# ViT

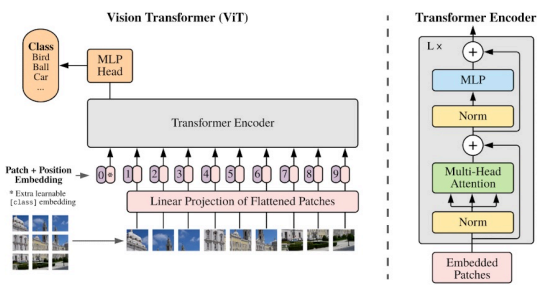
## 1. Concepts

CNNs inductive bias
- translation equivariance
- locality

'inductive bias': The inherent assumption or conditions that a model possesses by default.

Due to Transformer doesn't have these inductive biases, it has to learn from datas.
Therefore, ViT needs more datas.

## 2. Methods




### ① Patch Partitioning

$x \in \mathbb{R}^{H \times W \times C}$ (raw image) $\xrightarrow{\text{reshape}}$ $x \in \mathbb{R}^{N \times (P^2 \cdot C)}$ (flattened 2D Patches.)

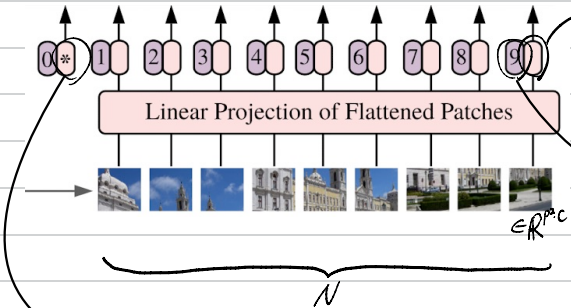
$N$: number of patches. / length of input sequence.
$P$: patches' $H, W$.

ex)


$N = 9$

$N = HW/P^2$

## ② Linear Projection of Flatten Patches.




$E \in \mathbb{R}^D$

"The Transformer uses constant latent vector D through all of its layers"

$E_{pos} \in \mathbb{R}^D$

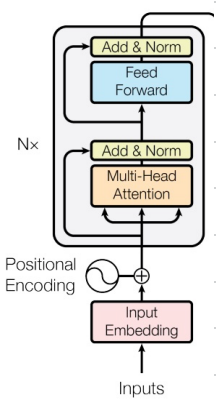
Learnable 1D position Embedding

• CNN의 inductive bias 중 하나인 locality를 보완하기 위한 방법.

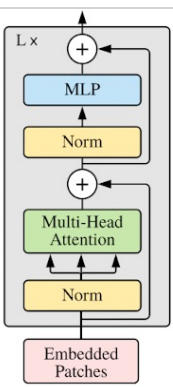
"Similar to BERT's [class] token" ($z_0^0 = x_{class}$)

## ③ Transformer Encoder Architecture.

Encoder of Trm



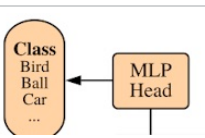

Encoder of ViT.



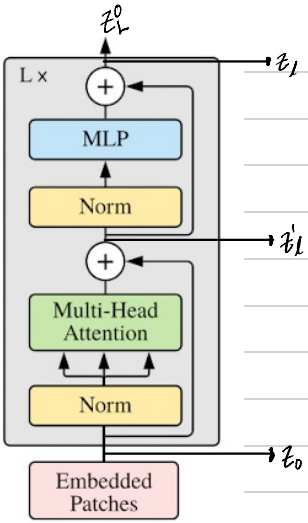

Pre-Layer Normalization's Strength

- Improvement training stainability.
- Residual Connection의 효과 유지.

## ④ Pre-training vs. Fine-tuning.




"MLP with one hidden layer at pre-training time, Single linear layer at fine-tuning time."




$x_{class}$

$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \cdots; \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}, \qquad \mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D},\ \mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D} \tag{1}$$

$$\mathbf{z}'_\ell = \text{MSA}(\text{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \qquad \ell = 1 \ldots L \tag{2}$$

$$\mathbf{z}_\ell = \text{MLP}(\text{LN}(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell, \qquad \ell = 1 \ldots L \tag{3}$$

$$\mathbf{y} = \text{LN}(\mathbf{z}_L^0) \tag{4}$$

$z_L^0 = \text{Encoder}(z_0)$

# Swin Transformer

## 1. Concepts

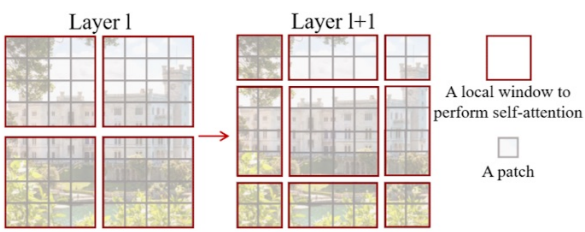
① hierarchical feature maps.

Swin transformer model can conveniently leverage advanced techniques for dense prediction such as feature pyramid networks (FPN) or U-Net.

dense prediction: tasks whose predictions are made for every location in the input data, such as images or videos.

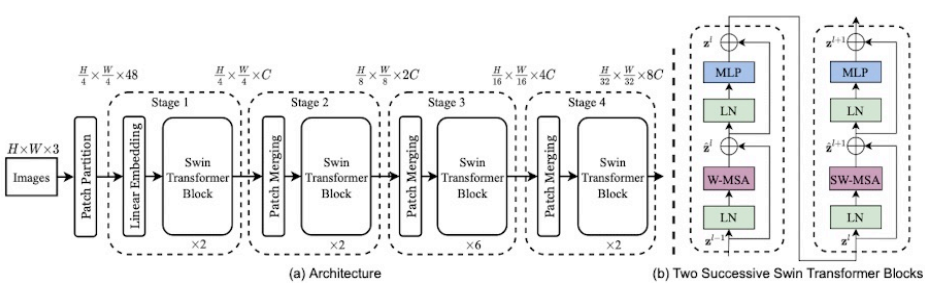
② linear computational complexity.

Linear computational complexity is achieved by computing self-attention locally within non-overlapping windows that partition an image.




This is an illustration of the shifted window. The self-attention computation in the new windows crosses the boundaries of the previous windows in layer $l$, providing connections among them.

## 2. Methods



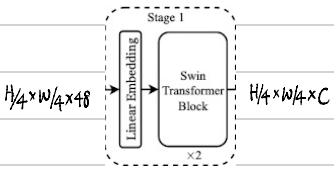

① Patch Partition.

- The input RGB image is devided into non-overlapping 4×4 patches. (a.k.a. tokens).

raw data $\in \mathbb{R}^{H\times W\times 3} \Rightarrow$ (patch partition) $\mathbb{R}^{H/4\times W/4\times 48} \Rightarrow$ (linear embedding within 1) $\mathbb{R}^{H/4\times W/4\times C}$.

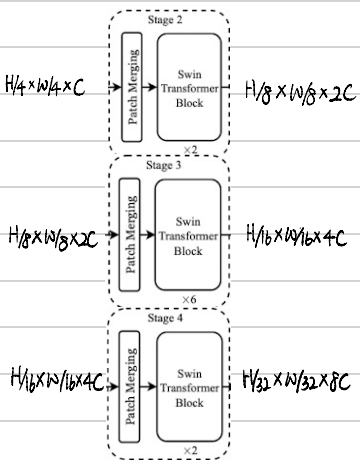
## ② Hierachical Representation.

### Stage 1.



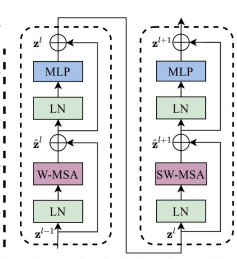

- Starts with the H/4 × W/4 token map created from the input image.
- Applies a linear embedding (within) layer to generate the initial feature ma

### Stage 2,3,4.




- In stage 2, 3, and 4, the resolution of the feature map is progressively halved, while the feature dimension is doubled at each stage using patch merging layers.
- This hierarchical process allows the model to capture increasingly abstract and high-level representations, with Swin Transformer blocks applied at each stage to refine the features.

## ③ Swin Transformer Blocks.



Two Successive Swin Transformer Blocks

1. Window-based Multi-head Self-Attention (W-MSA)

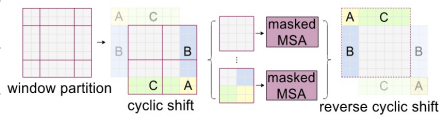
Computes self-attention within local windows to reduce computational complexity.

2. Shift Window Multi-head Self-Attention (SW-MSA)

Shifts the window partitions between layers to introduce cross-window connections

## ④ Cyclic Shift

The shift window partitioning increases the number of windows, leading to inefficient computation and memory usage.




1. Cyclic Shifting

Windows that are moved out of one side are wrapped around to the opposite side.

2. Masking

Ensure that self-attention calculations are restricted to the original shifted window regions, maintaining the correct connections

# Stable Diffusion

## 1. Concepts

### The problems of GAN

- mode-collapse : When a generative model is trained, it tends to focus on a specific distribution of the data, resulting in a lack of diversity. It repeatedly generates similar images.
- training instabilities.

### Solution by using DMs.

The paper emphasize that Diffusion Models can effectively learn various modes of the data distribution through their mode covering behavior.
- likelihood-based learning method. (ex. VAEs, AR model, DMs)
- gradual denoising process.
- training stability : DMs doesn't use adversarial loss.

### The problems of DMs.

- Requiring massive computational resources.
- Evaluating a trained model is also

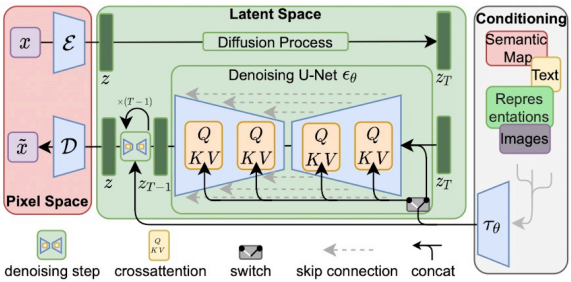
### Solution proposed in the paper. (LDMs)

As with any likelihood-based model, learning can be roughly divided into two stages:

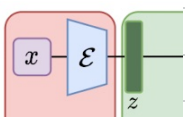
1. Perceptual compression.
   Training an autoencoder which provides a lower-dimensional representational space which is perceptually equivalent to the data space. (aka latent space)
2. Semantic compression.
   Training DMs in the learned latent space, which exhibits better scaling properties with respect to the spatial dimensionality.

"Moreover, we design a general-purpose conditioning mechanism based on cross-attention, enabling "multi-modal training"

# 2. Methods.



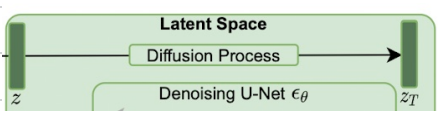

## ① Perceptual Compression.



Raw data $x \in \mathbb{R}^{H \times W \times C}$ → Encoder of autoencoder $\mathcal{E}$ → Latent representation. $z \in \mathbb{R}^{h \times w \times c}$

- The original resolution $H, W$ is reduced, and the number of channels $C$ is increased.

## ② Forward Diffusion Process.

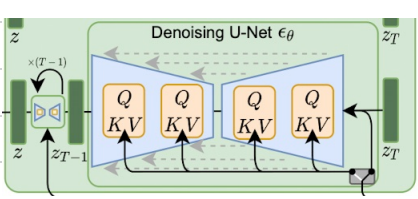



$z_t = \sqrt{1-\beta_t}\, z_{t-1} + \sqrt{\beta_t}\,\epsilon$ · $\beta_t$: A scheduling parameter controlling the noise level.

- The Forward Diffusion Process progressively adds noise $\epsilon \sim N(0,1)$ to the latent representation $z_0$ producing $z_t$

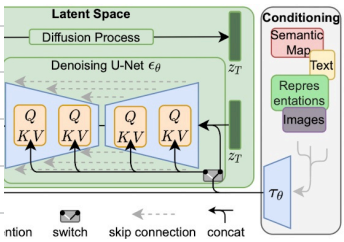
## ③ Reverse Diffusion Process.

### 1. Using a UNet Architecture.

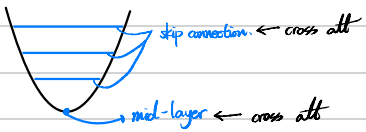



input: $z_t$ target: $\epsilon(z_t, t)$

## 2. Cross Attention.




- Conditioning (eg., Text, Images, ···) is incorporated into UNet via Cross Attention mechanism.
- Cross Attention is applied in the mid-layers and skip connections.

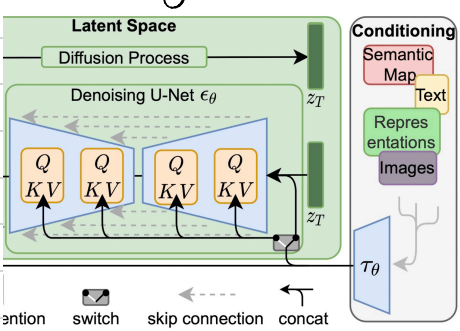



## 3. Loss Function.

- The UNet is trained by minimizing the L2 loss between the predicted noise $\epsilon_\theta(z_t, t)$ and the ground truth noise $\epsilon$:

$$L = E_{\epsilon(x), \epsilon \sim N(0,1), t}\left[\| \epsilon - \epsilon_\theta(z_t, t) \|_2^2\right]$$

## ④ Conditioning




- A domain specific encoder $\tau_\theta$ that projects $y$ to an intermediate representation $\tau_\theta(y) \in \mathbb{R}^{M \times d_\tau}$ — $M$: num of input tokens.

$$Q = W_Q^{(i)} \cdot \varphi_i(z_t) \quad K = W_K^{(i)} \cdot \tau_\theta(y) \quad V = W_V^{(i)} \cdot \tau_\theta(y).$$

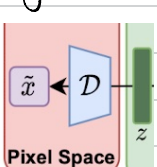
$\varphi_i(z_t) \in \mathbb{R}^{N \times d_\epsilon^i}$

$W_V^{(i)} \in \mathbb{R}^{d \times d_\epsilon^i}$, $W_Q^{(i)} \in \mathbb{R}^{d \times d_\tau}$, $W_K^{(i)} \in \mathbb{R}^{d \times d_\tau}$

$\Rightarrow \underset{d \times d_\tau \cdot d_\tau \times d \cdot d \times d_\epsilon^i}{\text{softmax}}\left(\frac{QK^T}{\sqrt{d}}\right) \cdot V \in \mathbb{R}^{d \times d_\epsilon^i}$

Maybe $W_0 \in \mathbb{R}^{N \times d_\epsilon^i}$

## ⑤ Image Generation.




Latent representation. Decoder of autoencoder Raw data

$z \in \mathbb{R}^{h \times w \times C}$ → $\mathcal{D}$ → $\tilde{x} \in \mathbb{R}^{H \times W \times C}$

- The Autoencoder's Decoder reconstructs the latent representation $z_0$ back into the original image space.