Transformers for Vision
Transformer

• The de facto neural architecture of many applications
  • GPT: generative pre-training Transformer
  • Gemini: multi-modal LLMs
  • Computer vision
  • Speech recognition
Attention

• Attention compute a weighted average of hidden state vectors.
• Formally:
  • Given N $k$-dimensional input vector $x^1 \ldots x^N$ and output vector $y^1 \ldots y^N$
  • Output $y^i$ is a weighted average of input vector; $w_{ij}$ is the attention weight from position $i$ over $x^j$
    • $y^i = \sum_{j=1}^{N} w_{ij} x^j$
Formulation of Self-attention

• Given $N$ $k$-dimensional input vector $x^1 \ldots x^N$ and output vector $y^1 \ldots y^N$

• Conventional projection layer:
  • Output $y^i$ is a weighted average of input vector
    
    $$y^i = \sum_{j=1}^{N} w_{ij} x^j$$

• Self-attention: self-dependent
  • The weight $w_{ij}$ depends on input itself $x^j$

    $$w'_{ij} = \sum_k x_i^k x_j^k$$

  • Followed by Softmax normalization: $w_{ij} = \frac{\exp(w'_{ij})}{\sum_j \exp(w'_{ij})}$
Self-attention with Query, Key, Value

• Self-attention: The weight $w_{ij}$ depends on input itself
  • $w'_{ij} = \sum_k x^i_k x^j_k$
  • the input $x^i$ is to matched the input $x^j$

• More flexible version: add learnable parameters
  • Linear projection layer with learnable weight matrix $W \in \mathbb{R}^{k \times k}$
  • Query: $Q^i = W_q x^i$, to match others
  • Key: $K^i = W_k x^i$, to be matched
  • Value: $V^i = W_v x^i$
Self-attention with Query, Key, Value

• Q, K, V are linearly projected
  • Query: $Q^i = W_q x^i$
  • Key: $K^i = W_k x^i$
  • Value: $V^i = W_v x^i$

• Compute attention weight based on $Q^i$ and $K$
  • $A_{ij} = \frac{\exp(Q^i K^j)}{\sum_j \exp(Q^i K^j)}$
Self-attention with Query, Key, Value

• Compute attention weight based on $Q^i$ and $K$
  
  \[ A_{ij} = \frac{\exp(Q^iK^i)}{\sum_j \exp(Q^iK^j)} \]

• Aggregated with value vector $V$
  
  • The new output for $i$-th position depends on the attention weights and value vectors of all input positions $j$
  
  \[ y^i = \sum_{j=1}^{T} A_j V^j \]
Scaled Dot-product Attention

- Scale factor $\sqrt{d}$ normalizes the dot product values $QK^T$, preventing their variance from becoming overly large.
- Scale dot-product: $A = \text{Softmax}(\frac{QK^T}{\sqrt{d}})$
- Matrix Multiplication: $y = AV$

scaled dot–product:

\[ A_{1,i} = \frac{QK^T}{\sqrt{d}} \]

- \( Q \): query (to match others)
- \( K \): key (to be matched)
- \( V \): value
- \( x \): input vector before projection
\[
\sigma(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}
\]
\[ y^1 = \sum_i \hat{A}_{1,i} V^i \]
Quadratic complexity

- Attention(Q, K, V) = $\text{Softmax} \left( \frac{QK^T}{\sqrt{d}} \right) V$
- Complexity: $O(n^2 \cdot d)$
- $n =$ sequence length, $d =$ hidden dimension
Cross-attention

• Given two different input vector \( x \) and \( m \)
• Input to Cross-attention
  • Query: \( Q^i = W_q x^i \)
  • Key: \( K^i = W_k m^i \)
  • Value: \( V^i = W_v m^i \)
• Source vector \( x \) is attended to target vector \( m \)
• Compute cross-attention weight based on \( Q^i \) and \( K \)
  \[ A_{ij} = \frac{\exp(Q^i K^i)}{\sum_j \exp(Q^i K_j)} \]
Rethink Cross-attention as Feature Clustering

• (not exactly the same)

• input vector $x \in \mathbb{R}^{N \times d}$ ($Q^i = W_q x^i$) is the feature centroids with N centroids

• input vector $m \in \mathbb{R}^{H \times W \times d}$ ($K^j = W_k m^j$, $V^i = W_v m^i$) is the pixel features

• Cross-attention weight v.s. Soft assignment based on feature distance
  • $A_{ij} = \frac{\exp(Q^i K^j)}{\Sigma_l \exp(Q^i K^l)}$

• Value aggregation v.s. Centroid update:
  • $O^i = \Sigma_{j=1}^M A^i j V^j$
Positional encoding

• Motivation:
  • While the attention mechanism is powerful, it doesn't inherently capture the order of elements within the sequence.
  • In language, the order of words is crucial for meaning. For example, "The cat sat on the mat" has a different meaning than "The mat sat on the cat."
  • Similarly, in computer vision, the permutation of image pixels matters.
Positional encoding

• append t to the input
  • This is not a great idea, because absolute position is less important than relative position
  • we want to represent position in a way that tokens with similar relative position have similar positional encoding

\[ \bar{x}_t = \begin{bmatrix} x_t \\ t \end{bmatrix} \]

• More advanced positional encoding (e.g., frequency-based, learning-based, relative positional encoding) are not discussed in this lecture.
\[ y^1 = \sum_i \hat{A}_{1,i} V^i \]

Positional encoding

\[ p_1, p_2, p_3, p_4 \]

\[ x_1, x_2, x_3, x_4 \]
Multi-head self-attention

• Run $h$ attention models in parallel on top of different linearly projected versions of $Q$, $K$, $V$; concatenate and linearly project the results
• Intuition: enables model to attend to different kinds of information at different positions

Transformer block

• A Transformer is a sequence of transformer blocks
  • Vaswani et al.:
    • 12 blocks, 512 embedding dimension, 6 attention heads
  • **Multi-Head Attention**: introduced before
  • **Add & Norm**: residual connection followed by layer normalization
  • **Feedforward (Multi-layer perceptron)**: two linear layers with ReLUs in between, applied independently to each vector
• Attention is the only interaction between inputs

Transformers

Transformers

Transformers

**Transformer Encoder:**
- Self-attention
- Serves as strong feature extracting/enhancing modules.

Transformers

Transformer Encoder:
- Vision Transformer (ViT)

Transformers

Transformers

Transformer Decoder:
- Cross-attention
- Serves as a bridge for sequence-to-sequence translation

Transformers

- Extracting pixel-level representation

**Transformer Encoder:**
- Self-attention
- Serves as strong feature extracting/enhancing modules.

**Transformer Decoder:**
- Cross-attention
- Serves as a bridge for sequence-to-sequence translation

Vision Transformers (ViTs)

ViT: the pioneer of Transformer architectures for vision

[1] Dosovitskiy, et al. An image is worth 16x16 words: Transformers for image recognition at scale. ICLR 2021
Vision Transformers (ViTs)
**ViT: overview**

- Divide an input image into 196 (14x14) small images of size (16x16)
- Treat it as embedding in NLP
- Use it as an input for traditional transformer encoder (like in BERT)
- Use 12 transformer layers (Norm, Multi-head attention, etc.)
- the last output, use it as input for Dense Layer with 1000
- you have a classification model

ViT: performance comparison to CNN

JFT-300M is an internal Google dataset with 300M labeled images.

If you pretrain on JFT and finetune on ImageNet, large ViTs outperform large ResNets.
ViT: strength and weakness

• Strength:
  • Long-range interaction via global attention
  • Strong performance/scalability with large scale of data

• Weakness:
  • Quadratic computational complexity
  • Lacking locality
  • Lacking low-level details due to large patch size (e.g., 16x16)
Key difference between Visual and Text Signals

- **multi-scale** (scale invariance)
- **locality** (spatial smoothness)
- **translation invariance**

I am a fat cat.
I like the green grass.
I am a fat cat. (invalid)

No scale variation
No spatial smoothness
Sensitive to absolute locations

Example adapted from Ross Girshick and Ze Liu
Hierarchical Vision Transformer

- Swin Transformer: Hierarchical Vision Transformer using Shifted Windows
  - Multi-scale (hierarchical) feature map
  - Use shifted window to get image patches

More friendly to visual signals

multi-scale / locality / translation invariance
Hybrid CNN-Transformer Model

• Formulation:
  • Micro-level Hybrid Model
    • Each basic block is designed with convolutional layer and self-attention layers
    • The model stack with several basic blocks repeatedly
  • Macro-level Hybrid Model
    • Each basic block is either pure CNN block or pure Transformer block
    • The model starts with few CNN blocks
    • The model further stacks several Transformer blocks

• Strength
  • Preserve local information due to CNN
  • Preserve translation invariance due to CNN
  • Efficient learning of spatial hierarchy due to CNN
  • Enable global interaction due to Transformer
Examples

• Hybrid Transformer for visual recognition: Yang et al., MOAT, ICLR 2023
• Hybrid Transformer for medical AI: Chen et al., TransUNet, arXiv 2021
• Hybrid Transformer for vision-language: Chen et al., ViTamin, CVPR 2024

Figure 1: **Block comparison.** (a) The MBConv block (Sandler et al., 2018) employs the inverted bottleneck design with depthwise convolution and squeeze-and-excitation (Hu et al., 2018) applied to the expanded features. (b) The Transformer block (Vaswani et al., 2017) consists of a self-attention module and a MLP module. (c) The proposed MOAT block effectively combines them. The illustration assumes the input tensor has channels $c$.

Hybrid Transformer + UNet: TransUNet

- Chen et al. 2021

Figure 3. **Overview of ViTamin architecture.** (a) ViTamin begins with a convolutional stem, followed by Mobile Convolution Blocks (MBConv) in stage 1 and 2, and Transformer Blocks (TFB) in stage 3. The 2D input to the stage 3 is flatten to 1D. For the macro-level designs, the three-stage layout generates the final feature map with output stride 16, similar to ViT/16 [31]. We set channels sizes for the three stages to be $(C, 2C, 6C)$. For the micro-level designs, the employed MBConv-LN modifies MBConv [115] by using a single LayerNorm [4]. TFB-GeGLU upgrades TFB’s FFNs [131] (Feed-Forward Networks) with GELU Gated Linear Units [117]. (b) In the CLIP framework, given $N$ image-text pairs, the vision model’s output $I_i$ is learned to align with its corresponding text Transformer’s output $T_j$. Our text Transformers are the same as OpenCLIP [62]. $+$: Addition. $*$: Multiplication.

Summary

• Transformers consist of several Transformer blocks with multi-head self-attention layer and feedforward layers.
• It is highly scalable and highly parallelizable
• Faster training, larger models, better performance across vision and language tasks
• Good capability in vision tasks.
Dual-Path Transformer: MaX-DeepLab

- Dual Path: Pixel Path + Memory Path
  - Memory to store global information
- Self-attention:
  - P2P: Pixel-to-pixel self-attention
    - Both query and key are pixel vector itself
  - M2M: Memory-to-memory self-attention
    - Both query and key are memory itself
- Cross-attention
  - P2M: Pixel-to-memory cross-attention
    - Query is pixel, key is memory
  - M2P: Memory-to-pixel cross-attention
    - Query is memory, key is pixel
Dual-Path Transformer: MaX-DeepLab

- Dual Path: Pixel Path + Memory Path
- Prediction: $N$ pairs of object class $p^{\text{class}}_j \in \mathbb{R}^1$ and mask $p^{\text{mask}}_j \in \mathbb{R}^{HW}$
  - Memory is decoded to $N$ object class
  - The dot product of Memory and Pixel result in $N$ object masks
Hungarian Matching for Prediction-Groundtruth Association

- **Goal**: match the groundtruth $y_i$ to a prediction $p_j$
  - Ground-truth: $M$ pairs of object class $y_i^{\text{class}} \in \mathbb{R}^1$ and mask $y_i^{\text{mask}} \in \mathbb{R}^{HW}$
  - Prediction: $N$ pairs of object class $p_j^{\text{class}} \in \mathbb{R}^1$ and mask $p_j^{\text{mask}} \in \mathbb{R}^{HW}$
    - Memory is decoded to $N$ object class
    - The dot product of Memory and Pixel result in $N$ object masks

- **Example**:
  - Successful match: $y_i = \{\text{dog}, \text{ } \}; \ p_j = \{\text{dog}, \text{ } \}$
    - Classification accuracy = 1, and mask IoU = 1
  - Failed match: $y_i = \{\text{dog}, \text{ } \}; \ p_j = \{\text{dog}, \text{ } \}$ -> low mask IoU
    - Classification accuracy = 1, but mask IoU = 0.1
  - Failed match: $y_i = \{\text{dog}, \text{ } \}; \ p_j = \{\text{cat}, \text{ } \}$ -> wrong classification
    - Mask IoU=1, but classification accuracy=0
Hungarian Matching for Prediction-Groundtruth Association

- **Goal:** match the groundtruth $y_i$ to a prediction $p_j$
  - Ground-truth: $M$ pairs of object class $y_i^{\text{class}}$ and mask $y_i^{\text{mask}}$
  - Prediction: $N$ pairs of object class $p_j^{\text{class}}$ and mask $p_j^{\text{mask}}$
    - Memory is decoded to $N$ object class
    - The dot product of Memory and Pixel result in $N$ object masks

- **Define cost matrix $C$ of size $M \times N$**
  - $C_{i,j} = -\text{accuracy}(y_i^{\text{class}}, p_j^{\text{class}}) - \text{IoU}(y_i^{\text{mask}}, p_j^{\text{mask}})$

- **Linear assignment problem to do association**
  - Formally, the task is to find a injection $f: \{1,2,\ldots,m\} \to \{1,2,\ldots,n\}$ that minimizes the total assignment cost: Minimize $\sum_{i=1}^{M} C_{i,f(i)}$
Cross-attention for Stereo Matching / Fusion

- The context feature of left and right images are fused with cross-attention
- Query is left context features and key is the right context feature, or vice versa
- Reference to context-enhanced stereo Transformer [1] to estimate disparity

**Context Enhanced Stereo Transformer**

Fig. 3. CSTR consists of two main components: (1) Context Enhanced Path that extracts long-range context information in low resolution feature. (2) Main Matching Path that use Axial-Attention to enhance context and Cross-Attention to compute raw disparity. Then a learnable Up Sampling block up restore the original scale of disparity and Context Adjustment block refines the disparity with context information across epipolar lines conditioned on the left image.

CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

Transformer decoder (cross-attention) is crucial in terms of an end-to-end problem formulation:

CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

Common pipeline:
Common pipeline:

- Image feature extraction through a backbone
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

Common pipeline:

- Image feature extraction through a backbone
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

Common pipeline:

- Image feature extraction through a backbone
- Image feature enhancement through transformer encoder (self-attention)
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

Common pipeline:

- Image feature extraction through a backbone
- Image feature enhancement through transformer encoder (self-attention)
- Learnable object queries aggregate pixel features through transformer decoder (cross-attention) -> box/mask embedding
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

Common pipeline:

- Image feature extraction through a backbone
- Image feature enhancement through transformer encoder (self-attention)
- Learnable object queries aggregate pixel features through transformer decoder (cross-attention) -> box/mask embedding
- Hungarian matching for supervision assignments
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

Common pipeline:

- Image feature extraction through a backbone
- Image feature enhancement through transformer encoder (self-attention)
- Learnable object queries aggregate pixel features through transformer decoder (cross-attention) -> box/mask embedding
- Hungarian matching for supervision assignments
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

Cross-Attention:

- Affinity logits are computed, with linear projections, between queries ($Q^c$) and pixels ($K^p$).

- A spatial-wise (HW) softmax is applied to convert the affinity logits map to attention weights.

- The attention weights are used to retrieve corresponding pixel features, with linear projection:

$$\hat{C} = C + \text{softmax}_{HW}(Q^c \times (K^p)^T) \times V^p,$$

- The update of queries is added in a residual manner.
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

Cross-Attention:

- Affinity logits are computed, with linear projections, between queries ($Q^c$) and pixels ($K^p$)

- A spatial-wise (HW) softmax is applied to convert the affinity logits map to attention weights

- The attention weights are used to retrieve corresponding pixel features, with linear projection

$$\hat{C} = C + \text{softmax}_{HW}(Q^c \times (K^p)^T) \times V^p,$$

- The update of queries is added in a residual manner.
Cross-Attention:

- Affinity logits are computed, with linear projections, between queries \((Q^c)\) and pixels \((K^p)\)

- A spatial-wise (HW) softmax is applied to convert the affinity logits map to attention weights

- The attention weights are used to retrieve corresponding pixel features, with linear projection.

\[
\hat{C} = C + \text{softmax}_{HW}(Q^c \times (K^p)^T) \times V^p,
\]

- The update of queries is added in a residual manner.
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

Cross-Attention:

- Affinity logits are computed, with linear projections, between queries ($Q^c$) and pixels ($K^p$)

- A spatial-wise (HW) softmax is applied to convert the affinity logits map to attention weights

- The attention weights are used to retrieve pixel features, with linear projection:

$$\hat{C} = C + \text{softmax} \left( Q^c \times (K^p)^T \right) \times V^p,$$

- The update of queries is added in a residual manner.
Cross-Attention:

- Affinity logits are computed, with linear projections, between queries ($Q^c$) and pixels ($K^p$).

- A spatial-wise (HW) softmax is applied to convert the affinity logits map to attention weights.

- The attention weights are used to retrieve corresponding pixel features, with linear projection:
  \[
  \hat{C} = C + \text{softmax}\left(\frac{Q^c \times (K^p)^T}{\text{HW}}\right) \times V^p,
  \]

- The update of queries is added in a residual manner.
Cross-Attention:

- Affinity logits are computed, with linear projections, between queries \( Q^c \) and pixels \( K^p \).

- A spatial-wise (HW) softmax is applied to convert the affinity logits map to attention weights.

- The attention weights are used to retrieve corresponding pixel features, with linear projection:

\[
\hat{C} = C + \text{softmax}_{HW}(Q^c \times (K^p)^T) \times V^p,
\]

As a result, the queries will be updated and converted to correspond to a specific object in prediction.

CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

- Inconsistency of using object queries for mask prediction and updating object queries

- Sparse attention map
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

- Inconsistency of using object queries for mask prediction and updating object queries

\[ Z = \text{softmax}\left( F \times C^T \right), \]

mask prediction:

- Sparse attention map
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

- Inconsistency of using object queries for mask prediction and updating object queries

  **mask prediction:**

  \[ Z = \text{softmax}(F \times C^T), \]

  **updating object queries:**

  \[ \hat{C} = C + \text{softmax}(Q_c \times (K^p)^T) \times V^p, \]

- Sparse attention map
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

- Inconsistency of using memory queries for mask prediction and updating memory queries
- Sparse attention map
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

\[
\hat{C} = C + \text{softmax}(Q^c \times (K^p)^T) \times V^p,
\]

Cross-attention -> a clustering process

object queries -> cluster centers

attention map -> clustering assignment

updating object queries -> updating clustering center
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

\[ \hat{C} = C + \text{softmax} \left( \frac{Q^c}{\text{HW}} \times (K^p)^T \right) \times V^p, \]

Cross-attention -> a clustering process

object queries -> cluster centers

attention map -> clustering assignment

updating object queries -> updating clustering center
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

\[
\hat{C} = C + \text{softmax}_{HW}(Q^c \times (K^p)^T) \times V^p,
\]

Cross-attention \(
\rightarrow \text{a clustering process}
\)

Object queries \(
\rightarrow \text{cluster centers}
\)

Attention map \(
\rightarrow \text{clustering assignment}
\)

Updating object queries \(
\rightarrow \text{updating clustering center}
\)
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

\[ \hat{C} = C + \text{softmax}(Q^c \times (K^p)^T) \times V^p, \]

Cross-attention -> a clustering process

object queries -> cluster centers

attention map -> clustering assignment

updating object queries -> updating clustering center
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

\[
\hat{C} = C + \text{softmax}_{HW}(Q^c \times (K^p)^T) \times V^p,
\]

**Machine Translation:** Each object query corresponds to a word in target language, and it will be assigned to one most affiliated word in source language as its update.

**Panoptic Segmentation:** Each object query corresponds to an object in prediction, and it will be assigned to one most affiliated pixel as its update.
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

\[ \hat{C} = C + \text{softmax}(Q^c \times (K^p)^T) \times V^p, \]

*Machine Translation*: Each object query corresponds to a word in target language, and it will be assigned to one most affiliated word in source language as its update.

*Panoptic Segmentation*: Each object query corresponds to an object in prediction, and it will be assigned to one most affiliated pixel as its update.
Machine Translation: Each object query corresponds to a word in target language, and it will be assigned to one most affiliated word in source language as its update.

Panoptic Segmentation: Each object query corresponds to an object in prediction, and it will be assigned to one most affiliated pixel as its update.

\[
\hat{C} = C + \text{softmax}(Q^c \times (K^p)^T) \times V^p,
\]
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

\[ \hat{C} = C + \text{softmax}_{HW}(Q^c \times (K^p)^T) \times V^p, \]

*Machine Translation*: Each object query corresponds to a word in target language, and it will be assigned to one most affiliated word in source language as its update.

*Panoptic Segmentation*: Each object query corresponds to an object in prediction, and it will be assigned to one most affiliated pixel as its update.
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

\[
\hat{C} = C + \text{softmax}(Q^c \times (K^p)^T) \times V^p;
\]

\[
\hat{C} = C + (\text{softmax}(\tilde{K}^p \times (\tilde{Q}^c)^T))^T \times V^p.
\]

*Panoptic Segmentation:* Each object query corresponds to an object in prediction, and it will be assigned to one most affiliated pixel as its update.

*Panoptic Segmentation:* Each pixel will choose one most affiliated object, all assigned pixels will serve as an update to corresponding object query.
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

- Include coordinates into the clustering

- Improve instance discrimination loss in MaX-DeepLab to pixel-wise contrastive loss
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

We build CMT-DeepLab upon previous SOTA method MaX-DeepLab:

COCO Panoptic:
- +4.2% val PQ
- +4.4% test PQ

Cityscapes Panoptic:
- +2.9% val PQ
CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

MaX-DeepLab

CMT-DeepLab
Summary

➢ A novel clustering view to better understand and design Transformer modules for object-centric representation learning

➢ We introduce CMT-DeepLab, which unifies cross-attention and panoptic segmentation from a clustering perspective
Overview

Extracting pixel-level representation:
- Vision Transformer (ViT)

Converting to object-level representation:
- kMaX-DeepLab [3] (ECCV 22)
**k-means Mask Transformer**

Cross-attention:

\[
\hat{C} = C + \text{softmax}\left(\frac{Q^c}{\sqrt{d_k}} \times (K^p)^T \right) \times V^p,
\]

Yu., et al. k-means Mask Transformer. ECCV 2022.
**k-means Mask Transformer**

Cross-attention:

\[
\hat{C} = C + \text{softmax}_{HW}(Q^c \times (K^p)^T) \times V^p,
\]
**$k$-means Mask Transformer**

Cross-attention:

\[
\hat{C} = C + \text{softmax}_{HW}(Q^c \times (K^p)^T) \times V^p,
\]

- Affinity logits with linear projections
- Spatial-wise softmax for attention weight
**k-means Mask Transformer**

Cross-attention:

\[
\hat{C} = C + \text{softmax}_{HW}(Q^c \times (K^p)^T) \times V^p,
\]

Affinity logits with linear projections

Spatial-wise softmax for attention weight

Retrieve update values with linear projections
**k-means** Mask Transformer

Cross-attention:

\[
\hat{C} = C + \text{softmax}_{HW}(Q^c \times (K^p)^T) \times V^p,
\]

- Affinity logits with linear projections
- Spatial-wise softmax for attention weight
- Retrieve update values with linear projections
- Residual update of object query
**k-means Mask Transformer**

k-means clustering algorithm:

\[
A = \arg\max_{N} (C \times P^T),
\]

\[
\hat{C} = A \times P,
\]
**k-means** Mask Transformer

k-means clustering algorithm:

$$A = \arg\max_N (C \times P^T),$$

$$\hat{C} = A \times P,$$

Affinity logits
*k*-means Mask Transformer

k-means clustering algorithm:

\[ A = \underset{N}{\text{argmax}} (C \times P^T), \]

\[ \hat{C} = A \times P, \]

Affinity logits \rightarrow Cluster-wise argmax for clustering assignment
**k-means Mask Transformer**

k-means clustering algorithm:

\[ A = \underset{N}{\text{argmax}} (C \times P^T), \]

\[ \hat{C} = A \times P, \]

Affinity logits \rightarrow Cluster-wise argmax for clustering assignment \rightarrow Retrieve update values
**k-means Mask Transformer**

k-means clustering algorithm:

\[
A = \arg\max_N (C \times P^T),
\]

\[
\hat{C} = A \times P,
\]

- **Affinity logits**
- **Cluster-wise argmax for clustering assignment**
- **Retrieve update values**
- **Replace cluster centers**
**k-means Mask Transformer**

cross-attention v.s. k-means clustering algorithm:

- Affinity logits with linear projections
- Spatial-wise softmax for attention weight
- Cluster-wise argmax for clustering assignment

Retrieve update values with linear projections

Residual update of object query

Retrieve update values

Replace cluster centers
**k-means Mask Transformer**

cross-attention v.s. k-means clustering algorithm:

- Affinity logits with linear projections
- Spatial-wise softmax for attention weight
- Cluster-wise argmax for clustering assignment
- Retrieve update values with linear projections
- Residual update of object query
- Replace cluster centers

Linear projections
Cross-attention v.s. \textit{k-means} clustering algorithm:

- **Affinity logits** with linear projections
- **Spatial-wise softmax for attention weight**
- **Cluster-wise argmax for clustering assignment**
- **Retrieve update values with linear projections**
- **Residual update of object query**
- **Retrieve update values**
- **Replace cluster centers**

**k-means Mask Transformer**

Linear projections

Residual update
**k-means Mask Transformer**

cross-attention v.s. k-means clustering algorithm:

- **Affinity logits**
  - with linear projections

- **Spatial-wise softmax**
  - for attention weight

- **Cluster-wise argmax**
  - for clustering assignment

- **Retrieve update values**
  - with linear projections

- **Residual update of object query**

- **Affinity logits**

- **Linear projections**

- **Residual update**

- **Cluster-wise argmax**
  - v.s.
  - Spatial-wise softmax
**k-means Mask Transformer**

cross-attention v.s. k-means clustering algorithm:

- Affinity logits with linear projections
  - Spatial-wise softmax for attention weight
  - Cluster-wise argmax for clustering assignment

- Retrieve update values with linear projections
  - Residual update of object query
  - Replace cluster centers

- Linear projections
  - Residual update

Cluster-wise argmax v.s. Spatial-wise
**k-means Mask Transformer**

A simple *change* for k-means cross-attention:

\[
\hat{C} = C + \frac{\text{softmax}}{HW}(Q^c \times (K^p)^T) \times V^p,
\]

\[
\hat{C} = C + \frac{\text{argmax}}{N}(Q^c \times (K^p)^T) \times V^p.
\]
A simple *change* for k-means cross-attention:
**k-means Mask Transformer**

A simple *change* for k-means cross-attention:

<table>
<thead>
<tr>
<th>pixel-cluster interaction module</th>
<th>ResNet-50</th>
<th></th>
<th></th>
<th>MaX-S</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>params</td>
<td>FLOPs</td>
<td>PQ</td>
<td>params</td>
<td>FLOPs</td>
<td>PQ</td>
</tr>
<tr>
<td>cross-attention [89]</td>
<td>56M</td>
<td>165G</td>
<td>47.5</td>
<td>73M</td>
<td>237G</td>
<td>52.0</td>
</tr>
<tr>
<td>dual-path cross-attention [92]</td>
<td>58M</td>
<td>175G</td>
<td>48.0</td>
<td>75M</td>
<td>247G</td>
<td>52.3</td>
</tr>
<tr>
<td>k-means cross-attention</td>
<td>57M</td>
<td>168G</td>
<td>52.7</td>
<td>74M</td>
<td>240G</td>
<td>56.1</td>
</tr>
<tr>
<td>dual-path <em>k</em>-means cross-attention</td>
<td>59M</td>
<td>176G</td>
<td>53.0</td>
<td>76M</td>
<td>248G</td>
<td>56.2</td>
</tr>
</tbody>
</table>
**k-means Mask Transformer**

A simple change for k-means cross-attention:

<table>
<thead>
<tr>
<th>Pixel-cluster interaction module</th>
<th>ResNet-50</th>
<th>MaX-S</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>params</td>
<td>FLOPs</td>
</tr>
<tr>
<td>cross-attention [89]</td>
<td>56M</td>
<td>165G</td>
</tr>
<tr>
<td>dual-path cross-attention [92]</td>
<td>58M</td>
<td>175G</td>
</tr>
<tr>
<td><strong>k-means cross-attention</strong></td>
<td><strong>57M</strong></td>
<td><strong>168G</strong></td>
</tr>
<tr>
<td>dual-path k-means cross-attention</td>
<td>59M</td>
<td>176G</td>
</tr>
</tbody>
</table>

5.2% (47.5% -> 52.7%) PQ improvement with one change and negligible extra cost
k-means Mask Transformer
### $k$-means Mask Transformer

<table>
<thead>
<tr>
<th>Method</th>
<th>params</th>
<th>FLOPs</th>
<th>FPS</th>
<th>PQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaX-DeepLab</td>
<td>451M</td>
<td>3692G</td>
<td>-</td>
<td>51.1% (-6.9%)</td>
</tr>
<tr>
<td>MaskFormer</td>
<td>212M</td>
<td>792G</td>
<td>5.2</td>
<td>52.7% (-5.3%)</td>
</tr>
<tr>
<td>K-Net</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>54.6% (-3.4%)</td>
</tr>
<tr>
<td>CMT-DeepLab</td>
<td>270M</td>
<td>1114G</td>
<td>3.2</td>
<td>55.3% (-2.7%)</td>
</tr>
<tr>
<td>kMaX-DeepLab</td>
<td>232M</td>
<td>749G</td>
<td>6.6</td>
<td>58.0%</td>
</tr>
</tbody>
</table>

COCO val set
**k-means Mask Transformer**

<table>
<thead>
<tr>
<th>Method</th>
<th>params</th>
<th>FLOPs</th>
<th>FPS</th>
<th>PQ</th>
<th>AP\text{mask}</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Panoptic-DeepLab</td>
<td>47M</td>
<td>548G</td>
<td>5.7</td>
<td>63.0% (-5.4%)</td>
<td>35.3% (-8.7%)</td>
<td>80.5% (-3.0%)</td>
</tr>
<tr>
<td>Axial-DeepLab</td>
<td>173M</td>
<td>2447G</td>
<td>-</td>
<td>64.4% (-4.0%)</td>
<td>36.7% (-7.3%)</td>
<td>80.6% (-2.9%)</td>
</tr>
<tr>
<td>SWideRNet</td>
<td>536M</td>
<td>10365G</td>
<td>1.0</td>
<td>66.4% (-2.0%)</td>
<td>40.1% (-3.9%)</td>
<td>82.2% (-1.3%)</td>
</tr>
<tr>
<td>kMaX-DeepLab</td>
<td>232M</td>
<td>1673G</td>
<td>3.1</td>
<td>68.4%</td>
<td>44.0%</td>
<td>83.5%</td>
</tr>
</tbody>
</table>

Cityscapes \textit{val} set
**k-means** Mask Transformer

![Image of k-means Mask Transformer process]

- **Image**
- **1st cluster assignment**
- **2nd cluster assignment**
- **3rd cluster assignment**
- **4th cluster assignment**
- **5th cluster assignment**
- **6th cluster assignment**
- **Panoptic prediction**
- **Panoptic label**
**k-means** Mask Transformer

![Image of k-means Mask Transformer](image)

- **Image**
- **1st cluster assignment**
- **2nd cluster assignment**
- **3rd cluster assignment**
- **4th cluster assignment**
- **5th cluster assignment**
- **6th cluster assignment**
- **Panoptic prediction**
- **Panoptic label**
**k-means** Mask Transformer
Summary

➢ Discuss the underlying similarity between cross-attention and k-means clustering algorithm.

➢ Propose k-means cross-attention, which designs cross-attention as a k-means clustering module, leading to better object-centric representation.

➢ A simple change on activation function with SOTA performance.