10417/10617

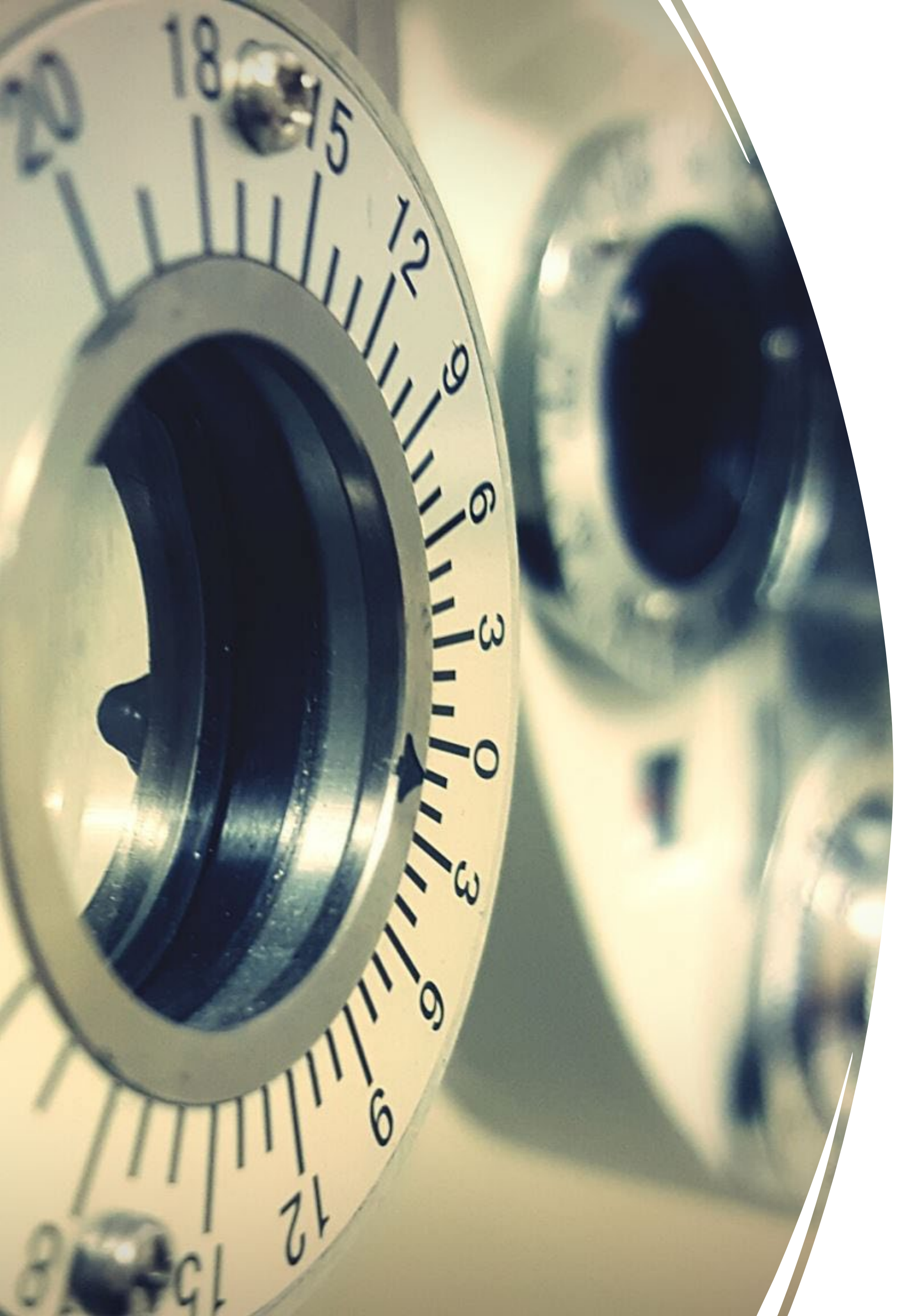
Vision Transformer: Attention is all you need

Convolution Neural Network

- So far, we have been focusing on using convolution neural networks for vision.
- Advantage: Convolution is a local operation, it is fast and parameter-efficient.
- Disadvantage: Convolution is a local operation, it can not capture global relations efficiently.
  - It is hard to use convolution on images with large global dependencies, or videos with long-time dependencies.

From Convolution to Vision Transformer

- Vision Transformer:
  - Still mainly uses local operation.
  - Add a “global attention” layer to gather information globally.



Vision Transformer

- Basic structure of ViT:
- Step 1, divide the input images ($d \times d \times C$) into (d^2/p^2) (disjoint)patches, each of size ($p \times p \times C$).
- Step 2:
  - Step 2.1. For each patch, flatten it, apply an MLP on it, and get an output of size d_{emb}
  - Step 2.2. For each vector on each patch, “mix” them together, to create new vectors on each patch.
- Repeat Step 2.

Vision Transformer

- The mixing operation is called "self-attention".
- Given vectors $v_1, v_2, \ldots, v_n$ on n patches, each of dimension D, a self-attention operator returns vectors $v'_1, v'_2, \ldots, v'_n$, each of dimension D.
- $v'_i = \sum_j \alpha_{i,j} v_j$ as a weighted average of the input vectors v's.
- $\alpha_{i,j}$ are also functions of $v_1, v_2, \ldots, v_n$.

Attention Layer

- $v'_i = \sum_j \alpha_{i,j} v_j$ as a weighted average of the input vectors v's.
  - Thus, for each i , the network mixes information from other j's to get the new output.
- $\alpha_{i,j}$ are also functions of $v_1, v_2, \dots, v_n$.
  - What functions?

Self Attention Layer

- Given vectors $v_1, v_2, \dots, v_n$, each in R^d, a self attention layer is defined as:
- $v'_i = V^T \sum_j \alpha_{i,j} v_j$
- Where $\left(\alpha_{i,j}\right)_{j \in [n]} = softmax\left(v_i^T Q K^T v_j + p_{i,j}\right)_{j \in [n]}$
- Here, Q is called the query matrix, K is called the key matrix, V is called the value matrix. They are all of dimension $d \times m$.
  - So, each v_i looks for the "most similar v_j , when projected to Q and K".
- $p_{i,j}$ is a bias term, also known as "relative positional encoding".
- They are all trainable.

Self Attention versus Convolution

- Convolution can be viewed as:
  - $v'_i = V^T \sum_j \alpha_{i,j} v_j$
  - Where $\alpha_{i,j}$ are parameters only depending on the index i , j.
  - $\alpha_{i,j}$ are non-zero only when i is “close” to j.
- Self-attention:
  - $\left(\alpha_{i,j}\right)_{j \in [n]} = softmax\left(v_i^T Q K^T v_j + p_{i,j}\right)_{j \in [n]}$ is a function of v_j’s.
  - Can be large even if index i is “far” from j.
- But can self-attention simulate a convolution?

Self-Attention can learn convolution

- If the ground-truth function is a convolution, then doing gradient descent on a self-attention layer can recover the convolution structure.
  - See the work [Samy Jelassi and Yuanzhi Li, 2022]: How vision transformer learns the patch associations.
- Intuitively, the network just learns $\left(\alpha_{i,j}\right)_{j\in[n]} = softmax\left(p_{i,j}\right)_{j\in[n]}$
- Where $p_{i,j}$ is more positive if index i is "closer" to index j.

Self-Attention can associate patches

- A function that can be easily represented by self-attention, but not convolution is the “association” function (suppose each v_i has norm one):
- $f_1(v_1, v_2, \dots, v_n) = \sum_i v_i 1_{v_i = U v_1}$
- Can be represented using $QK^T = \alpha U^{-1}$ for a large α.
- $\left(\alpha_{1,j}\right)_{j \in [n]} = softmax\left(\alpha v_1^T U^{-1} v_j\right)_{j \in [n]}$

Self-Attention can associate patches

- An function that can be easily represented by self-attention, but not convolution is the “double if” function (suppose each v_i has norm one):
- $f_1(v_1, v_2, \ldots, v_n) = \sum_i 1_{v_1=a, v_j=b}$
- Can be represented using $QK^T = \alpha a b^T$ for a large α.
- $\left(\alpha_{1,j}\right)_{j\in[n]} = softmax\left(\alpha(v_1^T a)(b^T v_j)\right)_{j\in[n]}$

Self-attention is more powerful than convolution

Self-attention is more powerful than convolution

Multi-Head Attention Layer

- The most fundamental layer in the transformer: Multi-head attention.
- Given vectors $v_1, v_2, \dots, v_n$, each in R^d, a multi-head attention layer is defined as:
- $v_i' = C \times concatenate\left(V_r \sum_j \alpha_{i,j}^r v_j\right)_{r \in [d/m]} + b$
- Where $\left(\alpha_{i,j}^r\right)_{j \in [n]} = softmax\left(v_i^T Q_r K_r^T v_j + p_{i,j}^r\right)_{j \in [n]}$
- Here, C is a $d \times d$ trainable matrix.
- Each v_i looks for the "most similar v_j , according to [d/m] many projection matrices Q_r and K_r.

Transformer Architecture

- A (post-layernorm) transformer block is defined as:
- Given input $V = v_1, v_2, \ldots, v_n$, each v_i in R^d.
  - (1). Apply Multi-Head Attention (input dimension d, output dimension d) on V to get $V^{(1)} = v_1^{(1)}, v_2^{(1)}, \ldots, v_n^{(1)}$.
  - (2). Apply layer-norm on each of the $v_i^{(1)}$ to get $v_i^{(2)}$.
  - (3). Apply residual link: $v_i^{(3)} = v_i^{(2)} + v_i$.
  - (4). Apply a one hidden layer MLP h (input dimension d, output dimension d) on each $v_i^{(3)}$ to get $v_i^{(4)} = h(v_i^{(3)})$ (all the v_i''' in the uses the same h per layer, different h for different layers).
  - (5). Apply layer-norm on each of the $v_i^{(4)}$ to get $v_i^{(5)}$.
  - (6). Apply residual link: $v_i^{(6)} = v_i^{(5)} + v_i^{(3)}$.
- The output $V^{(6)} = v_1^{(6)}, v_2^{(6)}, \ldots, v_n^{(6)}$, each $v_i^{(6)}$ in R^d.

Transformer Architecture

- A (pre-layernorm) transformer block is defined as:
- Given input $V = v_1, v_2, \ldots, v_n$, each v_i in R^d.
  - (1). Apply layer-norm on each of the v_i to get $v_i^{(1)}$.
  - (2). Apply Multi-Head Attention on $V^{(1)}$ to get $V^{(2)} = v_1^{(2)}, v_2^{(2)}, \ldots, v_n^{(2)}$.
  - (3). Apply residual link: $v_i^{(3)} = v_i^{(2)} + v_i$.
  - (4). Apply layer-norm on each of the $v_i^{(3)}$ to get $v_i^{(4)}$.
  - (5). Apply a one hidden layer MLP h on each $v_i^{(4)}$ to get $v_i^{(5)} = h(v_i^{(4)})$ (all the v_i''' in the uses the same h per layer, different h for different layers).
  - (6). Apply residual link: $v_i^{(6)} = v_i^{(5)} + v_i^{(3)}$.

Transformer Architecture

- A Vision Transformer consists of:
- 1. An embedding layer (linear layer), maps each input patch to a vector + layer-normalization.
- 2. Many transformer blocks.
- 3. A Flatten + Linear/MLP layer on top for classification.

VIT Large (the original vision transformer)

- Convert input image to 16x16 total patches (total 256 vectors).
- For each patch, apply an embedding layer to map it to dimension 1024.
- Apply 24 transformer blocks, each transformer block has a one-hidden-layer MLP of size 1024 -> 4096 -> 1024.
    - Each transformer block as a Multi-Head Attention layer with 16 heads.
- Total 307M parameters (very small for a transformer).