

Vision Transformer & Meta-architecture

2023.8.15 高鑫

Transformer-Based Visual Segmentation: A Survey

Xiangtai Li, Henghui Ding, Wenwei Zhang, Haobo Yuan, Jiangmiao Pang,
Guangliang Cheng, Kai Chen, Ziwei Liu, Chen Change Loy

— 1

Transformer

Self-attention, cross-attention, positional embedding, inductive bias

— 2

Meta-architecture

A unified model for classification, segmentation and detection

— 3

Methods in three aspects

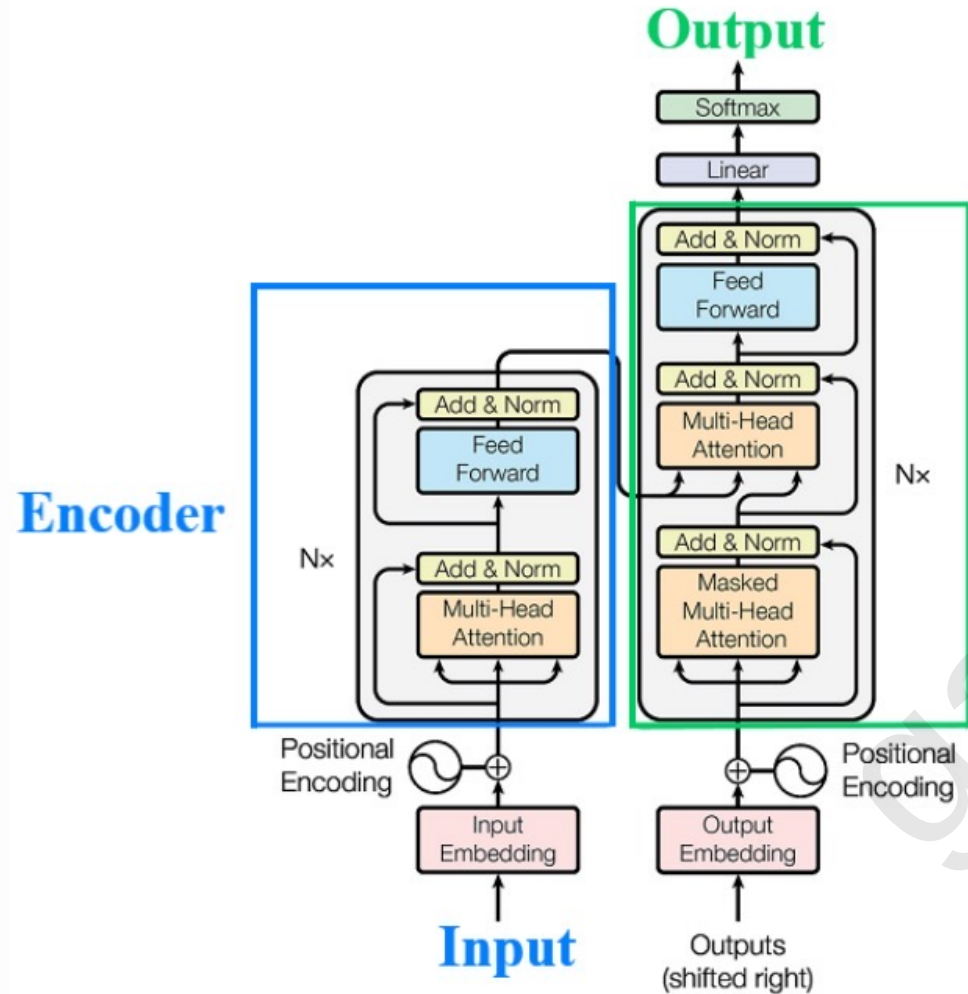
Backbone, decoder, object query

— 4

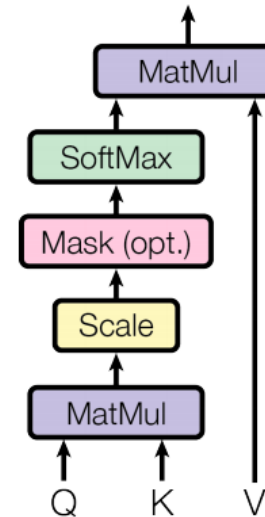
Future directions

Six future directions and possible applications

Think Again about Transformer



Scaled Dot-Product Attention

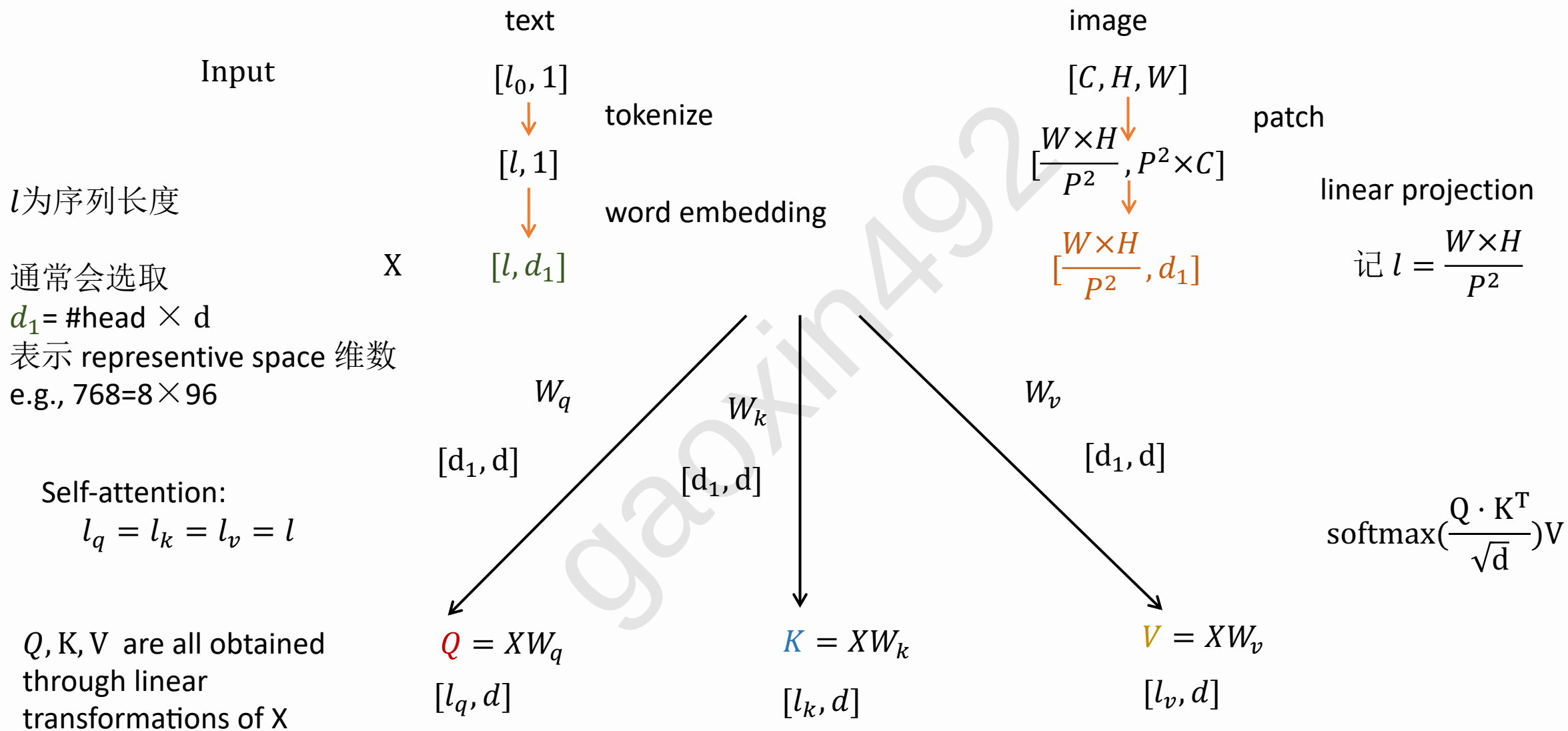


$$SA(X) = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d}}\right)V$$

Key point: Attention Mechanism

What do Q, K, and V actually represent?

Think Again about Transformer



先理解 $\text{softmax}(\frac{X \cdot X^T}{\sqrt{d}})X$

$$X = \begin{bmatrix} x_{11} & \dots & x_{1d} \\ \vdots & \ddots & \vdots \\ x_{l1} & \dots & x_{ld} \end{bmatrix} = [x_1, \dots, x_l]^T \in \mathbb{R}^{l \times d}, \text{ 其中 } x_i \in \mathbb{R}^d, i = 1, \dots, l$$

$$XX^T = \begin{bmatrix} x_1 \\ \vdots \\ x_l \end{bmatrix} [x_1 \quad \dots \quad x_l] = \begin{bmatrix} x_1 x_1^T & \dots & x_1 x_l^T \\ \vdots & \ddots & \vdots \\ x_l x_1^T & \dots & x_l x_l^T \end{bmatrix} \in \mathbb{R}^{l \times l}$$

x_1 与 $x_1, x_2, \dots, x_l$ 的相似度

某一token与其它token的相似度

两个向量 a, b 的内积 $ab^T = |a||b|\cos \angle a, b = \begin{cases} > 0, & \text{锐角 同向时最大} \\ = 0, & \text{正交} \\ < 0, & \text{钝角 反向时最小} \end{cases}$

向量 a 在 b 上的投影，表示两向量之间的相似度

$[x_1 x_1^T \quad \dots \quad x_1 x_l^T],$ 记 $z_i = \frac{x_1 x_i^T}{\sqrt{d}}$

$$\begin{bmatrix} z_1 & \dots & z_l \end{bmatrix} \xrightarrow{\text{softmax}} \begin{bmatrix} \frac{e^{z_1}}{\sum e^{z_i}} & \dots & \frac{e^{z_l}}{\sum e^{z_i}} \end{bmatrix}$$

Attention matrix

$$A = \text{softmax}\left(\frac{X \cdot X^T}{\sqrt{d}}\right) \Rightarrow \text{归一化后, 变成weight}$$

	机器学习			
机器学习	0.4	0.4	0.1	0.1
	0.4	0.5	0.1	0.0
	0.2	0.0	0.4	0.4
	0.1	0.1	0.3	0.5

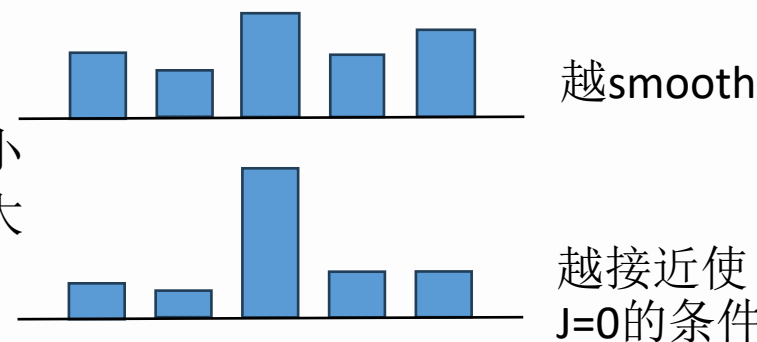
$$A = \text{softmax}\left(\frac{X \cdot X^T}{\sqrt{d}}\right) \quad \text{为什么要除以}\sqrt{d}?$$

$$\mathbb{R}^{l_k} \rightarrow \mathbb{R}^{l_k}$$

$$\text{Softmax: } [z_1 \dots z_{l_k}] \rightarrow [s_1 \dots s_{l_k}], \quad s_i = \frac{e^{z_i}}{\sum e^{z_i}}, i = 1, \dots, l_k \quad \text{对 Softmax 求导数}$$

$$\text{Jacobian matrix: } J = \begin{bmatrix} s_1(1-s_1) & -s_1s_2 & \dots & -s_1s_{l_k} \\ -s_2s_1 & s_2(1-s_2) & \dots & -s_2s_{l_k} \\ \vdots & \vdots & \ddots & \vdots \\ -s_{l_k}s_1 & -s_{l_k}s_2 & \dots & s_{l_k}(1-s_{l_k}) \end{bmatrix}, \quad J = 0 \Leftrightarrow s_i = 1, s_{j \neq i} = 0, i \text{ in } [1, \dots, l_k]$$

$$z_i \times \text{scaling factor}, i = 1, \dots, l_k \Rightarrow \left[\frac{e^{z_1}}{\sum e^{z_i}} \dots \frac{e^{z_{l_k}}}{\sum e^{z_i}} \right] \begin{cases} \text{scaling factor} < 1, \text{越小} \\ \text{scaling factor} > 1, \text{越大} \end{cases}$$



$X \cdot X^T$ 中的元素较大时，会使得softmax出现梯度消失，所以乘上一个scaling factor，乘什么呢？

假设两个独立的随机向量 X, Y ，向量中每个元素均值为0，方差为1

$$D(XY^T) = D(x_1y_1 + x_2y_2 + \dots + x_dy_d) = D(x_1y_1) + D(x_2y_2) + \dots + D(x_dy_d) = d$$

除以 $\sqrt{d}$ 对方差起到归一化作用

⁴To illustrate why the dot products get large, assume that the components of q and k are independent random variables with mean 0 and variance 1. Then their dot product, $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$, has mean 0 and variance d_k .

$$A = \text{softmax}\left(\frac{X \cdot X^T}{\sqrt{d}}\right) \in \mathbb{R}^{l_q \times l_k}, \quad AX \in \mathbb{R}^{l_q \times d}$$

$$AX = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1l_k} \\ a_{21} & a_{22} & \dots & a_{2l_k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{l_q1} & a_{l_q2} & \dots & a_{l_q l_k} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{l_v} \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + \dots + a_{1l_k}x_{l_v} \\ a_{21}x_1 + \dots + a_{2l_k}x_{l_v} \\ \vdots \\ a_{l_q1}x_1 + \dots + a_{l_q l_k}x_{l_v} \end{bmatrix}, \quad \begin{array}{l} \text{(必须满足 } l_k = l_v, \ l_q \text{ 可以不等)} \\ \text{(Self attention } l_k = l_v = l_q) \end{array}$$

$\text{softmax}\left(\frac{X \cdot X^T}{\sqrt{d}}\right)X$ 得到的结果

每一个元素都变成原来序列中所有token的加权和

$$\begin{bmatrix} \text{机} \\ \text{器} \\ \text{学} \\ \text{习} \end{bmatrix} \Rightarrow \begin{bmatrix} 0.4\text{机} + 0.4\text{器} + 0.1\text{学} + 0.1\text{习} \\ \vdots \\ \vdots \end{bmatrix}$$

即每个元素变成了与原token高度语义相关的token组合

那如果希望得到原token之后可能的下一个token呢？

或者其它映射规则？

增加可学习参数，
拟合映射规则

$$\text{softmax}\left(\frac{X \cdot X^T}{\sqrt{d}}\right)X$$



$$\text{softmax}\left(\frac{XW_q \cdot (XW_k)^T}{\sqrt{d}}\right)XW_v$$

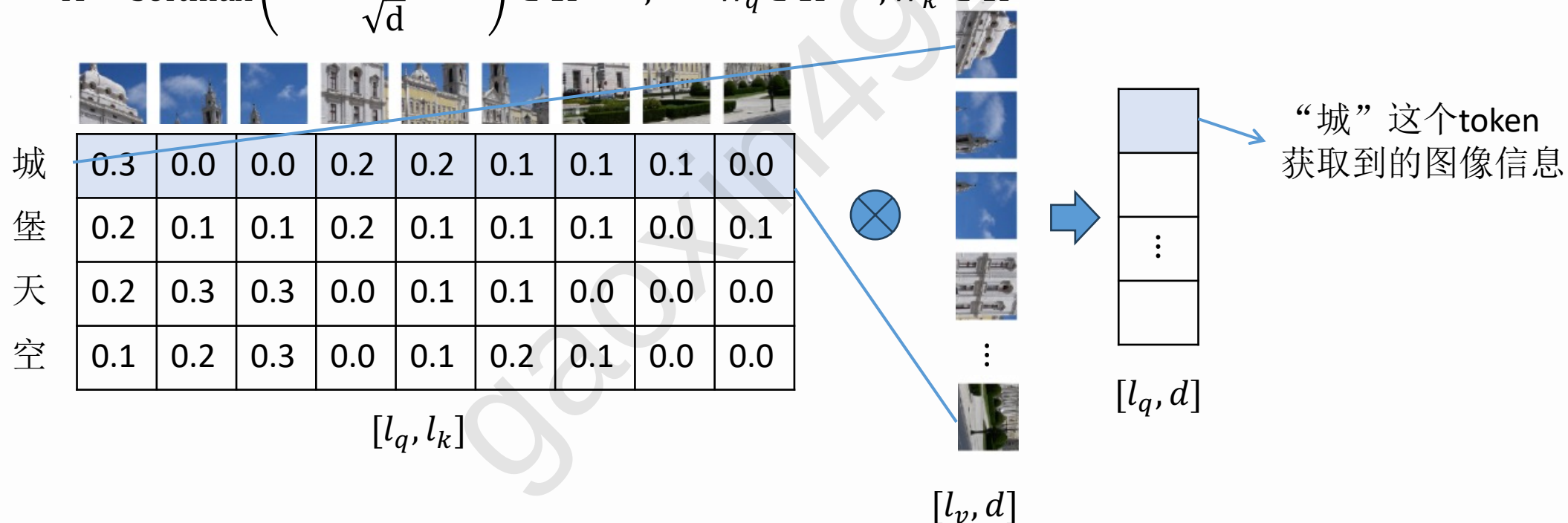
Self-attention in Transformer

Cross Attention

对于 $X \in \mathbb{R}^{l_q \times d}$, $Y \in \mathbb{R}^{l_k \times d}$, $l_k = l_v$,

X, Y 一般是不同模态的两个序列, 为了保证 $l_k = l_v$, K 和 V 一般都由 Y 得到

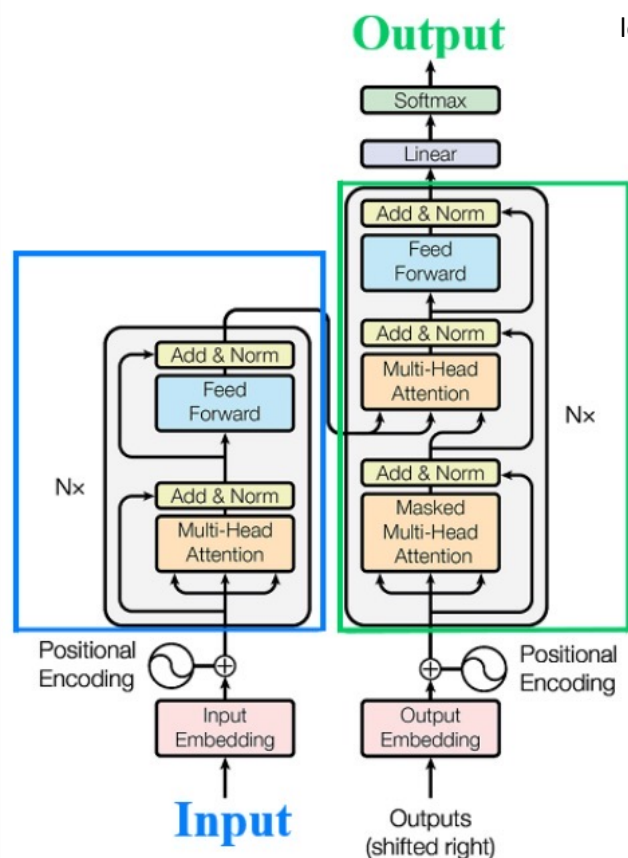
$$A = \text{softmax} \left(\frac{XW_q \cdot (YW_k)^T}{\sqrt{d}} \right) \in \mathbb{R}^{l_q \times l_k}, \quad W_q \in \mathbb{R}^{d \times d}, W_k \in \mathbb{R}^{d \times d}$$



$$\text{Output} = A(YW_v) \in \mathbb{R}^{l_q \times d}, \quad W_v \in \mathbb{R}^{d \times d}$$

Masked Attention

$$\text{mask} \odot \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d}}\right)V$$



	<s>	machine	learning
<s>	0.5	0.4	0.1
machine	0.4	0.5	0.1
learning	0.2	0.4	0.4

样本：翻译成英文：机器学习 machine learning

- Training

Input

Output(shifted right)

Output

Ground Truth

翻译成英文：机器学习

<s> machine learning
machine learning <ed>
machine learning <ed>

- Testing

1. Input

Output(shifted right)

Output

翻译成英文：机器学习

<s>
machine

2. Input

Output(shifted right)

Output

翻译成英文：机器学习

<s> machine
machine learning

3. Input

Output(shifted right)

Output

翻译成英文：机器学习

<s> machine learning
machine learning <ed>

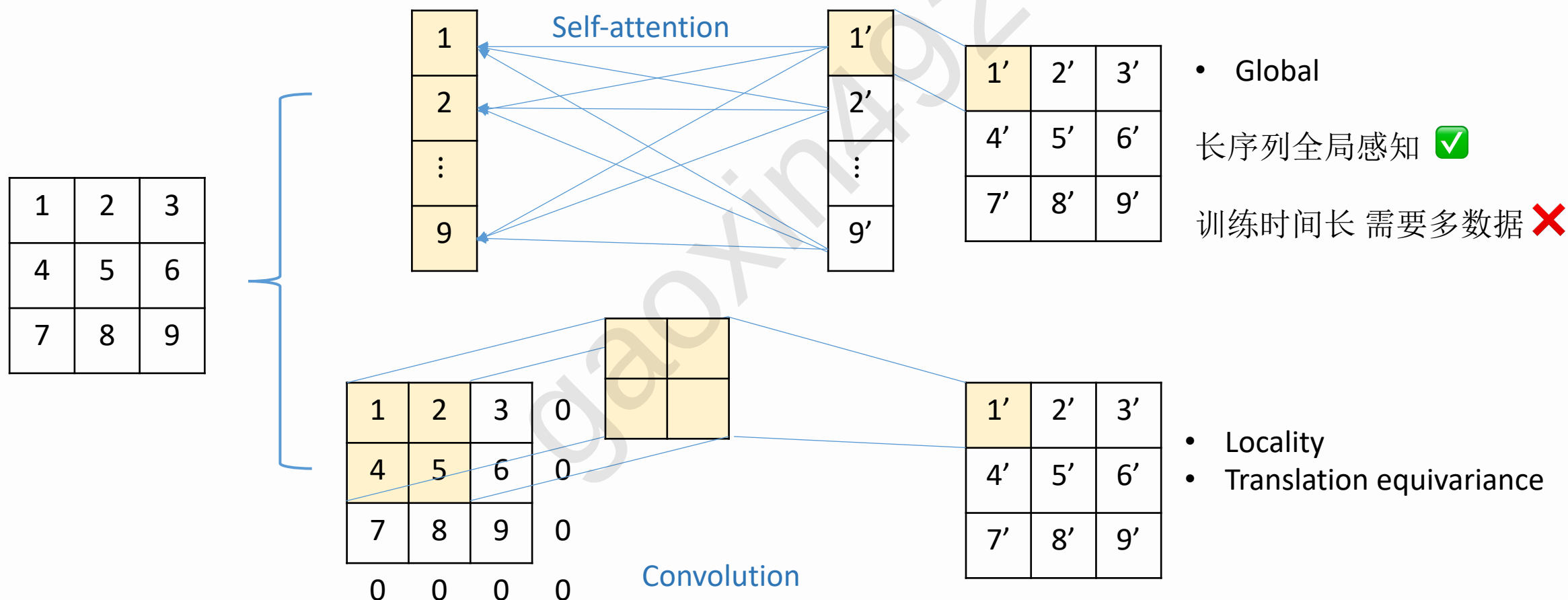
输出遇到<ed>，终止token-by-token的预测过程

	<s>	machine
<s>	0.5	
<s>	0.5	0.4
machine	0.4	0.5

Causal Mask

Image-specific Inductive bias

“We note that Vision Transformer has **much less image-specific inductive bias** than CNNs. In CNNs, **locality**, two-dimensional neighborhood structure, and **translation equivariance** are baked into each layer throughout the whole model. In ViT, only MLP layers are local and translationally equivariant, while the self-attention layers are global.”



Positional embedding in NLP Transformer and ViT

Transformer 位置不敏感(Position-insensitive)

1, 2, 3, 4,? 实际问题中可能比训练集的句子还要长, 会使得模型泛化性能较差

1. 为每个时间步 (token在序列中的位置) 输出唯一的编码;
2. 即便序列长度不一, 序列中两个时间步之间的距离应该是“恒定”的;
3. 模型可以轻易泛化到更长的序列上。

$P = [p_1, \dots, p_l]^T \in \mathbb{R}^{l \times d}$, 其中 $p_i = [p_{i1}, \dots, p_{id}]^T \in \mathbb{R}^d, i = 1, \dots, l$, 是一个token的位置编码

$$p_{ij} = \begin{cases} \sin(\omega_k \cdot i), & j = 2k \\ \cos(\omega_k \cdot i), & j = 2k + 1 \end{cases} \quad j = 0, \dots, d-1, \quad \omega_k = \frac{1}{T^{\frac{2k}{d}}}, \quad k = 0, 1, \dots, \frac{d}{2} - 1, T = 10000$$

$$p_i = [\sin(i), \cos(i), \sin\left(\frac{i}{10000^{\frac{2}{d}}}\right), \cos\left(\frac{i}{10000^{\frac{2}{d}}}\right), \dots, \sin\left(\frac{i}{10000^{\frac{d-2}{d}}}\right), \cos\left(\frac{i}{10000^{\frac{d-2}{d}}}\right)]^T$$

Think Again about Transformer

令 $d = 4, T = 4$, 序列的位置编码:

$$\omega_k = \frac{1}{T \frac{2k}{d}}$$

Token 0	$\sin(\frac{0}{4^{0/4}})$	$\cos(\frac{0}{4^{0/4}})$	$\sin(\frac{0}{4^{2/4}})$	$\cos(\frac{0}{4^{2/4}})$
Token 1	$\sin(\frac{1}{4^{0/4}})$	$\cos(\frac{1}{4^{0/4}})$	$\sin(\frac{1}{4^{2/4}})$	$\cos(\frac{1}{4^{2/4}})$
Token 2	$\sin(\frac{2}{4^{0/4}})$	$\cos(\frac{2}{4^{0/4}})$	$\sin(\frac{2}{4^{2/4}})$	$\cos(\frac{2}{4^{2/4}})$
Token 3	$\sin(\frac{3}{4^{0/4}})$	$\cos(\frac{3}{4^{0/4}})$	$\sin(\frac{3}{4^{2/4}})$	$\cos(\frac{3}{4^{2/4}})$

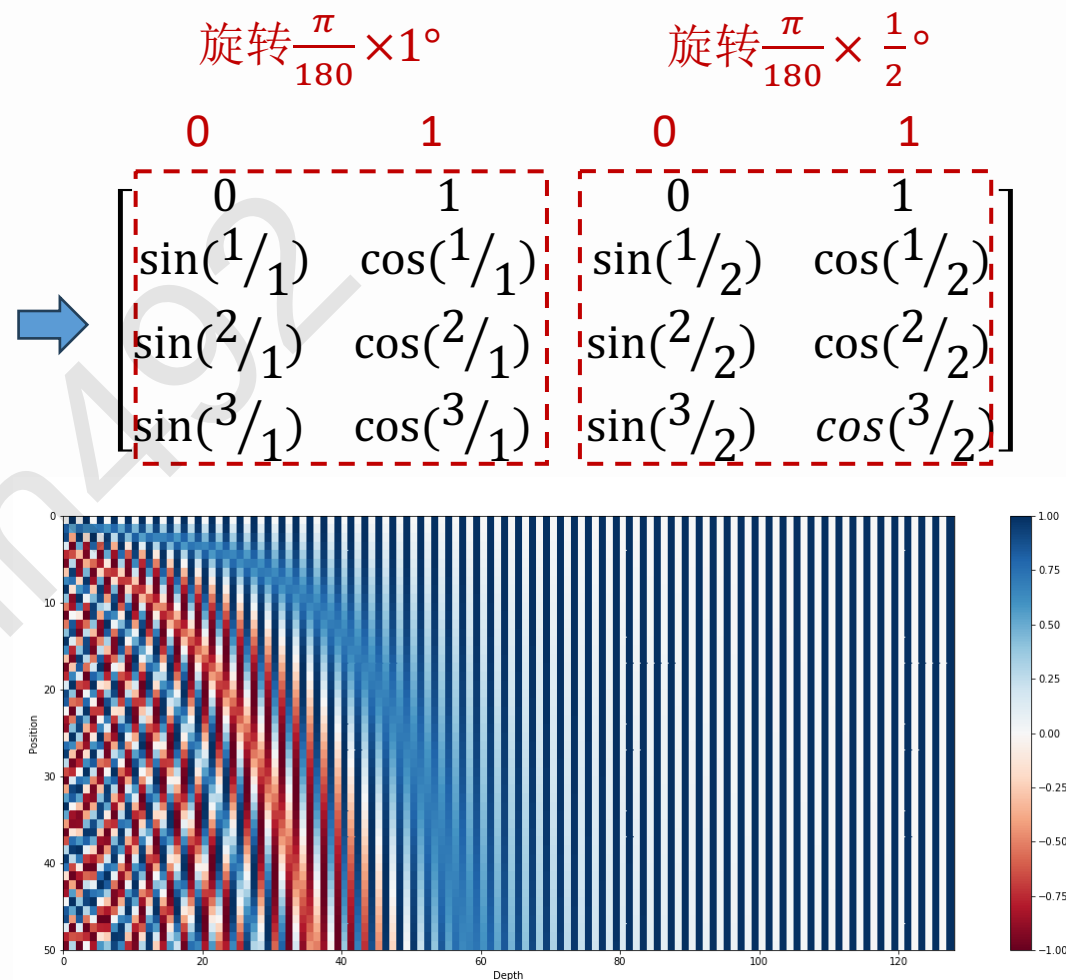
...

Token 2π	$\sin(\frac{2\pi}{4^{0/4}})$	$\cos(\frac{2\pi}{4^{0/4}})$	$\sin(\frac{2\pi}{4^{2/4}})$	$\cos(\frac{2\pi}{4^{2/4}})$
--------------	------------------------------	------------------------------	------------------------------	------------------------------

...

最小的周期是 2π , 最大是 $2 \times 2\pi$
 整个位置编码的周期是 $2 \times 2\pi$

Token $2 \times 2\pi$	$\sin(\frac{4\pi}{4^{0/4}})$	$\cos(\frac{4\pi}{4^{0/4}})$	$\sin(\frac{4\pi}{4^{2/4}})$	$\cos(\frac{4\pi}{4^{2/4}})$
-----------------------	------------------------------	------------------------------	------------------------------	------------------------------



- Position 周期 $T \times \pi$, 避免训练和测试的序列长度有差别, 提升泛化能力
- 位置信息基本储存在前面, 随着 depth 加深, 位置信息越来越少

Think Again about Transformer

- We chose this function because we hypothesized it would **allow the model to easily learn to attend by relative positions**, since for any fixed offset s , p_{i+s} can be represented as a **linear function** of p_i .

$$\begin{bmatrix} \sin(\omega_k(i+s)) \\ \cos(\omega_k(i+s)) \end{bmatrix} = \begin{bmatrix} \cos(\omega_k s) & \sin(\omega_k s) \\ -\sin(\omega_k s) & \cos(\omega_k s) \end{bmatrix} \begin{bmatrix} \sin(\omega_k i) \\ \cos(\omega_k i) \end{bmatrix}$$

p_{i+s} 和 p_i 位置向量的内积仅与相对位置有关

$$\begin{aligned} p_{i+s} \cdot p_i &= \sum_{k=0}^{\frac{d}{2}-1} \sin(\omega_k \cdot i) \cdot \sin(\omega_k(i+s)) + \cos(\omega_k \cdot i) \cdot \cos(\omega_k(i+s)) \\ &= \sum_{k=0}^{\frac{d}{2}-1} \cos(\omega_k(i - (i+s))) \\ &= \sum_{k=0}^{\frac{d}{2}-1} \cos(\omega_k s), \quad \text{where } \omega_k = \frac{1}{10000^{2k/d}} \end{aligned}$$

p_{i+s} 和 p_i 位置向量的欧式距离仅与相对位置有关

$$\begin{aligned} & \|p_{i+s} - p_i\|^2 \\ &= \sum_{k=0}^{\frac{d}{2}-1} (\sin(\omega_k \cdot i) - \sin(\omega_k(i+s)))^2 + (\cos(\omega_k \cdot i) - \cos(\omega_k(i+s)))^2 \\ &= 4 \sum_{k=0}^{\frac{d}{2}-1} [\cos(\omega_k(\frac{2i+s}{2})) \cdot \sin(\omega_k(\frac{s}{2}))]^2 + [\sin(\omega_k(\frac{2i+s}{2})) \cdot \sin(\omega_k(\frac{s}{2}))]^2 \\ &= 4 \sum_{k=0}^{\frac{d}{2}-1} \sin^2(\omega_k \cdot \frac{s}{2}), \quad \text{where } \omega_k = \frac{1}{10000^{2k/d}} \end{aligned}$$

- Position获取到了信息

$$QK^T = (XW_q)(YW_k)^T = XW_qW_k^T Y^T$$

表示X每个token与Y中所有token的联系

$$((X + p_1)W_q)((Y + p_2)W_k)^T = XW_qW_k^T Y^T + XW_qW_k^T p_2^T + p_1W_qW_k^T Y^T + p_1W_qW_k^T p_2^T$$

- 表示X每个token与Y中所有位置的联系
- 表示X每个位置与Y中所有token的联系
- 表示X每个位置与Y中所有位置的联系

ViT: Learnable Positional Embedding

```
self.pos_embedding = nn.Parameter(torch.randn(1, num_patches+1, dim))
```

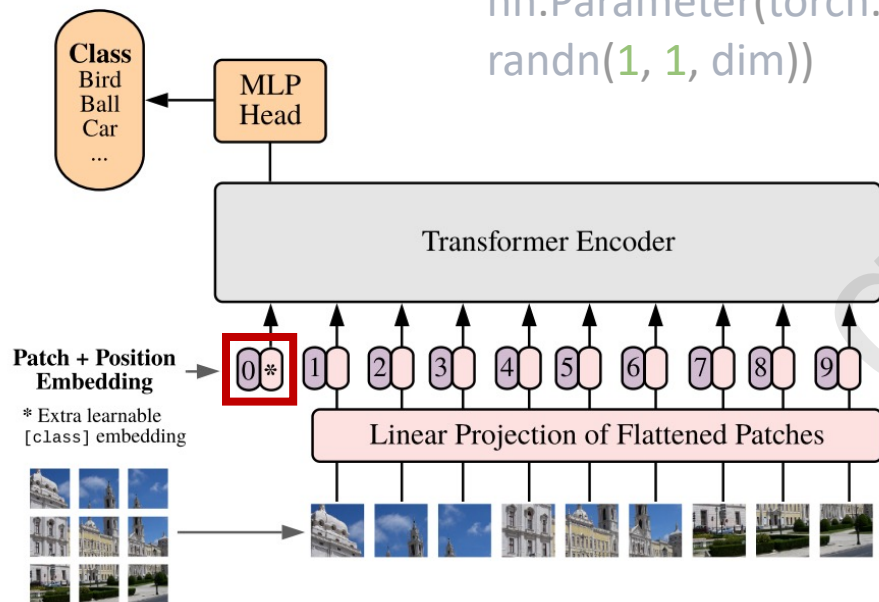
或者使用 `nn.Embedding`（将一个数字映射为指定维度的向量，其`weight`是可学习的参数）

```
self.pos_embed = nn.Embedding(num_patches+1, dim)
self.pos_embedding=self.pos_embed(range(num_patches+1)).weight
```

- 图像的序列长度基本固定
- 希望相对位置编码可以学习到图像的 inductive bias

Learnable cls token

```
self.cls_token =
nn.Parameter(torch.
randn(1, 1, dim))
```

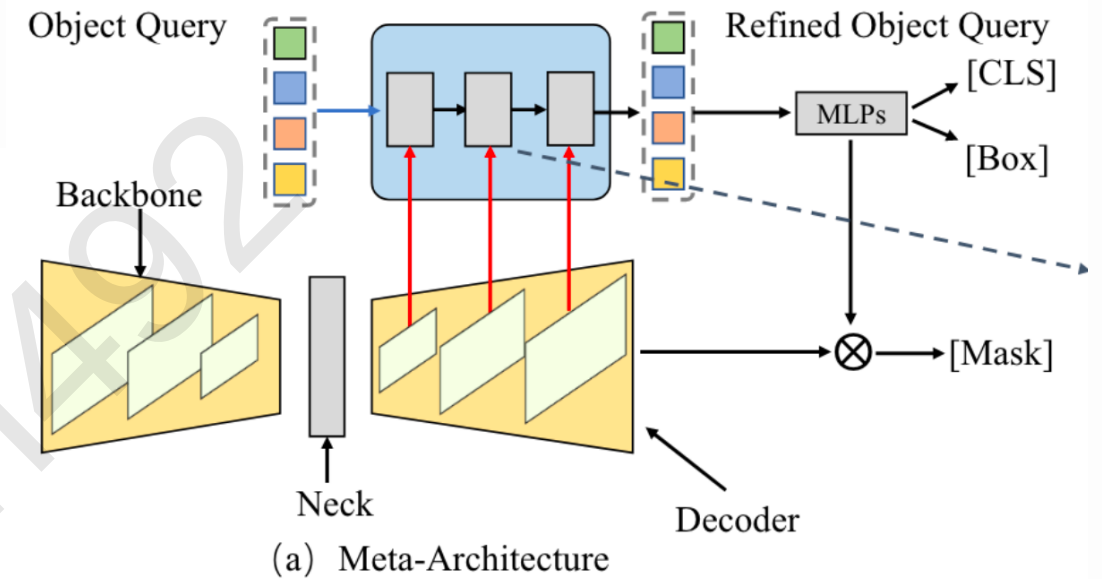
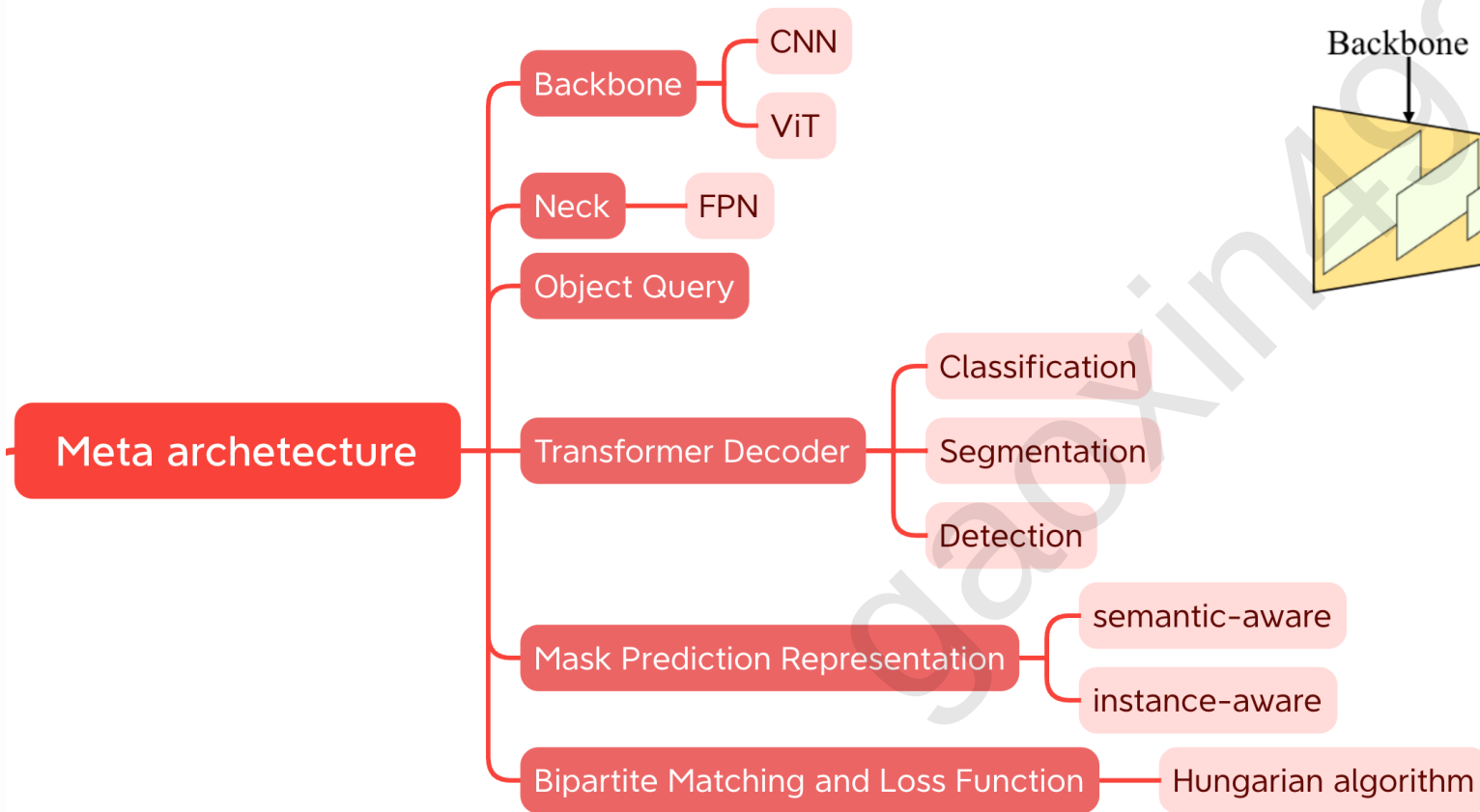


learnable query

用固定长度的Query序列对Key和Value序列采样是可学习的参数，随梯度下降更新

q1	0.3	0.0	0.0	0.2	0.2	0.1	0.1	0.1	0.0	Attention matrix between Q and Image
q2	0.2	0.3	0.3	0.0	0.1	0.1	0.0	0.0	0.0	
q3	0.0	0.0	0.0	0.0	0.0	0.1	0.4	0.3	0.2	
q4	0.2	0.1	0.1	0.2	0.1	0.1	0.1	0.0	0.1	

Meta architecture



How to do all the vision tasks in a unified model?

- Classification
- Segmentation(semantic, instance, panoptic)
- Detection

Meta architecture

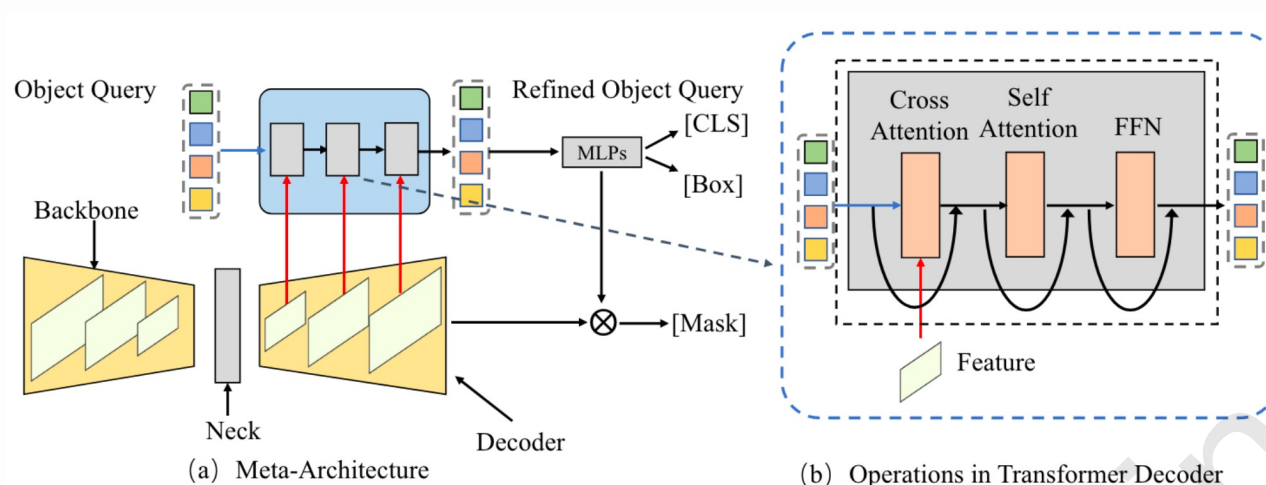


Fig. 3: Illustration of (a) meta-architecture and (b) common operations in the decoder.

- Neck

FPN(Feature Pyramid Network): Compute a multi-scale feature representation

$$F_1: \left[\frac{W}{P}, \frac{H}{P}, d \right] \Rightarrow F_2: [W', H', d] \Rightarrow F_3: [W, H, d]$$

- Backbone

$$[C, H, W] \Rightarrow \left[\frac{W \times H}{p^2}, d \right]$$

可以在序列前加上一个 **learnable cls token**,
变成 $\left[\frac{W \times H}{p^2} + 1, d \right]$

Backbone 输出的 **cls token** ($[1, d]$) 经过 MLP 进行分类, 也可以后面再分类

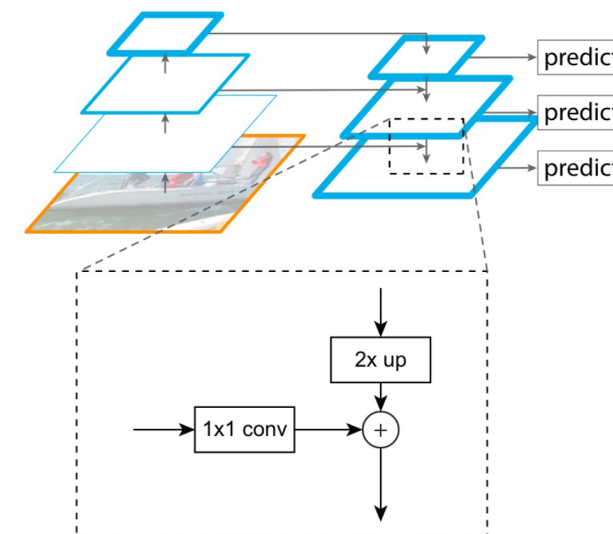


Figure 3. A building block illustrating the lateral connection and the top-down pathway, merged by addition.

Meta architecture

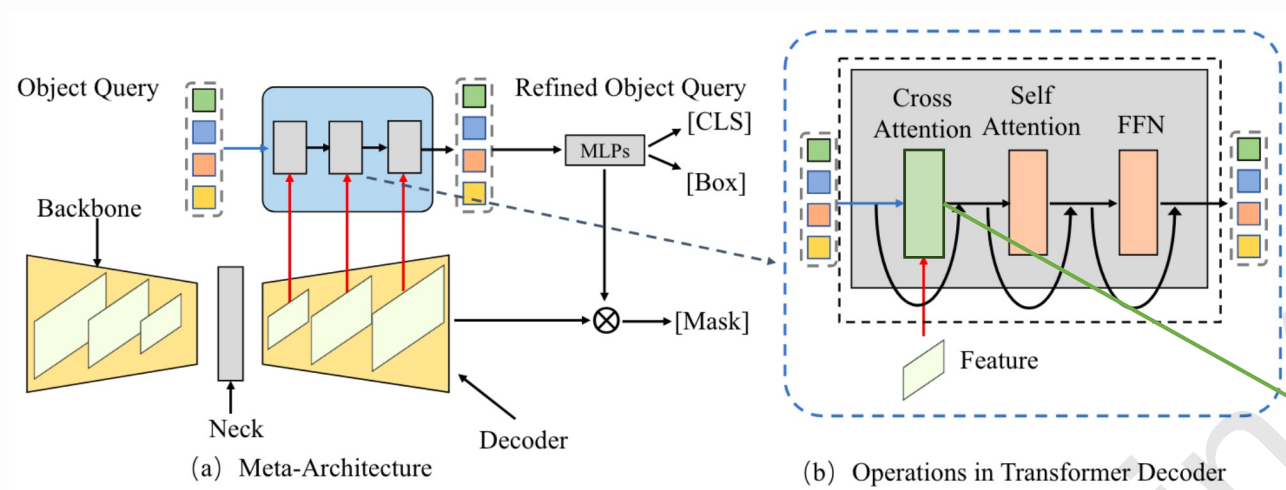


Fig. 3: Illustration of (a) meta-architecture and (b) common operations in the decoder.

• Learnable Object Query

Depending on the task and objectives, the number of learnable tokens is chosen, and the values of these tokens are altered through error-propagated gradient descent.

Attention matrix between Q and Image

q1	0.3	0.0	0.0	0.2	0.2	0.1	0.1	0.1	0.0
q2	0.2	0.3	0.3	0.0	0.1	0.1	0.0	0.0	0.0
q3	0.0	0.0	0.0	0.0	0.0	0.1	0.4	0.3	0.2
q4	0.2	0.1	0.1	0.2	0.1	0.1	0.1	0.0	0.1

• Transformer Decoder

$F_1: [\frac{W \times H}{P^2}, d]$ $F_2: [W' \times H', d]$ $F_3: [W \times H, d]$

Object query: $Q_0 \in \mathbb{R}^{l_q \times d}$, 0值初始化

$K_1 = F_1 W_{1q} \in \mathbb{R}^{\frac{W \times H}{P^2} \times d}, W_{1q} \in \mathbb{R}^{d \times d}$
 $V_1 = F_1 W_{1v} \in \mathbb{R}^{\frac{W \times H}{P^2} \times d}, W_{1v} \in \mathbb{R}^{d \times d}$
 $Q_1 = \text{softmax}\left(\frac{Q_0 \cdot K_1^T}{\sqrt{d}}\right) V_1 \in \mathbb{R}^{l_q \times d}$
.....
 $K_3 = F_1 W_{3q} \in \mathbb{R}^{(W \times H) \times d}, W_{3q} \in \mathbb{R}^{d \times d}$
 $V_3 = F_1 W_{3v} \in \mathbb{R}^{(W \times H) \times d}, W_{3v} \in \mathbb{R}^{d \times d}$
 $Q_3 = \text{softmax}\left(\frac{Q_3 \cdot K_3^T}{\sqrt{d}}\right) V_3 \in \mathbb{R}^{l_q \times d}$

Meta architecture

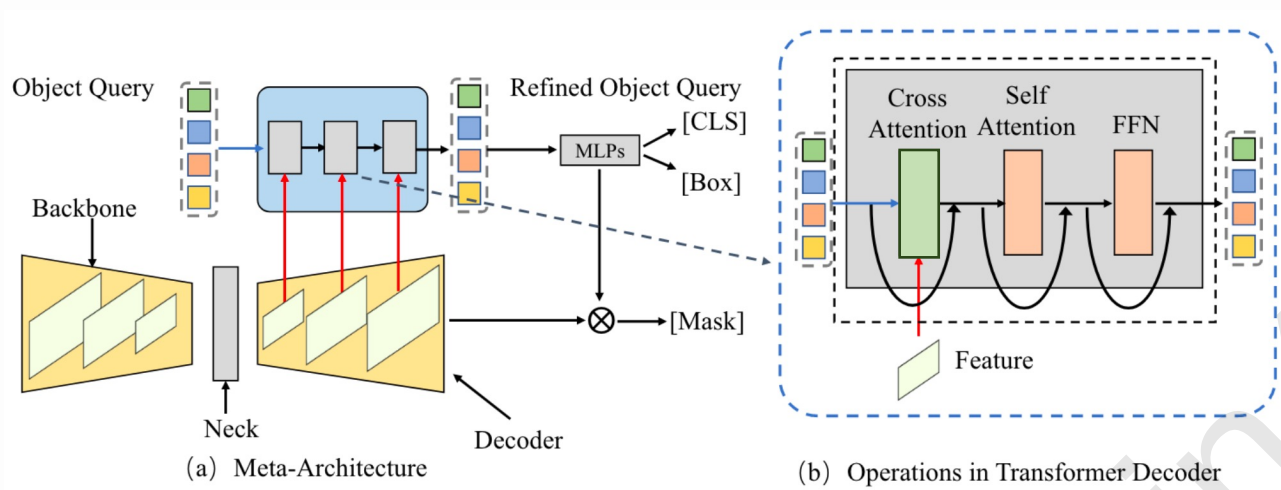


Fig. 3: Illustration of (a) meta-architecture and (b) common operations in the decoder.

- Bipartite Matching (Hungarian algorithm)

Instance Mask $\in \mathbb{R}^{l_q \times H \times W}$
 Ground Truth $\in \mathbb{R}^{C \times H \times W}$, $l_q > C$, 每个样本C不同

Cost matrix

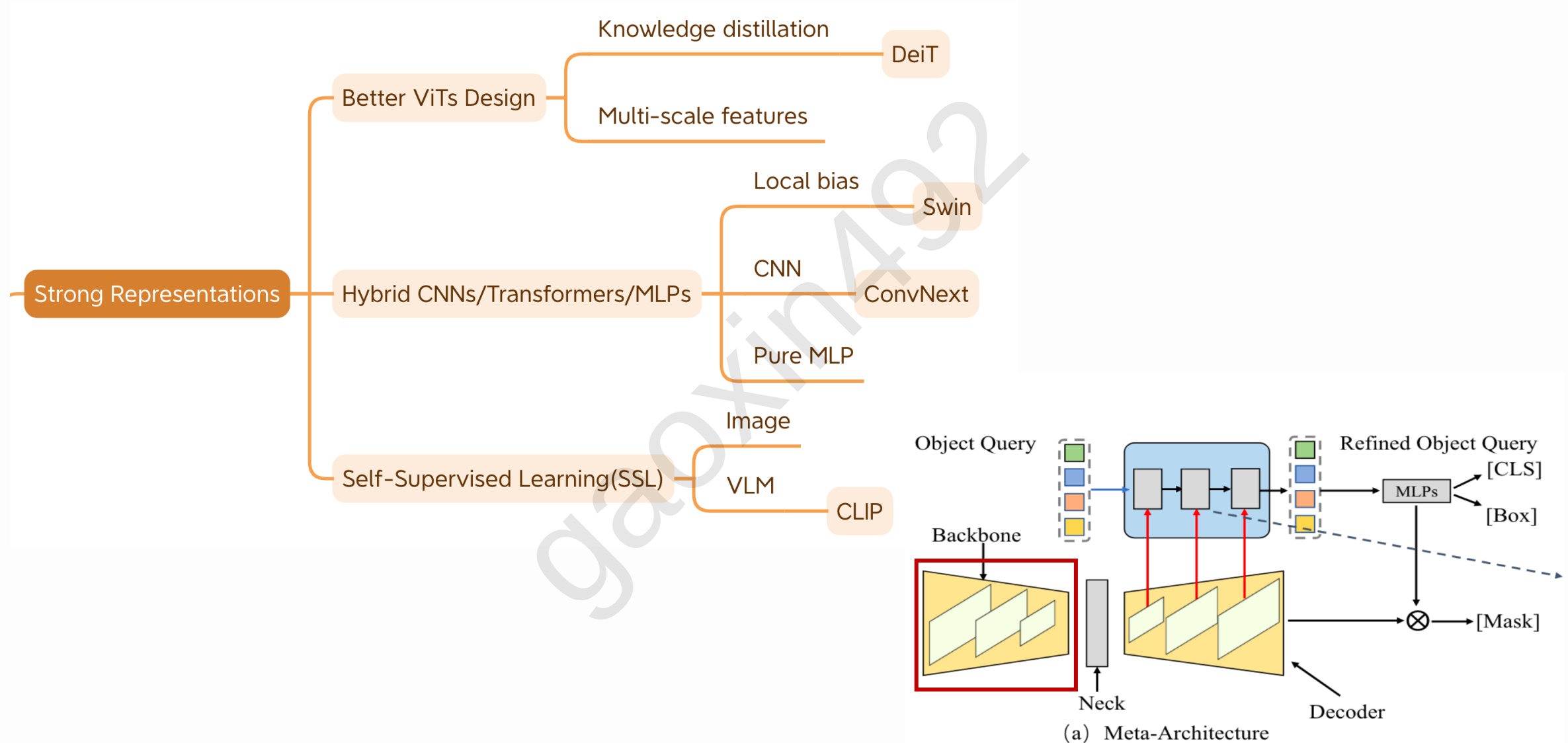
	M_1	M_2	M_3	M_4		M_{l_q-1}	M_{l_q}
G_1							
G_2							
$\vdots$							
G_c							

$\emptyset$ Detection是类似的, Cost则由cls和box定义

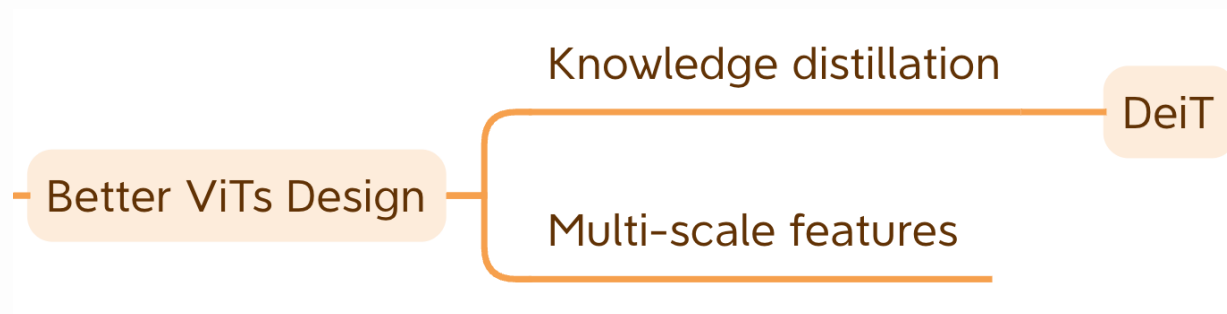
$Q_3 \in \mathbb{R}^{l_q \times d}$

- Classification $cls = mlp(Q_3)$
- Detection $cls, box = mlp(Q_3) \in \mathbb{R}^{l_q \times 5}$, l_q 为一个预设的数, 如100, 表示最多instance
- Segmentation
 - Instance Mask = $Sigmoid(Q_3 F_3^T) \in \mathbb{R}^{l_q \times H \times W}$, $F_3 \in \mathbb{R}^{H \times W \times d}$, l_q 为一个预设的数, 如100, 表示最多instance
 - Semantic Mask = $Sigmoid(Q_3 F_3^T) \in \mathbb{R}^{l_q \times H \times W}$, $F_3 \in \mathbb{R}^{H \times W \times d}$, l_q 为类别

1. Strong Representations



1. Strong Representations



- DeiT
- 提出了针对 ViT 的教师-学生蒸馏训练策略
- 提出了 token-based distillation 方法

"The transformers do not generalize well when trained on insufficient amounts of data."

$$\mathcal{L}_{\text{global}} = (1 - \lambda)\mathcal{L}_{\text{CE}}(\psi(Z_s), y) + \lambda\tau^2\text{KL}(\psi(Z_s/\tau), \psi(Z_t/\tau)). \quad (2)$$

$$\mathcal{L}_{\text{global}}^{\text{hardDistill}} = \frac{1}{2}\mathcal{L}_{\text{CE}}(\psi(Z_s), y) + \frac{1}{2}\mathcal{L}_{\text{CE}}(\psi(Z_s), y_t). \quad (3)$$

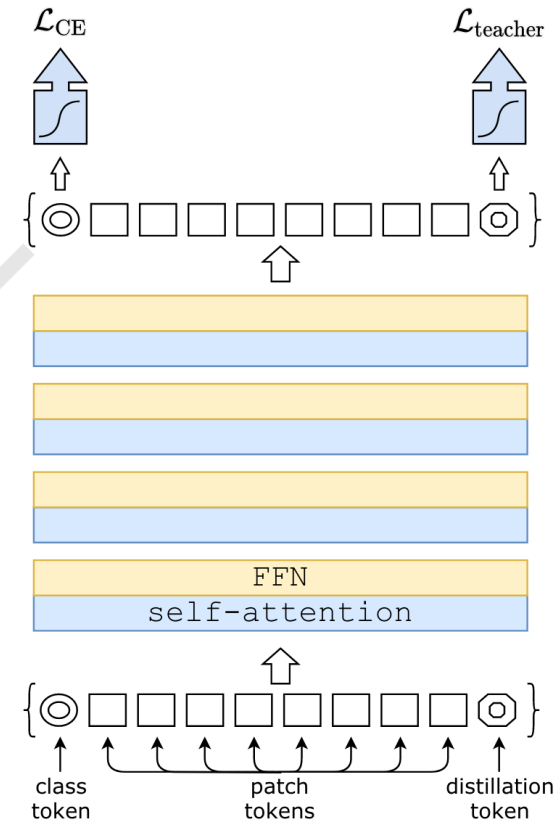
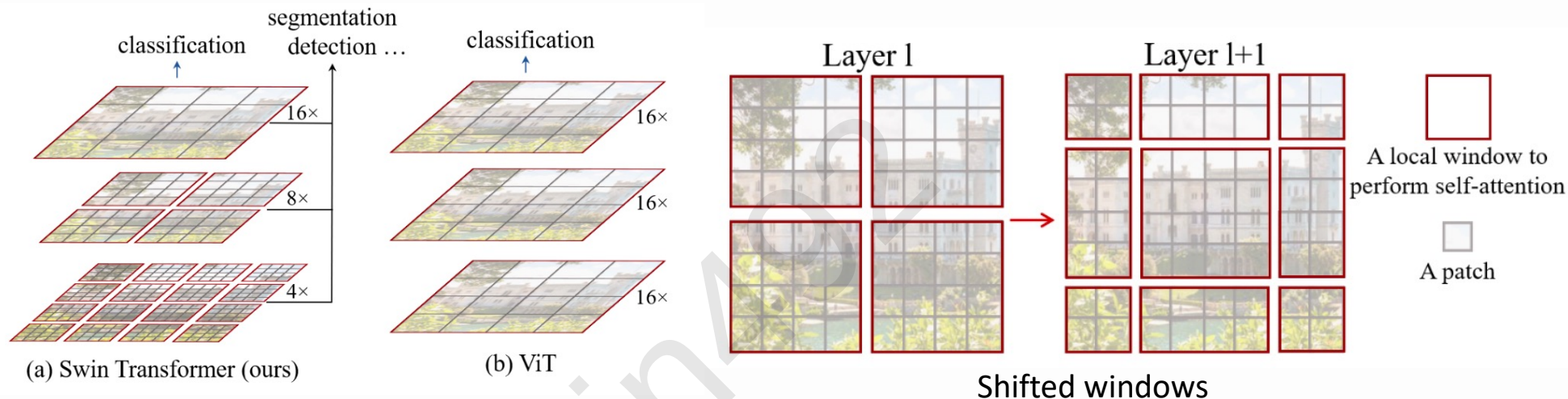


Figure 2: Our distillation procedure: we simply include a new *distillation token*. It interacts with the class and patch tokens through the self-attention layers. This distillation token is employed in a similar fashion as the class token, except that on output of the network its objective is to reproduce the (hard) label predicted by the teacher, instead of true label. Both the class and distillation tokens input to the transformers are learned by back-propagation.

1. Strong Representations

- Swin Transformer backbone
- General



- 在ViT中引入inductive bias
- Patch merging
- Shifted windows

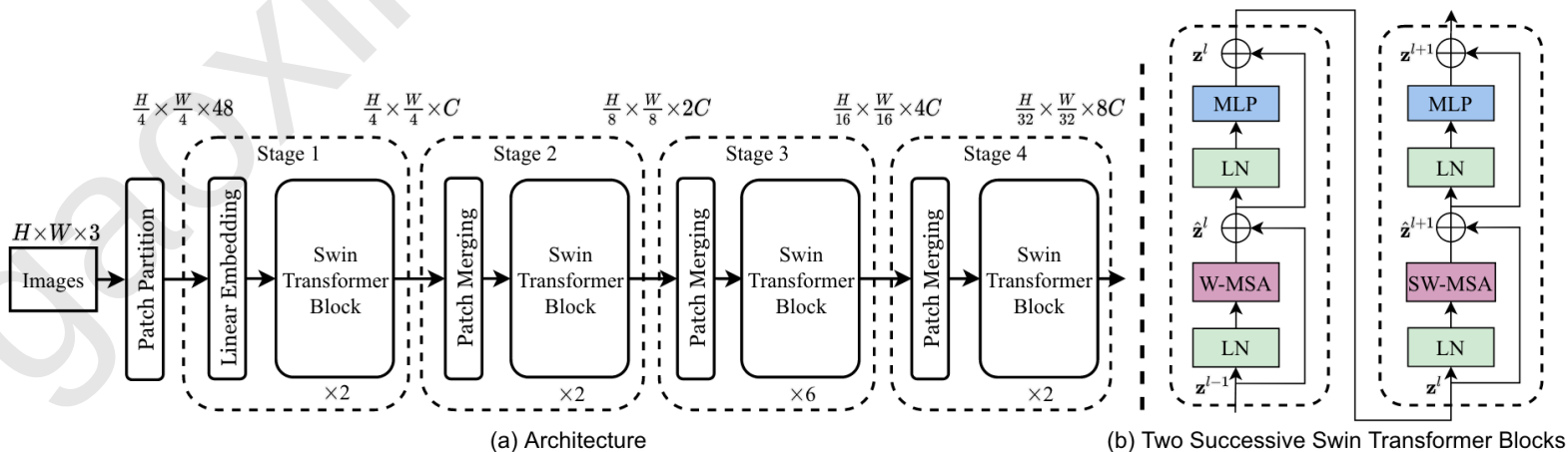
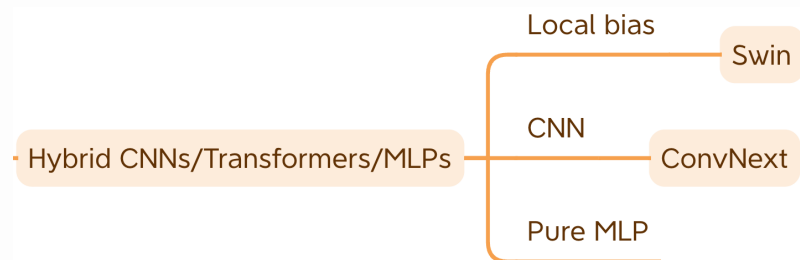


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

1. Strong Representations

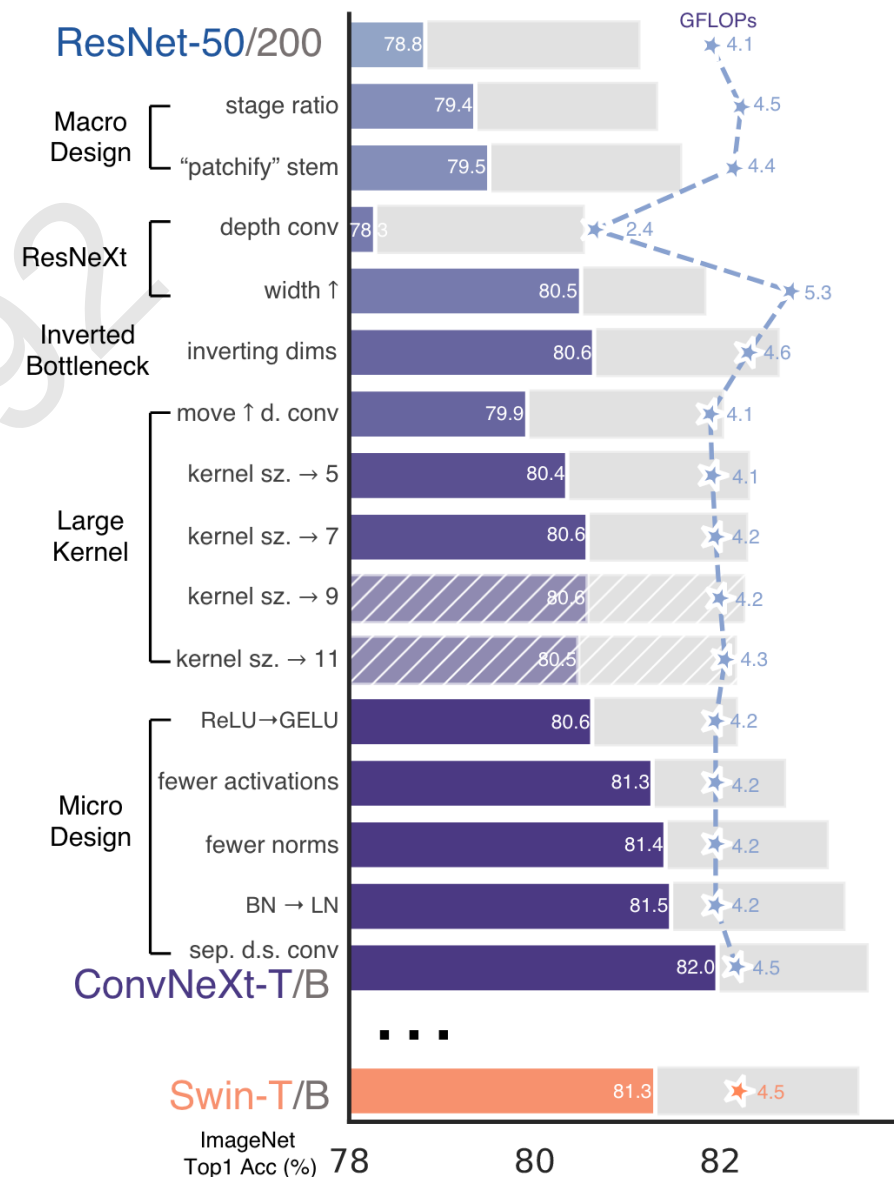
- ConvNeXt, facebook
 - 用训练ViT的一些灵感来改造CNN
- Training techniques: Optimization strategy, hyper-parameter settings (学习DeiT和Swin Transformer)

AdamW optimizer, data augmentation techniques such as Mixup, Cutmix, RandAugment, Random Erasing, and regularization schemes including Stochastic Depth and Label Smoothing.

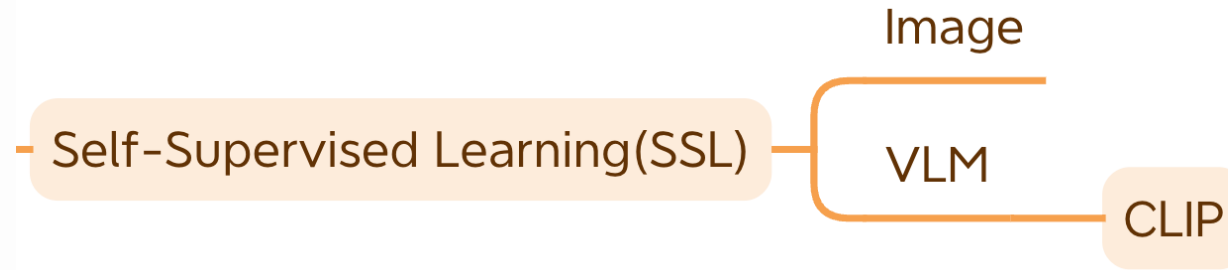
■ Macro Design

■ Micro Design

model	image size	#param.	FLOPs	throughput (image / s)	IN-1K top-1 acc.
ImageNet-1K trained models					
• RegNetY-16G [54]	224 ²	84M	16.0G	334.7	82.9
• EffNet-B7 [71]	600 ²	66M	37.0G	55.1	84.3
• EffNetV2-L [72]	480 ²	120M	53.0G	83.7	85.7
○ DeiT-S [73]	224 ²	22M	4.6G	978.5	79.8
○ DeiT-B [73]	224 ²	87M	17.6G	302.1	81.8
○ Swin-T	224 ²	28M	4.5G	757.9	81.3
• ConvNeXt-T	224 ²	29M	4.5G	774.7	82.1
○ Swin-S	224 ²	50M	8.7G	436.7	83.0
• ConvNeXt-S	224 ²	50M	8.7G	447.1	83.1
○ Swin-B	224 ²	88M	15.4G	286.6	83.5
• ConvNeXt-B	224 ²	89M	15.4G	292.1	83.8
○ Swin-B	384 ²	88M	47.1G	85.1	84.5
• ConvNeXt-B	384 ²	89M	45.0G	95.7	85.1
• ConvNeXt-L	224 ²	198M	34.4G	146.8	84.3
• ConvNeXt-L	384 ²	198M	101.0G	50.4	85.5



1. Strong Representations



Masked Autoencoders Are Scalable Vision Learners

Kaiming He^{*,†} Xinlei Chen^{*} Saining Xie Yanghao Li Piotr Dollár Ross Girshick

^{*}equal technical contribution [†]project lead

Facebook AI Research (FAIR)

Bootstrap Your Own Latent A New Approach to Self-Supervised Learning

Learning Transferable Visual Models From Natural Language Supervision

Alec Radford^{*,1} Jong Wook Kim^{*,1} Chris Hallacy¹ Aditya Ramesh¹ Gabriel Goh¹ Sandhini Agarwal¹
Girish Sastry¹ Amanda Askell¹ Pamela Mishkin¹ Jack Clark¹ Gretchen Krueger¹ Ilya Sutskever¹

Jean-Bastien Grill^{*,1}, Florian Strub^{*,1}, Florent Alché^{*,1}, Corentin Tallec^{*,1}, Pierre H. Richemond^{*,1,2}

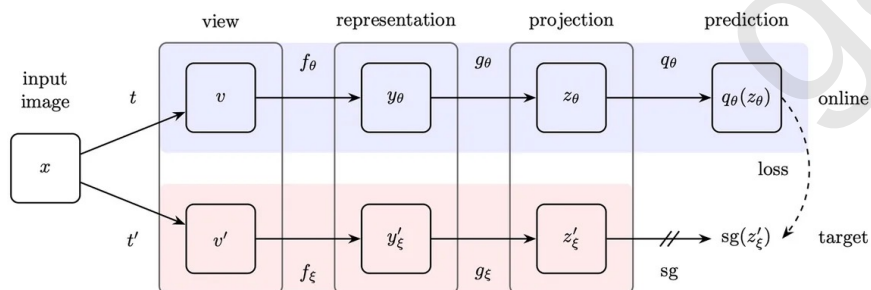
Elena Buchatskaya¹, Carl Doersch¹, Bernardo Avila Pires¹, Zhaohan Daniel Guo¹

Mohammad Gheshlaghi Azar¹, Bilal Piot¹, Koray Kavukcuoglu¹, Rémi Munos¹, Michal Valko¹

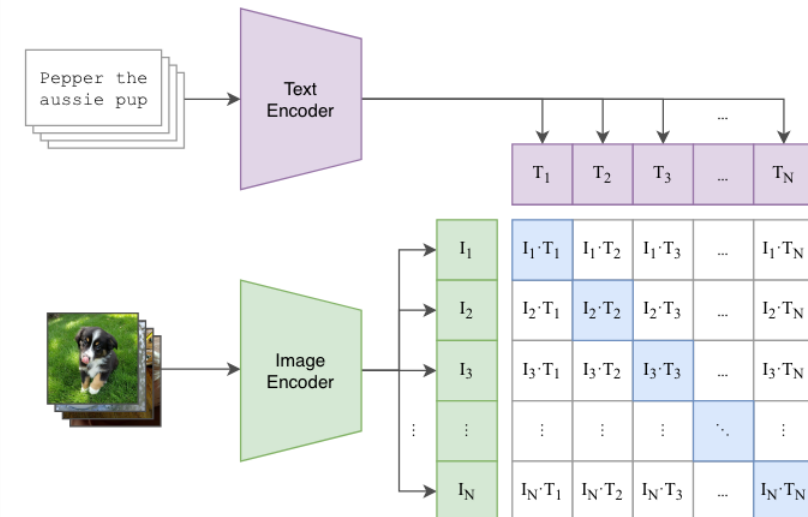
¹DeepMind

²Imperial College

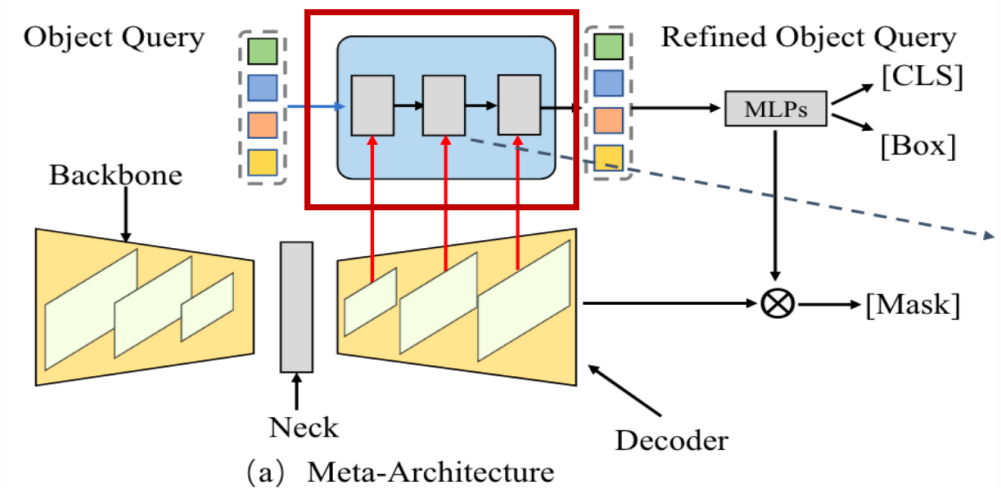
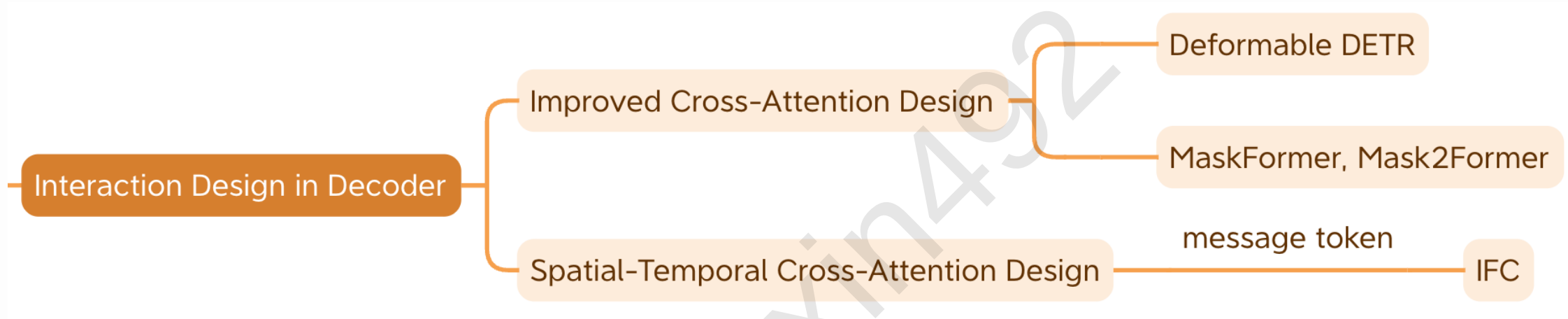
[jbgrill,fstrub,altche,corentint,richemond]@google.com



(1) Contrastive pre-training



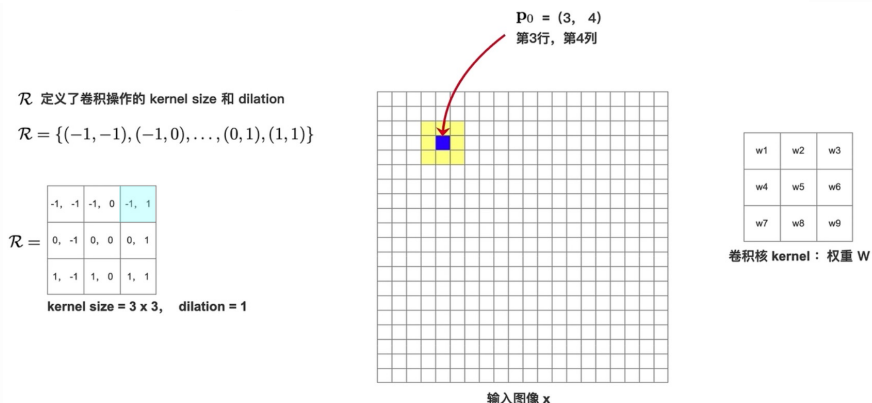
2. Interaction design in Decoder



2. Interaction design in Decoder

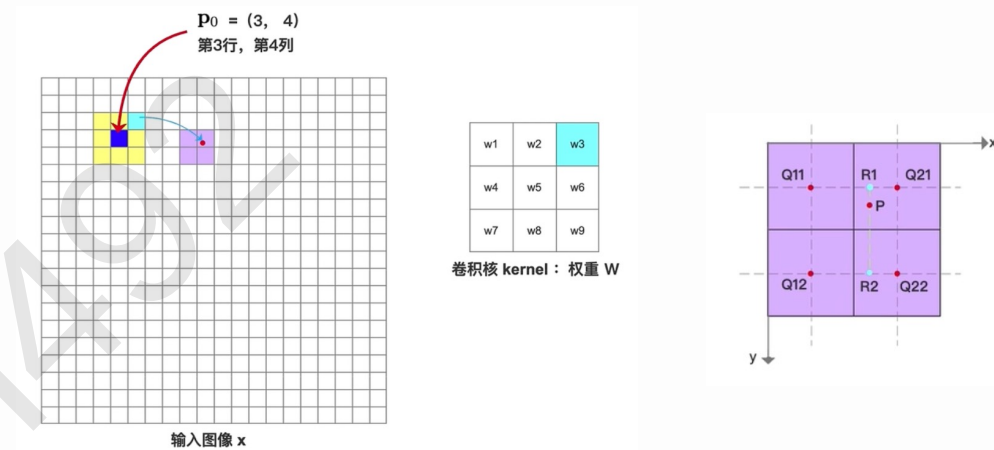
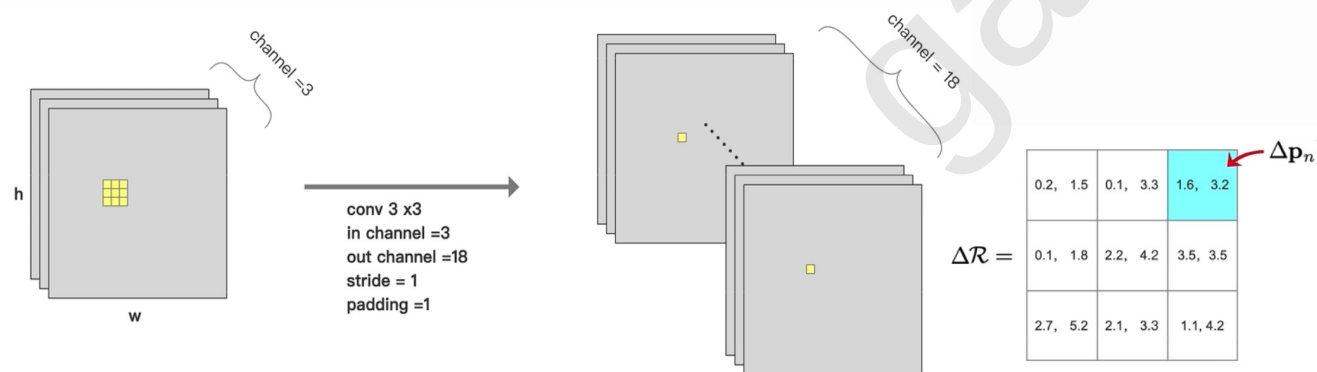
Deformable DETR • 提出了deformable Attention

普通卷积公式：
$$y(p_0) = \sum_{p_n \in \mathcal{R}} w(p_n) \cdot x(p_0 + p_n),$$



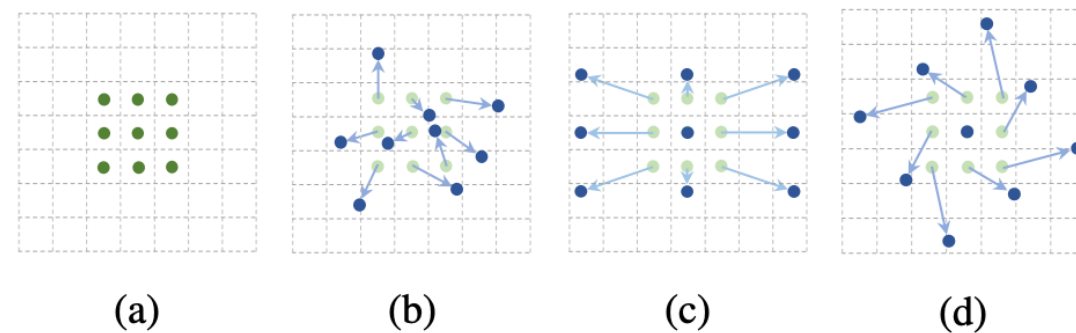
可变形卷积公式：
$$y(p_0) = \sum_{p_n \in \mathcal{R}} w(p_n) \cdot x(p_0 + p_n + \Delta p_n).$$

Δp_n 是学习到的值, 是浮点型数据, 由图像经过普通卷积计算得到



$$p = p_0 + p_n + \Delta p_n = (3, 4) + (-1, 1) + (1.6, 3.2) = (3.6, 8.2)$$

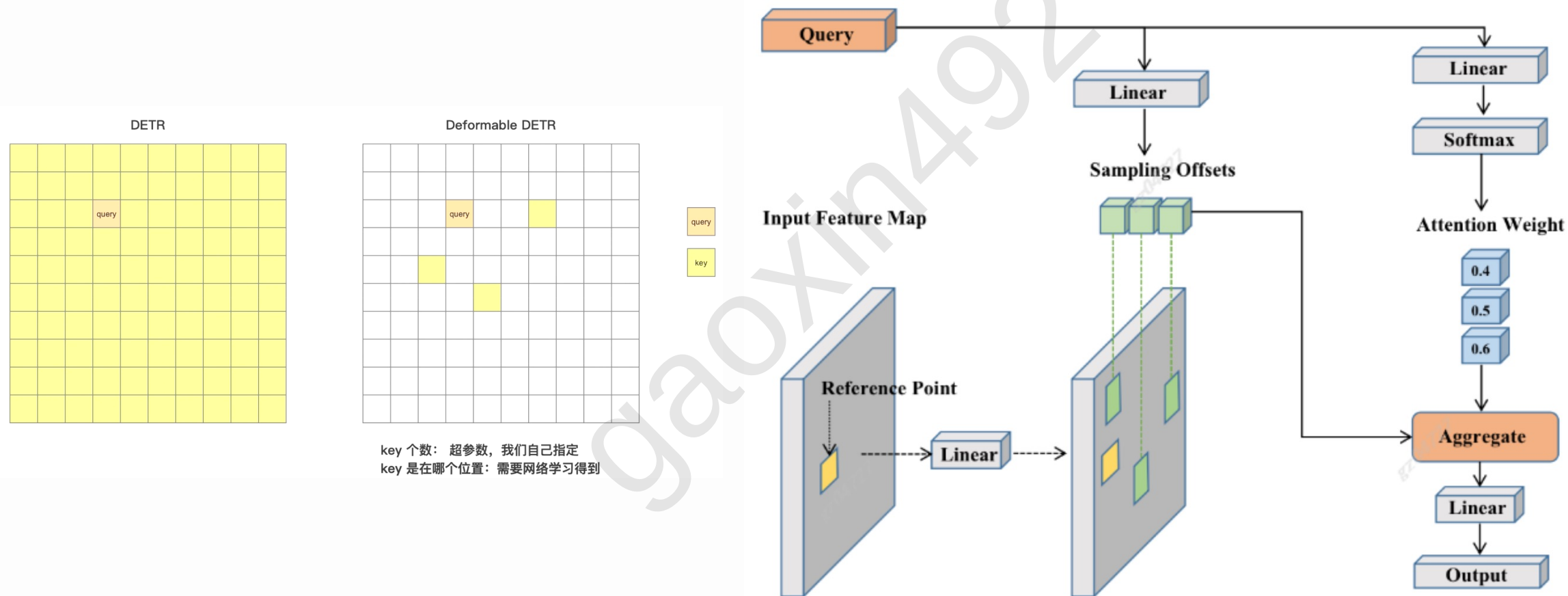
(3.6, 8.2) 临近的 4 个像素点分别为 (3, 8)、(3, 9)、(4, 8)、(4, 9)



2. Interaction design in Decoder

Deformable DETR • 提出了deformable Attention

Attention modules only attend to a small set of key sampling points around a reference.



2. Interaction design in Decoder

IFC message token $[f_t^0, m_t^0] \in \mathbb{R}^{(HW+M) \times C}, \quad t \in \{1, 2, \dots, T\},$

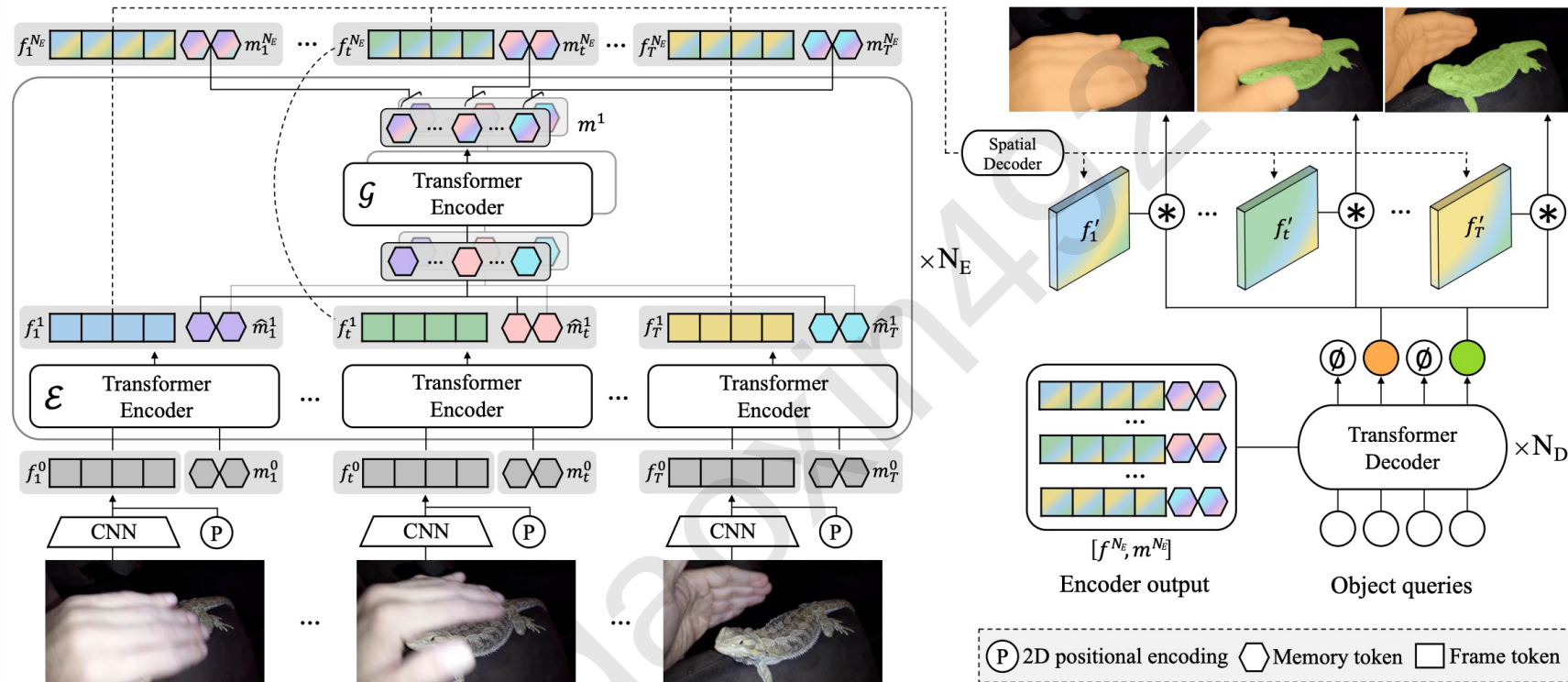
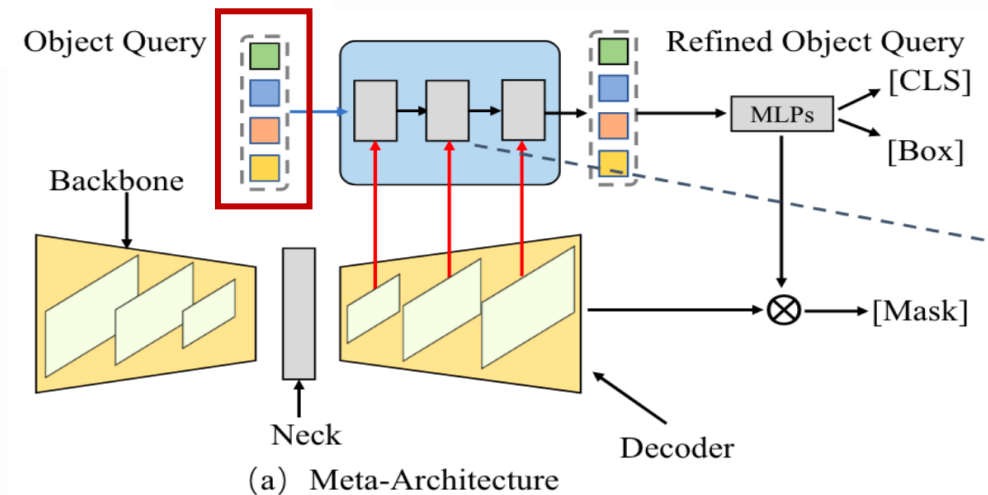
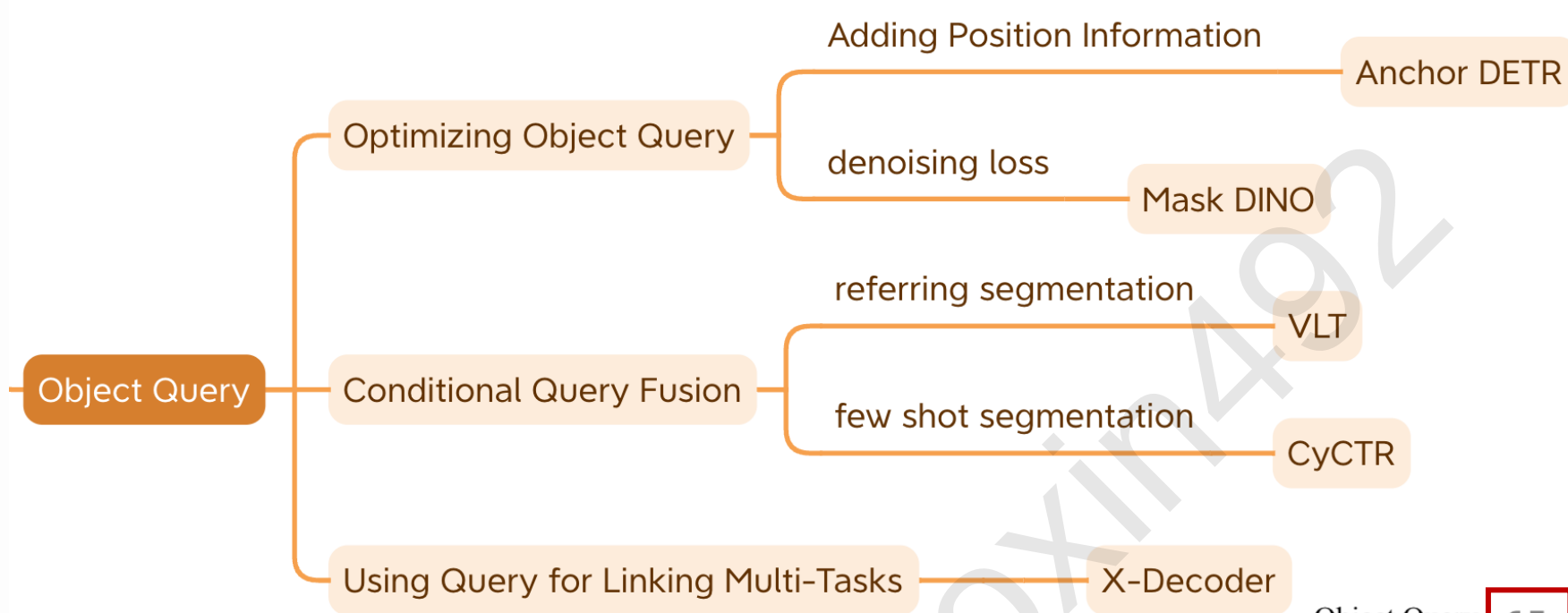


Figure 1: Overview of IFC framework. Our transformer encoder block has two components: 1) Encode-Receive ($\mathcal{E}$) simultaneously encodes frame tokens and memory tokens. 2) Only memory tokens pass Gather-Communicate ($\mathcal{G}$) to perform communications between frames. The outputs from the stack of N_E encoder blocks goes into two modules, spatial decoder and transformer decoder, to generate segmentation masks.

3. Object Query



3. Object Query

X-decoder

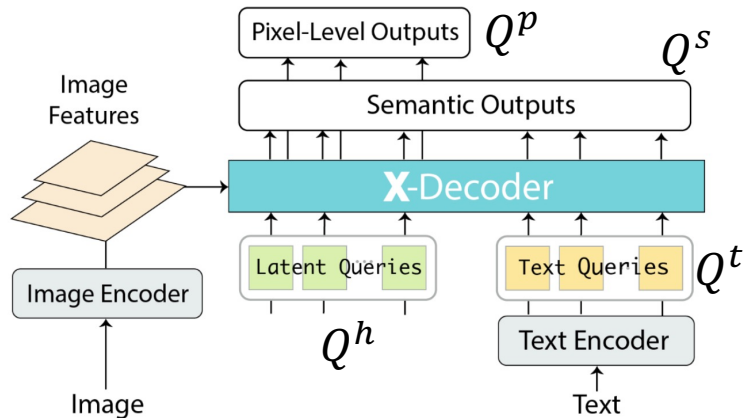
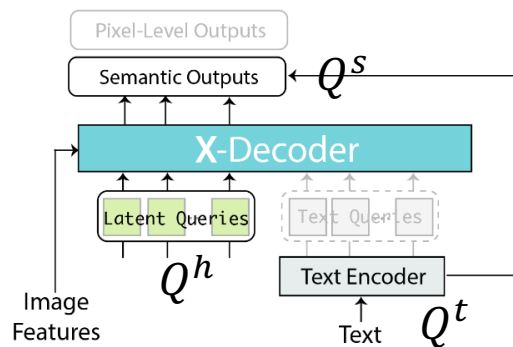


Figure 2. Overall pipeline for our model. It consists of an image encoder, a text encoder and our own designed X-Decoder.

$$Q^h \ Z \ Z \Rightarrow Q^s$$

• Image-text Retrieval



(c) Image-Text Retrieval

Input image $\mathbf{I} \in \mathcal{R}^{H \times W \times 3}$, image encoder $\mathbf{Enc}_I \Rightarrow$ features $\mathbf{Z} \in \mathcal{R}^{H'W' \times d}$

Textual query $\mathbf{T}$, text encoder $\mathbf{Enc}_T \Rightarrow \mathbf{Q}^t = \langle q_1^t, \dots, q_n^t \rangle$

Latent queries $\mathbf{Q}^h = \langle q_1^h, \dots, q_m^h \rangle$ are fed to our X-Decoder to predict the outputs:

$$\langle \mathbf{O}^p, \mathbf{O}^s \rangle = \mathbf{XDec}(\langle \mathbf{Q}^h, \mathbf{Q}^t \rangle; \mathbf{Z})$$

where $\mathbf{O}^p$ and $\mathbf{O}^s$ are the pixel-level masks and token-level semantics, respectively

Latent queries: m learnable queries, the last latent query to extract the global image representation and the remaining for generic segmentation

The last valid token feature of $\mathbf{Q}^t$ from the text encoder to represent a text as $\hat{q}^t$
The last entry in $\mathbf{O}^s$ derived from X-Decoder as $\hat{o}^s$

B pairs of features $\langle \hat{q}_i^t, \hat{o}_i^s \rangle_{i=1}^B$ for a minibatch of B image-text pairs

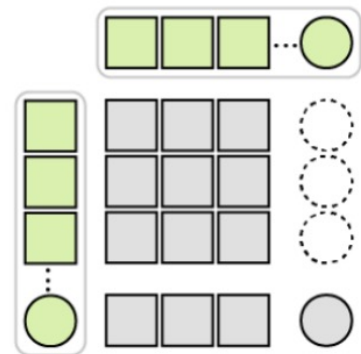
Dot-product between these $B \times B$ feature pairs $\Rightarrow$ affinity matrix $\mathbf{S}_{it} \in \mathcal{R}^{B \times B}$

Compute the bidirectional cross-entropy loss:

$$\mathcal{L}_{it} = \mathbf{CE}(\mathbf{S}_{it}, \mathbf{y}_{it}) + \mathbf{CE}(\mathbf{S}_{it}^T, \mathbf{y}_{it})$$

where $\mathbf{y}_{it}$ are the class labels corresponding to diagonal entries in $\mathbf{S}_{it}$

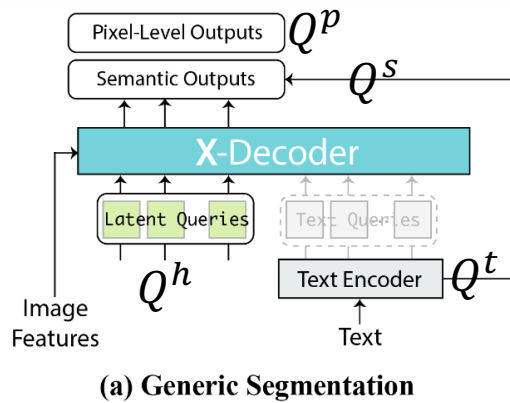
Self-attention mask



(a) Seg + Ret

3. Object Query

- Generic segmentation



Self attention $\overset{Q}{Q^h}, \overset{K}{Q^h}, \overset{V}{Q^h} \Rightarrow \overset{Q}{Q^{h'}}$

Cross attention $\overset{Q}{Q^{h'}}, \overset{K}{Z}, \overset{V}{Z} \Rightarrow \overset{Q}{O^s}$

$\text{Sigmoid}(\widehat{Q^s} Z^T) = O^p, \quad m - 1 \uparrow \text{instance mask}$

Bipartite Matching (Hungarian algorithm)

Mask classification

Encode all C class names including "background" into C text queries

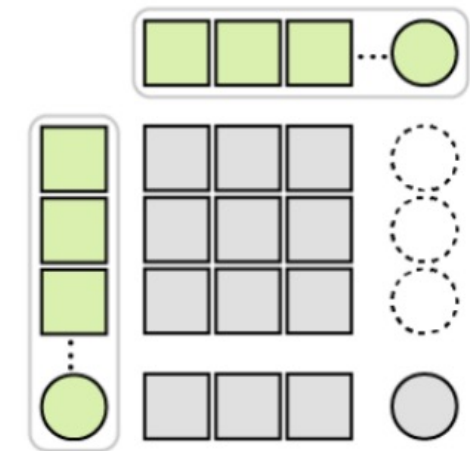
Take the last valid token feature from each to represent the concept

Use the first $(m - 1)$ query in O^s to compute the dot-product

Between these outputs and concept embeddings $\Rightarrow$ affinity matrix $\mathbf{S}_{cls} \in \mathcal{R}^{(m-1) \times C}$

Compute the loss $\mathcal{L}_{cls} = \mathbf{CE}(\mathbf{S}_{cls}, \mathbf{y}_{cls})$, with the ground-truth class $\mathbf{y}_{cls}$

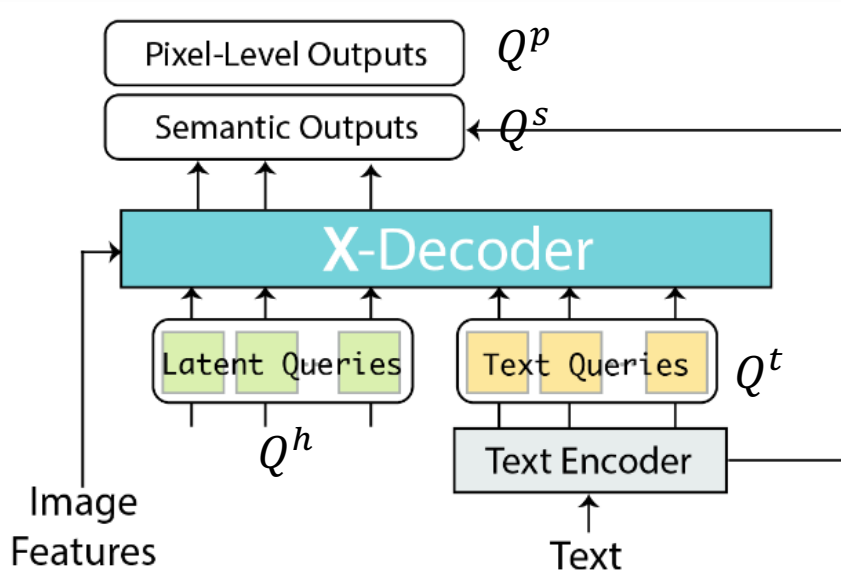
Semantic Seg: combine the instance seg output and class label



Self-attention mask

3. Object Query

- Referring Segmentation



(b) Referring Segmentation

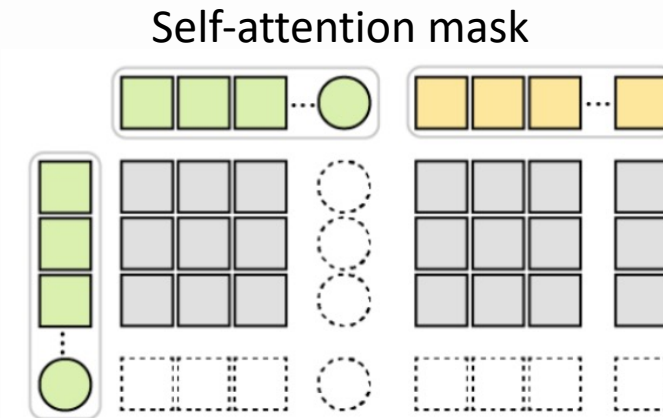
Conditional Self attention $\overset{Q}{Q^h}, \overset{K}{\langle Q^h, Q^t \rangle}, \overset{V}{\langle Q^h, Q^t \rangle} \Rightarrow \overset{Q}{Q^{h'}}$

Cross attention $\overset{Q}{Q^{h'}}, \overset{K}{Z}, \overset{V}{Z} \Rightarrow \overset{Q}{O^s}$

$\text{Sigmoid}(\widehat{Q^s} Z^T) = O^p, \quad m - 1 \uparrow \text{instance mask}$

Bipartite Matching (Hungarian algorithm)

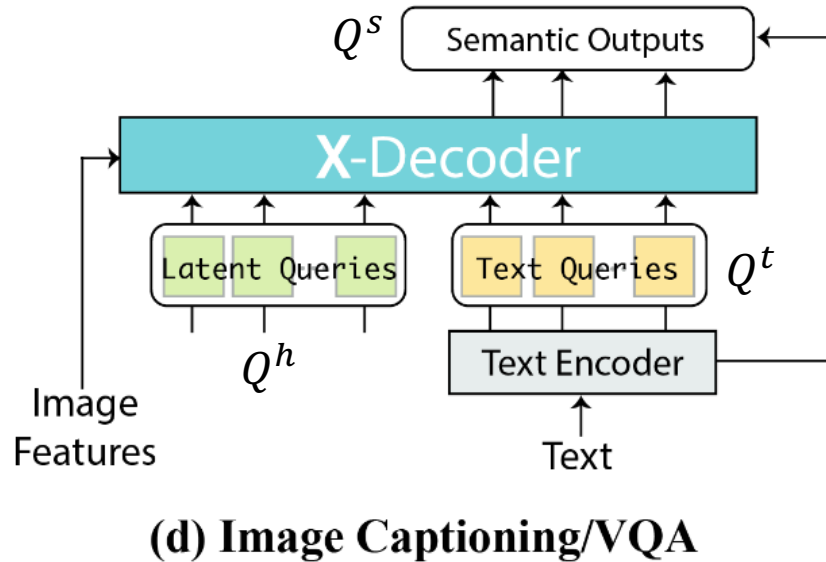
Other than the text used as a condition, everything else is the same as generic segmentation.



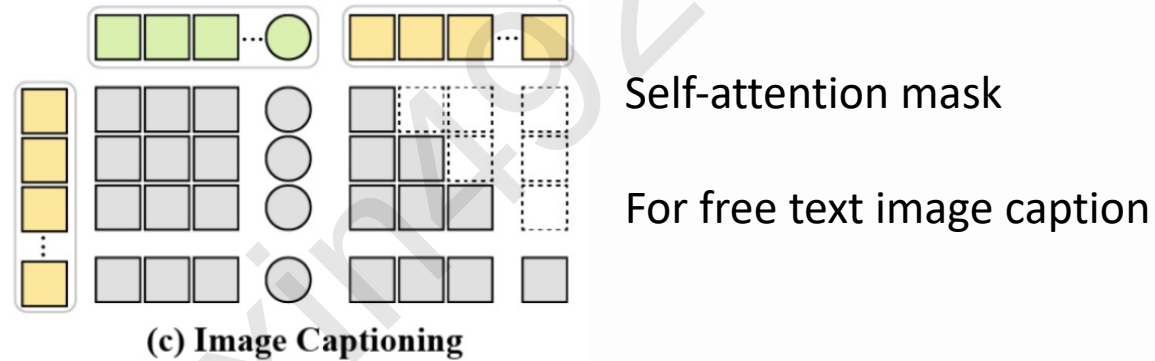
(b) Referring Segmentation

3. Object Query

- Image Captioning/VQA



Conditional Self attention $\overset{Q}{Q^t}, <\overset{K}{Q^h}, \overset{V}{Q^t}>, <\overset{K}{Q^h}, \overset{V}{Q^t}> \Rightarrow Q^{h'}$

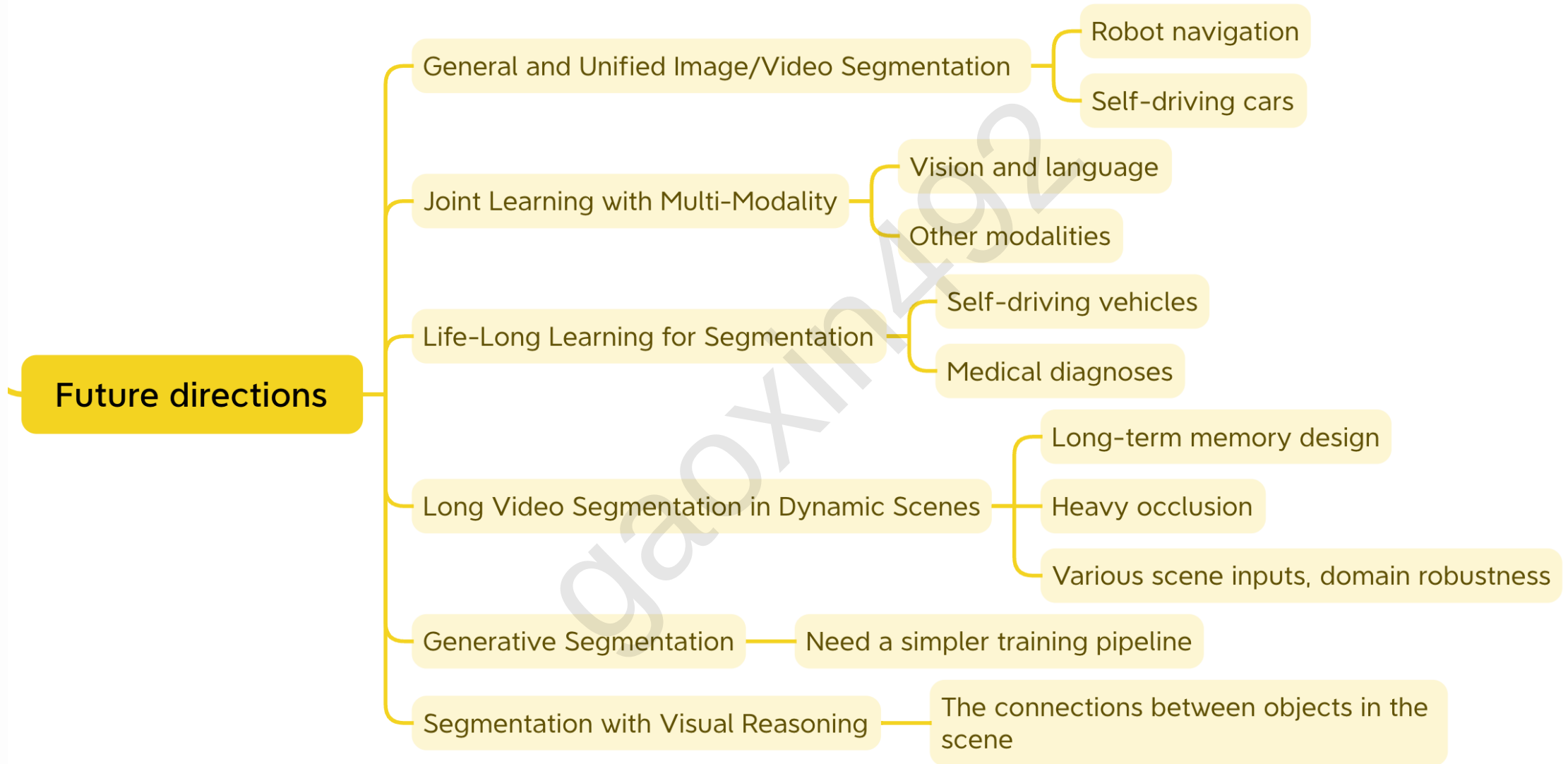


Cross attention $\overset{Q}{Q^{h'}}, \overset{K}{Z}, \overset{V}{Z} \Rightarrow \overset{Q}{O^s}$

The caption prediction follows a causal masking strategy while VQA use the last query in O^s to predict the answer for VQA.

All types of tasks are trained together, with the distinction lying solely in the combination of different modules and computation methods.

FUTURE DIRECTIONS



TRANSFORMER

Thanks

8.15 Journal Club