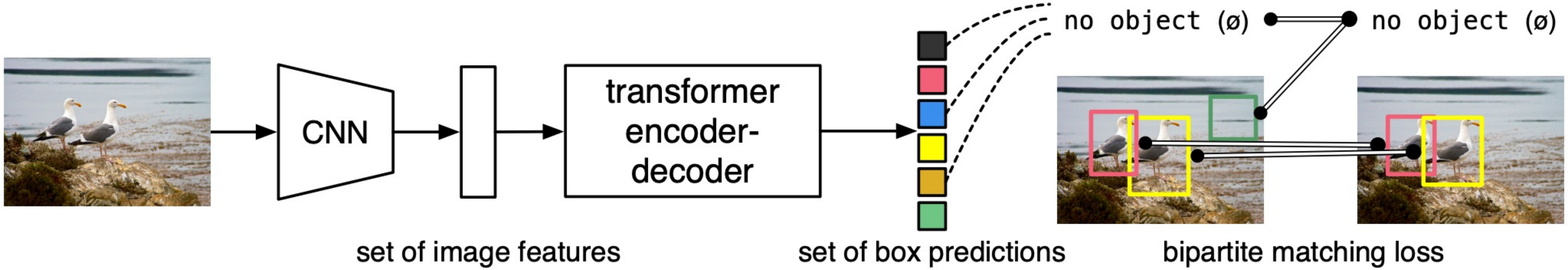
- Only in local neighborhoods (1)
  - Image Transformer, ICML 2018
  - Stand-alone self-attention in vision models, NeurIPS 2019
  - On the relationship between self-attention and convolutional layers, ICLR 2020
  - Exploring self-attention for image recognition, CVPR 2020
- Scalable approximations to 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

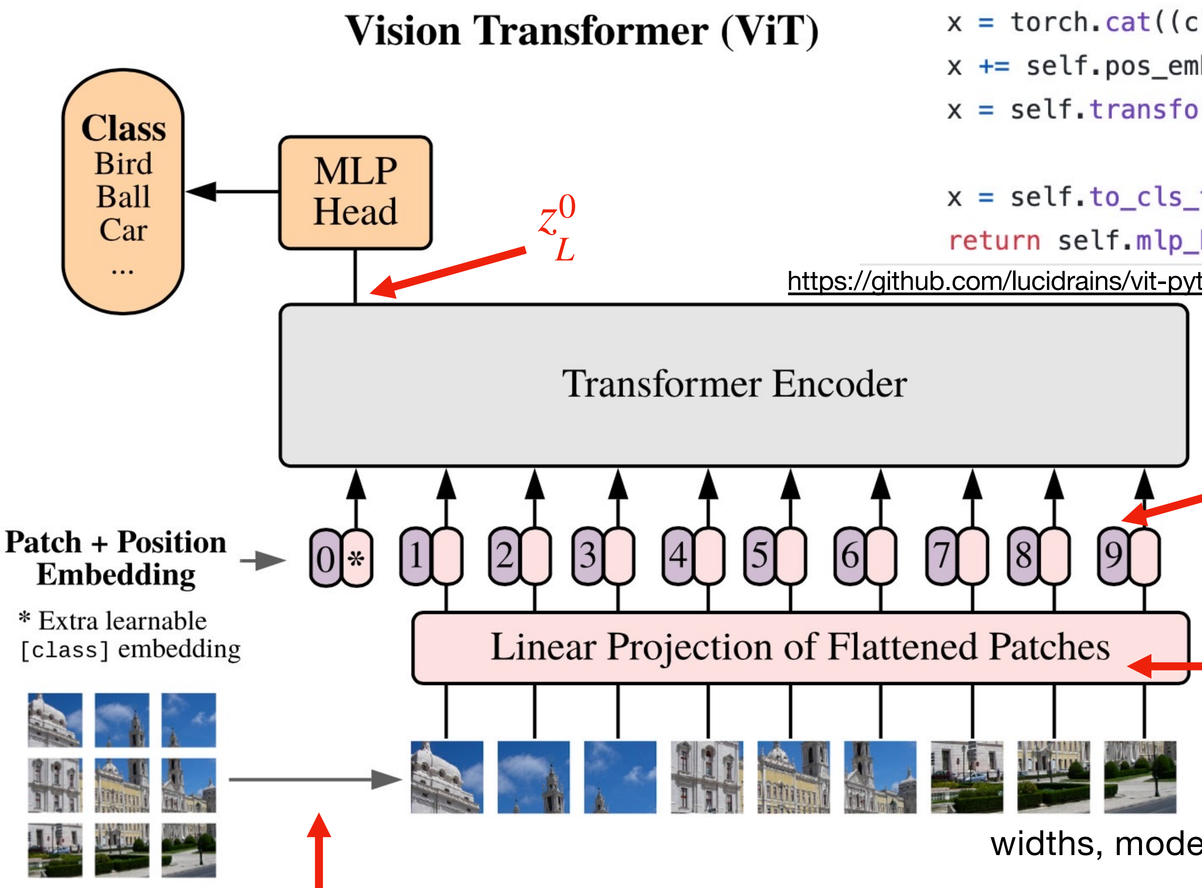
```
def forward(self, img, mask = None):
    p = self.patch_size

    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
```

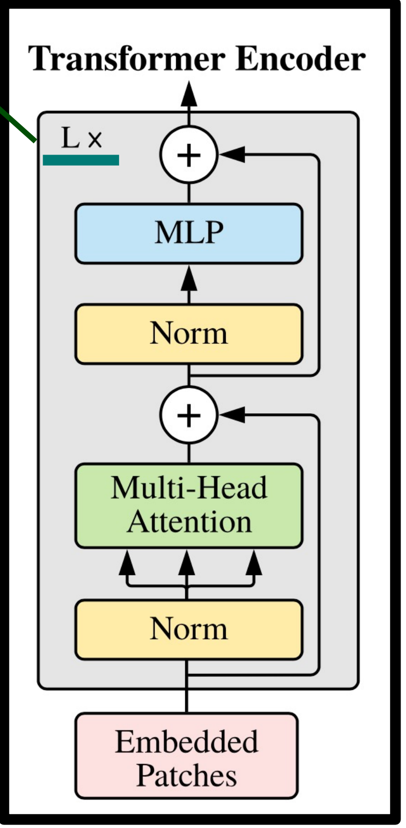
```python
def forward(self, img, mask = None):
    p = self.patch_size

    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

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

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

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

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

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

```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/**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 |

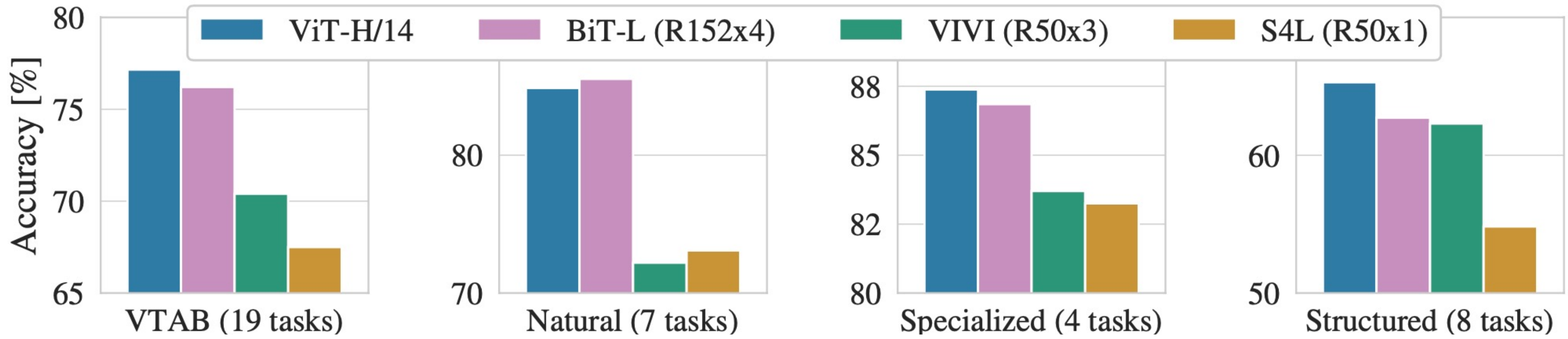



Figure 2: Breakdown of VTAB performance in *Natural*, *Specialized*, and *Structured* task groups.

# Experiments

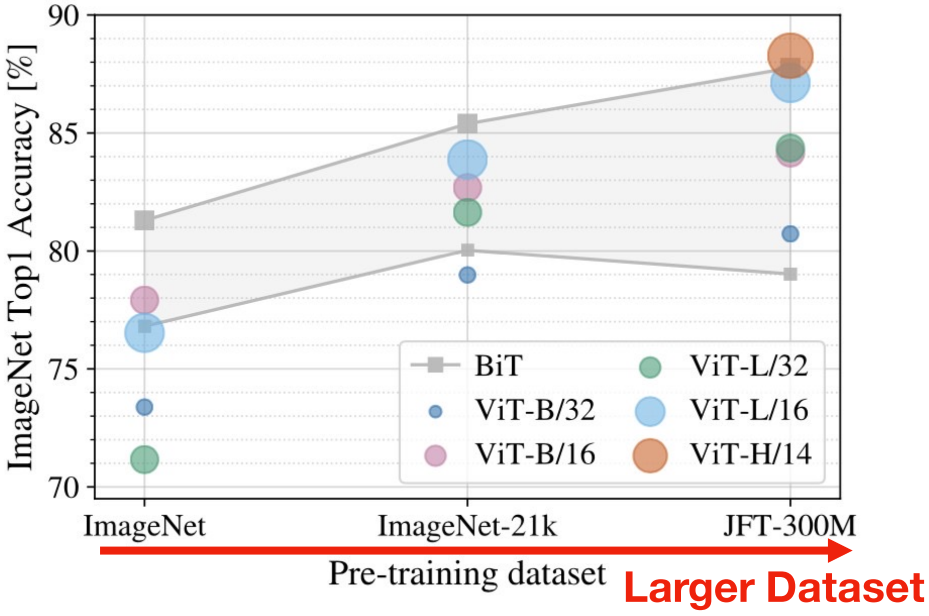



Figure 3: Transfer to ImageNet. While large ViT models perform worse than BiT ResNets (shaded area) when pre-trained on small datasets, they shine when pre-trained on larger datasets. Similarly, larger ViT variants overtake smaller ones as the dataset grows.

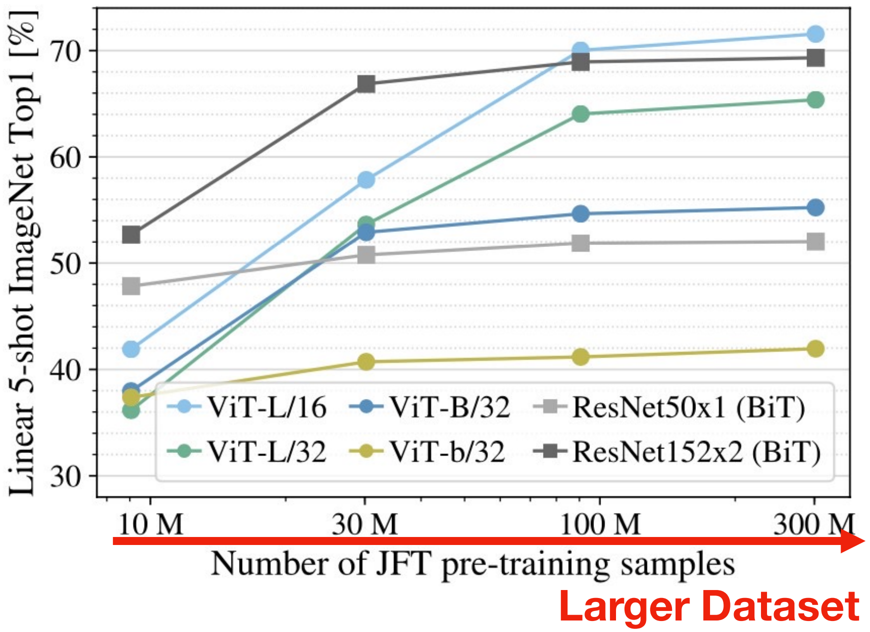
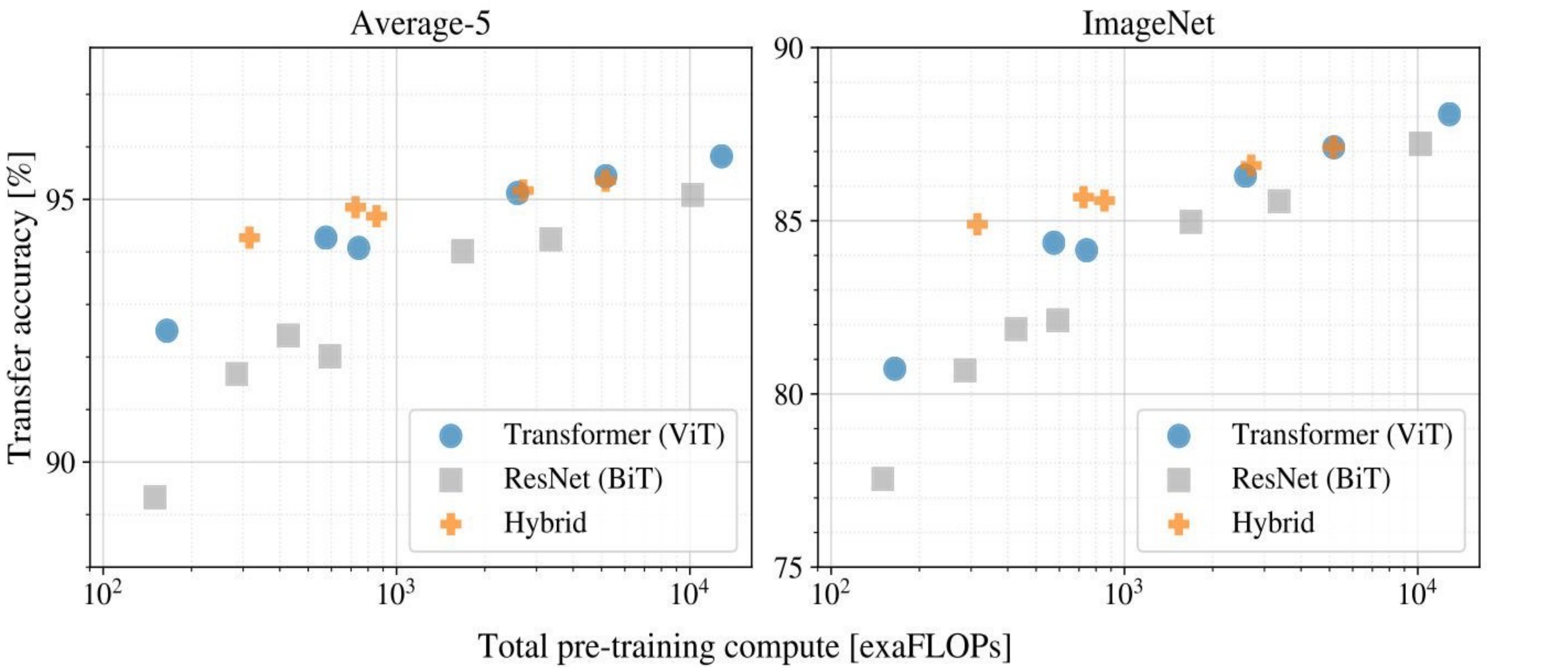


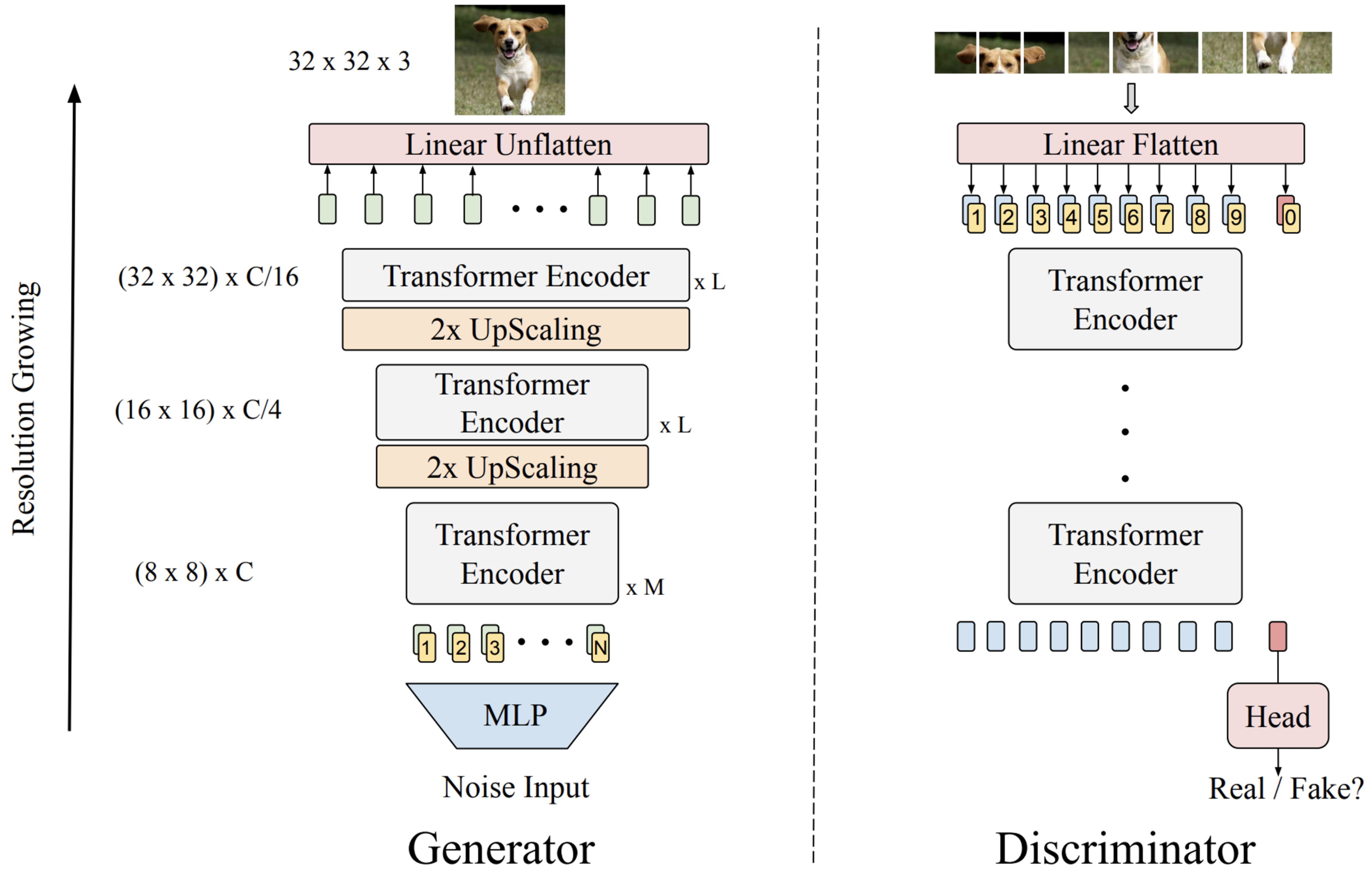

Figure 4: Linear few-shot evaluation on ImageNet versus pre-training size. ResNets perform better with smaller pre-training datasets but plateau sooner than ViT which performs better with larger pre-training. ViT-b is ViT-B with all hidden dimensions halved.

# Experiments

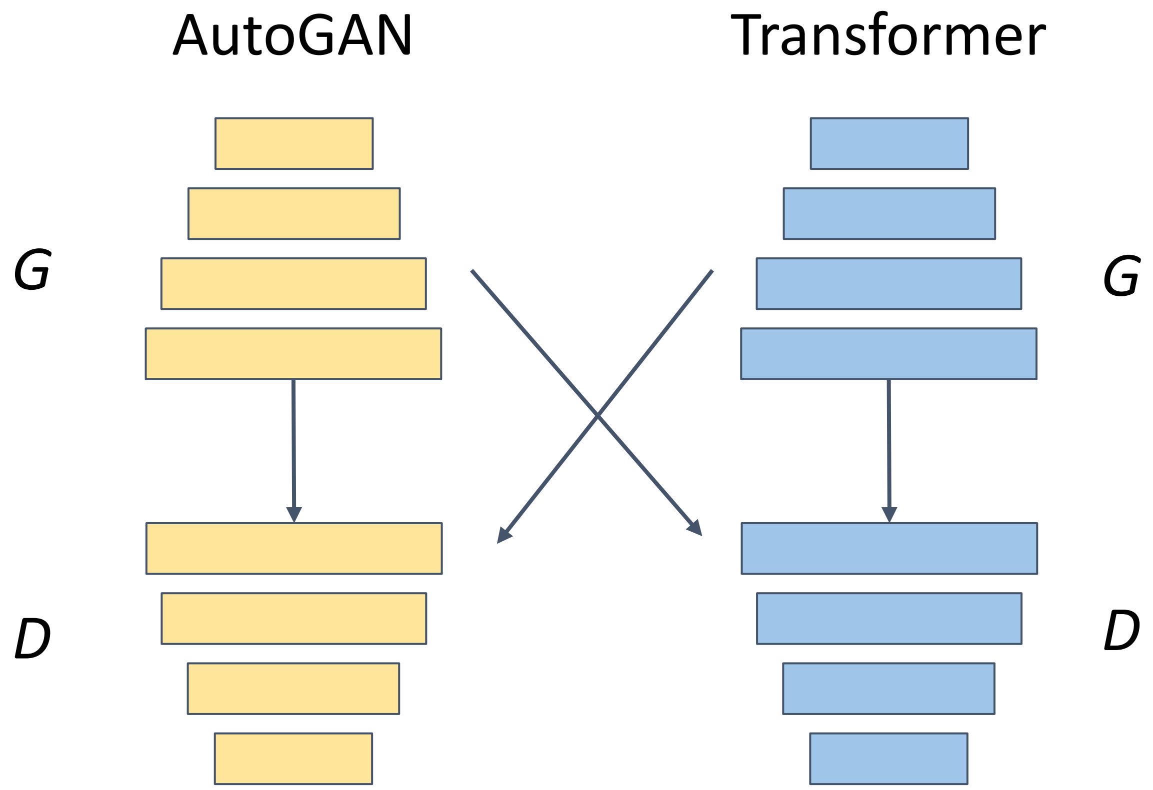



Performance versus cost for different architectures: Vision Transformers, ResNets, and hybrids. Vision Transformers generally outperform ResNets with the same computational budget. Hybrids improve upon pure Transformers for smaller model sizes, but the gap vanishes for larger models.

# Transformers Beyond Image Classification: TransGAN

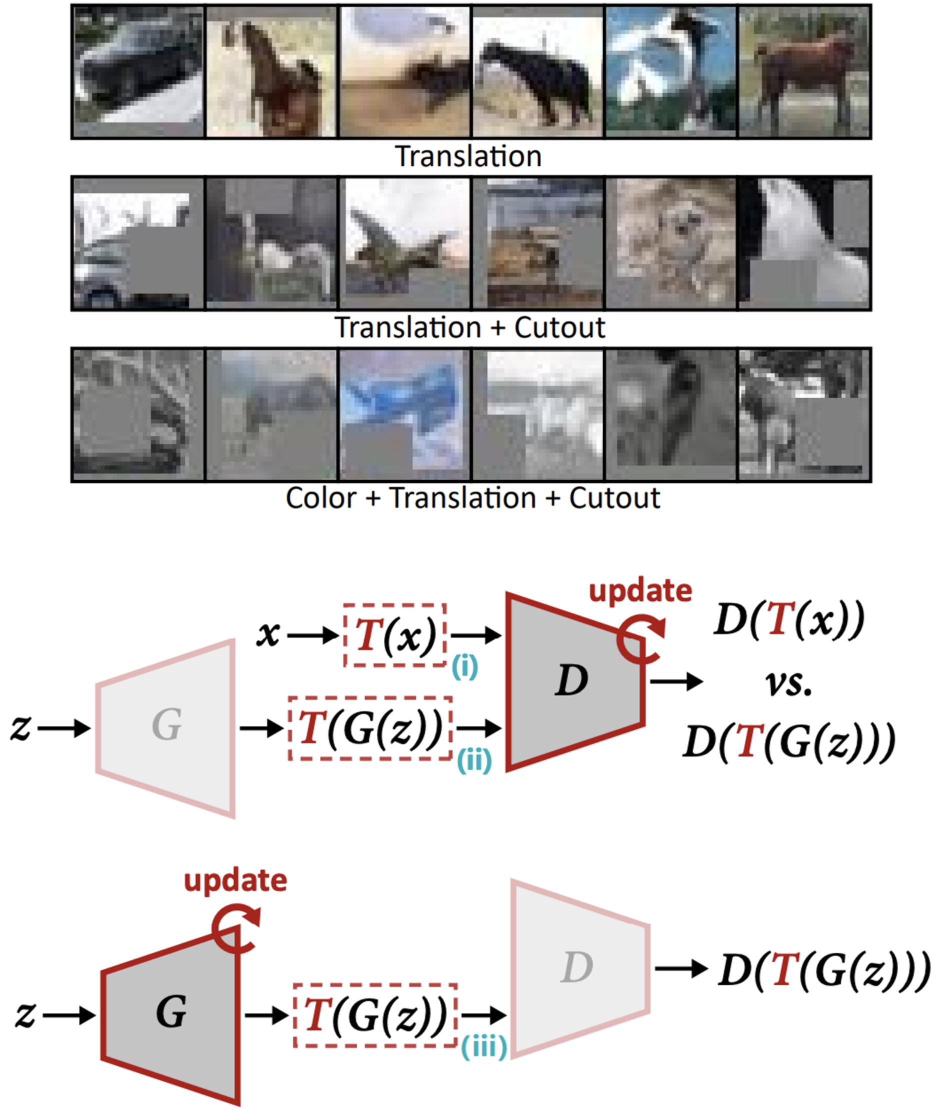


# Evaluating Transformer on Generator and Discriminator




| GENERATOR | DISCRIMINATOR | IS↑ | FID↓ |
|---|---|---|---|
| AUTOGAN | AUTOGAN | 8.55± 0.12 | **12.42** |
| TRANSFORMER | AUTOGAN | **8.59± 0.10** | 13.23 |
| AUTOGAN | TRANSFORMER | 6.17± 0.12 | 49.83 |
| TRANSFORMER | TRANSFORMER | 6.95 ± 0.13 | 41.41 |

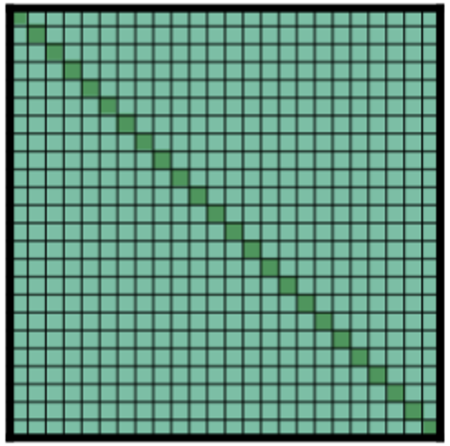
# Data Augmentation Matters A LOT



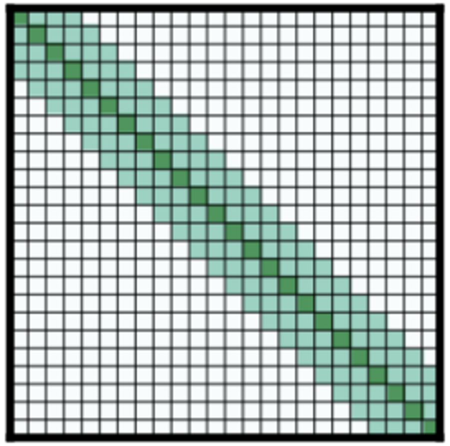

| METHODS | DA | IS ↑ | FID ↓ |
|---|---|---|---|
| WGAN-GP (GULRAJANI ET AL., 2017) | × | **6.49** ± 0.09 | 39.68 |
| | √ | 6.29 ± 0.10 | **37.14** |
| AUTOGAN (GONG ET AL., 2019) | × | 8.55 ± 0.12 | **12.42** |
| | √ | **8.60 ± 0.10** | 12.72 |
| STYLEGAN V2 (ZHAO ET AL., 2020B) | × | 9.18 | 11.07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

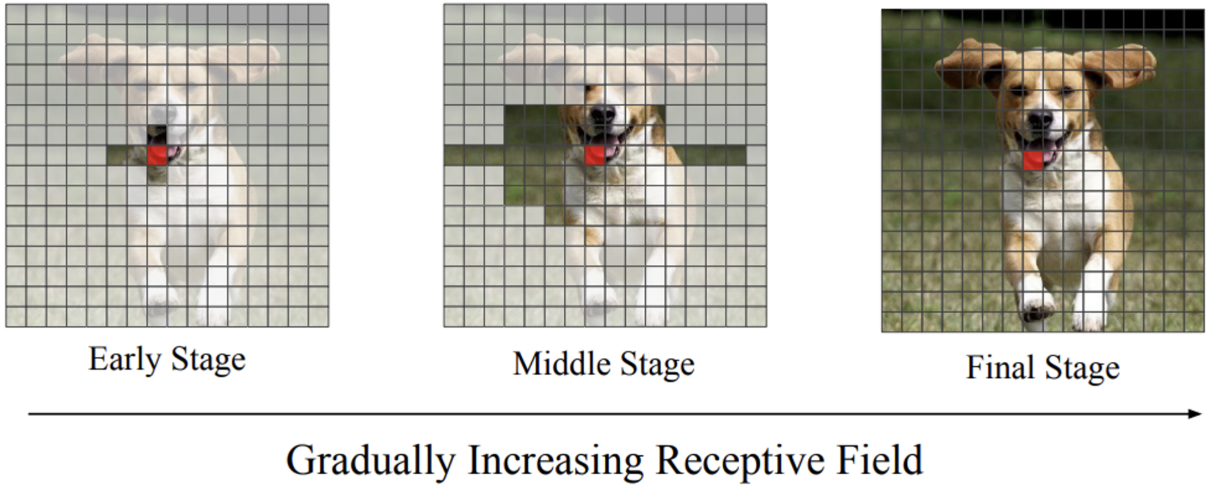
# Local Initialization For Self-Attention

Global

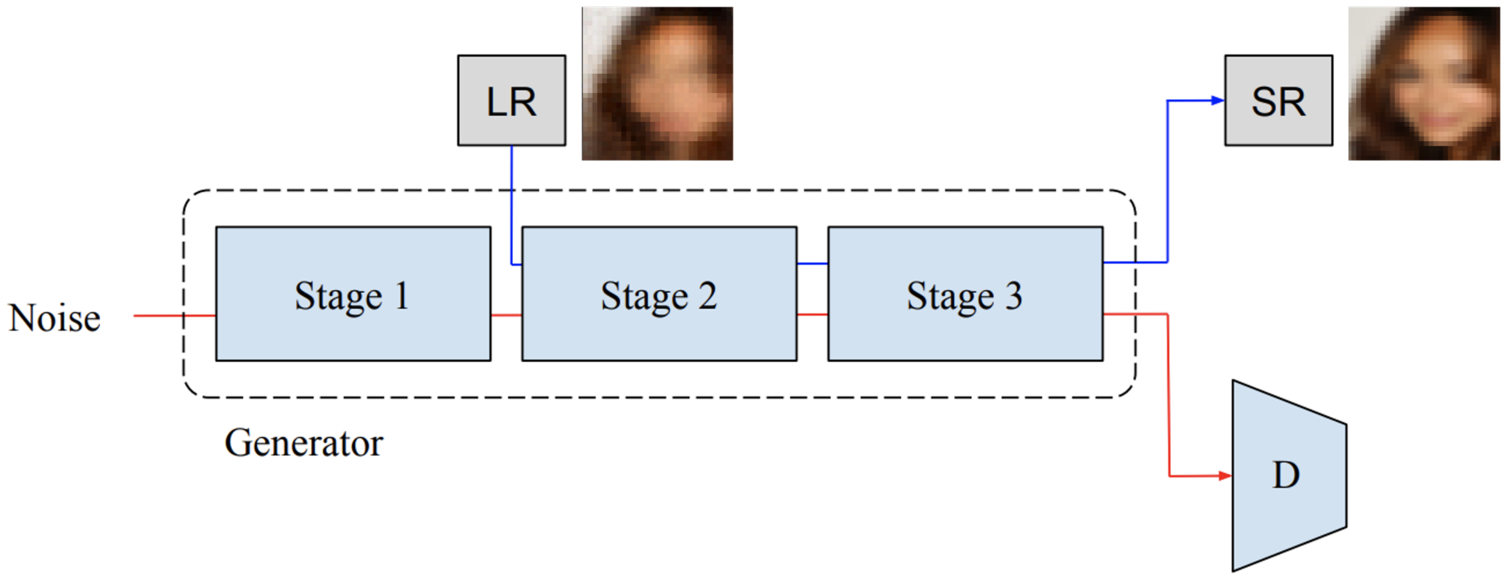


Local





# Multi-task Co-Training




| MODEL | IS↑ | FID↓ |
|---|---|---|
| TRANSGAN + DA (*) | 8.15± 0.14 | 19.85 |
| (*) + MT-CT | 8.20± 0.14 | 19.12 |
| (*) + MT-CT + LOCAL INIT. | **8.22± 0.12** | **18.58** |

# Scaling-Up the Generator

| Model | Depth | Dim | IS ↑ | FID ↓ |
|---|---|---|---|---|
| TransGAN-S | {5,2,2} | 384 | $8.22 \pm 0.14$ | 18.58 |
| TransGAN-M | {5,2,2} | 512 | $8.36 \pm 0.12$ | 16.27 |
| TransGAN-L | {5,2,2} | 768 | $8.50 \pm 0.14$ | 14.46 |
| TransGAN-XL | {5,4,2} | 1024 | $\mathbf{8.63 \pm 0.16}$ | **11.89** |

Efficiency Comparison:

| Methods | Flops (G) | IS | FID |
|---|---|---|---|
| SNGAN (Miyato et al., 2018) | 1.57 | $8.22 \pm 0.05$ | 21.7 |
| **TransGAN-S** | **0.68** | $\mathbf{8.22 \pm 0.14}$ | **18.58** |
| AutoGAN (Gong et al., 2019) | **1.77** | $8.55 \pm 0.10$ | 12.42 |
| Progressive-GAN (Karras et al., 2017) | 6.39 | $\mathbf{8.80 \pm 0.05}$ | 15.52 |
| **TransGAN-XL** | 2.83 | $8.63 \pm 0.16$ | **11.89** |

# Comparing with SOTA ConvNet-based GANs

| METHODS | IS | FID |
|---|---|---|
| WGAN-GP (GULRAJANI ET AL., 2017) | $6.49 \pm 0.09$ | 39.68 |
| LRGAN (YANG ET AL., 2017) | $7.17 \pm 0.17$ | - |
| DFM (WARDE-FARLEY & BENGIO, 2016) | $7.72 \pm 0.13$ | - |
| SPLITTING GAN (GRINBLAT ET AL., 2017) | $7.90 \pm 0.09$ | - |
| IMPROVING MMD-GAN (WANG ET AL., 2018A) | 8.29 | 16.21 |
| MGAN (HOANG ET AL., 2018) | $8.33 \pm 0.10$ | 26.7 |
| SN-GAN (MIYATO ET AL., 2018) | $8.22 \pm 0.05$ | 21.7 |
| PROGRESSIVE-GAN (KARRAS ET AL., 2017) | $8.80 \pm 0.05$ | 15.52 |
| AUTOGAN (GONG ET AL., 2019) | $8.55 \pm 0.10$ | 12.42 |
| STYLEGAN V2 (ZHAO ET AL., 2020B) | **9.18** | **11.07** |
| TRANSGAN-XL | $8.63 \pm 0.16$ | 11.89 |

CIFAR-10

| METHODS | IS ↑ | FID ↓ |
|---|---|---|
| DFM (WARDE-FARLEY & BENGIO, 2016) | $8.51 \pm 0.13$ | - |
| D2GAN (NGUYEN ET AL., 2017) | 7.98 | - |
| PROBGAN (HE ET AL., 2019) | $8.87 \pm 0.09$ | 47.74 |
| DIST-GAN (TRAN ET AL., 2018) | - | 36.19 |
| SN-GAN (MIYATO ET AL., 2018) | $9.16 \pm 0.12$ | 40.1 |
| IMPROVING MMD-GAN (WANG ET AL., 2018A) | $9.23 \pm 0.08$ | 37.64 |
| AUTOGAN (GONG ET AL., 2019) | $9.16 \pm 0.12$ | 31.01 |
| ADVERSARIALNAS-GAN (GAO ET AL., 2020) | $9.63 \pm 0.19$ | 26.98 |
| TRANSGAN-XL | $\mathbf{10.10 \pm 0.17}$ | **25.32** |

STL-10

*Transformer Eats Data !*

"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*

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