

Vision Transformer

relative to pyramid

Yinrui Li

December 15, 2023

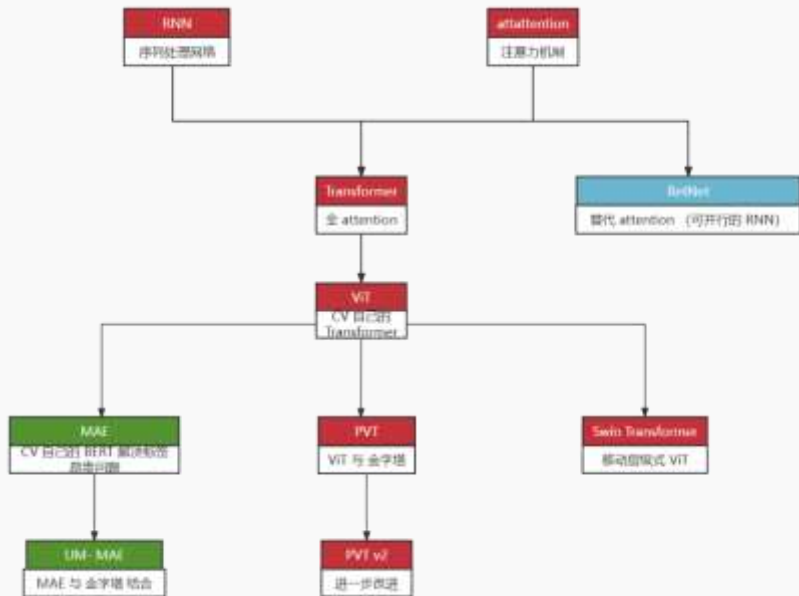


Fig. 1: mind map

1. Developments in NLP

1. motivation
2. RNN
3. Transformer

2. Developments in CV

1. motivation
2. Vision Transformer
3. PVT
4. swin transformer

1.Developments in NLP

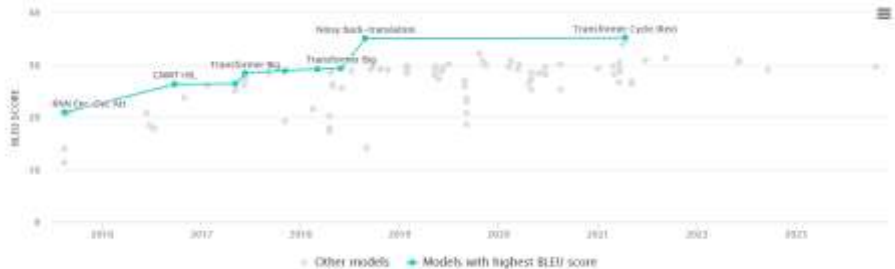
1.1

Machine Translation on WMT2014 English-German

Leaderboard

Dataset

View BLEU score by Date for All models



Rank	Model	score	achievable	# Params	Training Data	Paper	Live Demo	Result	Year	Tag
1	Transformer Cycle (Rev)	35.14	33.54		×	Lessons on Parameter Sharing across Layers in Transformers			2021	Transformer
2	Noisy back-translation	35.0	33.8		✓	Understanding Back-Translation at Scale			2018	
3	Transformer+Rep (Unl)	33.89	32.35		×	Rethinking Perturbations in Encoder-Decoders for Fast Training			2021	Transformer
4	TS-11B	32.1		11110M	×	Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer			2019	Transformer
5	BiBERT	31.26			×	BERT, mBERT, or BiBERT? A Study on Contextualized Embeddings for Neural Machine Translation			2021	
6	Transformer + R-Drop	30.91			×	R-Drop: Regularized Dropout for Neural Networks			2021	Transformer
7	Bi-SimCut	30.78			×	Bi-SimCut: A Simple Strategy for Boosting Neural Machine Translation			2022	
8	BERT-fused NMT	30.75			×	Incorporating BERT into Neural Machine Translation			2020	Transformer
9	Data Diversification - Transformer	30.7			×	Data Diversification: A Simple Strategy For Neural Machine Translation			2019	Transformer

1.2 RNN

Early NLP Models: Recurrent Neural Network (RNN)

- In NLP, RNN was the earliest used model.
- RNNs integrate current inputs with past states sequentially to determine outputs.

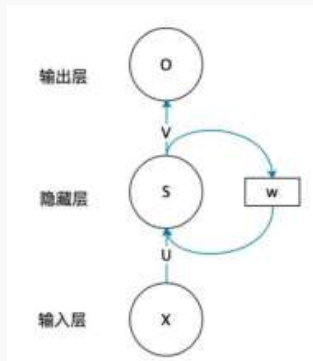


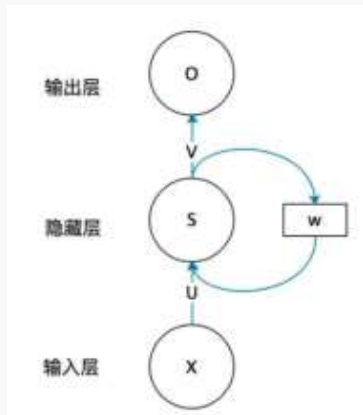
Fig. 2: The Basic RNN

Early NLP Models: Recurrent Neural Network (RNN)

(1)

$$O_t = g(V \cdot S_t)$$

$$S_t = f(U \cdot X_t + W \cdot S_{t-1})$$

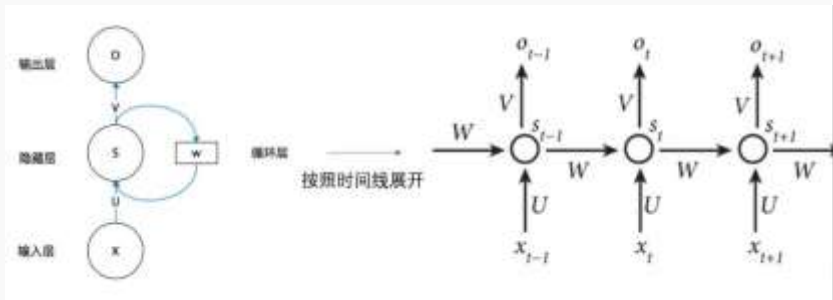


(a) Recurrent Neural Networks (RNN)

The value of S_t is contingent not only on X_t but also on S_{t-1} .

Early NLP Models: Recurrent Neural Network (RNN)

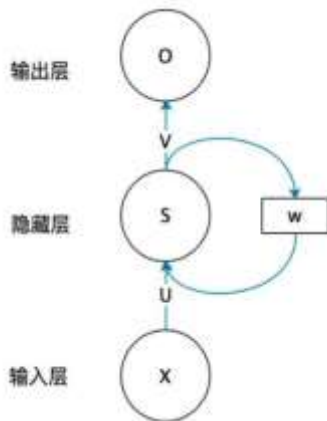
$$(1) \quad \begin{aligned} O_t &= g(V \cdot S_t) \\ S_t &= f(U \cdot X_t + W \cdot S_{t-1}) \end{aligned}$$



(a) Recurrent Neural Networks (RNN)

The value of S_t is contingent not only on X_t but also on S_{t-1} .

The Challenges of RNN



$$\frac{\partial L_t}{\partial U} = \sum_{k=0}^t \left(\frac{\partial L_t}{\partial o_t} \frac{\partial o_t}{\partial s_t} \left(\prod_{j=k+1}^t \frac{\partial s_j}{\partial s_{j-1}} \right) \frac{\partial s_k}{\partial U} \right)$$

$$\prod_{j=k+1}^t \frac{\partial s_j}{\partial s_{j-1}} = \prod_{j=k+1}^t \tanh(W)$$

issues:

- Gradient Vanishing and Gradient Explosion: Limitations in Scaling to Large Models

1.3 Transformer

Attention is All You Need

- Google's "Attention is All You Need" paper introduced Transformer.
- The goal was to replace RNN with attention for parallelism and efficiency.

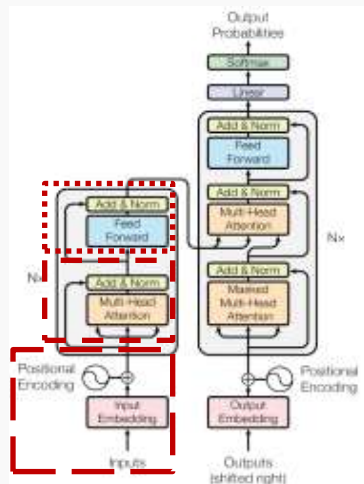


Fig. 6: Basic Transformer

1.3.2 attention

What is attention: Case I



Fig. 7: attention case I

- Attention means focusing on what's most important and "ignoring" the less significant.
- We invest more attention in things that are more important.

What is attention: Case I

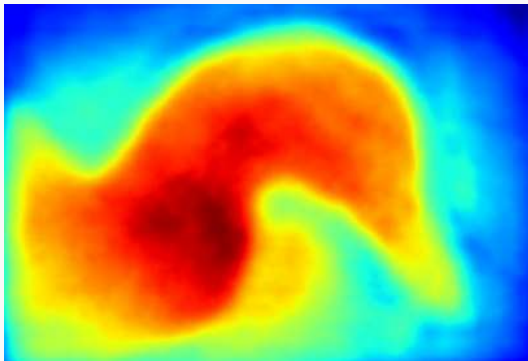


Fig. 8: attention case I

- Attention effectively preserves key information over distances.

What is attention: Case II

Hello, how are you?



Bonjour, comment ça va?

Fig. 9: attention case II

- Neural networks focus on inputs that are more likely to be accurate.

What is attention: Case II

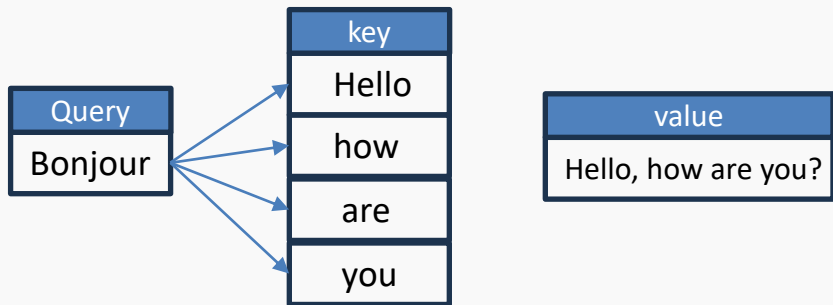


Fig. 10: attention case II

- Neural networks focus on inputs that are more likely to be accurate.

What is attention: Case II

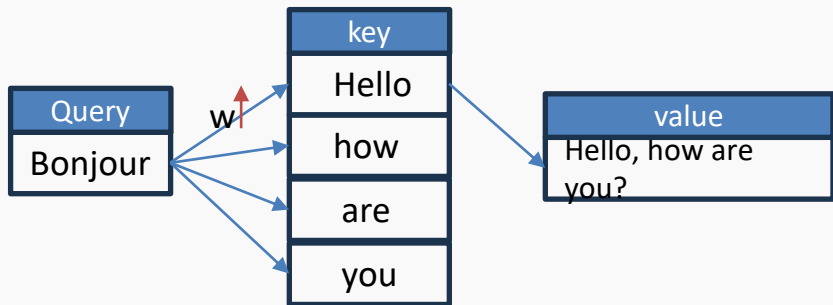


Fig. 10: attention case II

- Concentrating attention on crucial information saves resources and yields effective insights rapidly.

How to compute attention

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_K}} \right) V$$



Q = query K = key V = value

- Calculate similarity between query and key for a relevance score.
- Normalize the score with softmax to create an attention distribution.
- Use the attention distribution to compute the improved value.

How to compute attention

$$\text{Attention}(Q, K, V) = \text{softmax} \frac{QK^T}{\sqrt{d_K}} V \quad (3)$$

$$Q = X \times W_Q$$

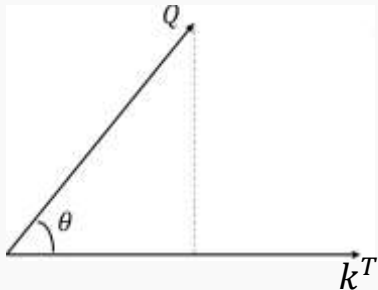
$$K = X \times W_K$$

$$V = X \times W_V$$

Self-attention is achieved by linearly transforming the input X into Q , K , and V .

How to compute attention

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



Why use QK^T

- The dot product is the product of the vector lengths and the cosine of their angle.
- The dot product reflects the similarity

How to compute attention

$$\text{Attention}(Q, K, V) = \text{softmax} \frac{QK^T}{\sqrt{d_K}} V \quad (6)$$

Why divide by $\sqrt{d_K}$?

- To prevent the value of QK^T from becoming too large
- Preventing overflow during softmax calculation.

multi-head attention

multi-head attention mechanism

- A single method of attention calculation is too **restrictive**.

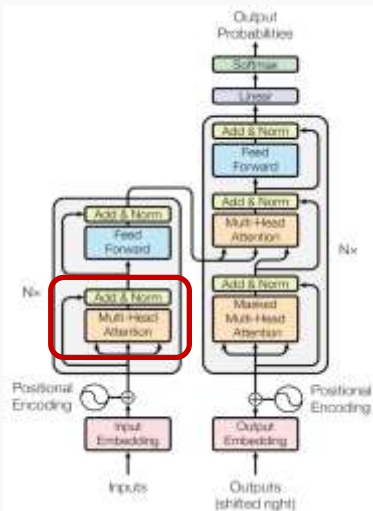


Fig. 12: Basic Transformer

multi-head attention mechanism

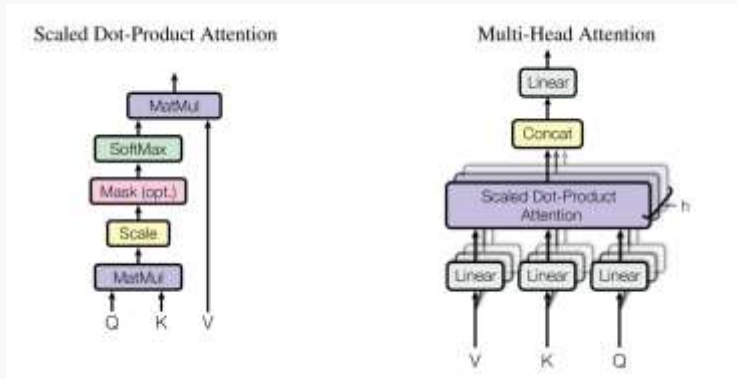
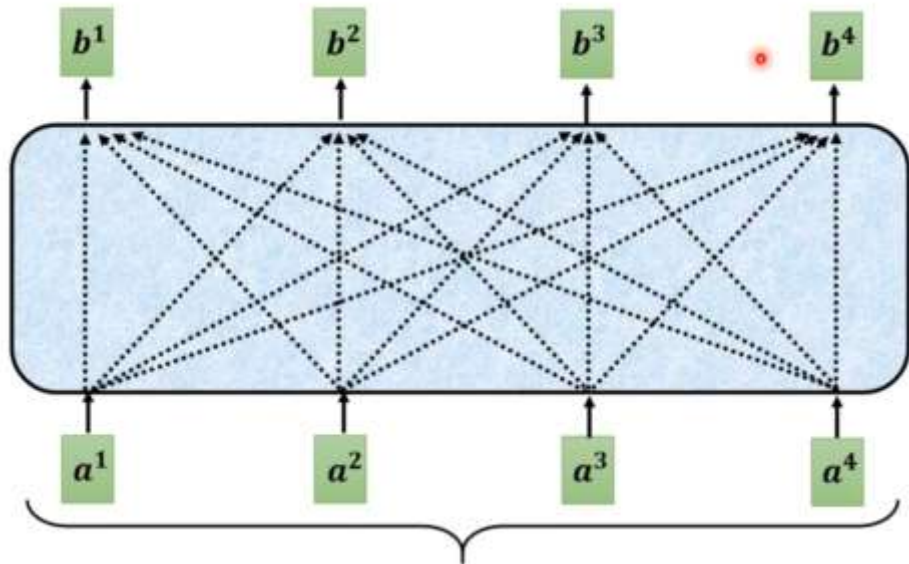


Fig. 13: Multi-Head Attention consists of several attention layers running in parallel.

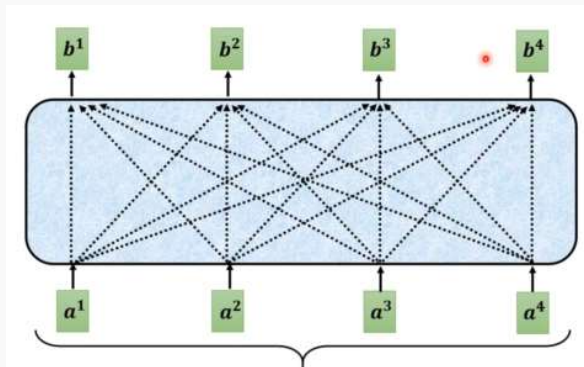
Multi-head attention uses subspaces for diverse attention learning.

attention



1.3.1 Positional Encoding

What is Positional Encoding



$$a^1 a^2 a^3 a^4 = a^2 a^4 a^1 a^3$$

High degree of parallelism

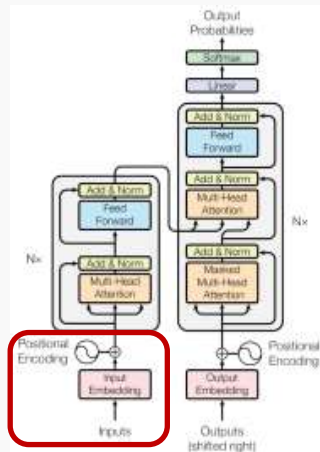


Fig. 15: Basic Transformer

What is Positional Encoding

我爱她 } different
她爱我 }

- Attention's parallelism neglect positional cues.

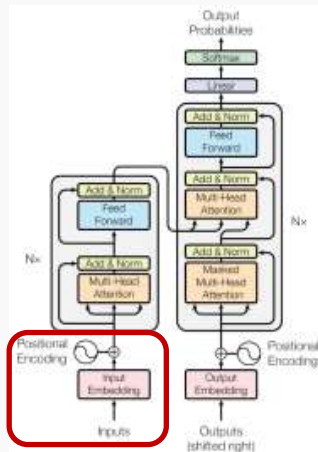


Fig. 15: Basic Transformer

What is Positional Encoding

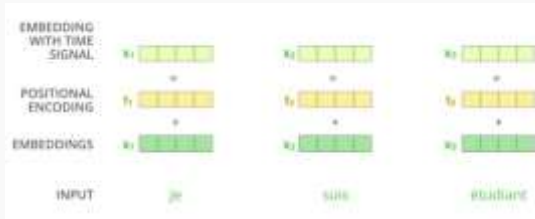


Fig. 16: Positional Encoding

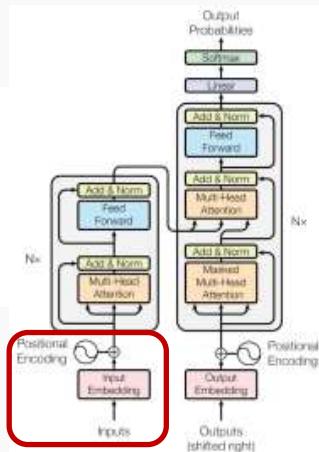


Fig. 17: Basic Transformer

What is Positional Encoding

The design principles of positional encoding are as follows:

- Expressing Absolute Position
- Expressing Relative Position
- Compatibility with Word Embeddings
- Generalizing Across Different Sequence Lengths

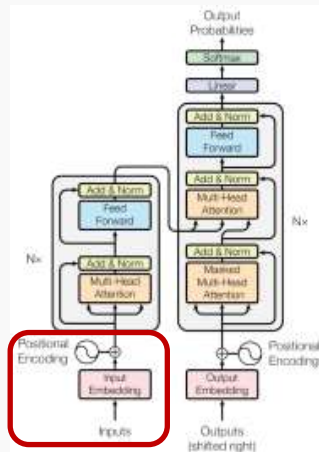


Fig. 18: Basic Transformer

Positional Encoding - Absolute Position

$$PE(pos, 2i) = \sin \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$

$$PE(pos, 2i + 1) = \cos \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$

- pos :The positions in the input sequence.
- i :Dimension index.
- d_{model} :Dimension.

Q : What is the capital of France?

A : France is a country in Western Europe.
Its capital is Paris.

Positional Encoding - Absolute Position

$$PE(pos, 2i) = \sin \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$

$$PE(pos, 2i + 1) = \cos \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$

- pos :The positions in the input sequence.
- i :Dimension index.
- d_{model} :Dimension.

- **Expressing Absolute Position**

Unique codes for each position are created by different combinations of position and dimension.

Positional Encoding- Relative Position

$$PE(pos, 2i) = \sin \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$

$$PE(pos, 2i + 1) = \cos \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$

- pos :The positions in the input sequence.
- i :Dimension index.
- d_{model} :Dimension.

Apple Inc. is a technology company.

Positional Encoding- Relative Position

$$\text{PE}(\text{pos}, 2i) = \sin \frac{\text{pos}}{10000^{\frac{2i}{d_{\text{model}}}}}$$

$$\text{PE}(\text{pos}, 2i + 1) = \cos \frac{\text{pos}}{10000^{\frac{2i}{d_{\text{model}}}}}$$

- pos :The positions in the input sequence.
- i :Dimension index.
- d_{model} :Dimension.

• Expressing Relative Position

Using Trigonometric Properties for Relative Positions

$$\cos(A + B) = \cos A \cdot \cos B - \sin A \cdot \sin B$$

$$\sin(A + B) = \sin A \cdot \cos B + \cos A \cdot \sin B$$

Positional Encoding- Relative Position

$$PE(pos, 2i) = \sin \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$

$$PE(pos, 2i + 1) = \cos \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$

- pos :The positions in the input sequence.
- i :Dimension index.
- d_{model} :Dimension.

- **Expressing Relative Position**

$$PE(pos + k, 2i) = PE(pos, 2i) \cdot PE(k, 2i + 1) + PE(pos, 2i + 1) \cdot PE(k, 2i)$$

$$PE(pos + k, 2i + 1) = PE(pos, 2i + 1) \cdot PE(k, 2i + 1) - PE(pos, 2i) \cdot PE(k, 2i)$$

Positional Encoding - Compatibility

$$PE(pos, 2i) = \sin \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$

$$PE(pos, 2i + 1) = \cos \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$

- pos :The positions in the input sequence.
- i :Dimension index.
- d_{model} :Dimension.

- **Compatibility with Word Embeddings**

Limited value range

The range of positional encoding needs to be limited to avoid dominating the embeddings.

Positional Encoding - Different Sequence Lengths

$$PE(pos, 2i) = \sin \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$

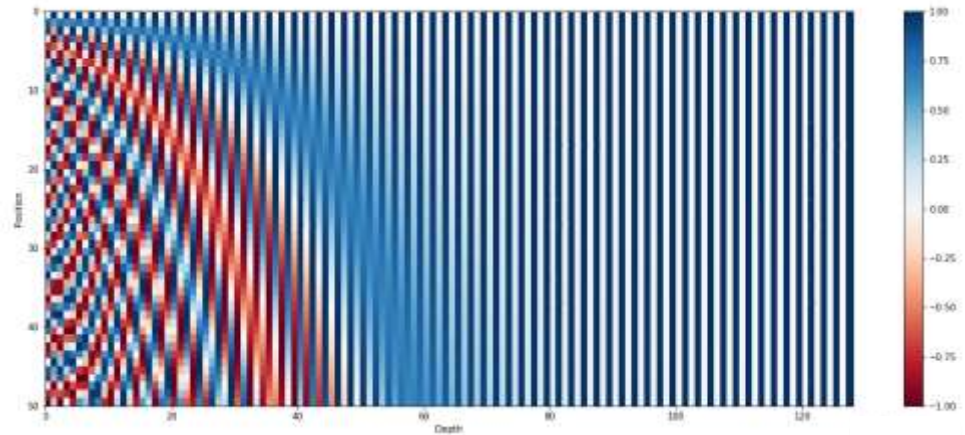
$$PE(pos, 2i + 1) = \cos \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$

- pos :The positions in the input sequence.
- i :Dimension index.
- d_{model} :Dimension.

• Generalizing Across Different Sequence Lengths

The formula can be applied to sequences of any length.

Positional Encoding - Different Sequence Lengths



Why addition instead of concatenation?

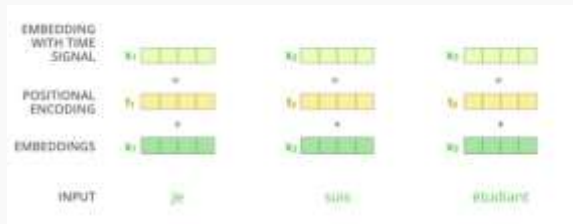


Fig. 16: Positional Encoding

The input to the transformer has already undergone a linear transformation.

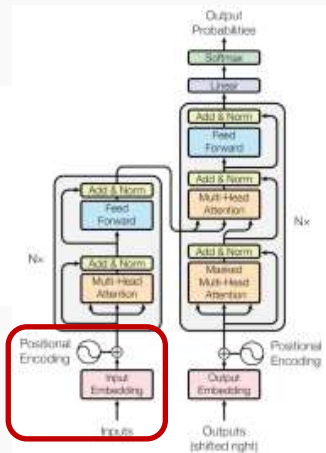


Fig. 17: Basic Transformer

Why addition instead of concatenation?

$$\begin{aligned} W \cdot x_p^i &= [W^I, W^P] \cdot [[x^i]^T, [x^p]^T]^T \\ &= W^I \cdot x^i + W^P \cdot x^p \\ &= \text{embed}^i + \text{pos}^i \end{aligned}$$

Addition \approx concatenation

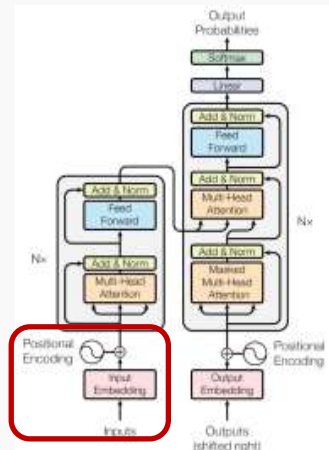


Fig. 17: Basic Transformer

1.3.3 Feed forward

Feed forward

```
class Feed_Forward(nn.Module):  
    def __init__(self, input_dim, hidden_dim=2048):  
        super(Feed_Forward, self).__init__()  
        self.L1 = nn.Linear(input_dim, hidden_dim)  
        self.L2 = nn.Linear(hidden_dim, input_dim)  
  
    def forward(self, x):  
        output = nn.ReLU()(self.L1(x))  
        output = self.L2(output)  
        return output
```

Fig. 19: Feed forward

- The feed-forward layer introduces non-linearity.

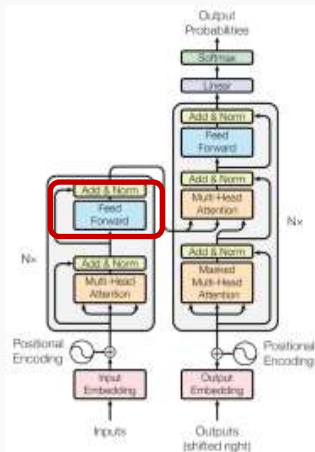


Fig. 20: Basic Transformer

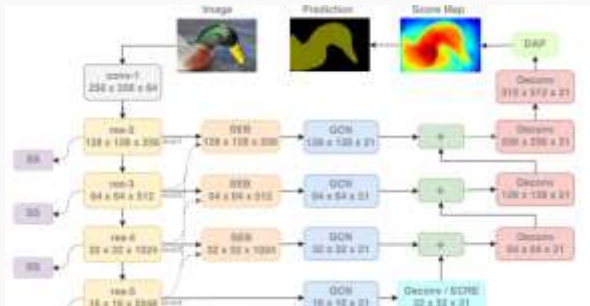
2.Developments in ViT

2.1 Motivation

**How can the transformer be used effectively
in computer vision?**

The Dominant Architecture in CV: CNN Series

例: ExFuse



In computer vision, the preservation of CNN architecture is a common practice. If attention is desired:

- Combine attention with convolutional neural networks.
- Use attention to replace a part of the CNN model.

The Dominant Architecture in CV: CNN Series


In computer vision, the preservation of CNN architecture is a common practice.

The dependence on CNNs is unnecessary

Challenges of Transformers in Computer Vision

Challenges of Transformers in Computer Vision

Image Data Structure

六一  节

六一 “儿童” 节？
六一 “中秋” 节？



这是小鸟！

Challenges of Transformers in Computer Vision

Enormous Computational Load



$$512 \times 512 = 262144$$



梅城

◎ 曹开第 · 梅城 · 社会悬疑 · 长篇

方言的养父母去世之后，方言凭着一串信息，去往梅城，寻找自己的亲生父母，找到了。就为了问他们一

26112万字 | 新书《说吧，梅城》发布

2.2 Vision Transformer

Vision Transformer

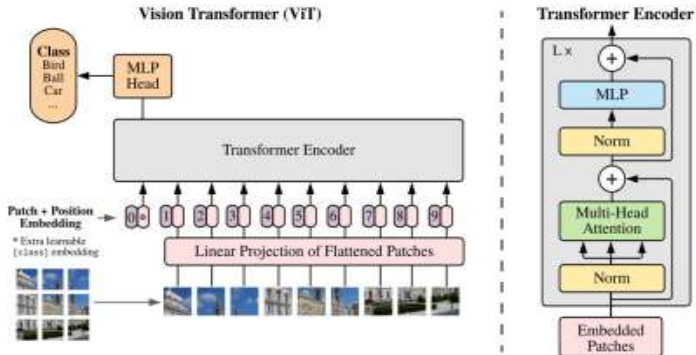


Fig. 21: ViT

Google introduced the Vision Transformer (ViT) model for large-scale image recognition.

Vision Transformer

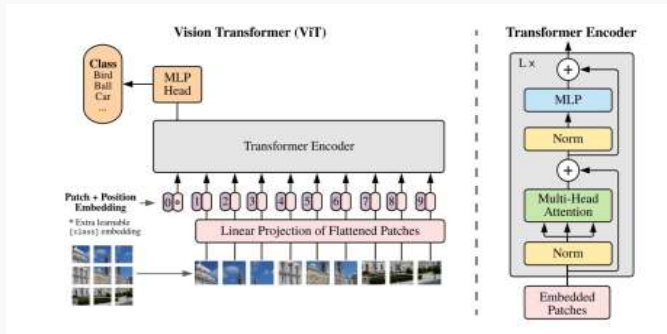


Fig. 22: ViT

In model design, we aim to closely follow the original transformer.

- This design allows easy use of scalable NLP transformer architectures.

Vision Transformer

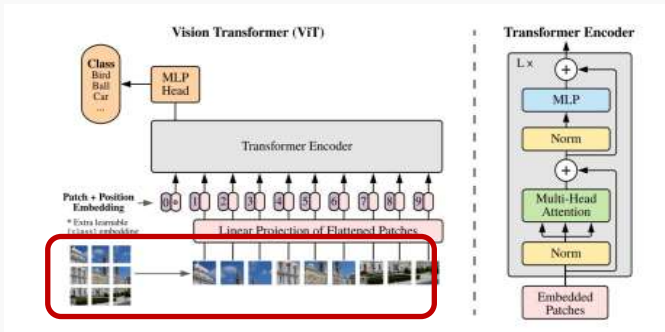


Fig. 23: ViT

Image Data Structure

- Images are segmented into patches for Transformer input.

Vision Transformer

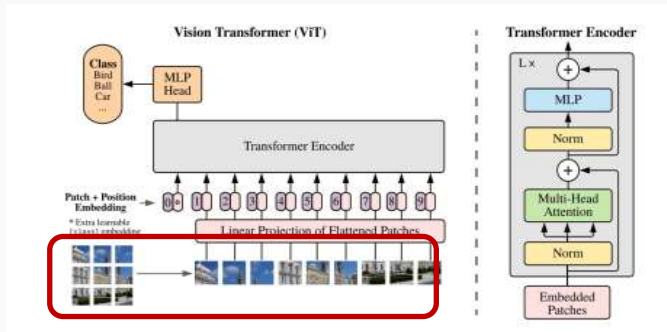


Fig. 26: ViT

Enormous Computational Load

- Segment an image into multiple patches

How to perform classification tasks

cls token

The CLS token first appeared in BERT

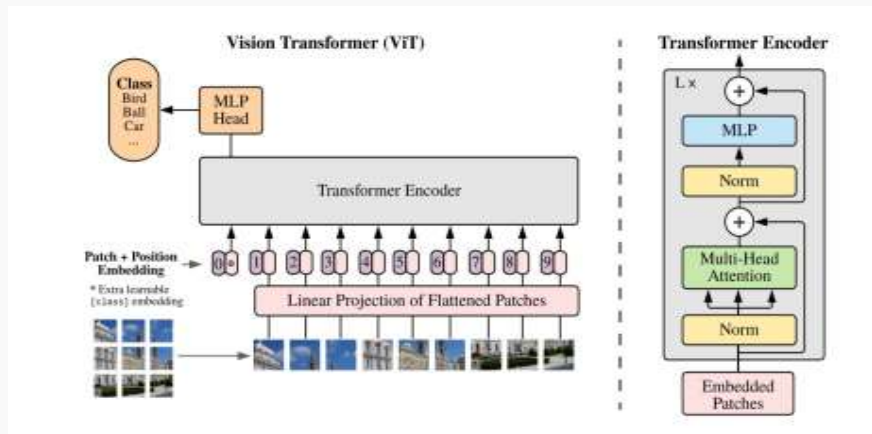


Fig. 27: ViT

cls token

The final CLS token will contain global information because of the self-attention

- Convenient for fine-tuning

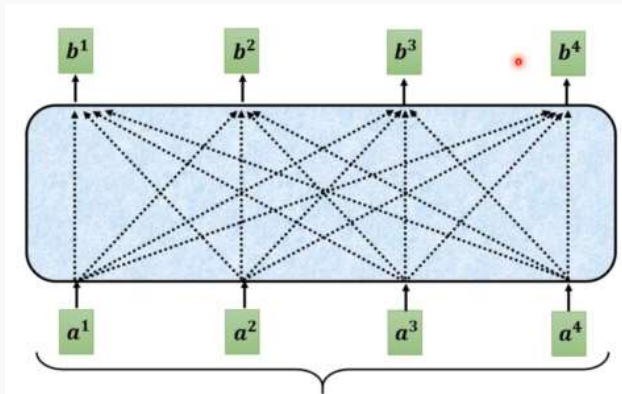


Fig. 28: attention

Vision Transformer

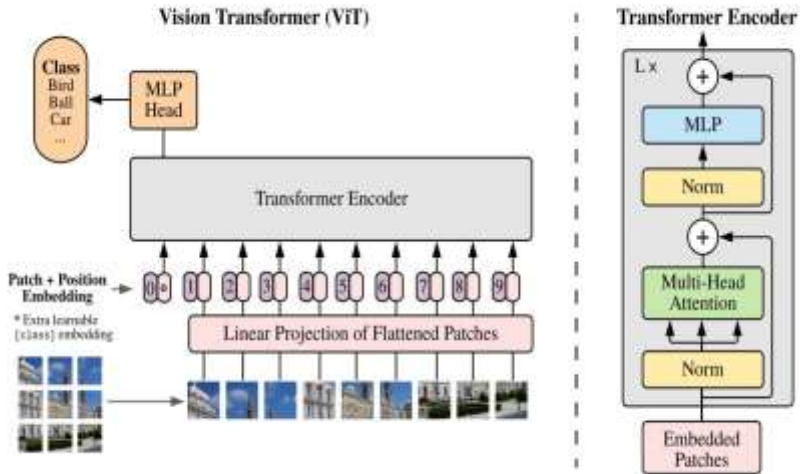


Fig. 29: ViT

Vision Transformer



Fig. 31: The ViT model performs well on large-scale datasets.

Vision Transformer

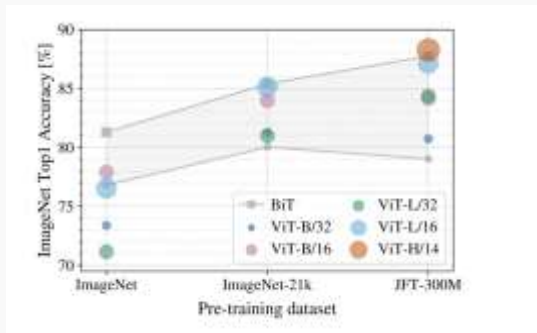


Fig. 32: The ViT model performs well on large-scale datasets.

Transformers lack some of the inherent biases found in CNNs

- locality (adjacent regions having similar features)
- invariance to translation

How to apply it to downstream tasks?

However, if ViT is applied to dense prediction tasks, it encounters several problems:

- Computational and memory consumption
- The scale problem

- The scale problem



2.4.1 PVT v1

Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions

Wenhai Wang¹, Enze Xie², Xiang Li³, Deng-Ping Fan^{4,5},
Kaitao Song³, Ding Liang⁵, Tong Lu^{1,5}, Ping Luo², Ling Shao⁴

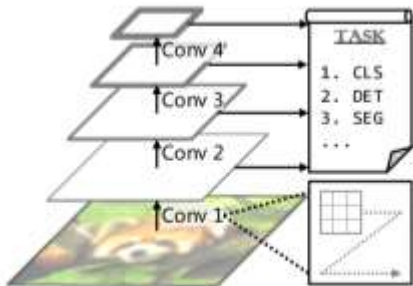
¹Nanjing University ²The University of Hong Kong

³Nanjing University of Science and Technology ⁴IIAI ⁵SenseTime Research

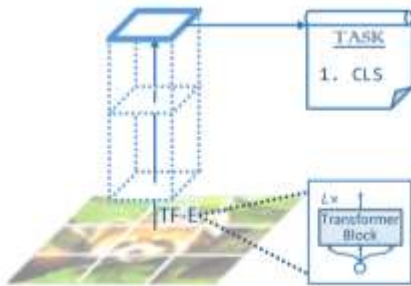
<https://github.com/whai362/PVT>

Fig. 33: PVT v1

Motivation

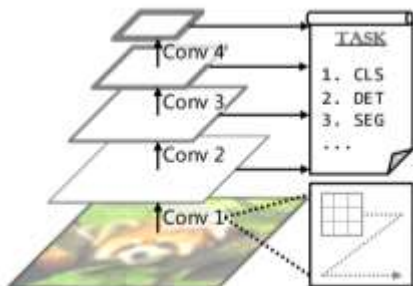


(a) CNNs: VGG [54], ResNet [22], etc.

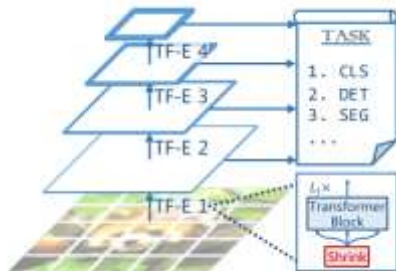


(b) Vision Transformer [13]

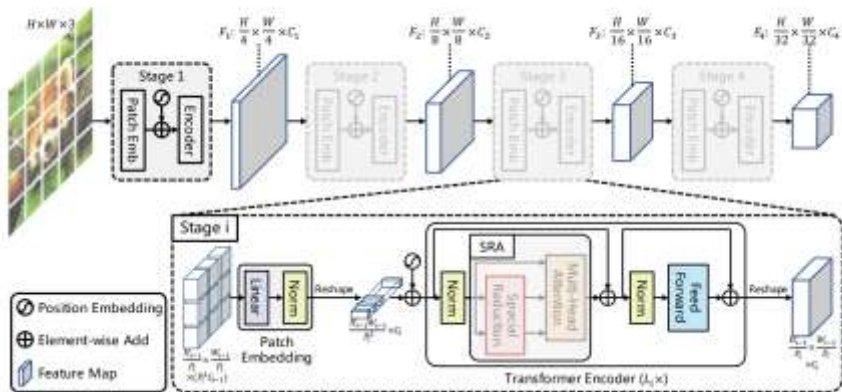
分辨率高的图像细节信息多但语义信息少
强语义信息的图像细节少，分辨率低



(a) CNNs: VGG [54], ResNet [22], etc.



(c) Pyramid Vision Transformer (ours)



PVT v1 - patch embedding

Through patch embedding Generate features at different scales

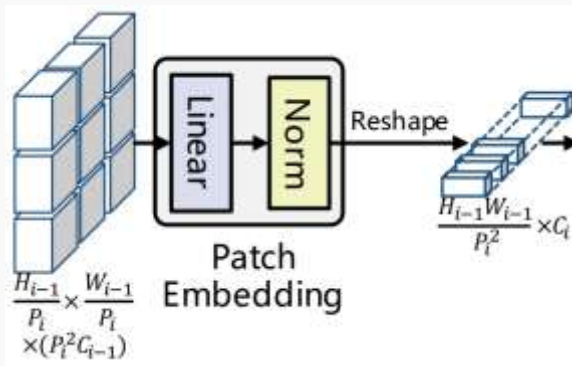


Fig. 35: patch embedding

Patch embedding significantly increases computational complexity.

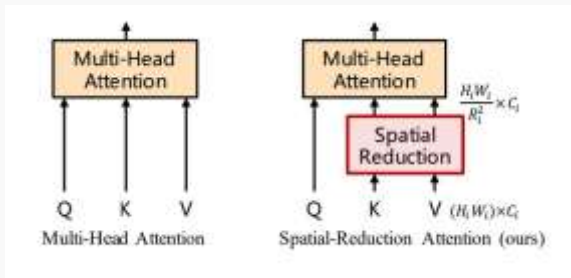


Fig. 36: SRA

Design SRA to reduce computational complexity.

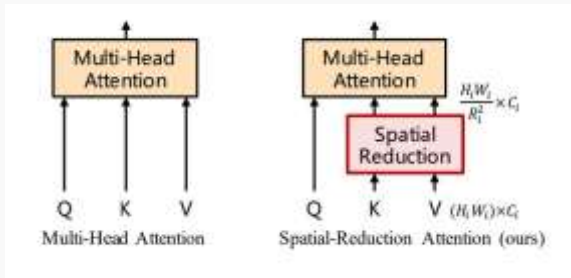


Fig. 37: SRA

Spatial Reduction Attention, adds a Spatial Reduction step to the conventional multi-head self-attention.

$$SRA(Q, K, V) = \text{Concat}(\text{head}_0, \dots, \text{head}_N) W^O \quad (7)$$

$$\text{head}_j = \text{Softmax} \frac{QW_j^Q (SR(K)W_j^K)^T}{\sqrt{d_k}} SR(V)W_j^V \quad (8)$$

$$SR(x) = \text{Norm}(\text{Reshape}(x, R_i) W^S) \quad (9)$$

SRA downsamples K and V before performing self-attention, changing feature map resolution.

However, PVT v1 encounters several problems:

1. PVT v1's use of 4x4 patches can neglect local image continuity.
2. PVT v1's fixed-size positional encodings are less effective for varying image sizes.
3. These methods still entail a high computational cost.

2.4.2 PVT v2

PVT v2: Improved Baselines with Pyramid Vision Transformer

Wenhai Wang^{1,2}, Enze Xie³, Xiang Li⁴, Deng-Ping Fan⁵,

Kaitao Song⁴, Ding Liang⁶, Tong Lu², Ping Luo³, Ling Shao⁵

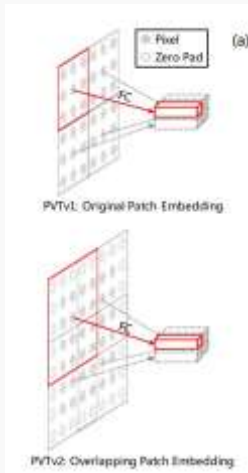
¹Shanghai AI Laboratory ²Nanjing University ³The University of Hong Kong

⁴Nanjing University of Science and Technology ⁵IIAI ⁶SenseTime Research

wangwenhai@pjlab.org.cn,

Fig. 38: PVT v2

Improve the patch embedding module of PVT v1 to enhance the interaction between neighboring patches.



By enlarging the patch window, the overlap area between adjacent windows is reduced by half.

Depth-wise convolution in the FeedForward module replaces positional encoding, reducing computation.

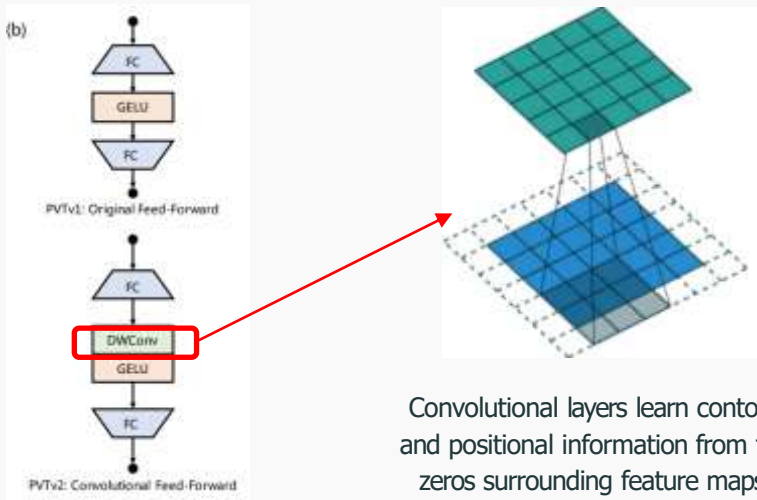


Fig. 40: pos

Pooling replaces convolution in SRA to reduce complexity.

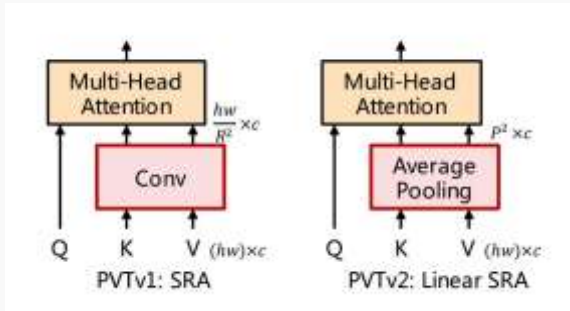


Fig. 41: linear SRA

v1: Conv + linear

v2: pooling + 1×1 Conv + linear + GELU

2.3 Swin transformer

Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

Ze Liu^{1*} Yutong Lin^{1*} Yue Cao¹ Han Hu^{2,1} Yixuan Wei¹
Zheng Zhang¹ Stephen Lin¹ Baining Guo¹
Microsoft Research Asia

{v-zeliu1, v-yutlin, yucaoc, hanhu, v-yixwe, zhez, stevelin, bainguo}@microsoft.com

Fig. 42: swin transformer

Swin transformer

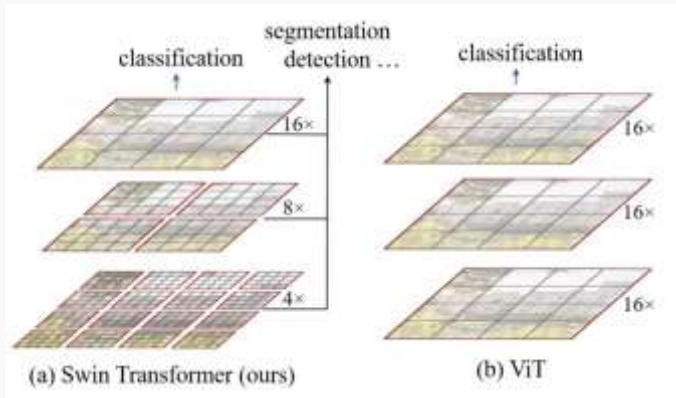


Fig. 43: Swin transformer

The Swin Transformer introduces hierarchical, multi-scale feature extraction similar to CNNs.

Patch merging:

Similar to the pooling operation in CNN

Swin transformer - patch merging

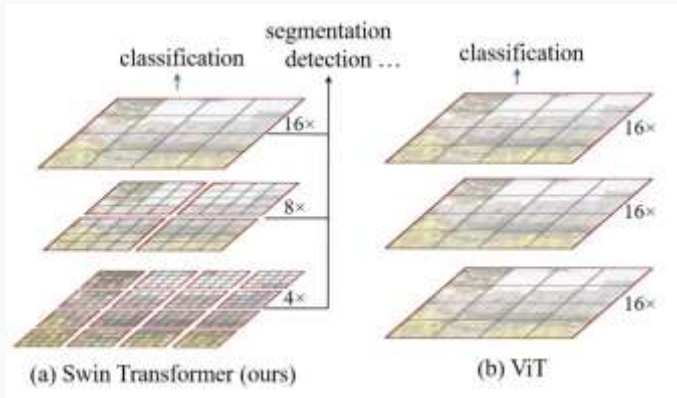


Fig. 44: swin transformer

It merges smaller patches into larger ones for enhanced receptive field and multi-scale feature capture.

Swin transformer - patch merging

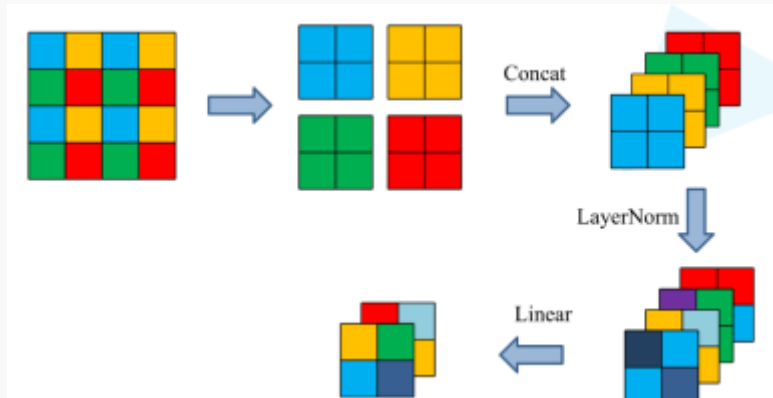


Fig. 45: patch merging

The Transformer's Patch Merging layer reduces feature map dimensions while increasing depth.

The small patches result in an unbearable
sequence length.

Swin transformer – W- MSA

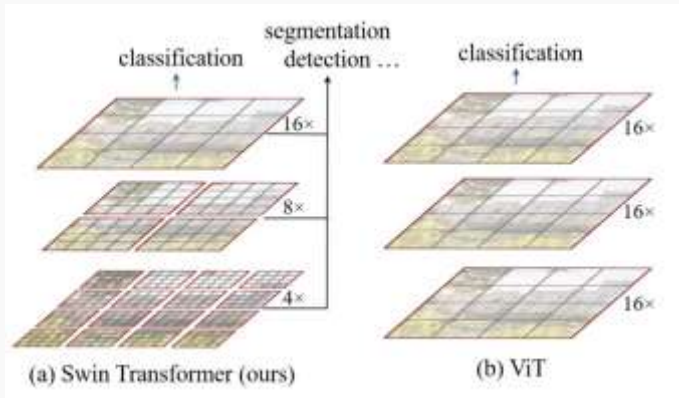


Fig. 46: swin transformer

It uses a windowing approach for multi-head attention in non-overlapping regions of the feature map.

Swin transformer – W-MSA

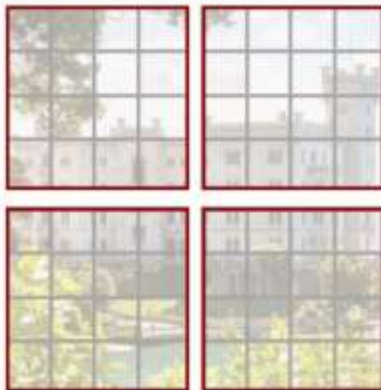


Fig. 47: Window-MSA

The W-MSA module divides the map into $M \times M$ windows for separate Self-Attention in each.

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

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

Fig. 48: W-MSA

The amount of computation is greatly reduced !

**Issue with W-MSA: Limited Information
Interaction and Reduced Receptive Field.**

Swin transformer – SW-MSA

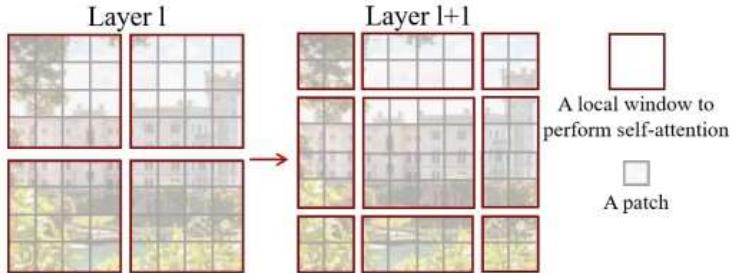


Fig. 49: Shifted Window-MSA

- Window movement in the Transformer enables interactive learning and information exchange between patches.

Swin transformer

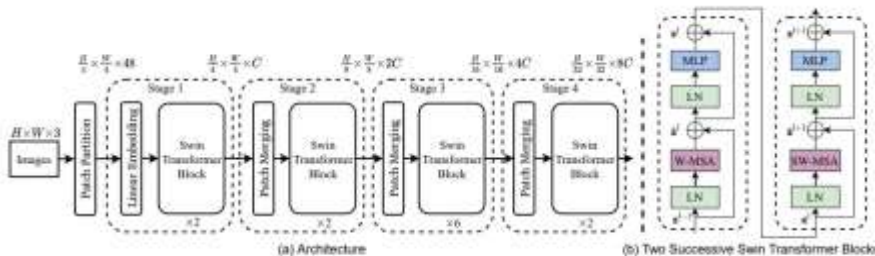


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

Fig. 50: swin transformer

RetNet



Questions?

Thanks for listening!