TransGAN

Two Pure Transformers Can Make One Strong GAN, and That Can Scale Up

2021.11.10 박승주

Table of Contents

01 Abstract & Introduction

Contributions

Can we build a strong GAN completely free of convolutions?

CNN

- with strong inductive bias for natural images

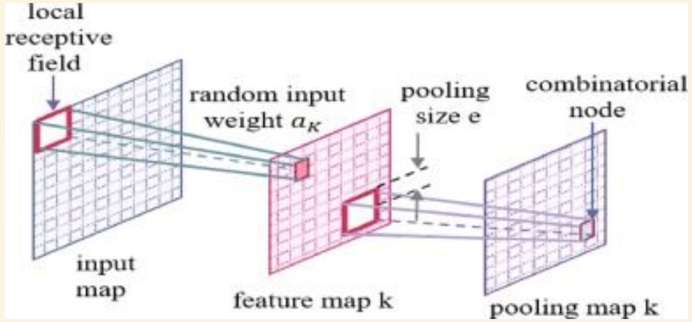
  → contribute to the appealing visual results
  → rich diversity achieved by modern GANs

- GAN's commonsense

- Convolution → local receptive field

- cannot process long-range dependencies
unless passing through a sufficient layers

- cause **loss of feature resolution & details**
- difficulty of optimization
(unstable and prone to mode collapse)



- Vanilla CNN-based models
  → not well suited for capturing 'global' statistics

- adopting self-attention / non-local operations

Contributions

Novel Architecture Design

- First GAN using purely transformers
  → **NO convolution**
- Memory friendly **generator**
  → gradually increasing
  feature map resolution
- Multi-scale **discriminator**
  → patches of varied size as inputs
  → balance between globlal & local
- New **grid self-attention** mechanism
  → alleviates memory bottleneck

New Training Recipe

- leveraging **data augmentation**
  → Differential augmentation
  (Translation, Cutout, Color)
- modifying **layer normalization**
  → token-wise scaling layer
  → prevent too high magnitude
  in transformer blocks
- adopting **relative position encoding**
  → exploiting lags
  → learns a stronger relationship

Performance & Scalability

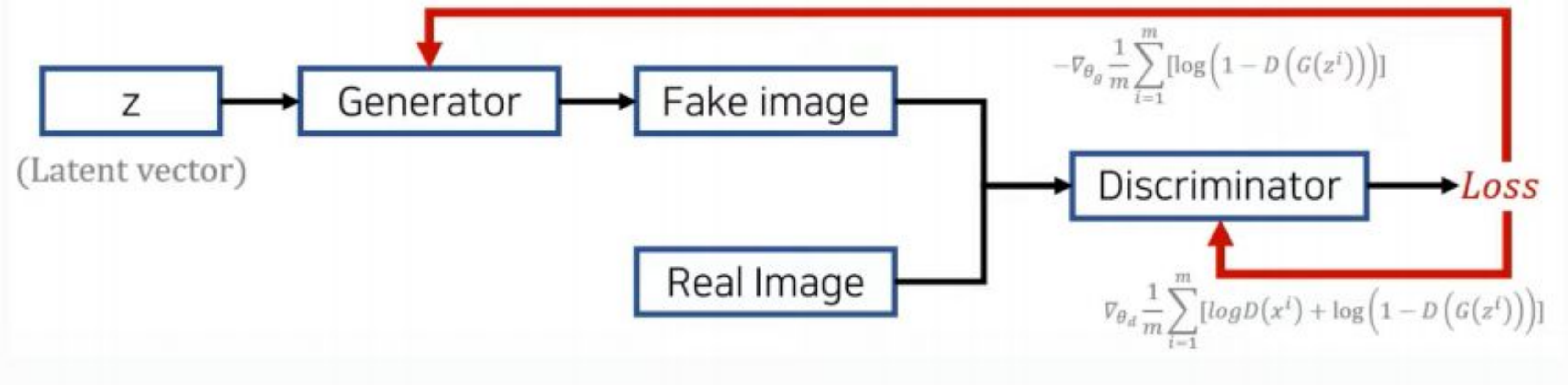
- competitive performance compared to sota GANS

  → IS 10.43, FID 18.28 on STL-10
  → IS 9.02, FID 9.26 on CIFAR-10
  → FID 5.28 on CelebA 128

02 Related Works

GAN, Transformer, BERT, DiffAug

GAN



- Generator (G(z)) : new data instance

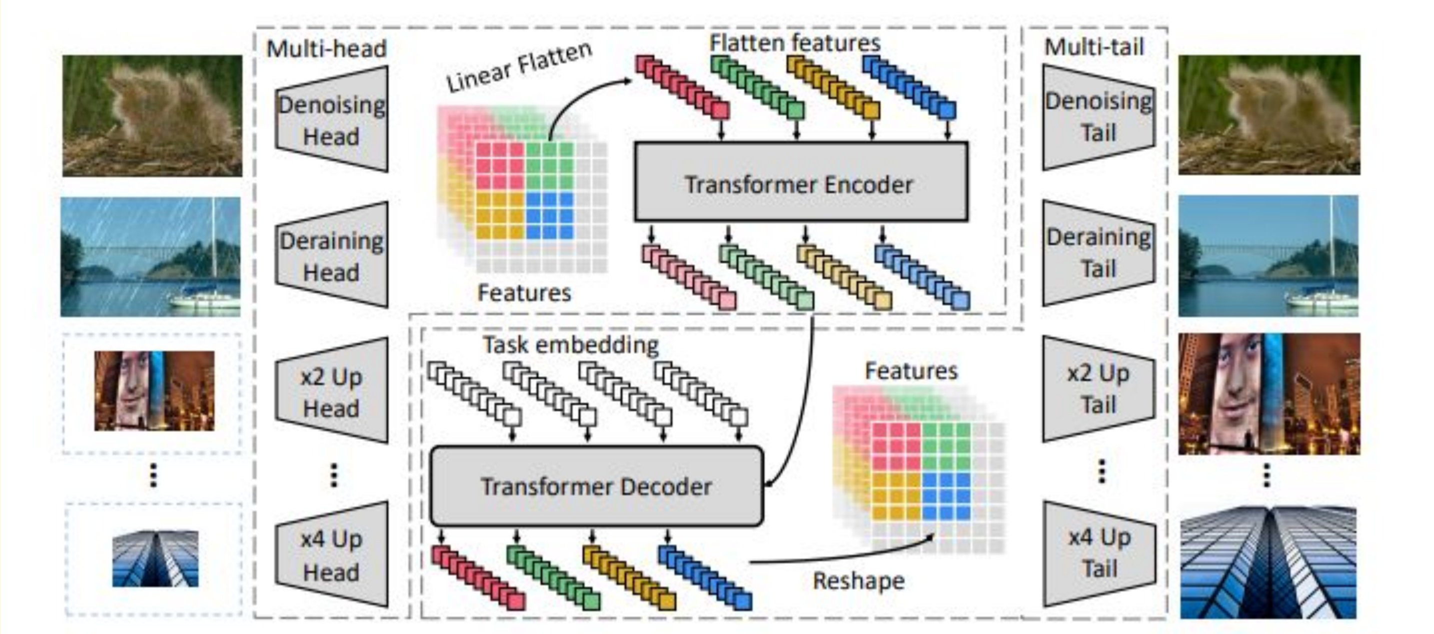
- Discriminator (D(x)) : probability - a sample came from the real distribution
  (Real : 1 / Fake : 0)

$$\min_G \max_D V(D,G) = \boxed{\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})]} + \boxed{\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]}.$$

Transformer

- Success of original vision transformer
  → relies on **pretraining on large-scale external data**

- pyramid / hierarchical structure to transformer or + convolutional layers



https://arxiv.org/abs/2012.00364 : Pre-Trained Image Processing Transformer

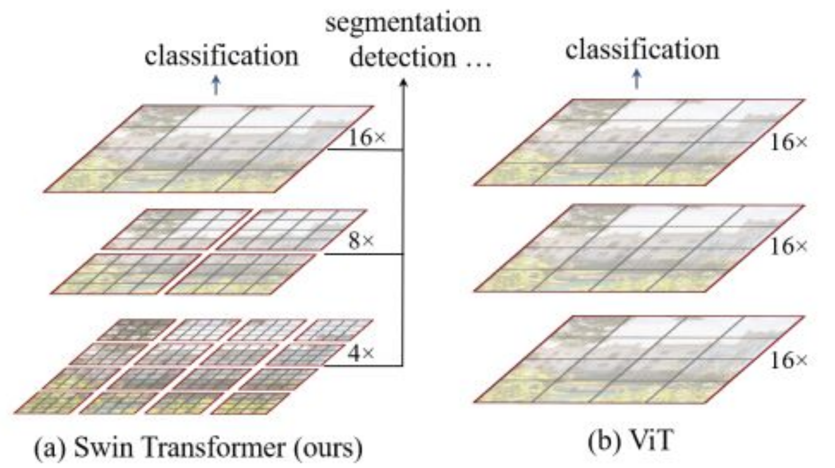


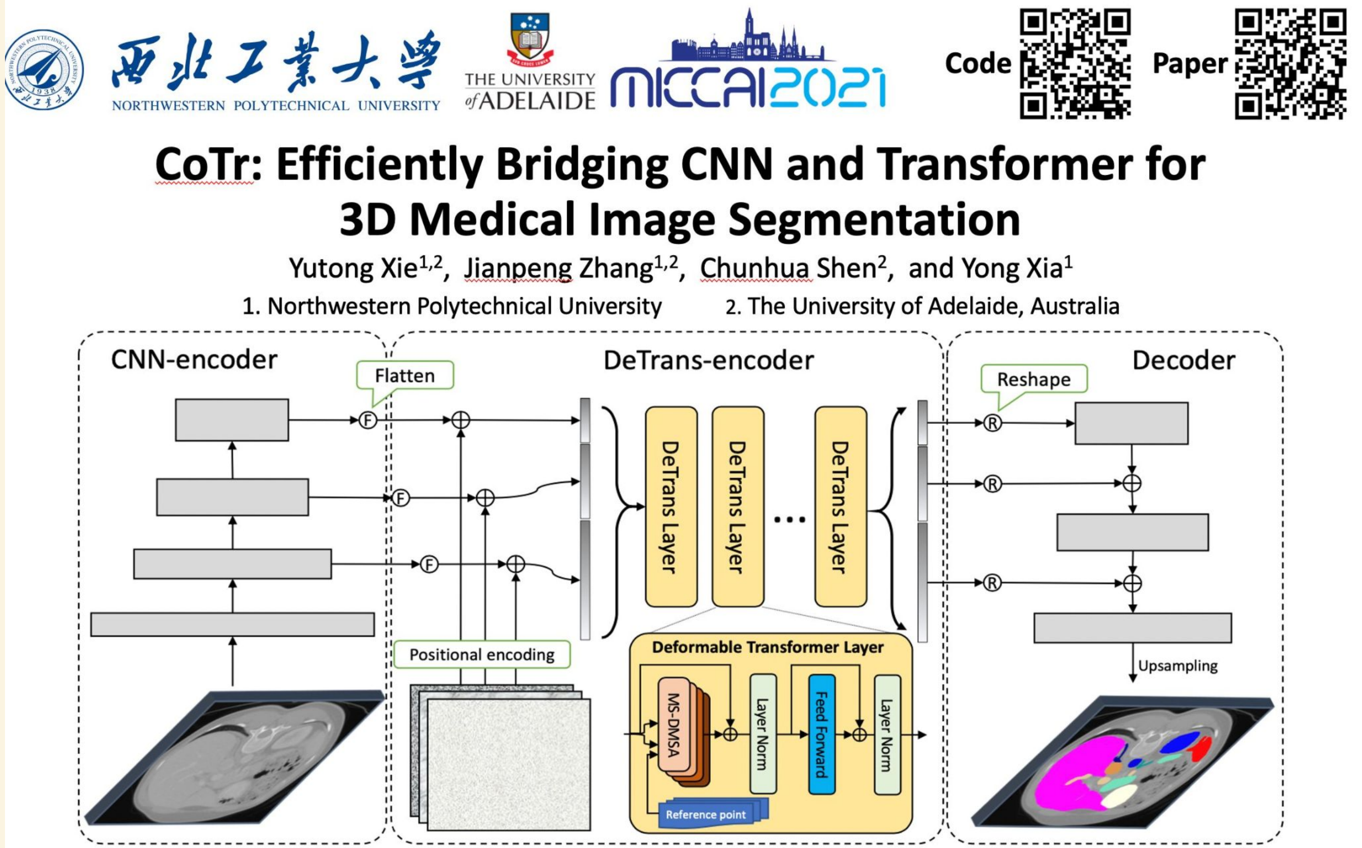
Figure 1. (a) The proposed Swin Transformer builds hierarchical feature maps by merging image patches (shown in gray) in deeper layers and has linear computation complexity to input image size due to computation of self-attention only within each local window (shown in red). It can thus serve as a general-purpose backbone for both image classification and dense recognition tasks. (b) In contrast, previous vision Transformers [19] produce feature maps of a single low resolution and have quadratic computation complexity to input image size due to computation of self-attention globally.

https://arxiv.org/abs/2103.14030 : Swin Transformer

Transformer

- Success of original vision transformer
  → relies on **pretraining on large-scale external data**

- pyramid / hierarchical structure to transformer or + convolutional layers



https://arxiv.org/pdf/2103.03024 : CoTR

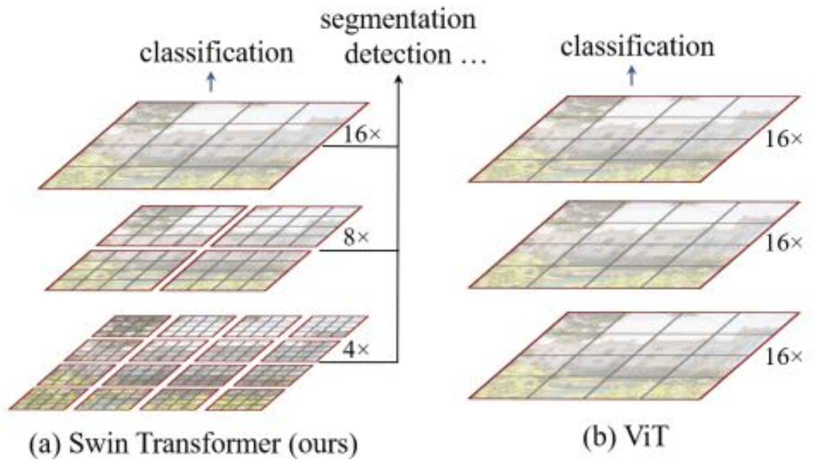
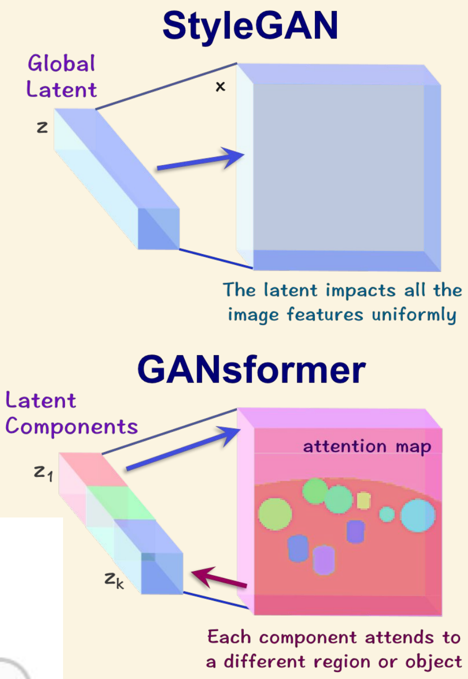


Figure 1. (a) The proposed Swin Transformer builds hierarchical feature maps by merging image patches (shown in gray) in deeper layers and has linear computation complexity to input image size due to computation of self-attention only within each local window (shown in red). It can thus serve as a general-purpose backbone for both image classification and dense recognition tasks. (b) In contrast, previous vision Transformers [19] produce feature maps of a single low resolution and have quadratic computation complexity to input image size due to computation of self-attention globally.

https://arxiv.org/abs/2103.14030 : Swin Transformer

Transformer module for image generation

- Overall CNN architecture remains
- Customized designs (codebook, quantization) → limits models' versatility
- GANsformers - using StyleGAN
  → main structure is still convolutional



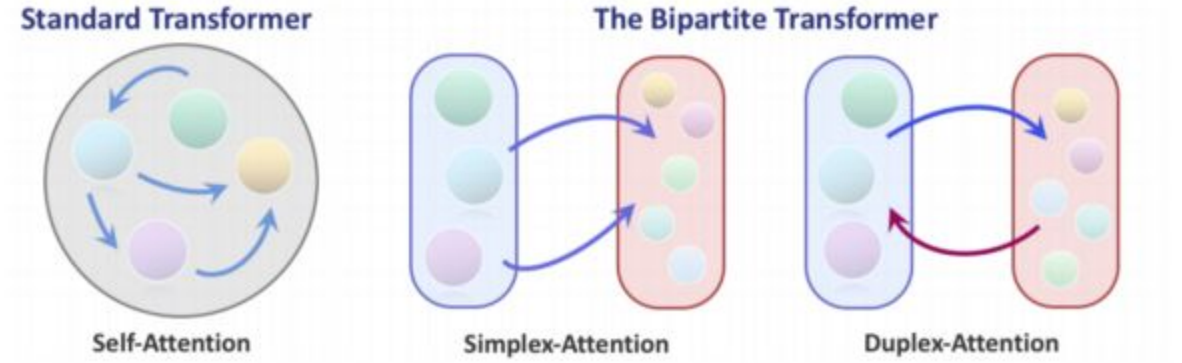
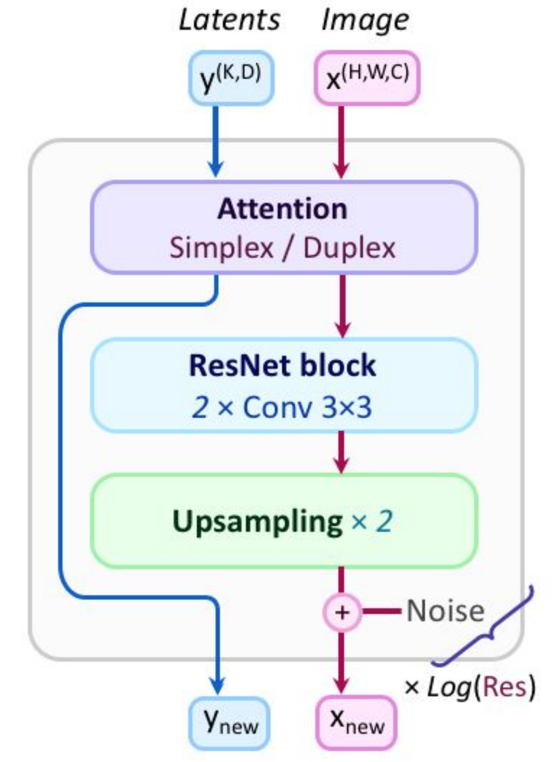


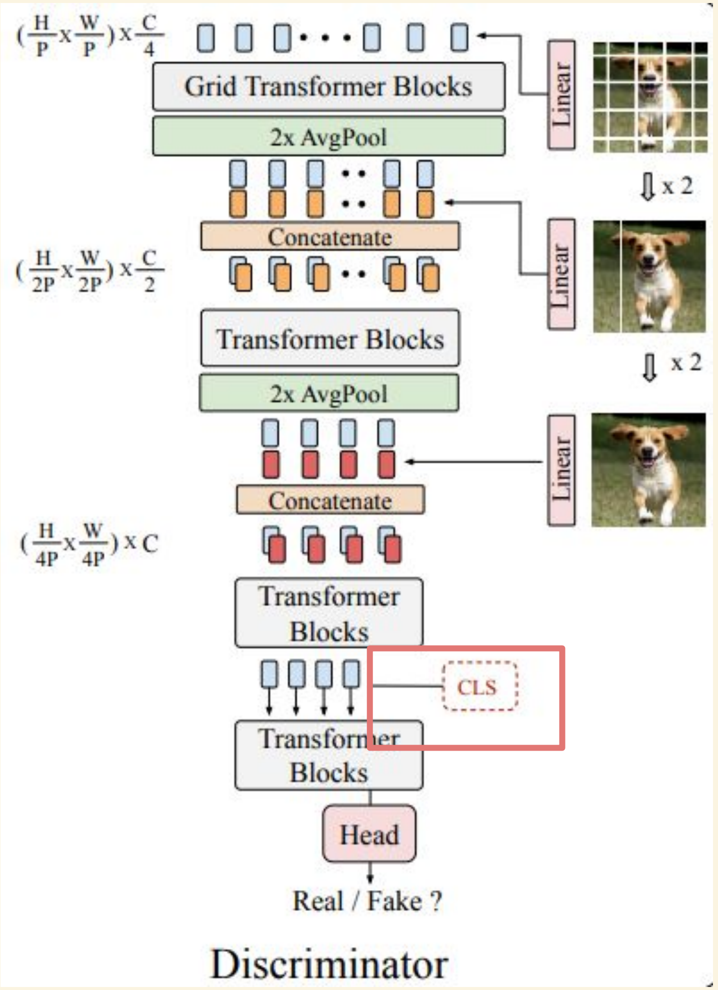
Figure 2. We introduce the GANsformer network, that leverages a bipartite structure to allow long-range interactions, while evading the quadratic complexity standard transformers suffer from. We present two novel attention operations over the bipartite graph: *simplex* and *duplex*, the former permits communication in one direction, in the generative context – from the latents to the image features, while the latter enables both top-down and bottom up connections between these two variable groups.

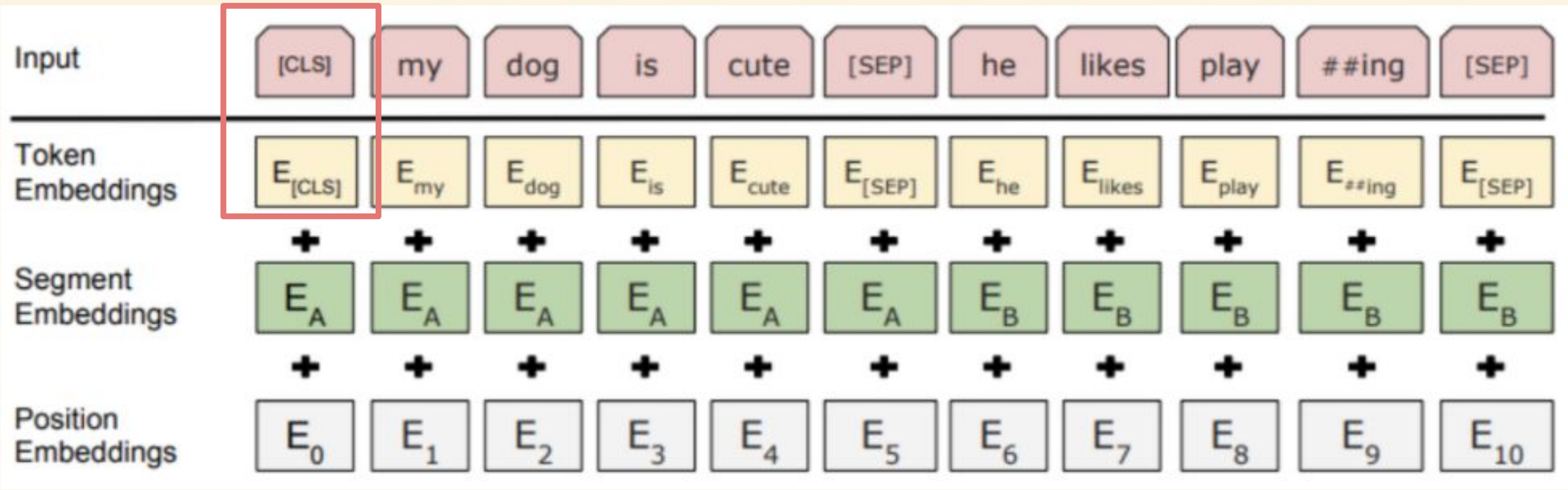


https://arxiv.org/abs/2103.01209 : Generative Adversarial Transformers

BERT

- CLS (special classification token) → 문장 가장 첫번째 토큰으로 삽입
  → 전체 layer 거치면 token sequence 결합된 의미 가짐





https://arxiv.org/abs/1810.04805 : BERT

Differential Augmentation

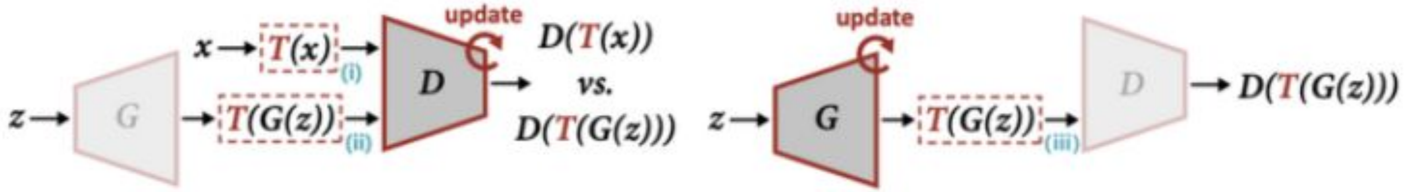
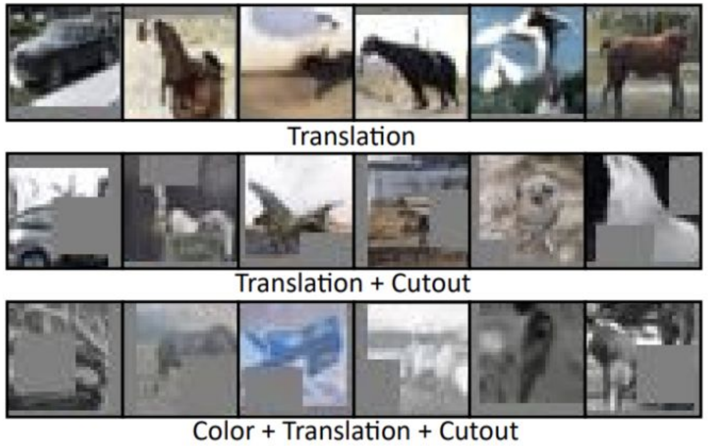


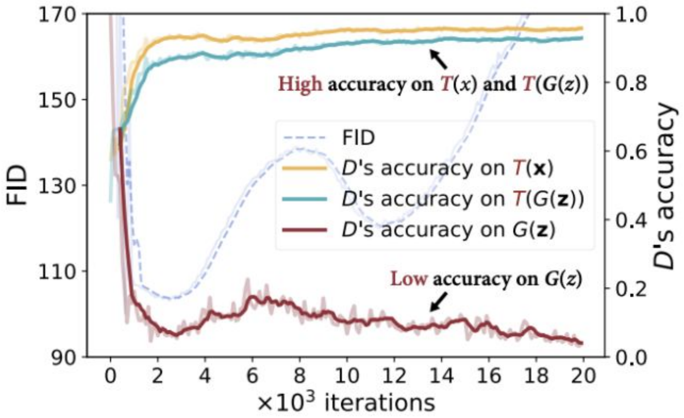
Figure 4: **Overview of DiffAugment** for updating D (left) and G (right). DiffAugment applies the augmentation T to both the real samples $\boldsymbol{x}$ and the generated output $G(\boldsymbol{z})$. When we update G, gradients need to be back-propagated through T, which requires T to be differentiable w.r.t. the input.

- real / generated image 모두
  differential augmentation 적용

→ gradient가 generator로 전파
→ distribution 이동 없이 discriminator 정규화



(a) "Augment reals only": the same augmentation artifacts appear on the generated images.



(b) "Augment D only": the unbalanced optimization between G and D cripples training.

- Transformation이 real에만

→ generator
: augmented image 분포와 일치하도록
→ distribution이동 + artifact

- real / generated image 둘 다 augmentation

→ G, D 균형 깨져서 convergence 감소

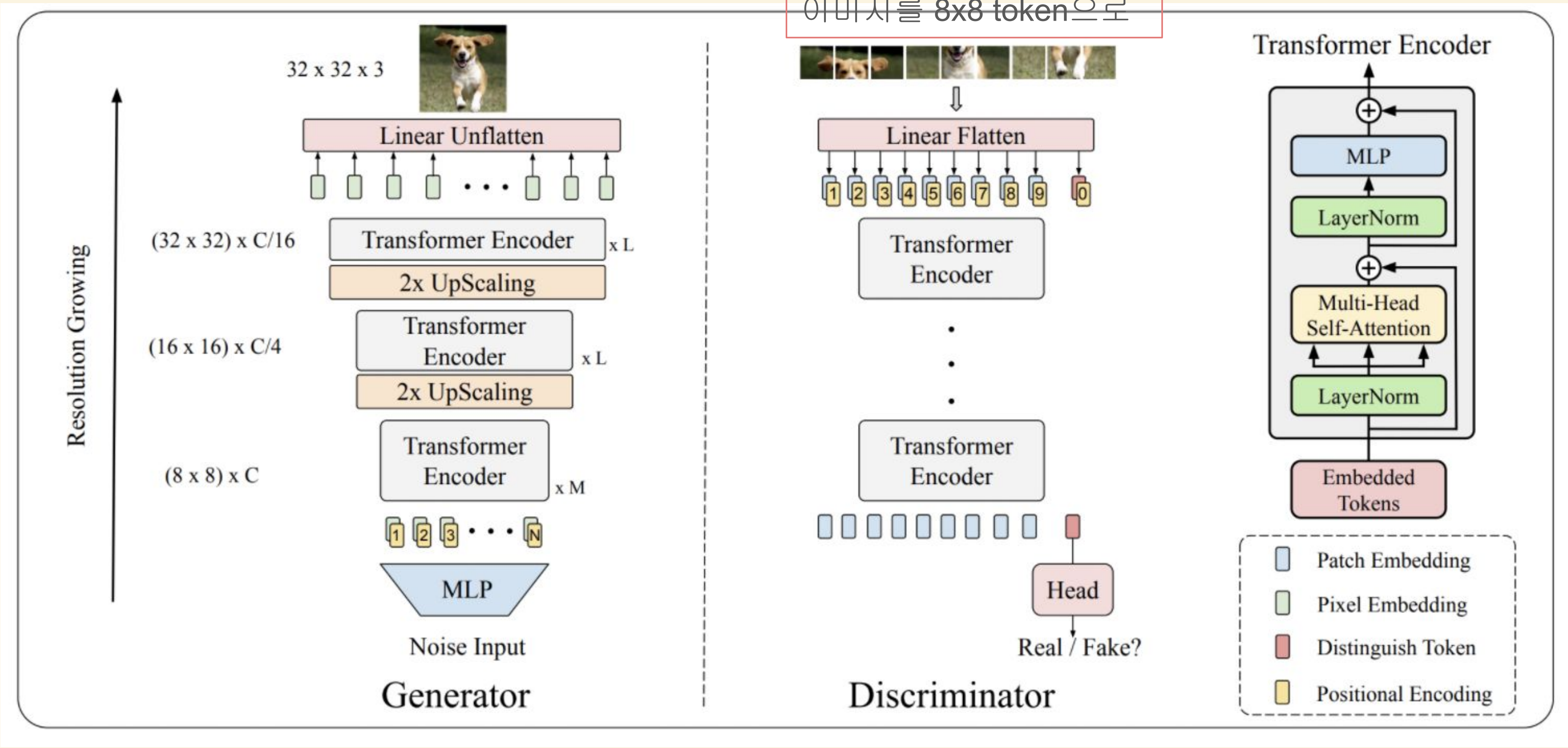
03

Technical approach

Memory-friendly generator, Multi-scale dicriminator, Grid self-attention, Training recipe, Relative position encoding

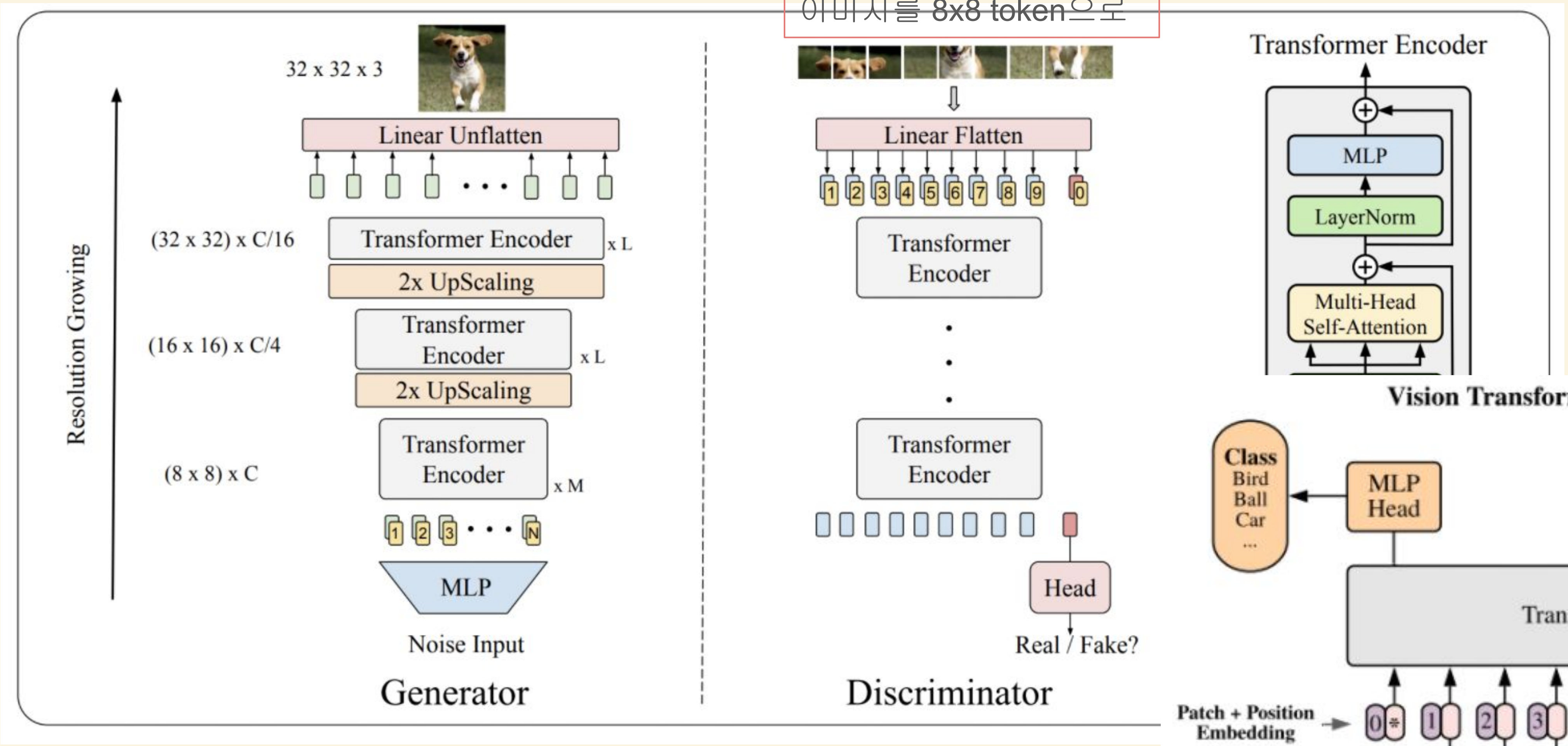
Previous architecture

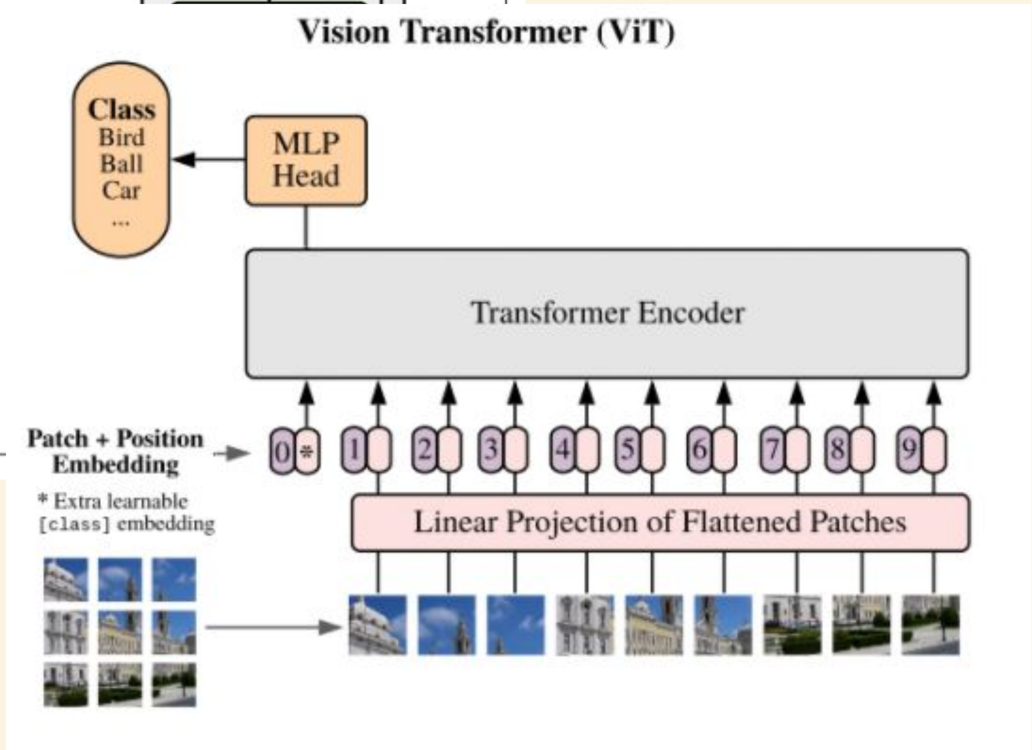
이미지를 8x8 token으로



Previous architecture

이미지를 8x8 token으로





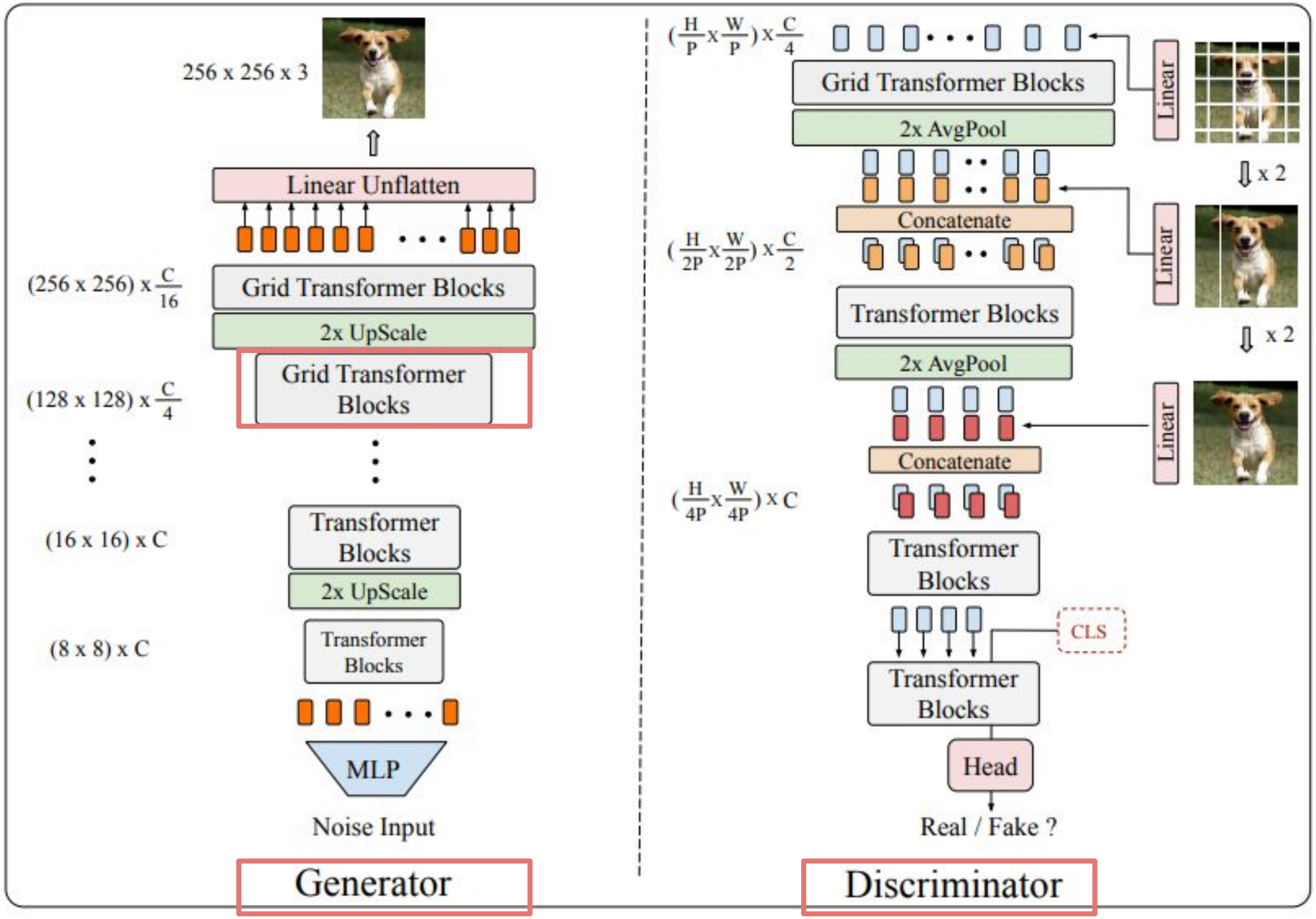
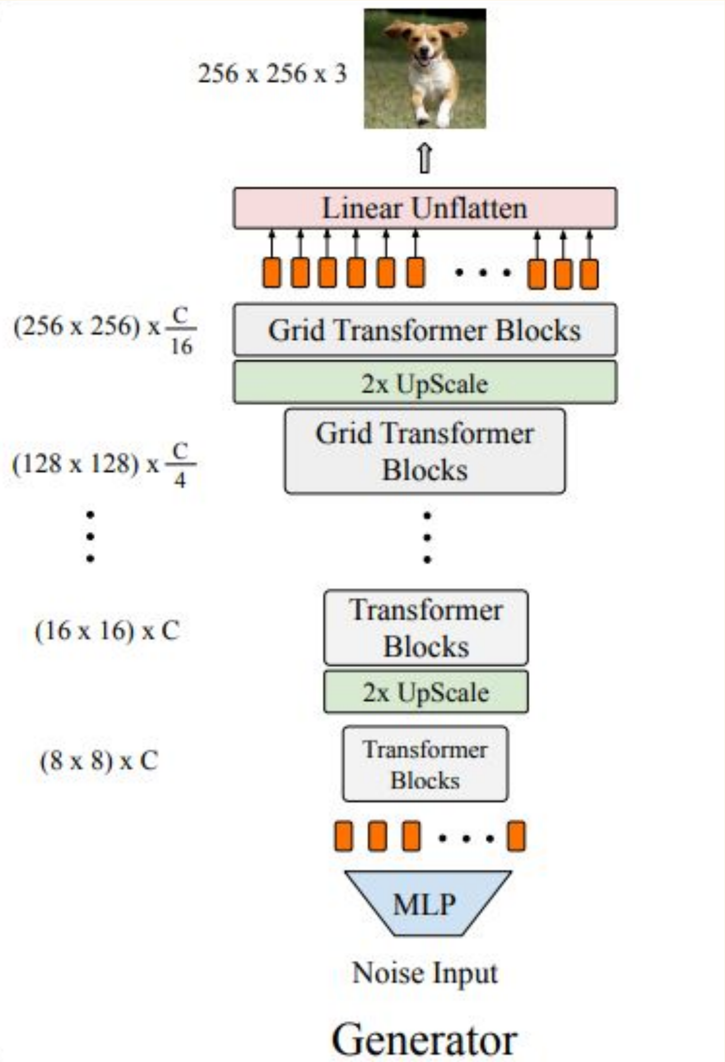


Figure 2: The pipeline of the pure transform-based generator and discriminator of TransGAN. We take 256×256 resolution image generation task as a typical example to illustrate the main procedure. Here patch size p is set to 32 as an example for the convenience of illustration, while practically the patch size is normally set to be no more than 8×8, depending on the specific dataset. `Grid Transformer Blocks` refers to the transformer blocks with the proposed grid self-attention. Detailed architecture configurations are included in Appendix C.

Memory - friendly Generator



- Transformer → daunting cost !
  32x32 image --- 1024 sequence !

- Use iteratively upscale the resolution at multiple stages
  → increase input sequence & reduce embedding dimension

- Input = random noise → MLP (H0 x W0 x C) + positional encoding

- Each stage → stacks several transformer blocks
  - gradually increase feature map resolution ~ H x W

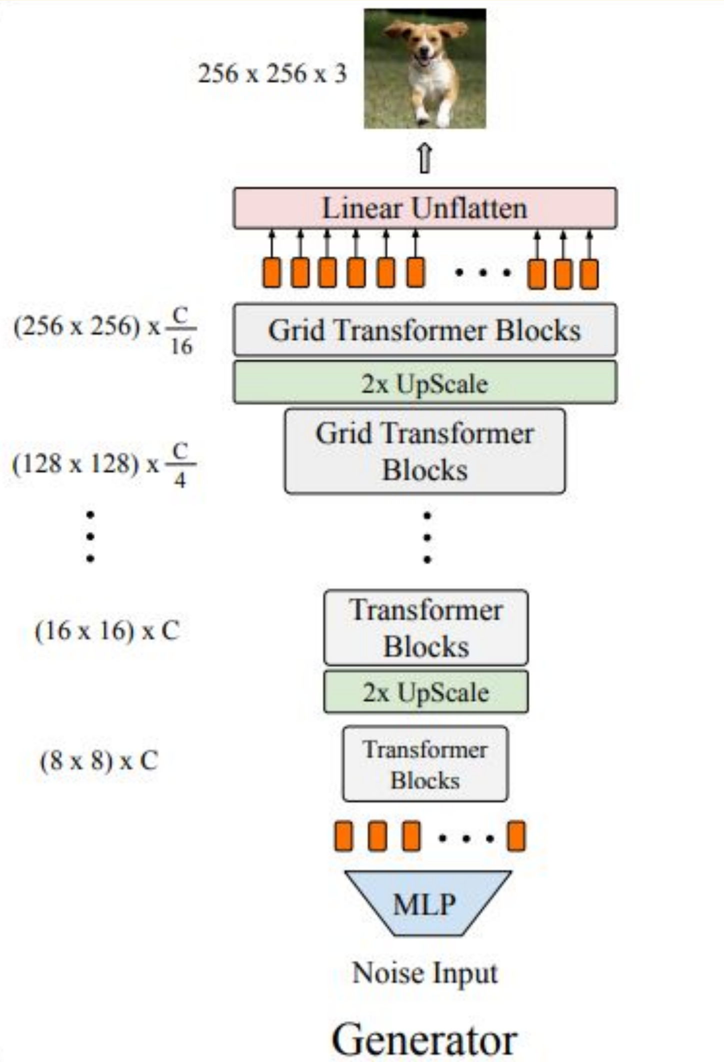
  - <u>upsampling module</u> (reshaping + resolution upscaling)
  ~64x64 resolution : 1D → 2D embedding + bicubic layer

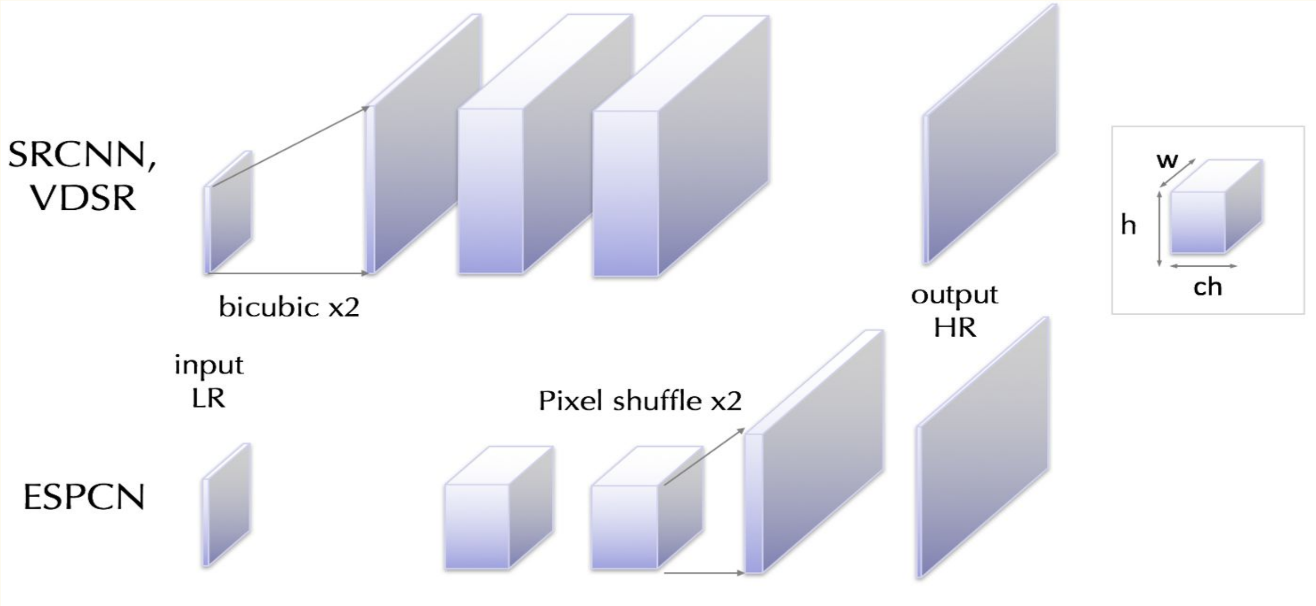
$X_i \in \mathbb{R}^{H_i \times W_i \times C}$ → $X_i' \in \mathbb{R}^{2H_i \times 2W_i \times C}$

64x64 resolution ~ : **<u>pixel shuffle module</u>**
(feature map resolution x2 / reduce embedding dimension 1/4)

- target resolution → projection (obtain RGB images)

Memory - friendly Generator





- upsampling module (reshaping + resolution upscaling)

~64x64 resolution : 1D → 2D embedding + bicubic layer

$X_i \in \mathbb{R}^{H_i \times W_i \times C}$ ⇨ $X_i' \in \mathbb{R}^{2H_i \times 2W_i \times C}$

64x64 resolution ~ : pixelshuffle module

(feature map resolution x2 / reduce embedding dimension 1/4)

- target resolution → projection (obtain RGB images)

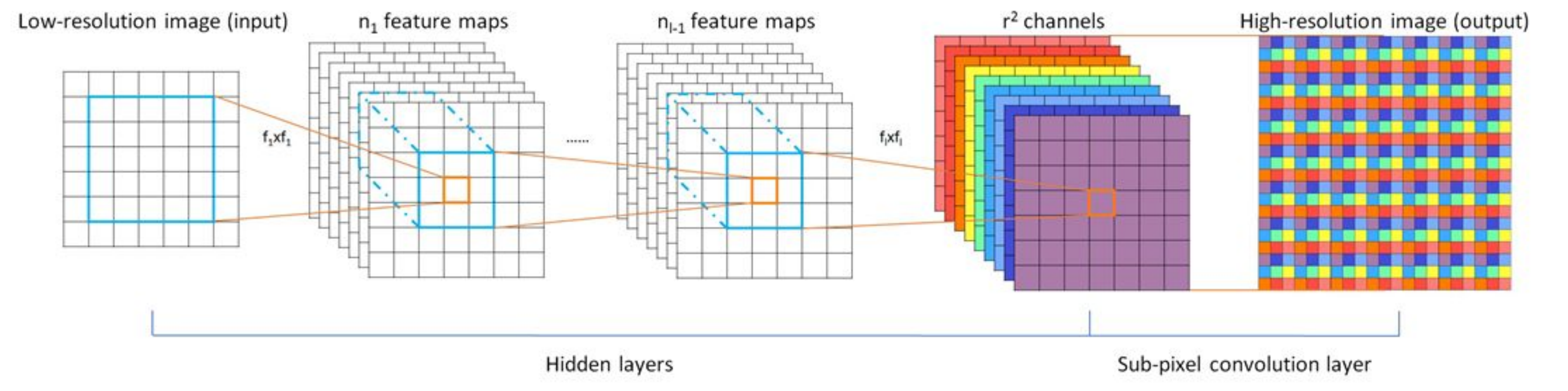
Memory - friendly Generator

Pixel shuffle

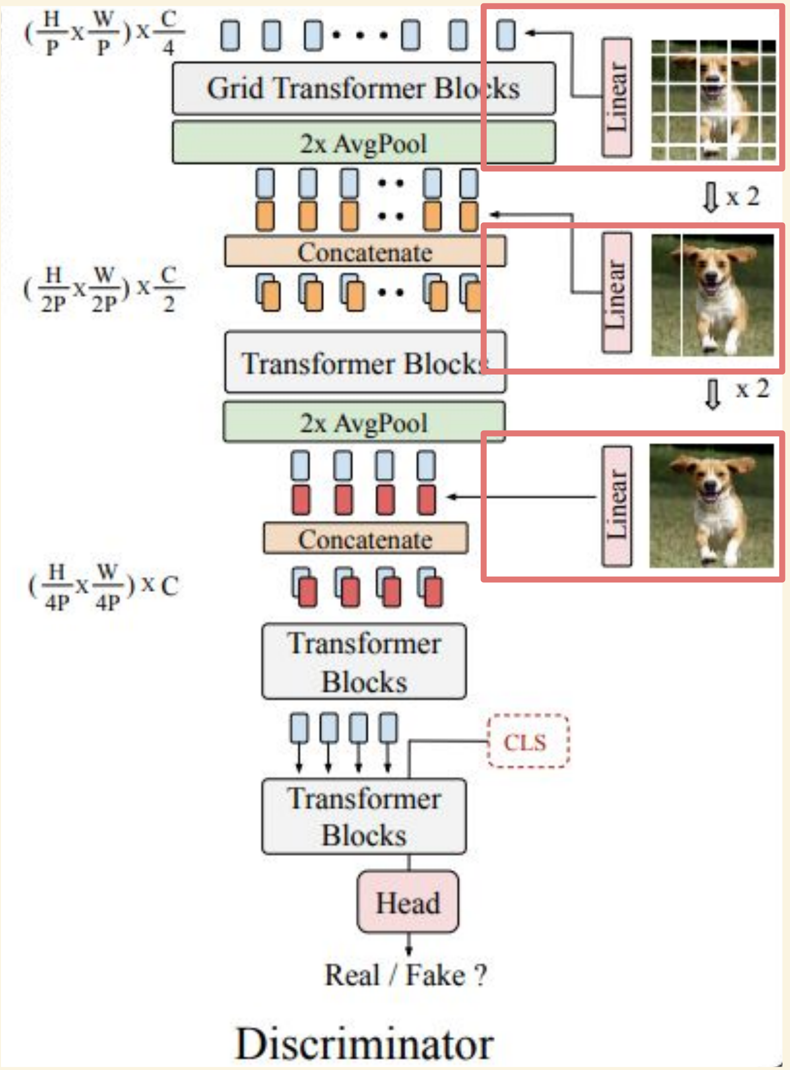
→ (H x W) -- upscaling x r -- (rH x rW)

→ **efficient sub-pixel convolution layer - r^2 channels**

→ **feature map 순서대로 배치**



Multi-scale Discriminator



- distinguish between real/fake images
  → large patch size = sacrifices low-level texture detail
  small patch size = results in a longer sequence (more memory)

- **Multi-scale discriminator**
  → take varying size of patches as input (3가지 size)
  1. size P
  2. size 2P
  3. size 4P

  → extract both semantic structure and texture details

- Average pooling layer
  → downsample the feature

- CLS token (beginning of the 1D sequence)
  → classification head

Grid Self-Attention

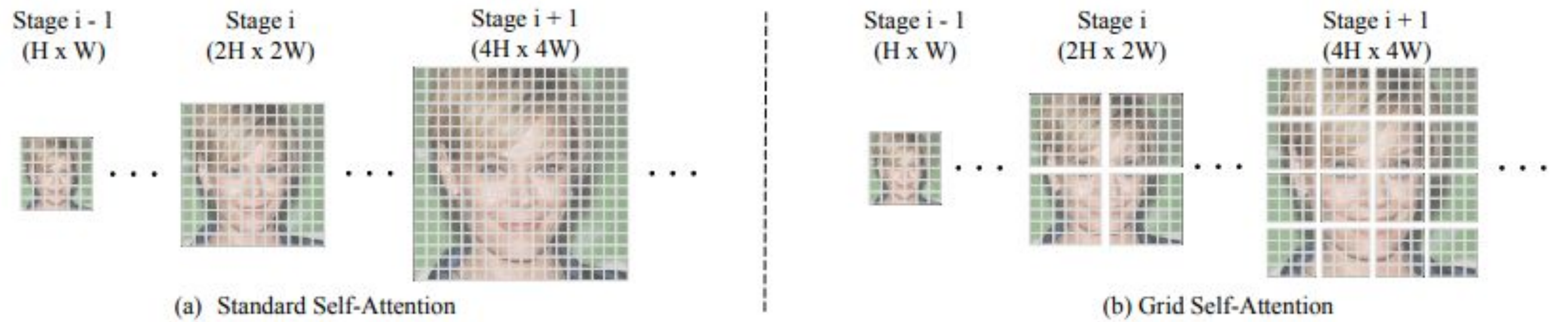


Figure 3: Grid Self-Attention across different transformer stages. We replace Standard Self-Attention with Grid Self-Attention when the resolution is higher than 32×32 and the grid size is set to be 16×16 by default.

- self attention → capture global correspondence
  → impedes efficiency when modeling higher resolution

- **Grid Self-Attention for high-resolution image generation (32x32 ~)**
  → full size feature map --- > partition into several non-overlapped grids
  → calculate token interactions inside each local grid
  → balance local + global

- Boundary artifact ? → enough training iterations 해결!
  → owing to larger, multi scale receptive field discriminator

Training recipe
- Data augmentation

- Transformer-based + not large data → inferior to CNNs in image recognition task
  → data augmentation is found to be crucial

- ‘Translation, Cutout, Color’ - using differential augmentation
  → surprising performance improvement for TransGAN

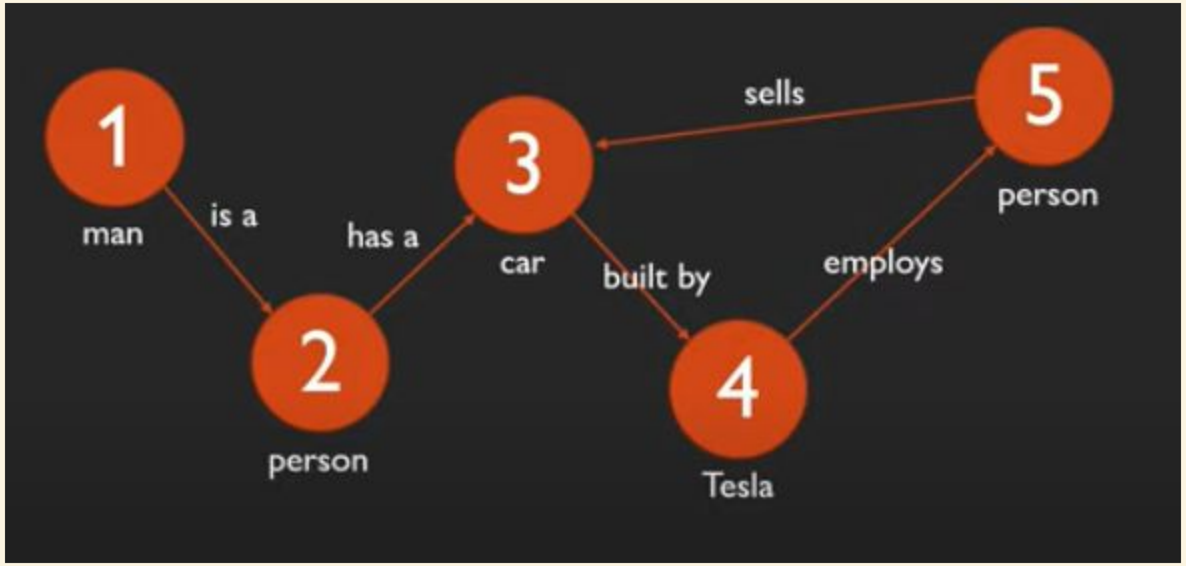
Table 2: The effectiveness of Data Augmentation on both CNN-based GANs and TransGAN. We use the full CIFAR-10 training set and DiffAug [68].

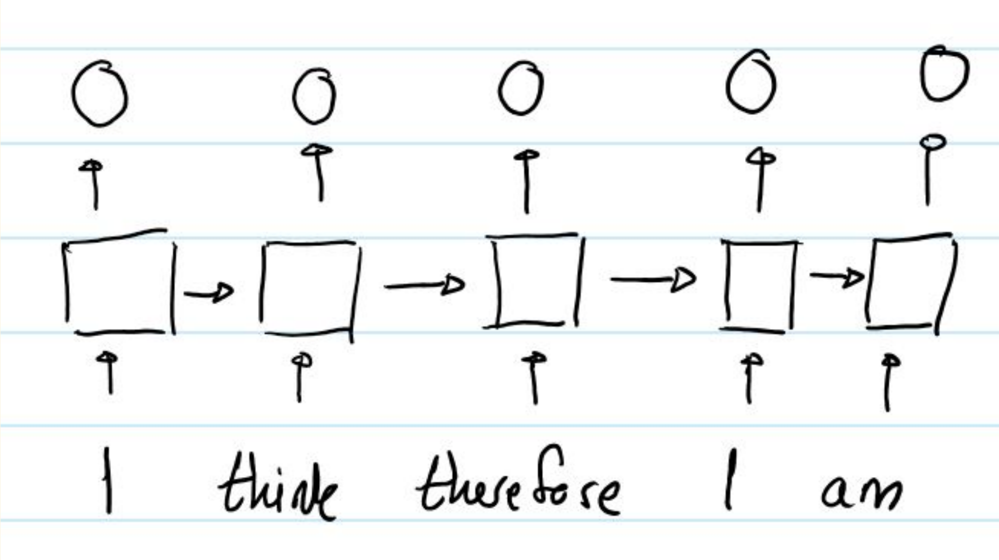
Methods	WGAN-GP		AutoGAN		StyleGAN-V2		TransGAN	
	IS ↑	FID ↓	IS ↑	FID ↓	IS ↑	FID ↓	IS ↑	FID ↓
Original	**6.49**	39.68	8.55	**12.42**	9.18	11.07	8.36	22.53
+ DiffAug [68]	6.29	**37.14**	**8.60**	12.72	**9.40**	**9.89**	**9.02**	**9.26**

Training recipe
- Relative Position Encoding

- Classical transformers → use deterministic position encoding

$$Attention(Q,K,V) = softmax((\frac{QK^T}{\sqrt{d_k}}V)$$





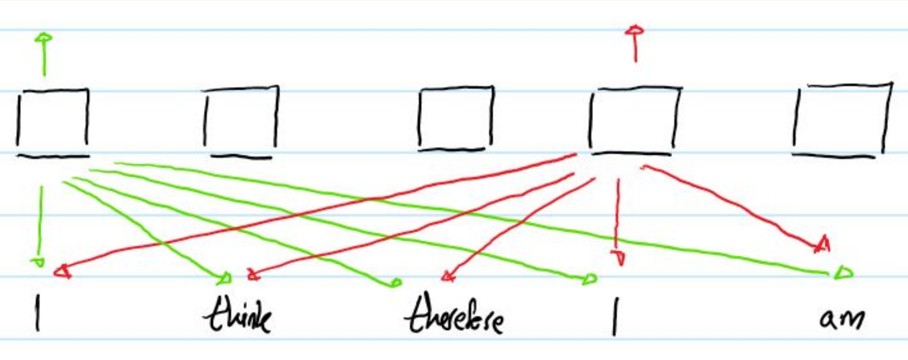
Training recipe

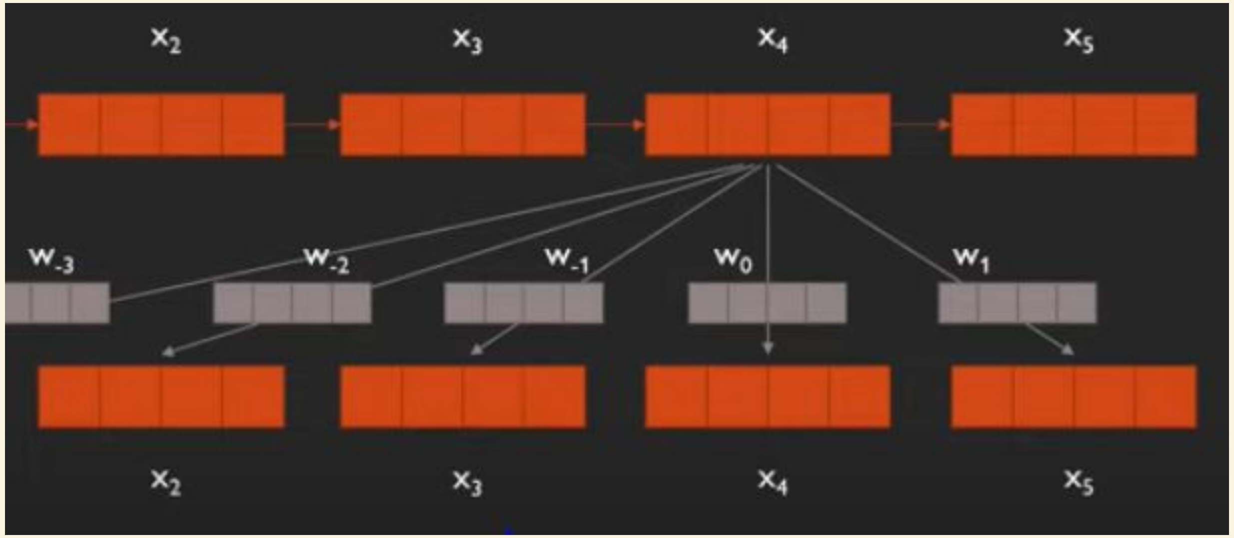
- Relative Position Encoding

- Relative position encoding → learns a stronger relationship between local contents

$$Attention(Q,K,V) = softmax(((\frac{QK^T}{\sqrt{d_k}} + E)V)$$

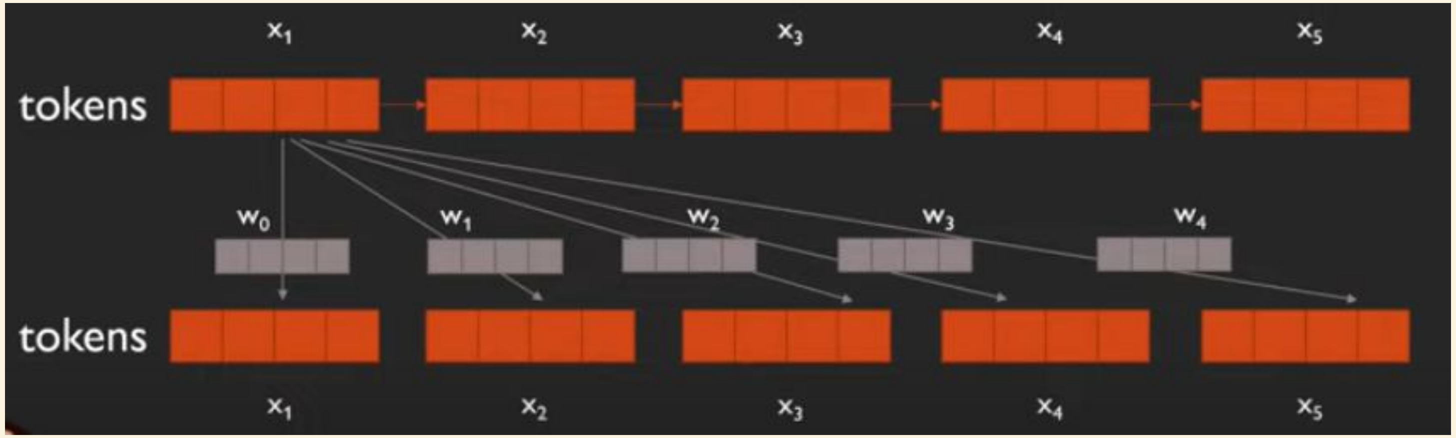
where $Q,K,V \in \mathbb{R}^{(H\times W)\times C}$ represent query, key, value matrices, H,W,C denotes the height, width, embedded dimension of the input feature map. The difference in coordinate between each query and key on H axis lies in the range of $[-(H-1), H-1]$, and similar for W axis. By simultaneously considering both H and W axis, the relative position can be represented by a parameterized matrix $M \in \mathbb{R}^{(2H-1)\times(2W-1)}$. Per coordinate, the relative position encoding E is taken from matrix M and added to the attention map QK^T as a bias term, shown as following,

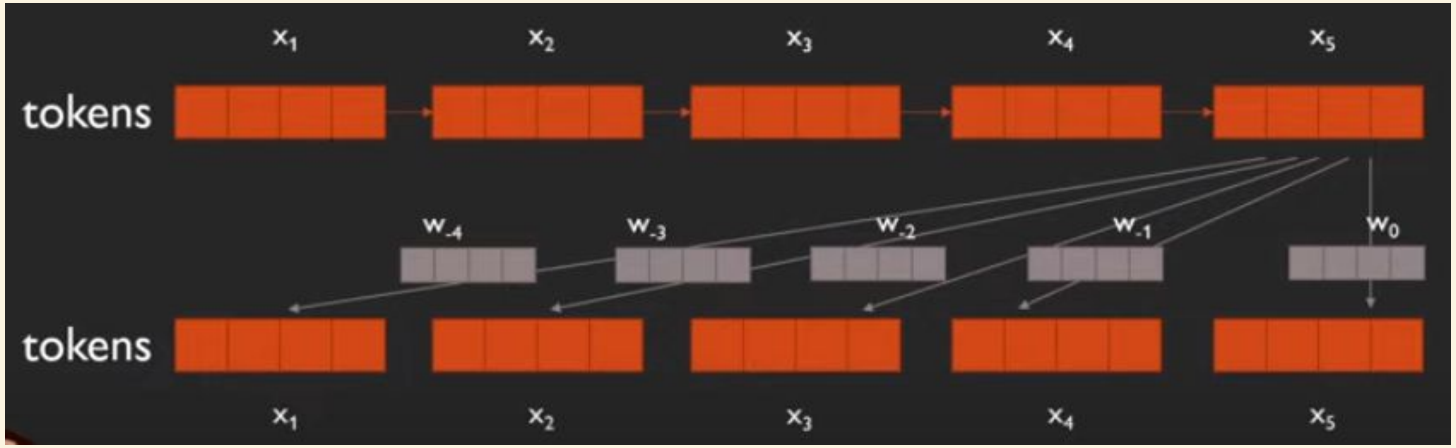




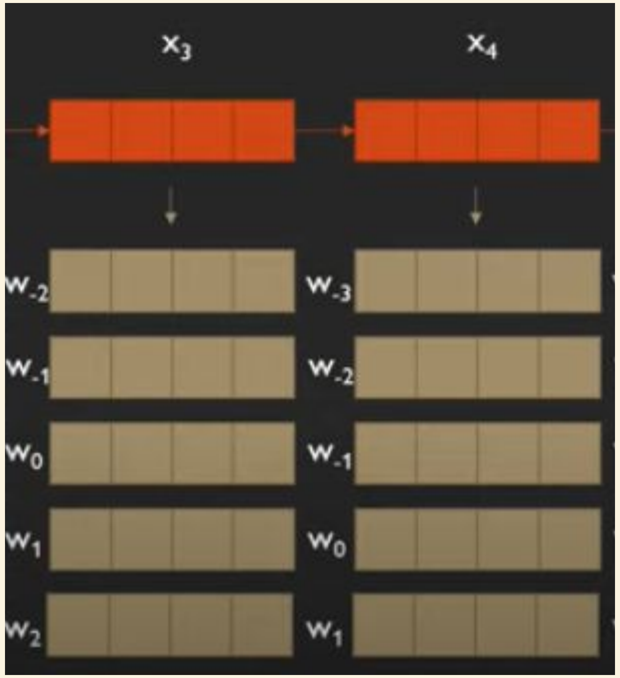
Training recipe

- Relative Position Encoding





- 5 tokens
  → -4 ~ 4 (9 embeddings)



$$e_{ij} = \frac{x_i W^Q (x_j W^K)^T + x_i W^Q (a_{ij}^K)^T}{\sqrt{d_z}} \qquad (5)$$

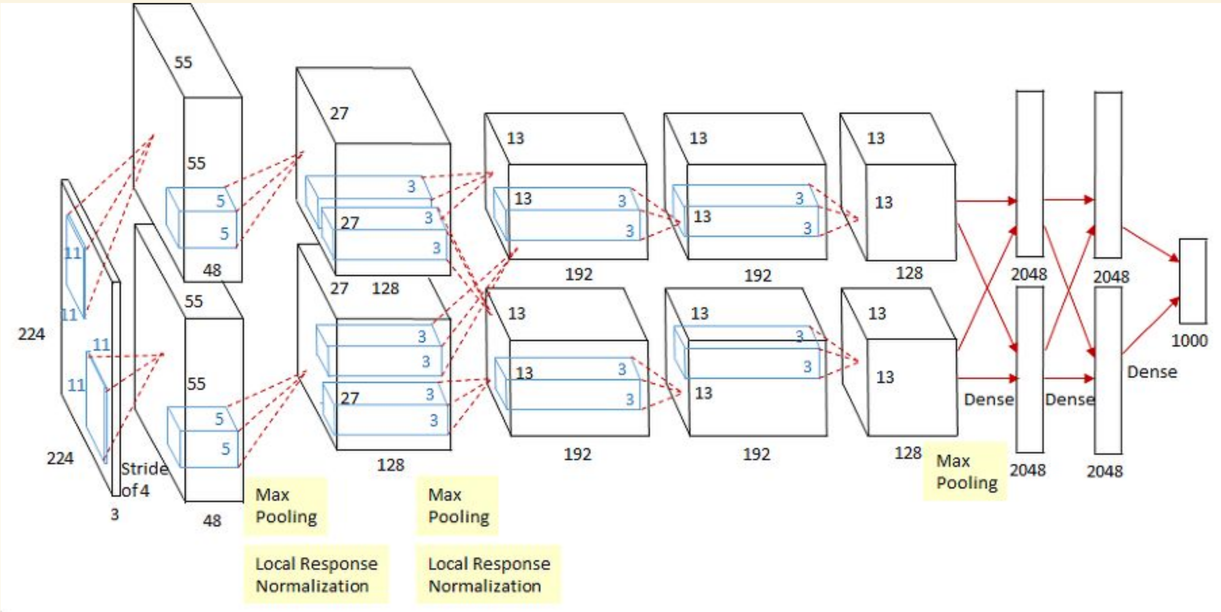
Training recipe

- Modified Normalization

- token-wise scaling layer → prevent transformer blocks' magnitude being too high
  → simple re-scaling without learnable parameters suffices

$$Y = X/\sqrt{\frac{1}{C}\sum_{i=0}^{C-1}(X^i)^2 + \epsilon}$$

- Local Response Normalization (in AlexNet)
  → RELU results normalize (lateral inhibition)
  (양수값 받으면 그 값을 그대로 전달 → 강한 자극이 약한 자극 전달 막음)



04

Table of Contents

Lorem Ipsum is simply dummy text of the printing and typesetting industry.

Comparison with State-of-the-art 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**

- Datasets : CIFAR-10(32), STL-10(48), CelebA(128)
- Implementation : generator batch 128, discriminator batch 64, 16 V100 GPUs

Scaling Up to Higher-Resolution

- CelebA-HQ (256), LSUN Church (256)
  → FID 10.28 / FID 8.94

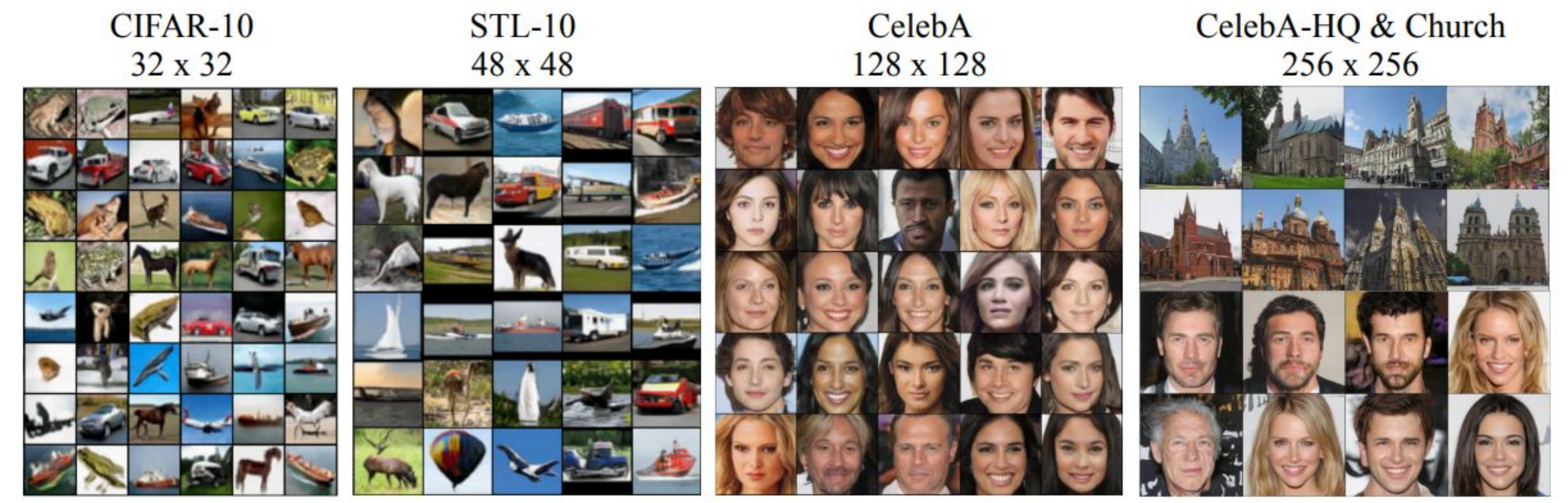


Figure 4: Visual results produced by TransGAN on different datasets, as resolution grows from 32 × 32 to 256 × 256. More visual examples are included in Appendix F.

Ablation

Table 3: The ablation study of proposed techniques in three common dataset CelebA(64×64), CelebA(128×128, and LSUN Church(256×256)). "OOM" represents out-of-momery issue.

Training Configuration	**CelebA** (64x64)	**CelebA** (128x128)	**LSUN Church** (256x256)
(A). Standard Self-Attention	8.92	**OOM**	**OOM**
(B). Nyström Self-Attention [62]	13.47	17.42	39.92
(C). Axis Self-Attention [65]	12.39	13.95	29.30
(D). Grid Self-Attention	9.89	10.58	20.39
+ Multi-scale Discriminator	9.28	8.03	15.29
+ Modified Normalization	7.05	7.13	13.27
+ Relative Position Encoding	6.14	6.32	11.93
(E). Converge	**5.01**	**5.28**	**8.94**

Ablation

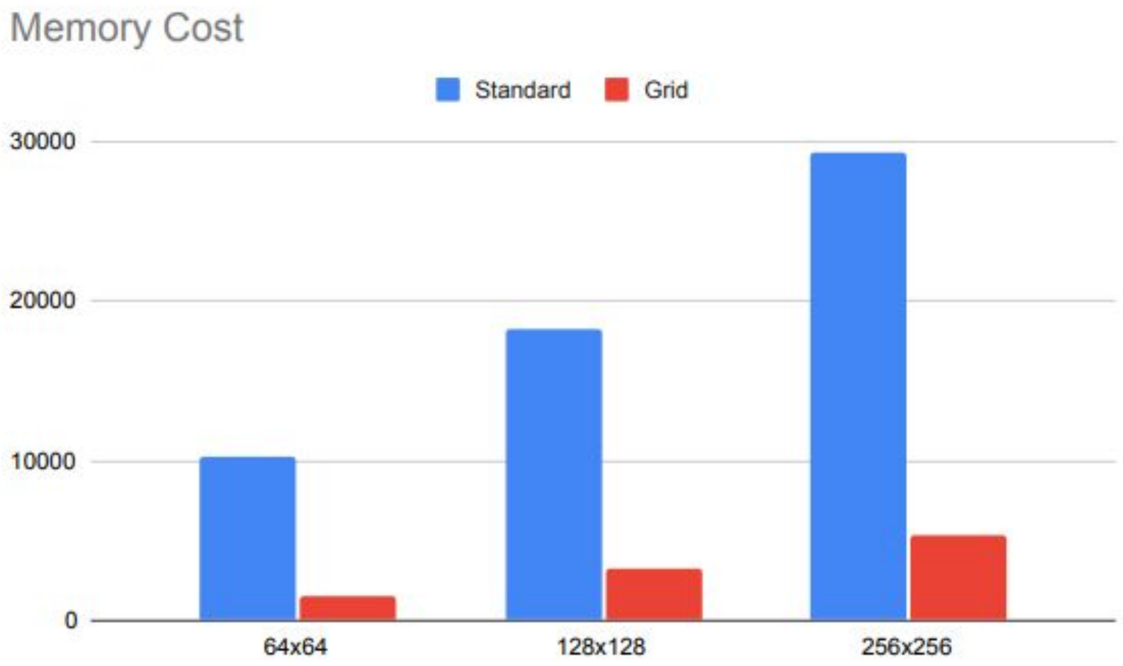


Figure 8: Memory cost comparison between standard self-attention and grid self-attention

Q & A

Thank you!