Speech & Vision Transformer

서울대학교 IDEA 연구
실
이해영

Vision Transformer

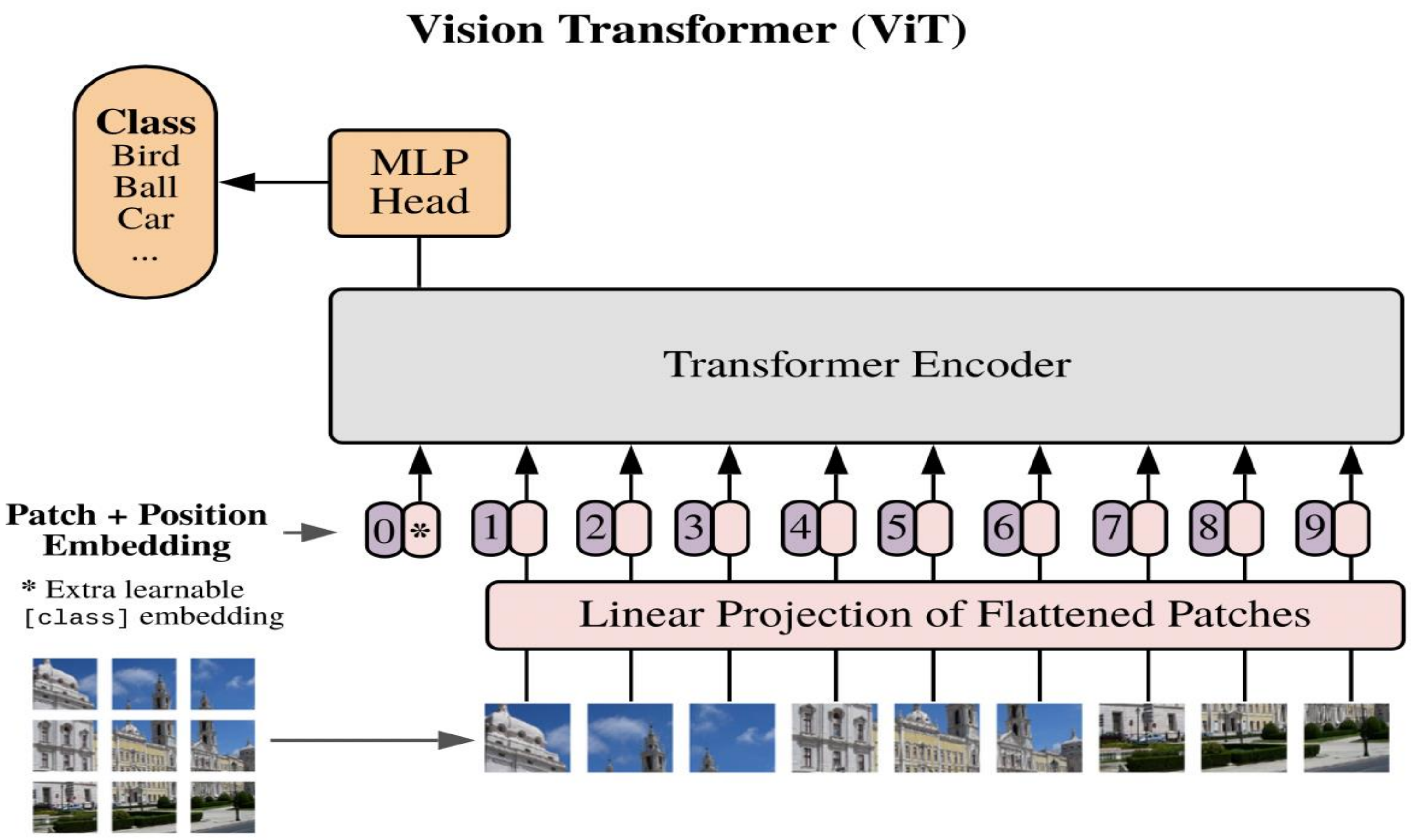
Outline

01 Vision Transformer (ViT)

02 Swin Transformer

01. Vision Transformer

An Image is Worth 16 x 16 Words: Transformers for Image Recognition at Scale (ICLR 2021)



- ViT is used for image classification tasks.
- ViT use Encoder part of Transformer.
- ViT divides an image into multiple patches, flattens them, and feeds these patches as input sequences into the Transformer.

01. Vision Transformer

Notation

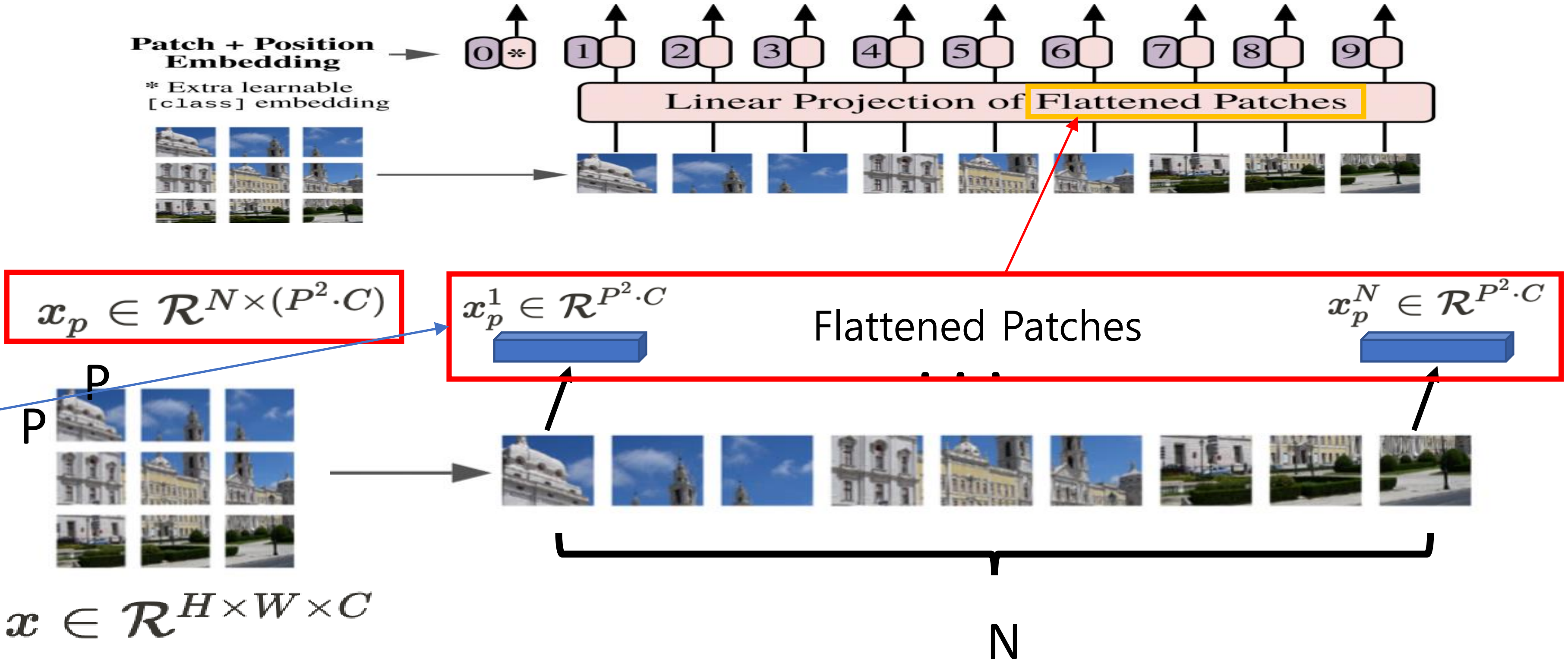
- H, W : Height and width of the input image
- C : Number of channels in the image (c=3 for rgb)
- P : Patch size (resolution of each patch is $P \times P$)
- N : Number of patches, $N = HW/P^2$
- D : Latent vector size
- x : Input image ($x \in \mathbb{R}^{H\times \mathrm{W} \times C}$)
- x_p : Flattened image patches ($x_p \in \mathbb{R}^{\mathrm{N}\times (P^2 \cdot C)}$)
- E : Patch embedding projection (trainable)
- $Epos$: Positional embeddings
- z : Feature embedding at different layers.

01. Vision Transformer

Model Architecture

- Converts 2D image $x \in \mathbb{R}^{H \times W \times C}$ into a 1D sequence $x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$
- Split the input image into 9 patches. A positional embedding is added at the beginning of each patch to preserve its order in the sequence.

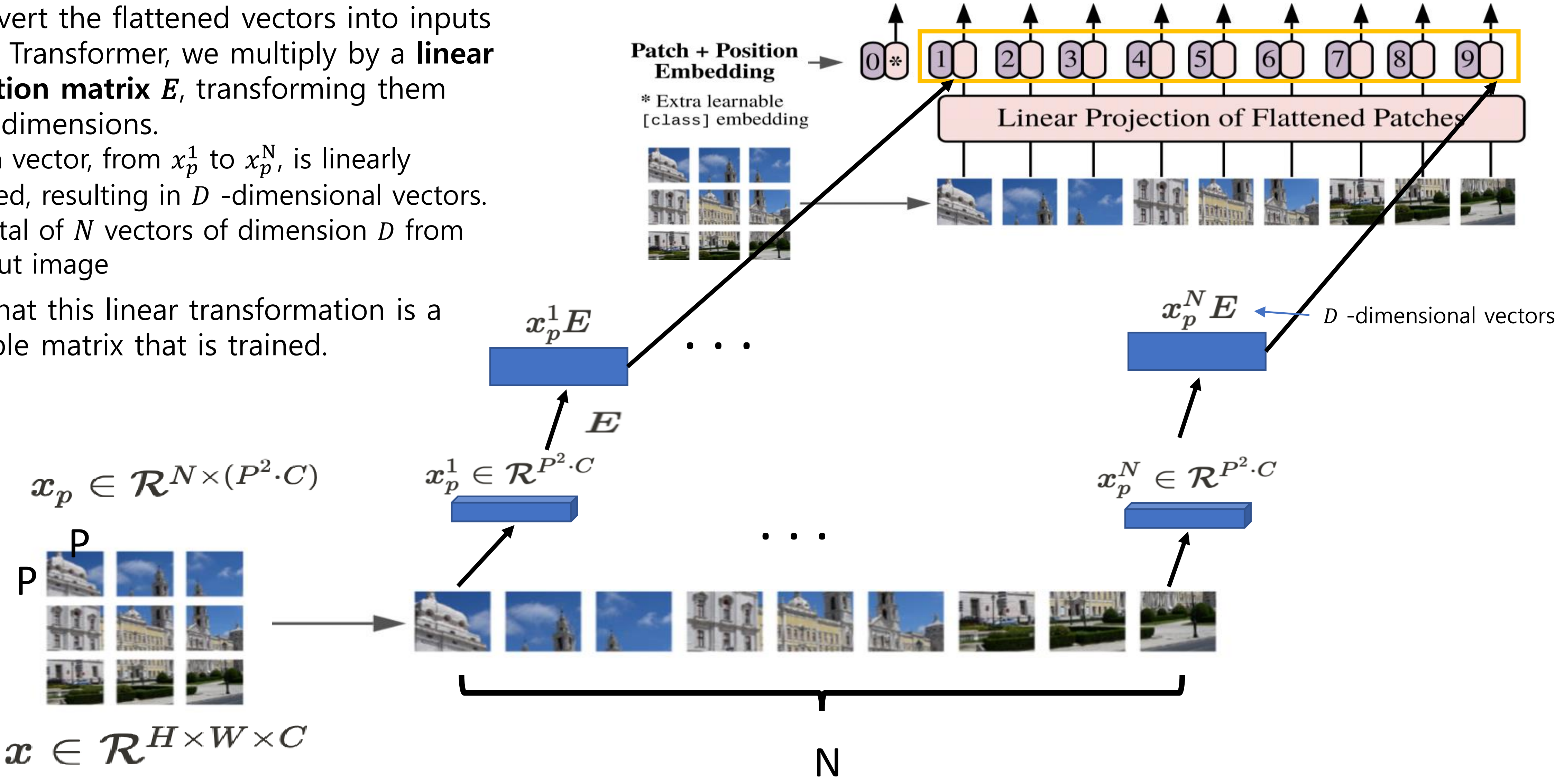
ex) for a 48*48*3 image
and patch size $P = 16$
→ there are 9 patches
(3*3=9), each flattened
into a 16*16*3 vector.
→ total of 9 vectors



https://www.youtube.com/watch?v=7lxKirsixuQ

01. Vision Transformer

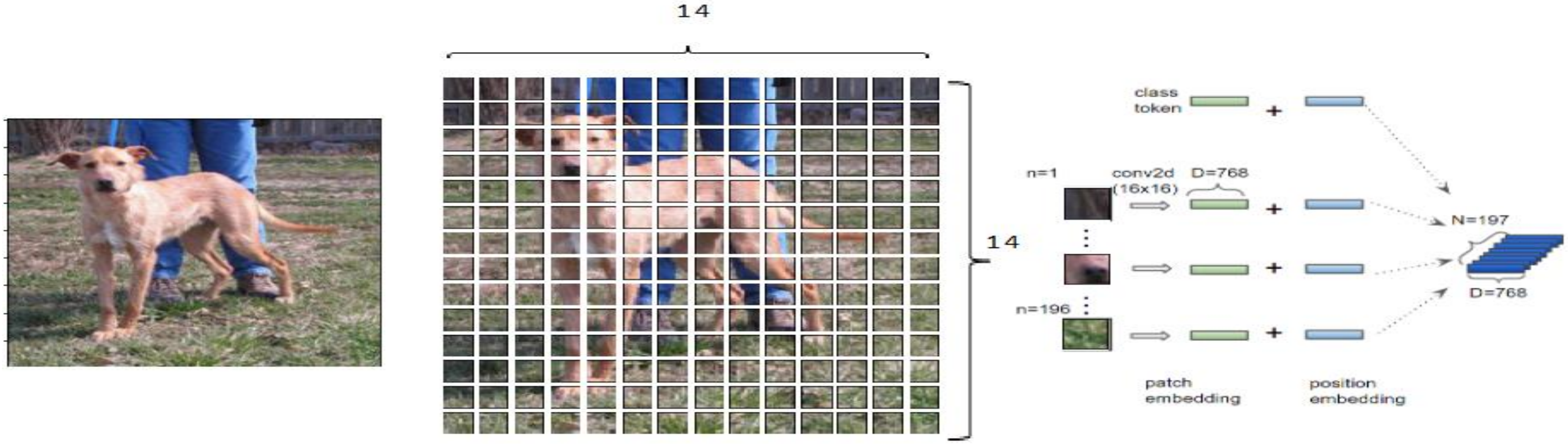
- To convert the flattened vectors into inputs for the Transformer, we multiply by a **linear projection matrix** E, transforming them into D dimensions.
  → each vector, from x_p^1 to x_p^N, is linearly projected, resulting in D -dimensional vectors.
  → A total of N vectors of dimension D from the input image
- Note that this linear transformation is a learnable matrix that is trained.



01. Vision Transformer

ex) input image size = 224*224*3, patch size = 16*16 → results in 14*14 patches = 196 patches in total.

- Eatch patch (16*16*3) is linearly projected to D dimension (e.g., D =768).
- Class embedding is added for image classification (like the CLS token in BERT).
- Position embeddings are added to maintain patch locations.



$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}};\, \mathbf{x}_p^1\mathbf{E};\, \mathbf{x}_p^2\mathbf{E};\, \cdots;\, \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}$$

https://dkamatblog.home.blog/2021/08/05/vision-transformers-vit/

Final transformer input (z_0)
= flattened patches → linear projection → class embedding → position embedding

01. Vision Transformer

Training

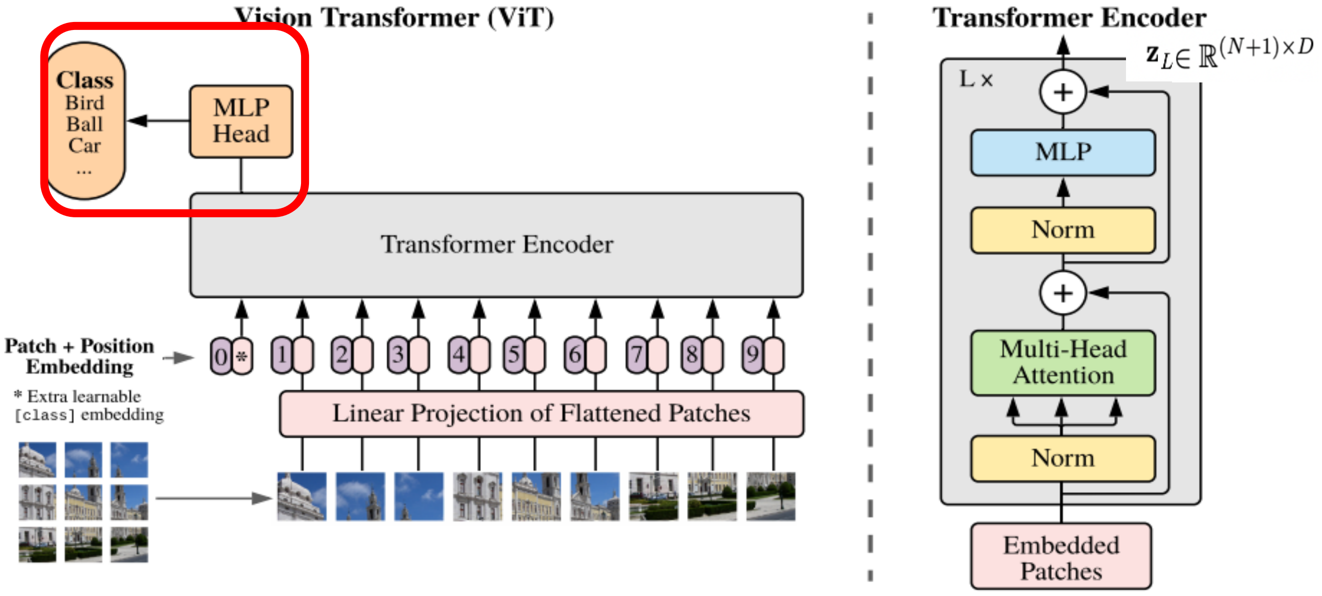
- Input : z_0
- Output : z_L
- Objective function : Cross-Entropy Loss Function
- L x Transformer Encoder : normalization → MSA → residual connection → normalization → MLP → skip connection → z_L
- Use CLS token for classification through MLP head
- Pre-training & Fine-tuning
  → During fine-tuning, the entire model, including the new classification head, is updated without freezing any layers.

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** $\pm$ 0.04	87.76 $\pm$ 0.03	85.30 $\pm$ 0.02	87.54 $\pm$ 0.02	88.4/88.5*
ImageNet ReaL	**90.72** $\pm$ 0.05	90.54 $\pm$ 0.03	88.62 $\pm$ 0.05	90.54	90.55
CIFAR-10	**99.50** $\pm$ 0.06	99.42 $\pm$ 0.03	99.15 $\pm$ 0.03	99.37 $\pm$ 0.06	–
CIFAR-100	**94.55** $\pm$ 0.04	93.90 $\pm$ 0.05	93.25 $\pm$ 0.05	93.51 $\pm$ 0.08	–
Oxford-IIIT Pets	**97.56** $\pm$ 0.03	97.32 $\pm$ 0.11	94.67 $\pm$ 0.15	96.62 $\pm$ 0.23	–
Oxford Flowers-102	99.68 $\pm$ 0.02	**99.74** $\pm$ 0.00	99.61 $\pm$ 0.02	99.63 $\pm$ 0.03	–
VTAB (19 tasks)	**77.63** $\pm$ 0.23	76.28 $\pm$ 0.46	72.72 $\pm$ 0.21	76.29 $\pm$ 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

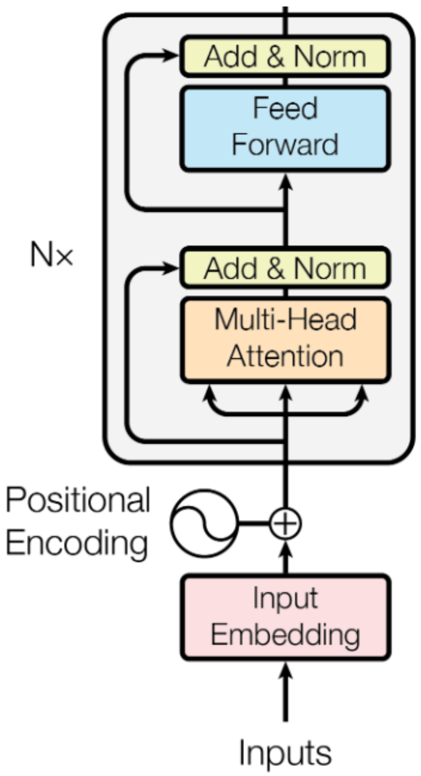
$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}};\, \mathbf{x}_p^1\mathbf{E};\, \mathbf{x}_p^2\mathbf{E};\, \cdots;\, \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}, \qquad \mathbf{E} \in \mathbb{R}^{(P^2 \cdot C)\times D},\, \mathbf{E}_{pos} \in \mathbb{R}^{(N+1)\times D}$$

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

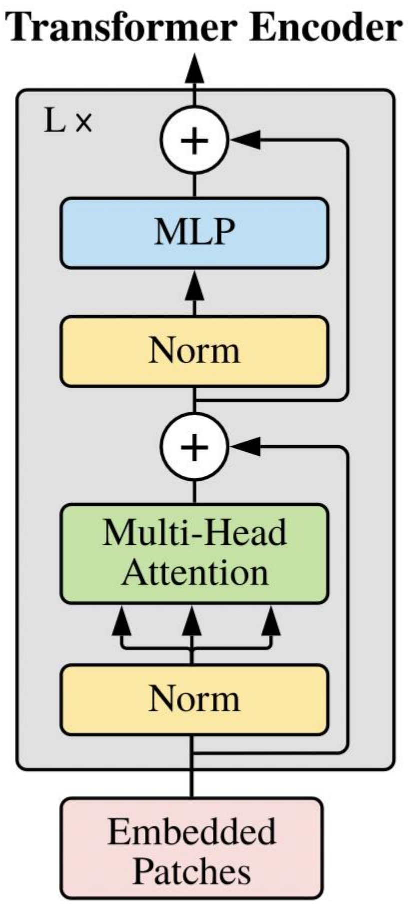
$$\mathbf{z}_\ell = \mathrm{MLP}(\mathrm{LN}(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell, \qquad \ell = 1 \ldots L$$



01. Vision Transformer



Original Transformer Encoder



ViT Transformer Encoder

- While the original transformer applies normalization after the attention block, ViT applies normalization before the attention block.
- While the original transformer uses the ReLU function in the MLP process, ViT uses GeLU.
- In the original transformer, positional embeddings are fixed vectors, but in ViT, they are learnable parameters.

02. Swin Transformer

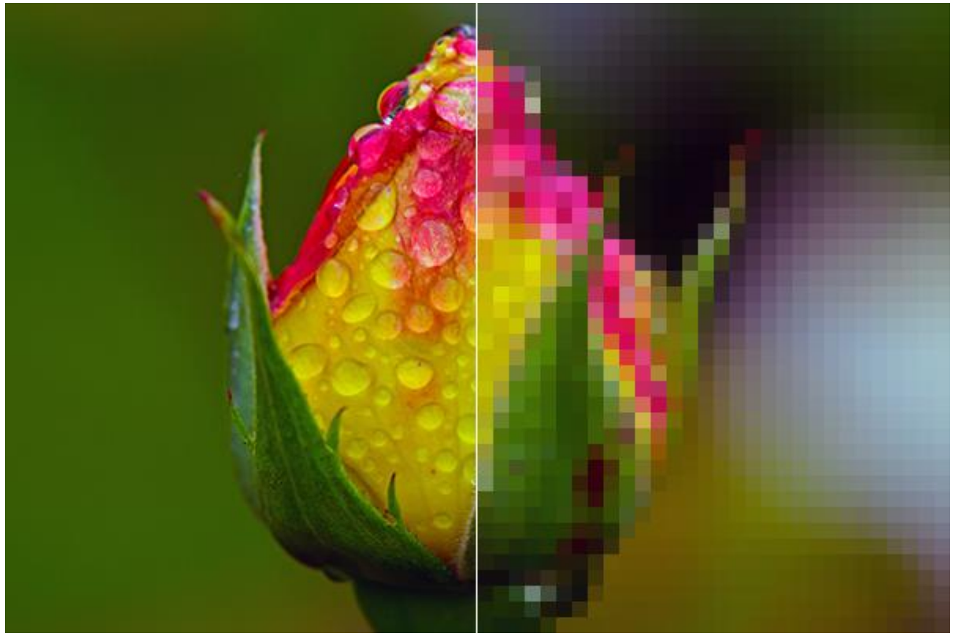
Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

- Limitation
  - The original Vision Transformer (ViT) was designed to solve **classification** problems.
  - Unlike text, ViT lacks specific characteristics suited for processing **images**.
  - The **computational cost** increases quadratically as the number of tokens grows.
- Solution
  - Proposes a model that can be used as a **backbone for various tasks** beyond classification.
  - Introduces a method that **incorporates image-specific characteristics** into the transformer architecture.
  - **Reduces computational complexity** compared to the original ViT model, making it more efficient.

02. Swin Transformer

Image-Specific Characteristics

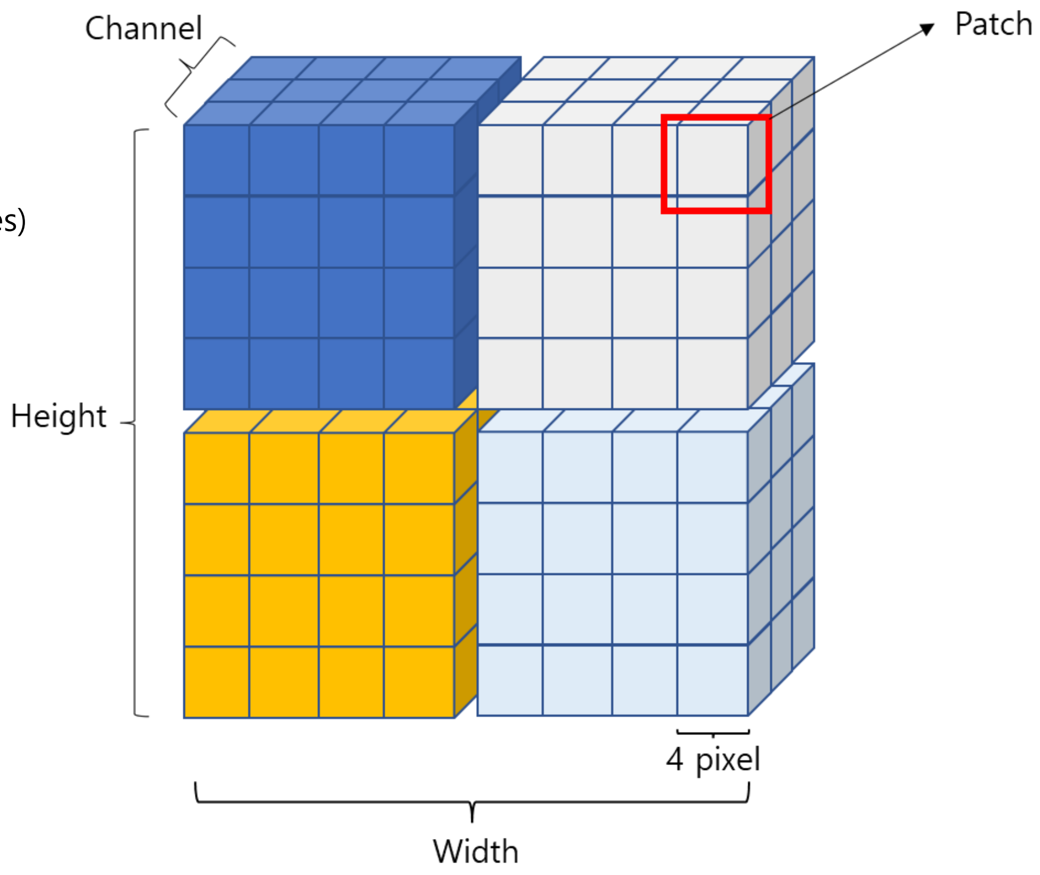
- Images differ from text due to their unique characteristics, such as **resolution** and **the scale of visual entities**.
- Proposed method : Apply **Local Windows** and a **Hierarchical Structure** to the model.



02. Swin Transformer

Notation

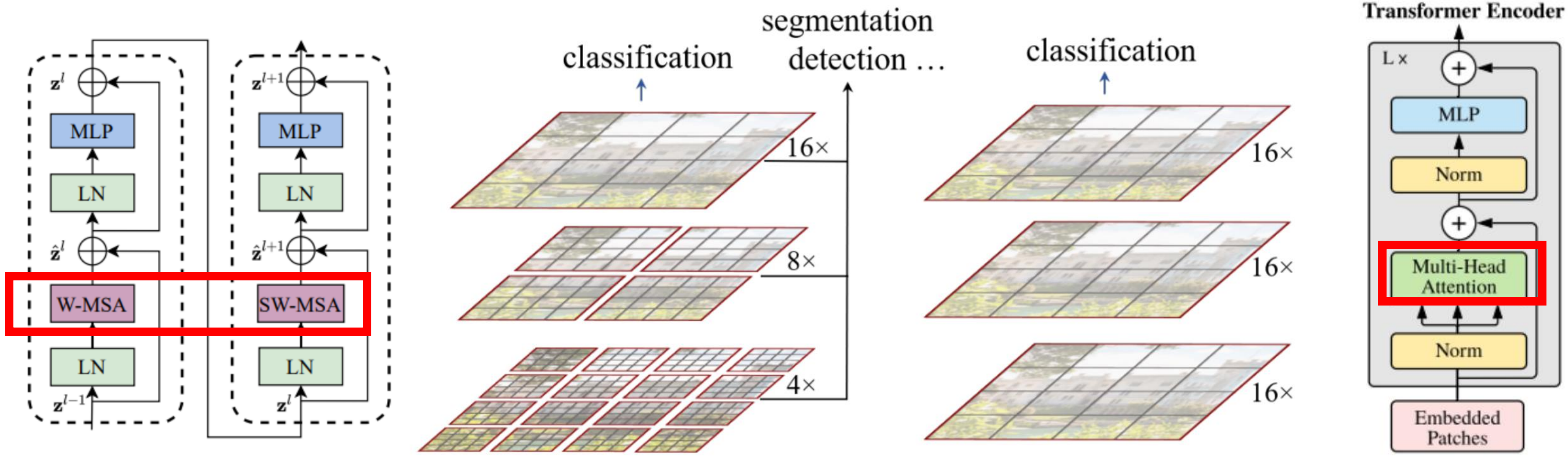
- H : Height of the input image
- W : Width of the input image
- C : Number of channels in the input image (e.g., 3 for RGB images)
- B : Batch size
- M : Size of the local window
- $Patch$: A tokenized portion of the image
- P_h, P_w : Patch dimensions
- $N(=N_h \times N_W)$: Total number of patches (height × width)



02. Swin Transformer

Proposed Method

- Apply **Local Windows** and a **Hierarchical Structure** to the model.



$\Omega(\text{W-MSA}) = 4hwC^2 + 2M^2hwC,$

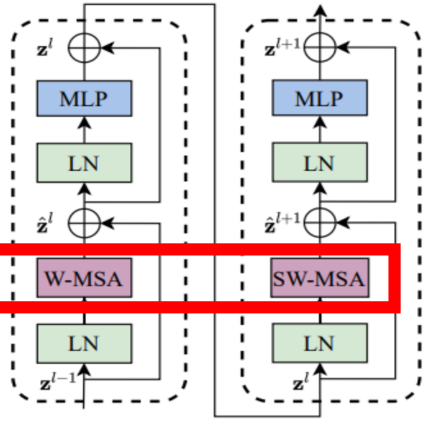
(a) Swin Transformer (ours)

(b) ViT

$\Omega(\text{MSA}) = 4hwC^2 + 2(hw)^2C,$

h : Height of the input image
w : Width of the input image
C : arbitrary dimension size for an image token

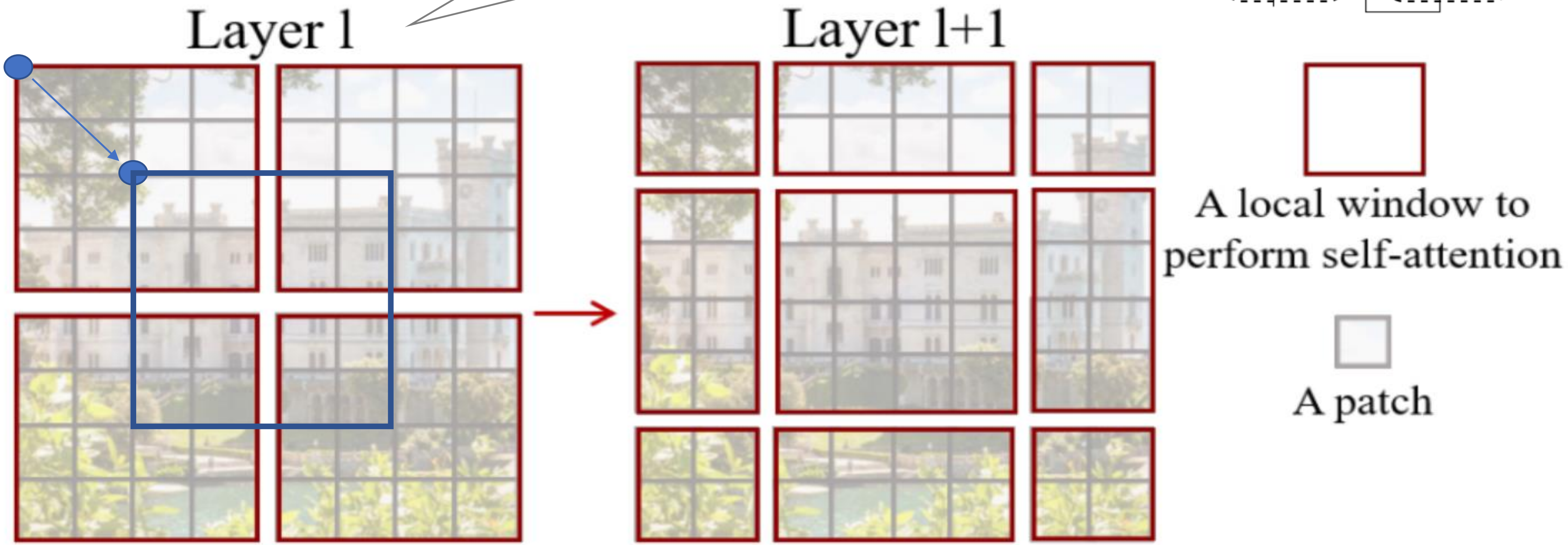
02. Swin Transformer



Proposed Method

Shifted Window Approach

Efficient batch computation



W-MSA (Window Multi-head Self Attention)

SW-MSA (Shifted Window Multi-head Self Attention)

02. Swin Transformer

Proposed Method

$$\Omega(\mathrm{MSA}) = 4hwC^2 + 2(hw)^2C, \quad (1)$$

$$\Omega(\text{W-MSA}) = 4hwC^2 + 2M^2hwC, \quad (2)$$

Cyclic Shift

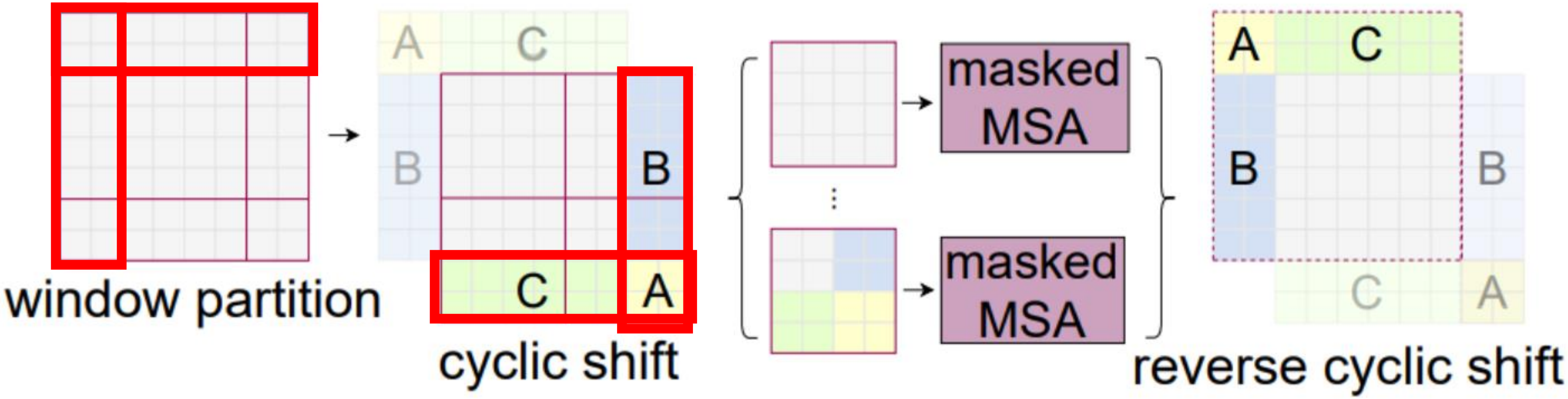


Figure 4. Illustration of an efficient batch computation approach for self-attention in shifted window partitioning.

02. Swin Transformer

Overall Architecture

- Patch Merging
- Swin Transformer Block

W-MSA (Window Multi-head Self Attention)
SW-MSA (Shifted Window Multi-head Self Attention)

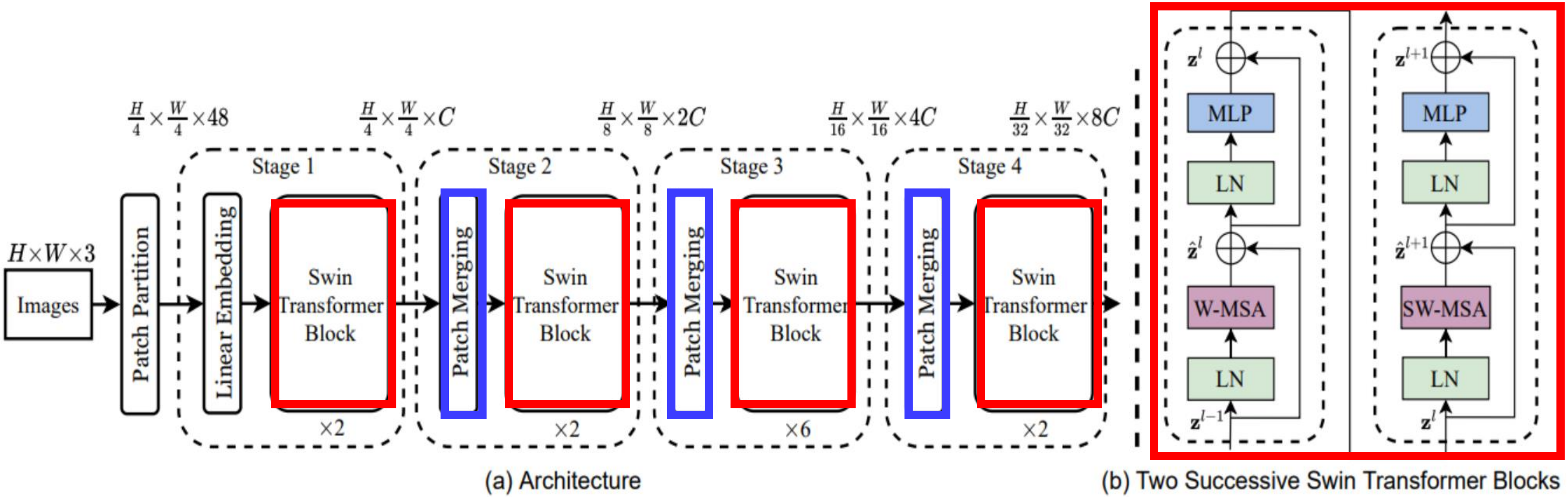


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

02. Swin Transformer

Overall Architecture

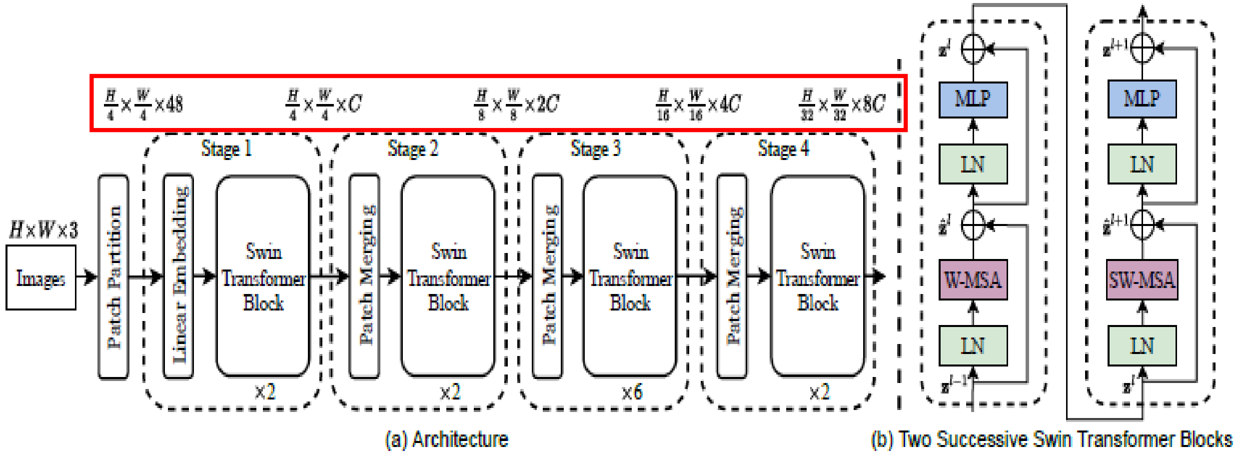
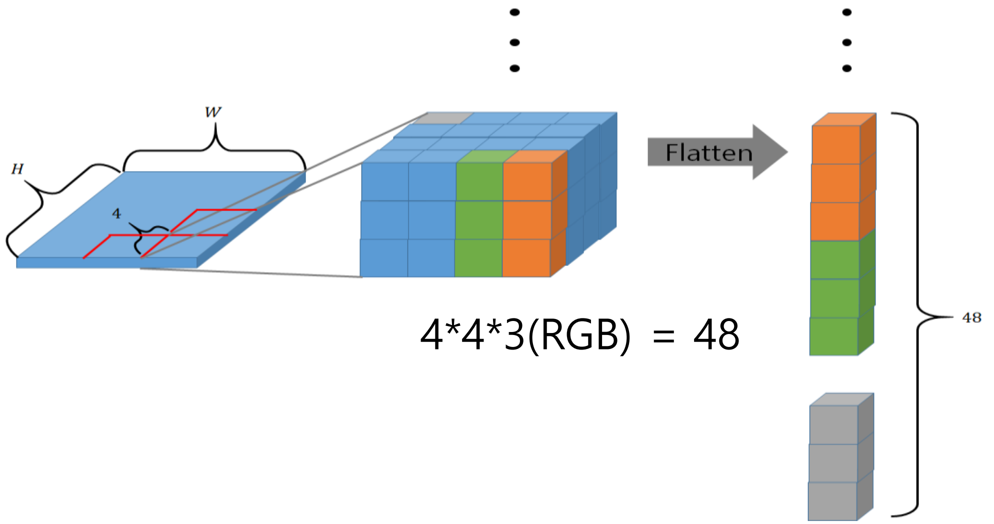


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

Patch Partition



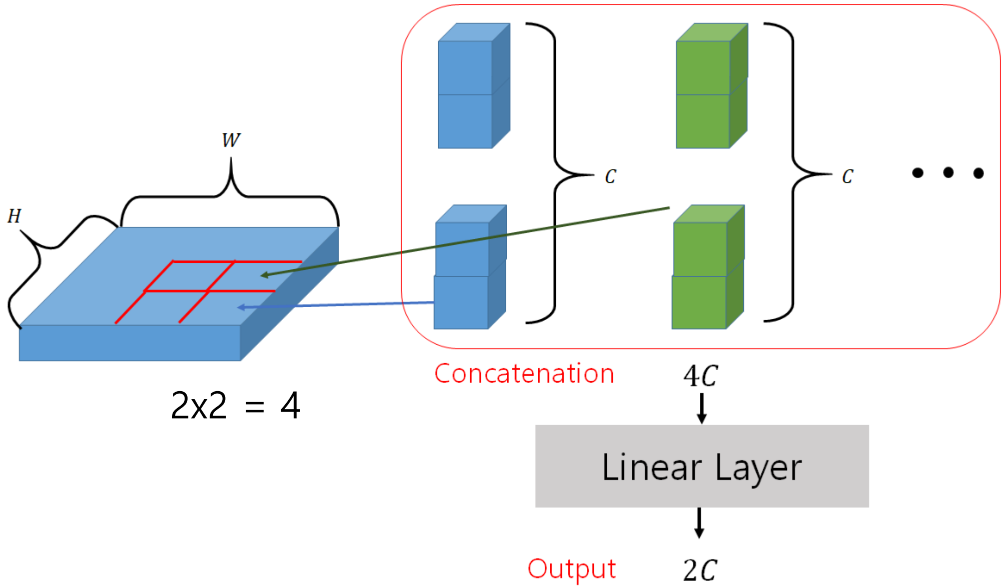
4*4*3(RGB) = 48

Linear Embedding

$$\frac{H}{4}\times\frac{W}{4}\times 48 \rightarrow \frac{H}{4}\times\frac{W}{4}\times C$$

Patch Merging

$\frac{H}{4}\times\frac{W}{4}\times 48$ $\frac{H}{4}\times\frac{W}{4}\times C$ $\frac{H}{8}\times\frac{W}{8}\times 2C$ $\frac{H}{16}\times\frac{W}{16}\times 4C$ $\frac{H}{32}\times\frac{W}{32}\times 8C$



2x2 = 4

02. Swin Transformer

Training

- **Pre-training** : Image Classification
  Objective function : Cross-Entropy Loss
- **Fine-tuning** : Image Classification, Object Detection, Semantic Segmentation

(b) ImageNet-22K pre-trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
R-101x3 [34]	384^2	388M	204.6G	-	84.4
R-152x4 [34]	480^2	937M	840.5G	-	85.4
ViT-B/16 [19]	384^2	86M	55.4G	85.9	84.0
ViT-L/16 [19]	384^2	307M	190.7G	27.3	85.2
Swin-B	224^2	88M	15.4G	278.1	85.2
Swin-B	384^2	88M	47.0G	84.7	86.4
Swin-L	384^2	197M	103.9G	42.1	87.3

(a) Regular ImageNet-1K trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
RegNetY-4G [44]	224^2	21M	4.0G	1156.7	80.0
RegNetY-8G [44]	224^2	39M	8.0G	591.6	81.7
RegNetY-16G [44]	224^2	84M	16.0G	334.7	82.9
ViT-B/16 [19]	384^2	86M	55.4G	85.9	77.9
ViT-L/16 [19]	384^2	307M	190.7G	27.3	76.5
DeiT-S [57]	224^2	22M	4.6G	940.4	79.8
DeiT-B [57]	224^2	86M	17.5G	292.3	81.8
DeiT-B [57]	384^2	86M	55.4G	85.9	83.1
Swin-T	224^2	29M	4.5G	755.2	81.3
Swin-S	224^2	50M	8.7G	436.9	83.0
Swin-B	224^2	88M	15.4G	278.1	83.5
Swin-B	384^2	88M	47.0G	84.7	84.5

4.2. Object Detection on COCO

Settings Object detection and instance segmentation experiments are conducted on COCO 2017, which contains 118K training, 5K validation and 20K test-dev images. An ablation study is performed using the validation set, and a system-level comparison is reported on test-dev. For the ablation study, we consider four typical object detection frameworks: Cascade Mask R-CNN [29, 6], ATSS [79], RepPoints v2 [12], and Sparse RCNN [56] in mmdetection [10]. For these four frameworks, we utilize the same settings: multi-scale training [8, 56] (resizing the input such that the shorter side is between 480 and 800 while the longer side is at most 1333), AdamW [44] optimizer (initial learning rate of 0.0001, weight decay of 0.05, and batch size of 16), and 3x schedule (36 epochs). For system-level comparison, we adopt an improved HTC [9] (denoted as HTC++) with instaboost [22], stronger multi-scale training [7], 6x schedule (72 epochs), soft-NMS [5], and ImageNet-22K pre-trained model as initialization.

ADE20K Method	Backbone	val mIoU	test score	#param.	FLOPs	FPS
DLab.v3+ [11]	ResNet-101	44.1	-	63M	1021G	16.0
DNL [65]	ResNet-101	46.0	56.2	69M	1249G	14.8
OCRNet [67]	ResNet-101	45.3	56.0	56M	923G	19.3
UperNet [63]	ResNet-101	44.9	-	86M	1029G	20.1
OCRNet [67]	HRNet-w48	45.7	-	71M	664G	12.5
DLab.v3+ [11]	ResNeSt-101	46.9	55.1	66M	1051G	11.9
DLab.v3+ [11]	ResNeSt-200	48.4	-	88M	1381G	8.1
SETR [73]	T-Large‡	50.3	61.7	308M	-	-
UperNet	DeiT-S†	44.0	-	52M	1099G	16.2
UperNet	Swin-T	46.1	-	60M	945G	18.5
UperNet	Swin-S	49.3	-	81M	1038G	15.2
UperNet	Swin-B‡	51.6	-	121M	1841G	8.7
UperNet	Swin-L‡	**53.5**	**62.8**	234M	3230G	6.2

Table 3. Results of semantic segmentation on the ADE20K val and test set. † indicates additional deconvolution layers are used to produce hierarchical feature maps. ‡ indicates that the model is pre-trained on ImageNet-22K.

02. Swin Transformer

Conclusion

- Using a similar architecture for both NLP and computer vision could significantly accelerate the research process.

- Advantages over CNNs:
    - Higher accuracy on large datasets.
    - Higher modeling capacity.
    - Lower inductive biases and global receptive fields.

- Modern ViTs (e.g., Swin) are becoming more CNN-like by:
    - Reducing receptive fields.
    - Using hierarchical, pyramidal feature maps.

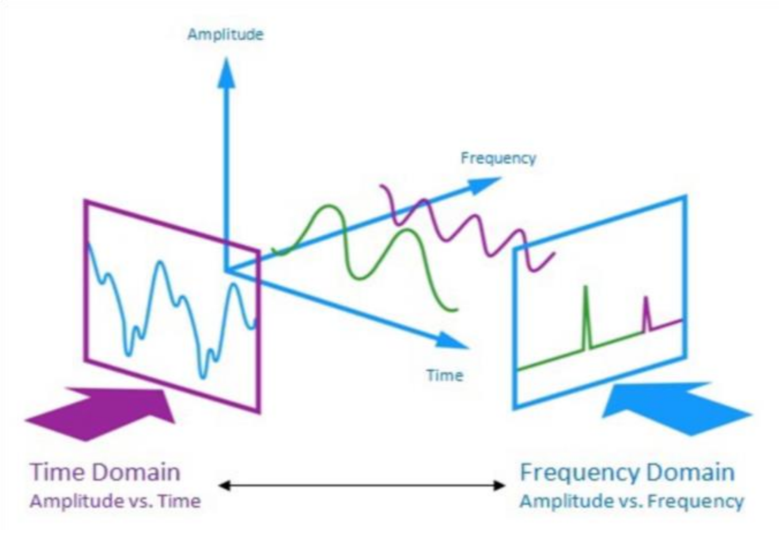
- However, CNNs still perform on-par or better than state-of-the-art ViTs (e.g., ImageNet) when:
    - Comparing model complexity/size versus accuracy.
    - Trained without knowledge distillation or additional data, particularly at lower accuracy targets.

Speech Transformer

Outline

01. Introduction

Speech

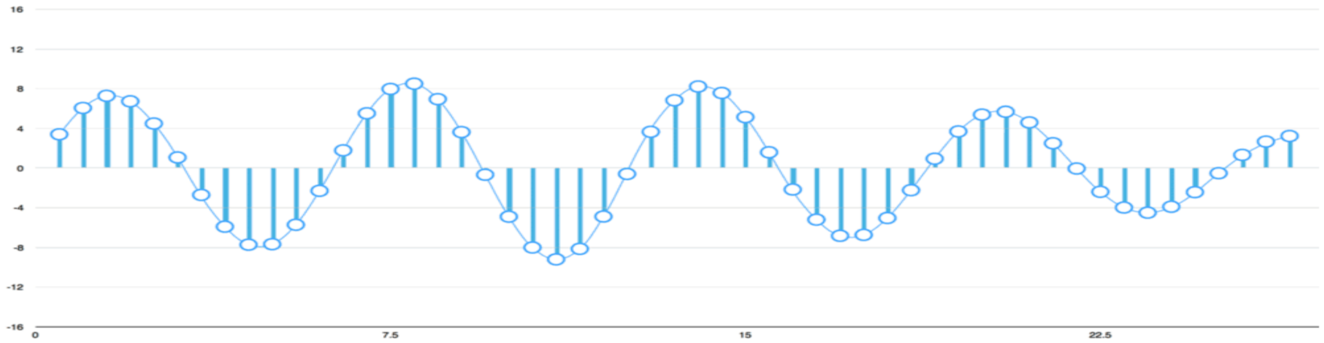



```
import scipy.io.wavfile
sample_rate, signal = scipy.io.wavfile.read('example.wav')
```

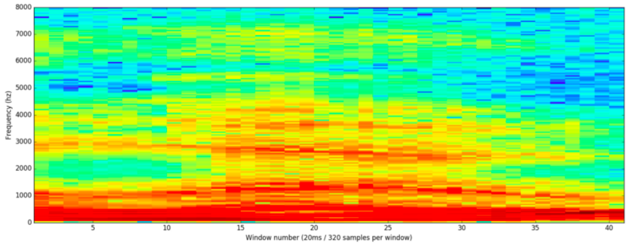
```
>>> sample_rate
16000
>>> signal
array([36, 37, 60, ...,  7,  9,  8], dtype=int16)
>>> len(signal)
183280
>>> len(signal) / sample_rate
11.455
```

https://ratsgo.github.io/speechbook/docs/fe/ft

- Speech signals are composed of various frequencies.
- Complex waveforms can be split into different frequency components.
- Fourier transform helps to analyze these signals in the frequency domain.
- To process speech in computers, analog signals are converted into digital data.
- Sampling : determines how often to capture data points.
  → Commonly 16,000 Hz for speech, meaning 16,000 samples per second.
- Often, transformations are used to extract useful features.ㅍ



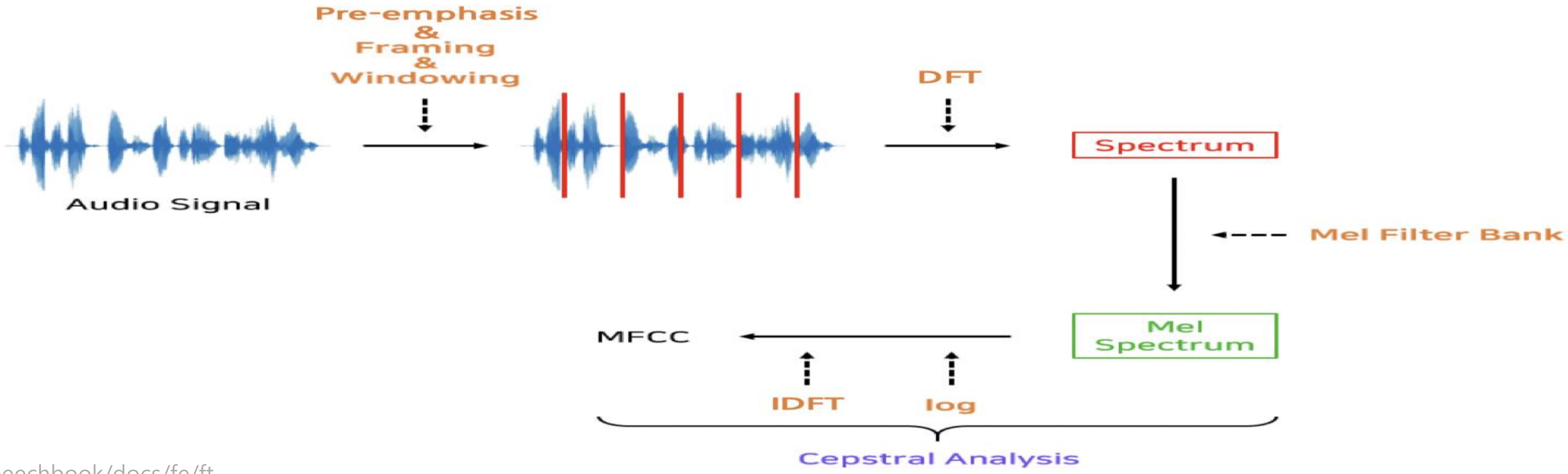
01. Introduction



"hello" 음성에 대한 전체 음향 스펙트럼(spectrogram)

Speech

- Fourier Transform applied to 25ms windows of audio segments.
- Spectrogram : visualizes frequency components over time, resulting in 2D data.
- MFCC (Mel-frequency Cepstral Coefficients) : focuses on low-frequency details based on human hearing sensitivity.
- Features like spectrogram of MFCC are converted into 2D tensors or 1D sequences for input.
- Traditional speech recognition : handcrafted feature extraction for better speech recognition.

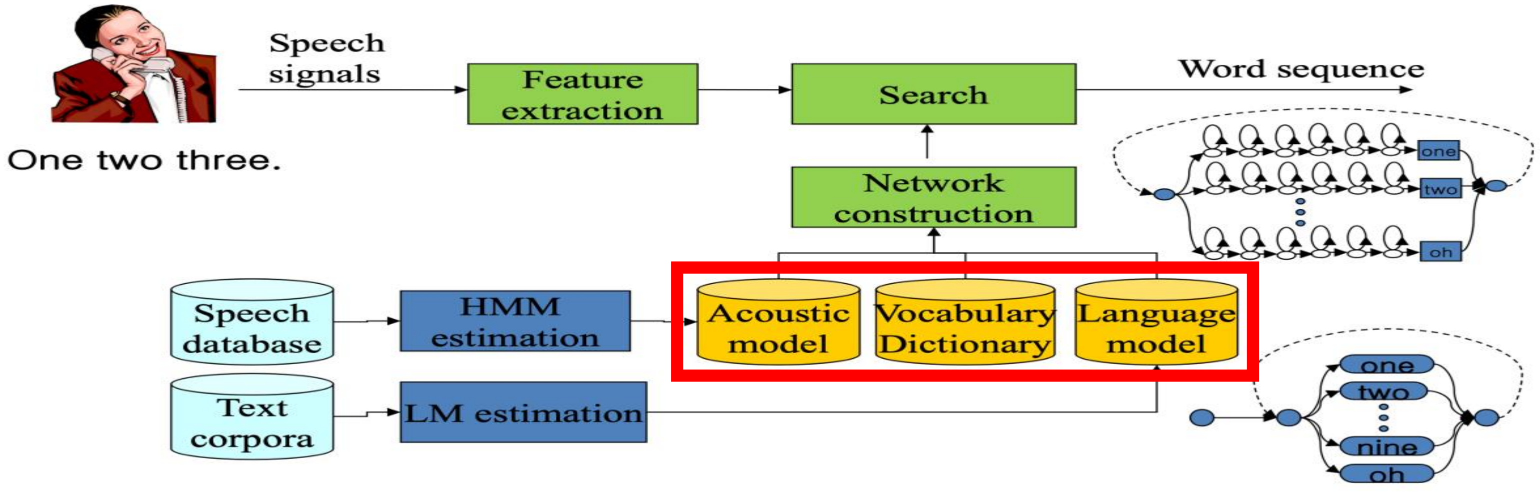


https://ratsgo.github.io/speechbook/docs/fe/ft

01. Introduction

Automatic Speech Recognition (ASR)

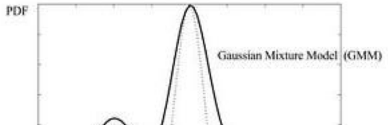
- Acoustic model : recognizes sounds or phonemes like /a/, /e/, /i/ in speech
- Language model : captures statistical connections between words and ensures natural word sequences (e.g., I go to school vs. I sell school)
- Vocabulary dictionary : Maps phoneme sequences to words, using a pronunciation lexicon
- Combines these three components to convert speech signals into word sequences through decoding and searching.

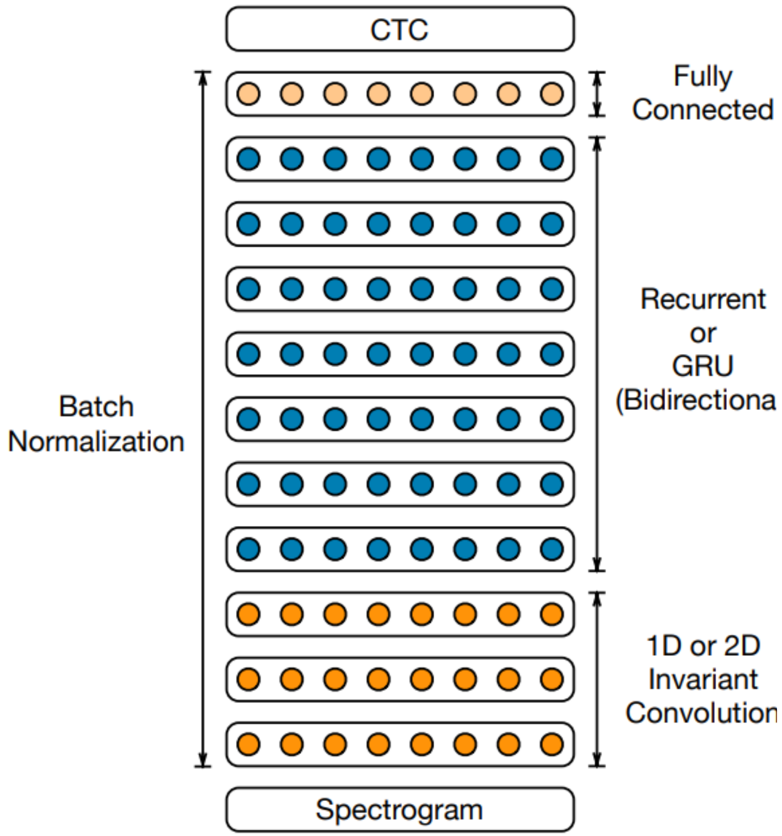


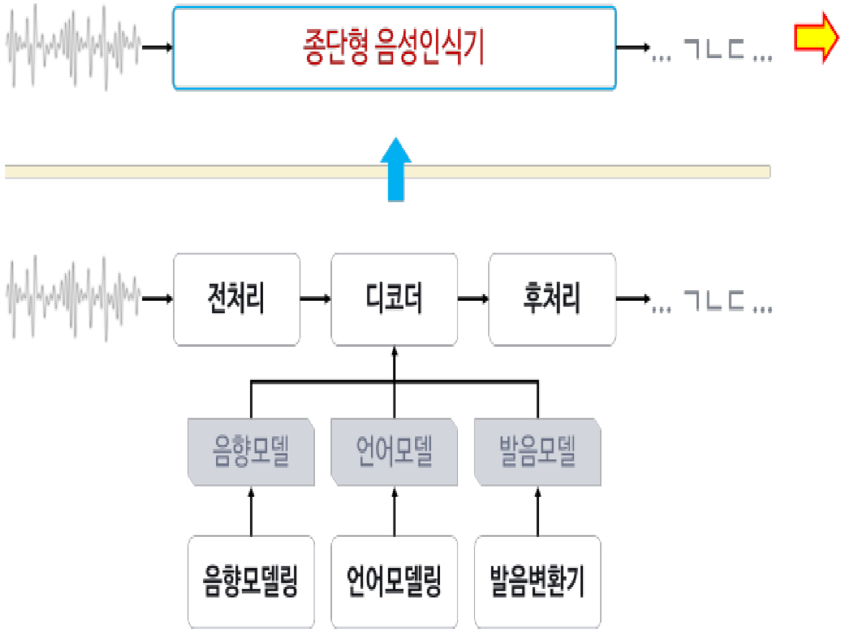
02. Pre-Transformer Speech Models

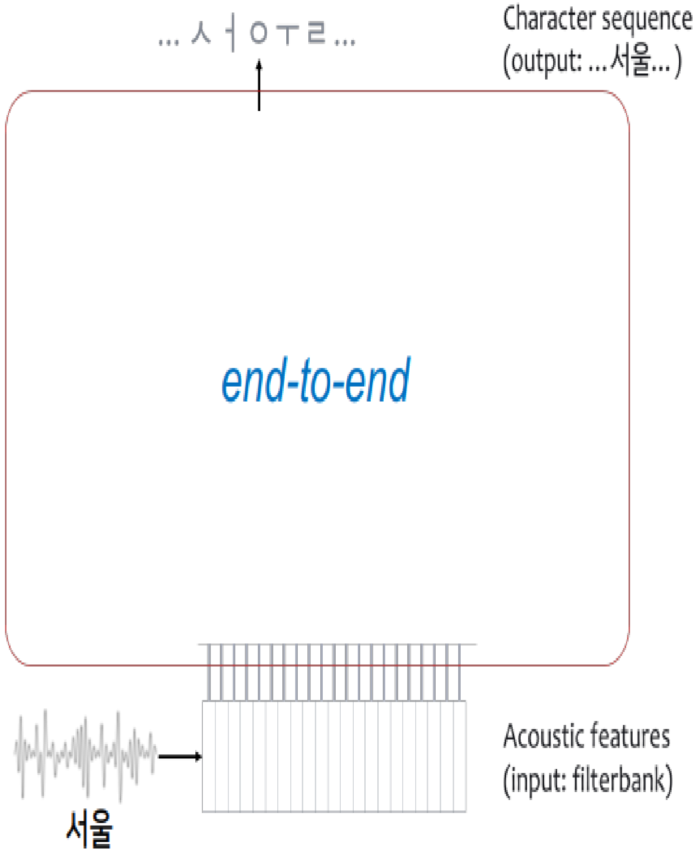
Traditional Approaches

- GMM-HMM → DNN-HMM → End-to-End → Pre-trained model









https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/lectures/cs224n-2017-lecture12.pdf
Deep Speech 2: End-to-End Speech Recognition in English and Mandarin

02. Pre-Transformer Speech Models

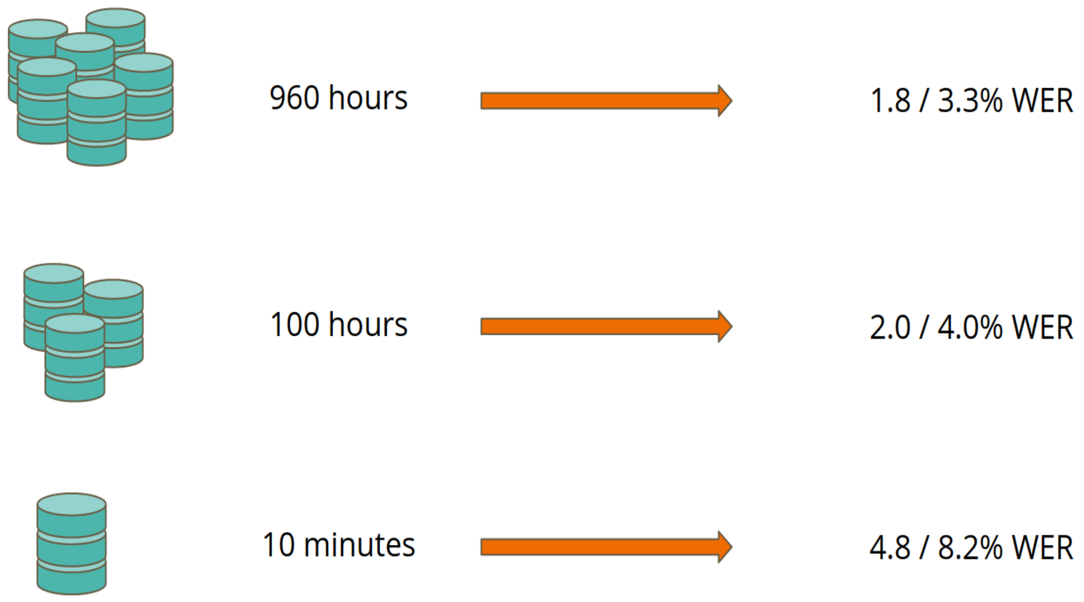
Limitations

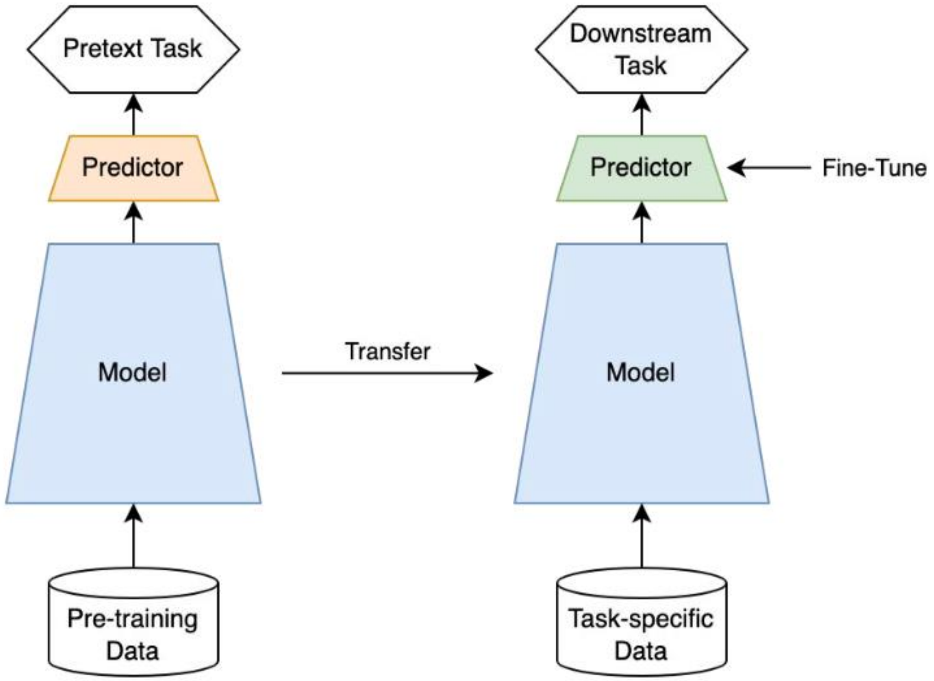
- RNN/LSTM models process sequences step by step, which limits parallelization and leads to slow training, especially for long speech sequences.
- Each time step depends on the previous one, making it difficult to compute multiple steps in parallel.
- Speech sequences are often long, and RNN/LSTM models struggle to handle these efficiently, leading to increased computational cost.
- **Transformer** solves these issues by using **Self-Attention**, allowing the model to process all positions in the sequence simultaneously, enabling faster training and better parallelization.

03. Speech Transformer – Wav2Vec2.0

Wav2Vec2.0 (2020)

- Wav2Vec 2.0 is a self-supervised learning-based acoustic model, utilizing a Transformer architecture.
- **Pre-training** : 53k hours of unlabeled speech data
  Fine-tuning : 10 minutes of labeled data for downstream tasks : 4.8% PER, 8.2% WER

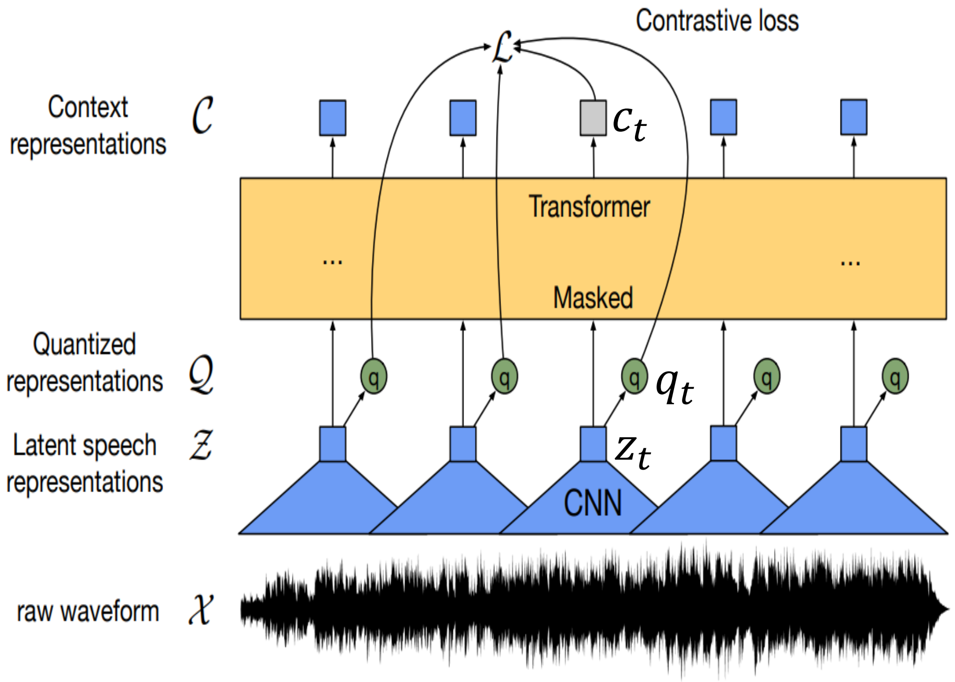




03. Speech Transformer – Wav2Vec2.0

Notation

- x_i : Raw audio signal at index I where $x_i \in X$.
- z_t : Latent speech representation at time step t.
- q_t : Quantized representation at time step t.
- c_t : Contextualized representation at time step t.
- G : Codebook size (the number of groups of codewords).
- V : Number of codewords per group.
- e : Codeword vector representation from the codebook.

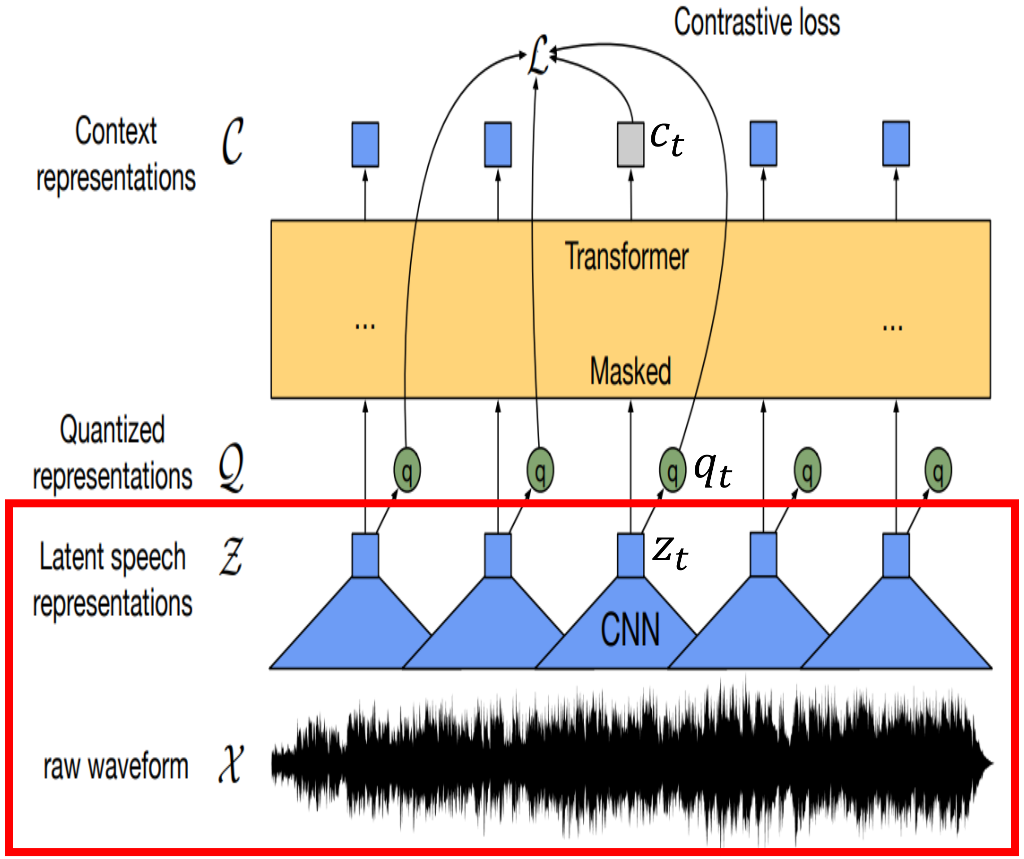


03. Speech Transformer – Wav2Vec2.0

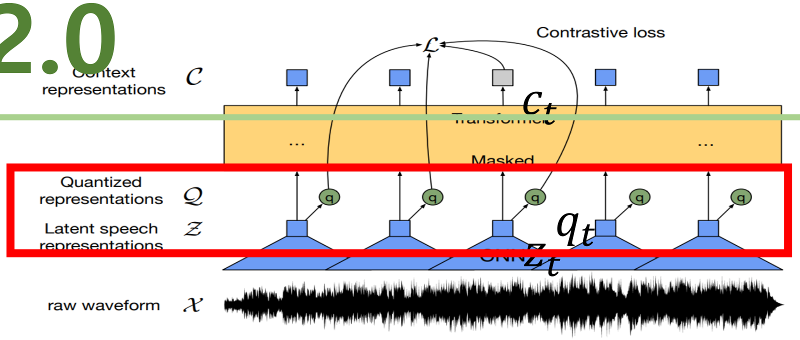
Model Architecture

<1> CNN Feature Encoder < $f: X \rightarrow Z$ >

- **Input** : Raw audio waveform
- **Process** : Passed through a multi-layer CNN encoder
- **Output** : Converted into 25ms representation vectors
- At each time step T , the model outputs latent speech representations (z_1, ..., z_T).
- These representations (z_1, ..., z_T) are used as input for the quantizer and transformer modules.



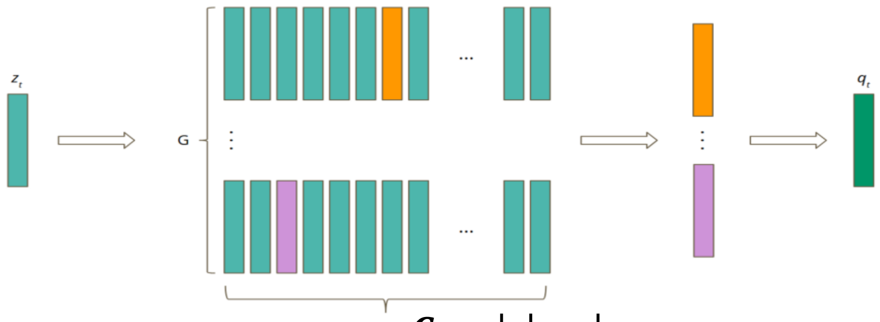
03. Speech Transformer – Wav2Vec2.0



Model Architecture

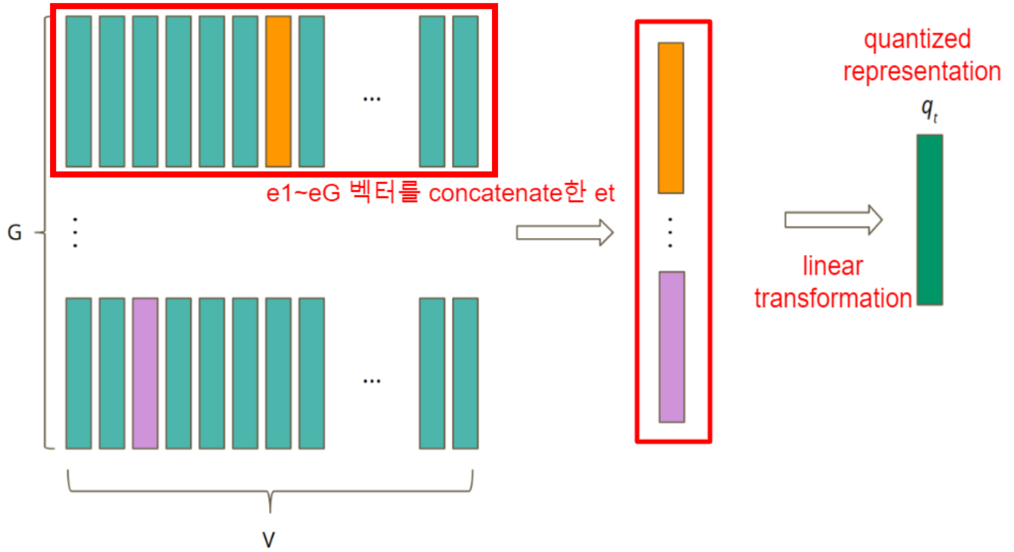
<2> Quantization Module < $Z \rightarrow \mathbf{Q}$ >

- **Input** : Latent speech representation z
- **Process** : z_t (from CNN encoder) is discretized using the quantizer $Z \rightarrow \mathbf{Q}$ to output q_t
- **Output** : q_t
- Finds the closest matching codeword vector to the current time-step vector z_t and converts it into the corresponding quantized representation q_t
- Codeword vector represents universal human phonemes, shared across languages.
- Codebook matrix consists of multiple learnable parameters.

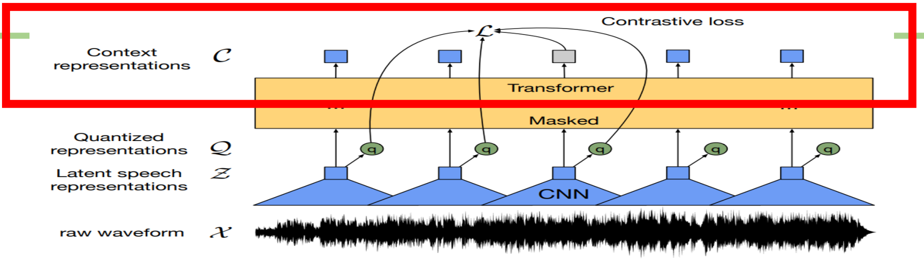


G codebooks
each containing ***V*** code words

codebook ***e***



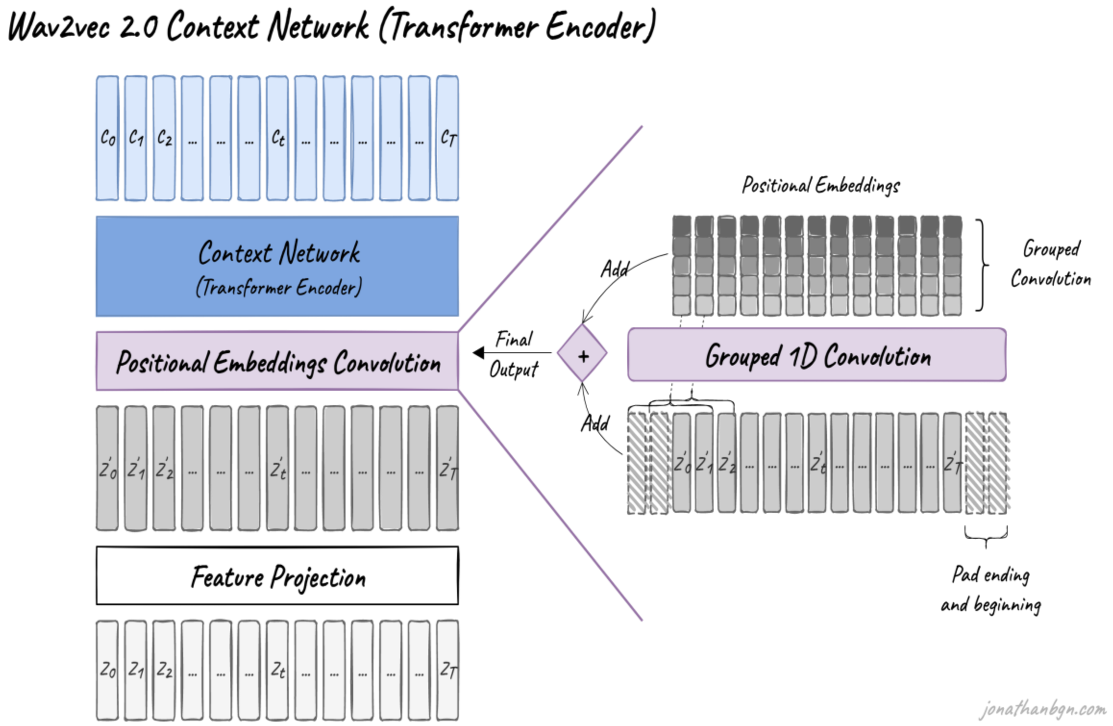
03. Speech Transformer – Wav2Vec2.0



Model Architecture

<3> Transformer Module < $g: Z \rightarrow C$ >

- **Input** : Latent speech representations $z_1, \ldots, z_T$
- **Process** : Transformer (encoder) block captures contextual information across the entire sequence, outputting $c_1, \ldots, c_T$
- **Output** : Context representations $c_1, \ldots, c_T$
- 12 Transformer blocks for the *BASE* version of the model, or 24 blocks for the *LARGE* version.
- No absolute positional embeddings used.
  The wav2vec model instead uses a new grouped convolution layer to learn relative positional embeddings by itself.



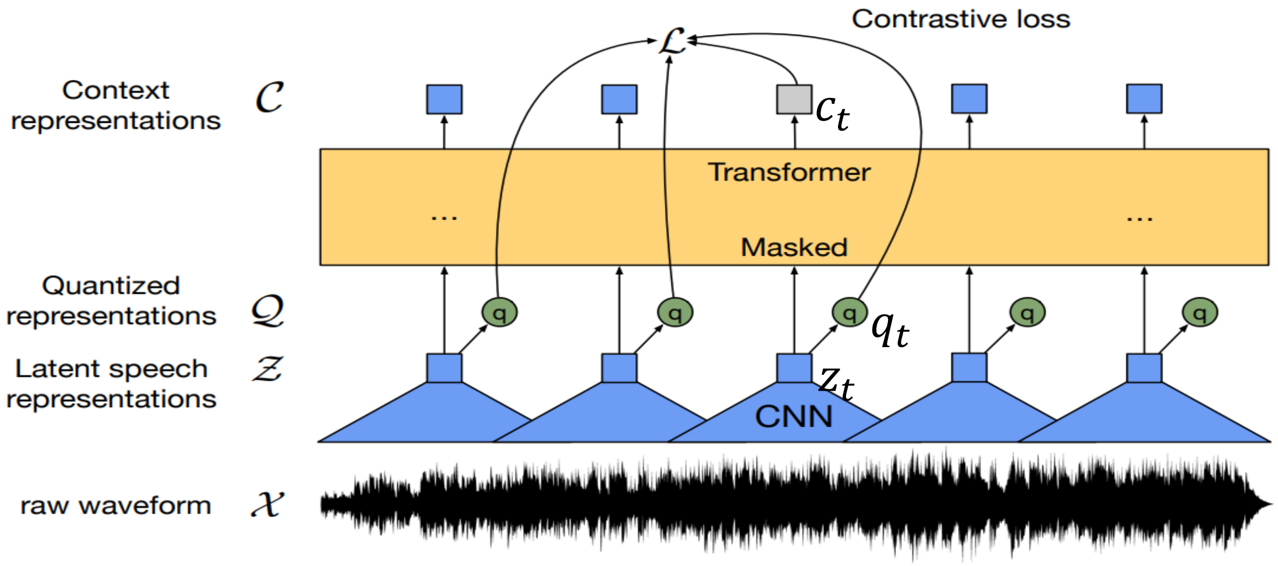
03. Speech Transformer – Wav2Vec2.0

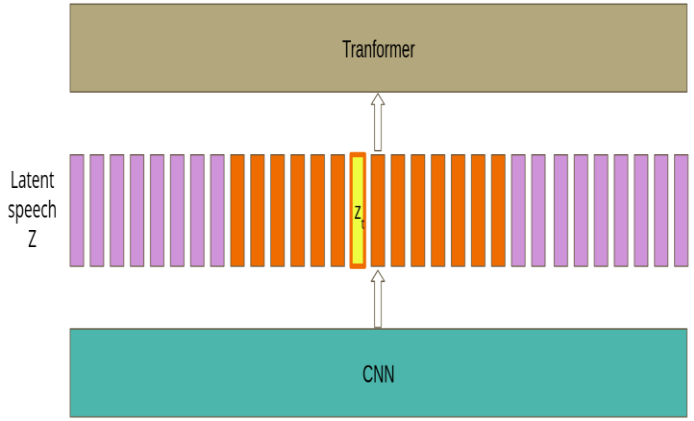
Training

$$L_m = -\log \frac{\exp(sim(\mathbf{c}_t, \mathbf{q}_t)/\kappa)}{\sum_{\tilde{\mathbf{q}} \sim \mathbf{Q}_t} \exp(sim(\mathbf{c}_t, \tilde{\mathbf{q}})/\kappa)}$$

- **Objective function** : Contrastive Loss
  - Half of latent representations are masked before entering Transformer.
  - Learns to make **context representations** similar to the **quantized representation** of the masked position, while pushing away representations of other positions

  → maximizes similarity between q_t and c_t at the same position & minimizes similarity with other positions.





Conclusion

- The introduction of transformer models in speech processing helps overcome the bottlenecks and performance limitations of traditional models.

- Their parallel processing capabilities and attention mechanisms enable efficient handling of the complexity in speech data.

- The adoption of self-supervised learning techniques has significantly improved the performance of transformer-based speech models.

End