Computer Vision - 2025

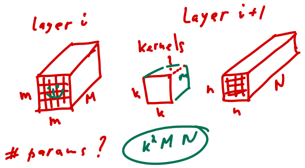
Lecture #03. Visual Transformers

Lectures by Alexei Kornaev [1,2,3]
Practical sessions by Kirill Yakovlev [2]

[1]AI Institute, Innopolis University (IU), Innopolis
[2]Robotics & CV Master's Program, IU, Innopolis
[3]RC for AI, National RC for Oncology, Moscow

February 3, 2025

Agenda



Section 1. Outcomes

Outcomes

This week's lecture on Vision Transformers (ViTs) aims to provide an understanding of transformer-based architectures in computer vision. By the end of this week, students will be able to:

1. Understand the core concepts of Transformer architecture: embeddings, positional encoding, self-attention, and multi-head attention.
2. Explain Vision Transformers (ViTs), including patch embeddings, transformer blocks, and classification heads.
3. Compare CNNs and ViTs for vision tasks, highlighting their strengths and weaknesses.
4. Analyze modern ViT architectures such as DeiT and hybrid CNN-ViT models.

Key Takeaway: Vision Transformers (ViTs) provide an alternative to CNNs by leveraging self-attention, enabling global context modeling in images .

Section 2. Transformer Architecture Overview

INNOPOLIS UNIVERSITY

Transformer Architecture

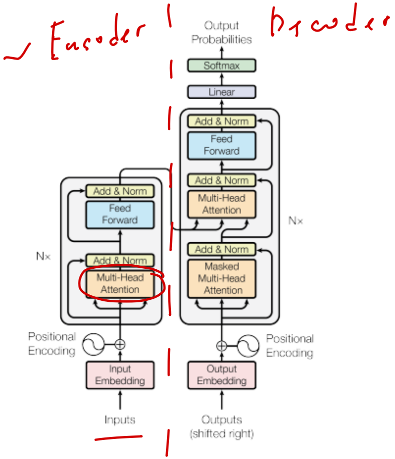


Figure: High-level Transformer architecture, consisting of an encoder and decoder stack [Vaswani et al., 2023].

Self-Attention Mechanism

Self-attention allows the model to weigh different parts of the input sequence when making predictions.

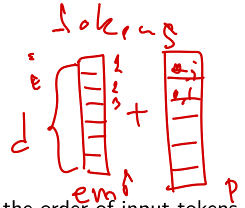
Scaled Dot-Product Attention

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

where:

- $\underline{Q, K, V}$ are the query, key, and value matrices.
- d_k is the dimensionality of the key vectors.

Positional Embedding in Transformers



Why Positional Embedding?

Transformers, unlike CNNs or RNNs, do not inherently understand the order of input tokens. Positional embeddings are added to input embeddings to inject information about the position of tokens in the sequence.

- **Input Embedding**: Represents the content of each token.
- **Positional Embedding**: Represents the position of each token.
- **Combined**: Input embedding + Positional embedding.

$$\mathbf{E} = \mathbf{E}_{\text{input}} + \mathbf{E}_{\text{position}} \quad (1)$$

Key Takeaway: Positional embeddings enable Transformers to process sequences with order-awareness.

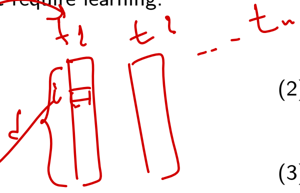
Sinusoidal Positional Encoding

Sinusoidal Encoding

In the original Transformer paper [Vaswani et al., 2023], sinusoidal functions are used to generate positional encodings. These encodings are deterministic and do not require learning.

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \tag{2}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \tag{3}$$



- *pos*: Position of the token in the sequence.
- *i*: Dimension index of the embedding.
- *d*: Dimensionality of the embedding.

Key Takeaway: Sinusoidal encoding allows the model to generalize to sequences longer than those seen during training.

Positional Embedding in Vision Transformers (ViTs)

Patch Embeddings and Positional Encoding

In Vision Transformers (ViTs), the input image is divided into patches, and each patch is treated as a token. Positional embeddings are added to these patch embeddings to preserve spatial information.

$$\mathbf{E}_{\text{patch}} = \text{Linear}(\text{Flatten}(\text{Patches})) \tag{4}$$

$$\mathbf{E} = \mathbf{E}_{\text{patch}} + \mathbf{E}_{\text{position}} \tag{5}$$

- **Patches**: Image divided into fixed-size patches (e.g., 16x16).
- **Positional Embedding**: Added to patch embeddings to encode spatial location.

Key Takeaway: Positional embeddings in ViTs help the model understand the spatial arrangement of patches in the image.

Section 3. Vision Transformers (ViTs)

Image to Patch Splitting

Key Idea

ViTs divide the image into **non-overlapping patches** of fixed size.

- Given an image $x \in \mathbb{R}^{H \times W \times C}$ (height H, width W, channels C), we divide it into N patches of size $P \times P$.
- The number of patches is:

$$N = \frac{HW}{\underline{P^2}}.$$

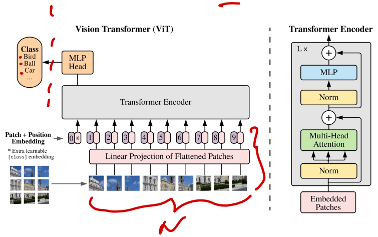


Figure: ViT intuition [Dosovitskiy et al., 2021].

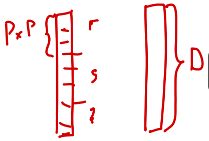
Linear Projection of Flattened Patches

Patch Embeddings

Each patch is flattened into a vector and projected into a higher-dimensional space using a linear layer.

- Each patch $\underline{x_p} \in \mathbb{R}^{P \times P \times C}$ is flattened into a vector $x_p \in \mathbb{R}^{P^2 C}$.
- A trainable weight matrix $E \in \underline{\mathbb{R}^{D \times P^2 C}}$ projects it into a D-dimensional embedding:

$$\underline{z_i = Ex_i,} \quad i = 1, \ldots, N.$$



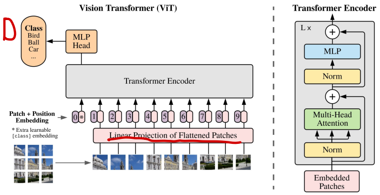


Figure: ViT intuition [Dosovitskiy et al., 2021].

Positional Embeddings

Why Positional Encoding?

Transformers process input as a set of tokens without spatial order. Positional embeddings restore spatial information.

- Each patch embedding receives a learned positional encoding E_p.
- The final input to the Transformer is: .

$$z_i = \underline{Ex_i} + \underline{E_p(i)}, \quad i = 1, \ldots, N$$

- A [CLS] token is prepended for classification tasks.

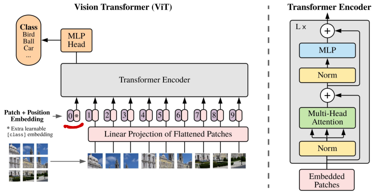


Figure: ViT intuition [Dosovitskiy et al., 2021].

Learned Positional Embeddings in ViTs

Instead of fixed sinusoidal functions, Vision Transformers often use

learned positional embeddings:

$$\mathbf{z}_0 = \texttt{class token}, \quad \mathbf{z}_i = E(x_i) + E_p(i), \quad i = 1, \ldots, N$$

where:

- $E(x_i)$ is the patch embedding, $E_p(i)$ is the learned positional embedding for patch i,
- $\mathbf{z}_0$ is an extra classification token.

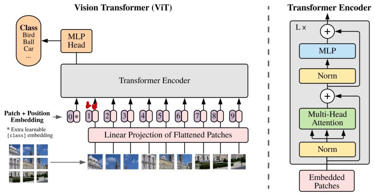
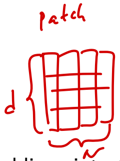


Figure: ViT intuition [Dosovitskiy et al., 2021].

Input Representation: Query, Key, and Value



What Are Q, K, V?

Self-attention transforms input embeddings into Query (Q), Key (K), and Value (V) vectors using trainable weight matrices.

- **Input sequence:** $\underline{\boxed{X} \in \mathbb{R}^{N\times d}}$ (where N is the number of patches, and d is the embedding dimension).
- **Linear projections:**

$$Q = \underline{X}\overset{\checkmark}{W_Q}, \quad K = \underline{X}\overset{\checkmark}{W_K}, \quad V = \underline{X}\overset{\checkmark}{W_V}$$

  where $W_Q, W_K, W_V \in \underline{\mathbb{R}^{d\times d}}$ are learnable.
- **Roles of Q, K, V:** - **Query:** What we are looking for. - **Key:** What is stored in memory. - **Value:** What information is retrieved.

Similarity Computation: Scaled Dot-Product Attention

A.Kornaev,
K.Yakovlev

How Does Attention Work?

Each query compares itself with all keys to measure similarity.

- **Compute dot-product similarity between Q and K:**

$$S = QK^T \in \mathbb{R}^{N \times N}$$

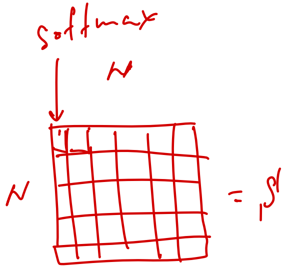
- **Normalize using $\sqrt{d}$ to stabilize gradients:**

$$\underline{A = \frac{S}{\sqrt{d}}}$$

- **Apply the softmax function:**

$$A_{ij} = \frac{\exp(S_{ij}/\sqrt{d})}{\sum_k \exp(S_{ik}/\sqrt{d})}$$

- The resulting **attention matrix** A defines how much focus each query has on every key.



Weighted Sum: Generating the Output

How Is the Output Computed?

Each patch's representation is updated based on the weighted sum of all patches.

- **Multiply the attention weights A by the Value matrix V:**

$$\underline{Z = AV}$$

- Each output row is a **weighted sum of value vectors**.
- Self-attention enables **global interactions between patches**.

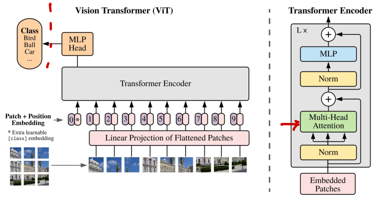


Figure: ViT intuition [Dosovitskiy et al., 2021].

Multi-Head Self-Attention

Why Use Multiple Heads?

Multi-head attention enables the model to capture multiple patterns in data.

- Instead of one set of Q, K, V, use **multiple independent projections:**

$$\underline{Q_i} = XW_{Q_i}, \quad \underline{K_i} = XW_{K_i}, \quad \underline{V_i} = XW_{V_i}$$

- Compute attention separately for each head:

$$\text{head}_i = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_h}}\right) V_i$$

- **Concatenate outputs** from all h heads and apply a final transformation:

$$\text{MSA}(X) = \text{Concat}(\underline{\text{head}_1}, \ldots, \underline{\text{head}_h})\underline{W_O}$$

Layer Normalization and Residual Connection

Why Normalize and Add Residuals?

Normalization ensures stable training, and residual connections help gradient flow.

- **Residual connection around self-attention:**

$$X' = \underline{X + \text{MSA}(X)}$$

- **Apply Layer Normalization:**

$$X'' = \underline{\text{LayerNorm}(X')}$$

- This prevents **vanishing gradients** and stabilizes training.

INNOPOLIS UNIVERSITY

Feedforward Network (MLP) and Output

What Happens After Self-Attention?

Each token representation is refined using a feedforward network (MLP).

- **Apply a 2-layer MLP with activation:**

$$Y = \underline{\mathrm{MLP}(X'')}$$

- Includes two linear transformations and a GELU activation:

$$Y = \mathrm{ReLU}(X''W_1 + b_1)W_2 + b_2$$

- **Final output:**

$$Z = X'' + Y$$

- This structure **Self-Attention** $\rightarrow$ **MLP** $\rightarrow$ **Residuals** defines the **Transformer Encoder Block**.

Comparison: CNNs vs. ViTs

Key Differences

- CNNs learn spatial hierarchies via local receptive fields and weight sharing.
- ViTs model long-range dependencies using self-attention but lack inductive biases like locality and translation invariance.

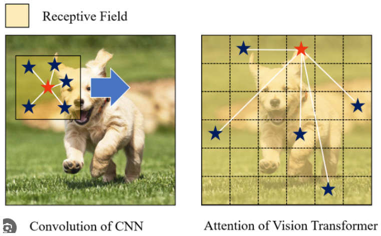


Figure: Comparison of CNNs and ViTs in feature extraction and learning patterns [Baek et al., 2022]

ViT Models Zoo

The following resources should be met:

1. ViTs, Community Computer Vision Course by Hugging Face
2. Vision Transformer (ViT) by Hugging Face

Section 4. Conclusion

Strengths and Weaknesses of ViTs

Strengths

- **Long-range dependencies** captured effectively.
- **Scalability** to large datasets.
- **Flexibility** to adapt across domains (text, vision, multimodal).

Weaknesses

- **Data-hungry** - requires large-scale training.
- **Computationally expensive** due to self-attention complexity.
- **Lack of inductive bias** â struggles with small datasets.

Conclusion

Key Takeaways

- Vision Transformers (ViTs) treat images as sequences of patches and use self-attention for feature extraction.
- Unlike CNNs, ViTs do not rely on convolution but instead learn global relationships from data.
- While powerful, ViTs require large datasets and high computational resources to generalize well.
- Hybrid architectures (CNN + ViT) help balance efficiency and performance.

Looking Ahead: Modern architectures like DeiT, Swin Transformer, and hybrid models address ViT limitations.

Bibliography

A. Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," NeurIPS 2020. [arXiv:2010.11929](https://arxiv.org/abs/2010.11929)

A. Vaswani et al., "Attention Is All You Need," NeurIPS 2017. [arXiv:1706.03762](https://arxiv.org/abs/1706.03762)

H. Touvron et al., "Training Data-efficient Image Transformers Distillation through Attention," ICML 2021. [arXiv:2012.12877](https://arxiv.org/abs/2012.12877)

Z. Liu et al., "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," ICCV 2021. [arXiv:2103.14030](https://arxiv.org/abs/2103.14030)

S. Prince, *Understanding Deep Learning*, MIT Press, 2023.

Bibliography

Sihun Baek, Jihong Park, Praneeth Vepakomma, Ramesh Raskar, Mehdi Bennis, and Seong-Lyun Kim. Visual transformer meets cutmix for improved accuracy, communication efficiency, and data privacy in split learning, 2022. URL https://arxiv.org/abs/2207.00234.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.