Transformer-based method with Deformable Convolution and Central Difference Convolution for Object Detection

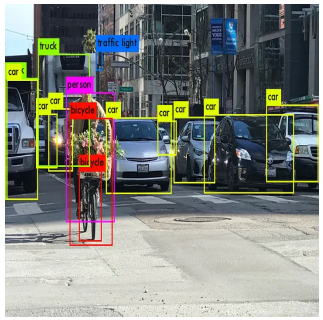
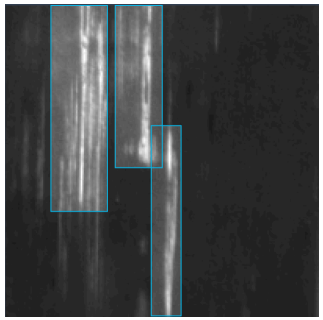
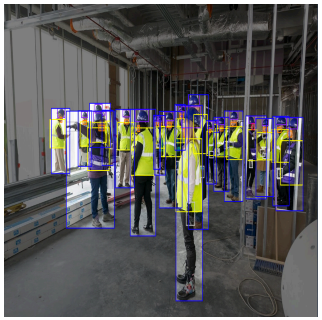
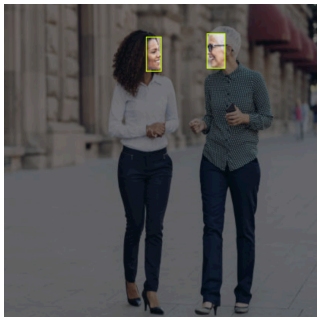
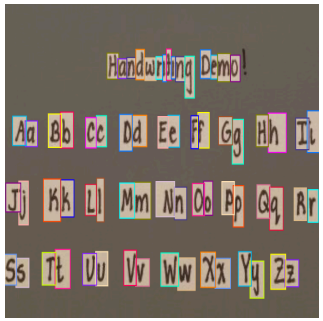
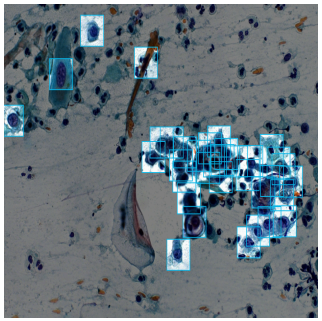
基於 Transformer 網路及可變形卷積與中心差分卷積的物件偵測方法

劉律奇

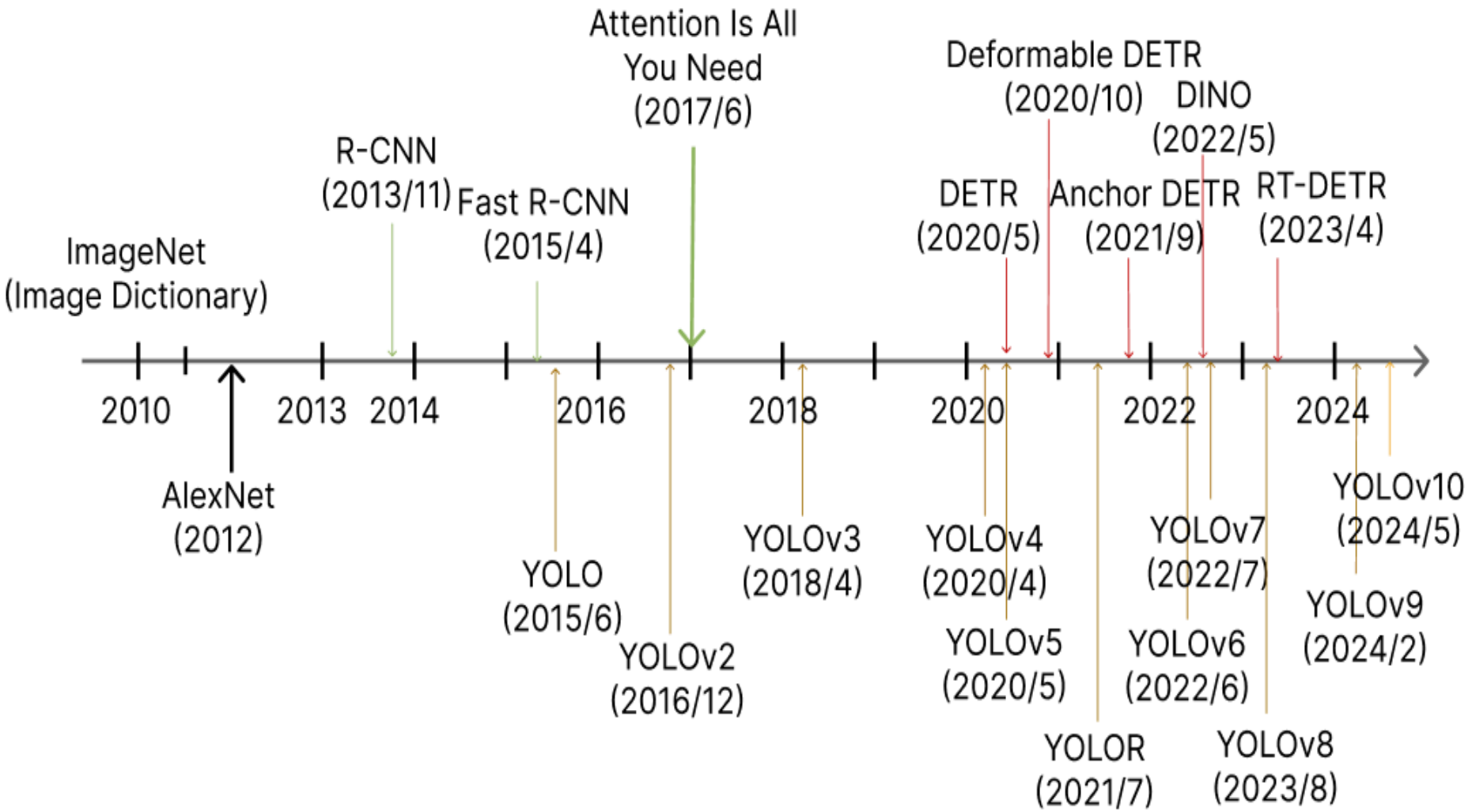
June 21, 2024

Table of contents

Object Detection Application

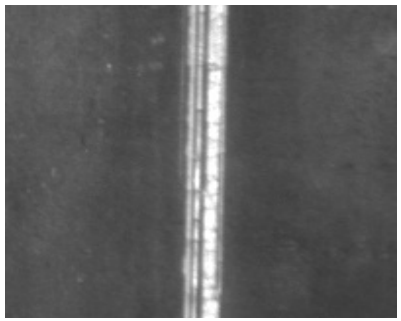
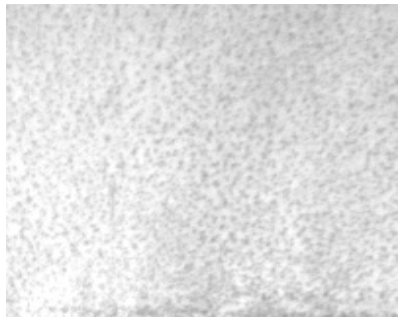
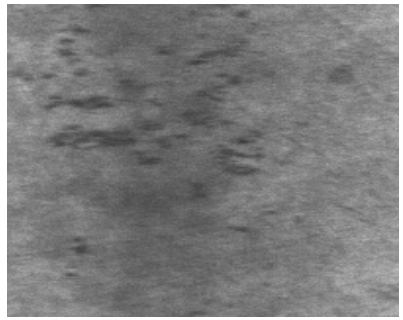
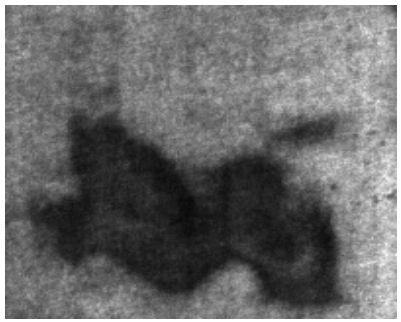
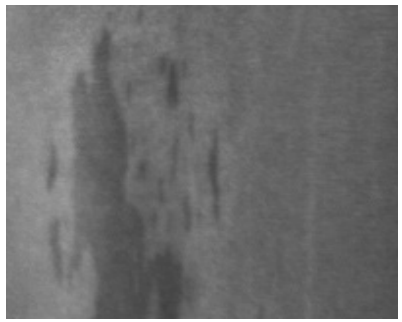
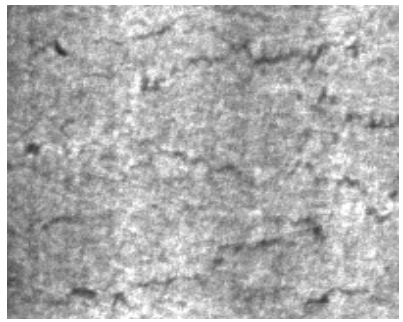
			
Vehicle	Defect	Safety Helmet	Face
			
Hand Writting	Wound	Pap-smear	Pills

History of Object Detection



Research Objective

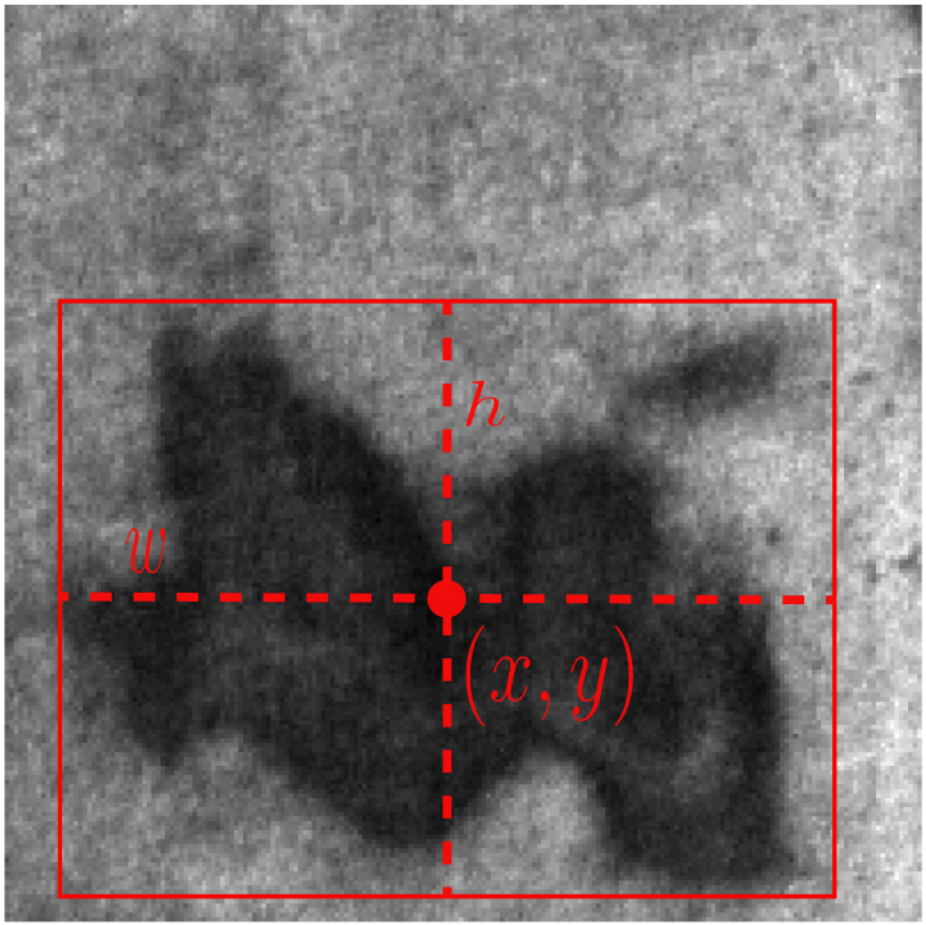
- Improve the detection capabilities of real-time detection transformer (RT-DETR).
- Performing defect detection[1]with the RT-DETR model.

		
Scratches	Pitted surface	Rolled in scale
		
Patches	Inclusion	Crazing

[1] Kechen Song and Yunhui Yan. "A noise robust method based on completed local binary patterns for hot-rolled steel strip

Introduction of Object Detection

Object detection is a branch of computer vision that empowers computers to identify and locate objects within images or videos.



$$\text{model(Image)} = \begin{bmatrix} \vdots \\ P_{Patches} \\ \vdots \\ x \\ y \\ w \\ h \end{bmatrix}$$

CNN-based Detector YOLO

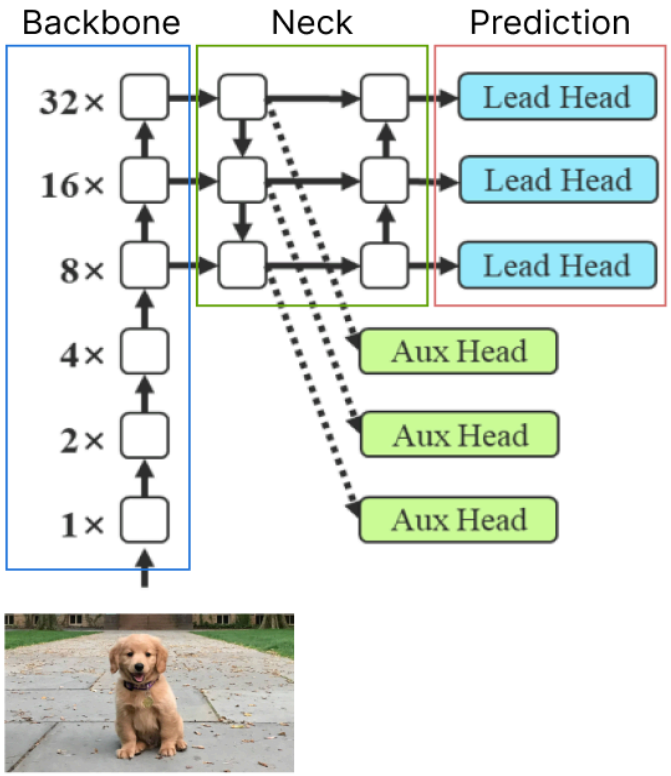


Figure: The main structure of YOLO.[3]

- Backbone
  Extracting and encoding features from the image
- Neck
  Further transforming and refining the features
- Prediction
  Designed to produce the task-specific prediction

[3] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. *YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information*. 2024. arXiv: 2402.13616 [cs.CV].

YOLO Detecting Head

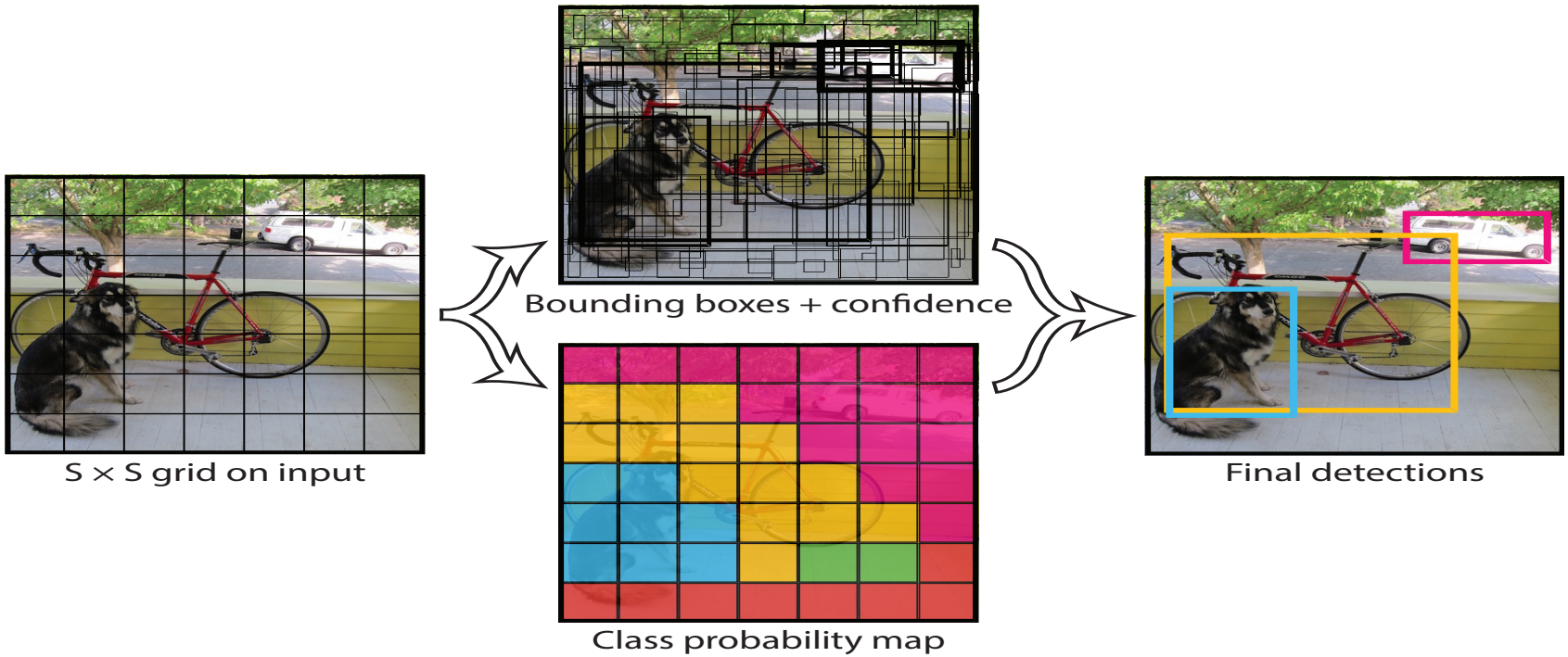


Figure: **Detecting Head.** YOLO detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.[5]

[5] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV].

YOLO Detecting Head

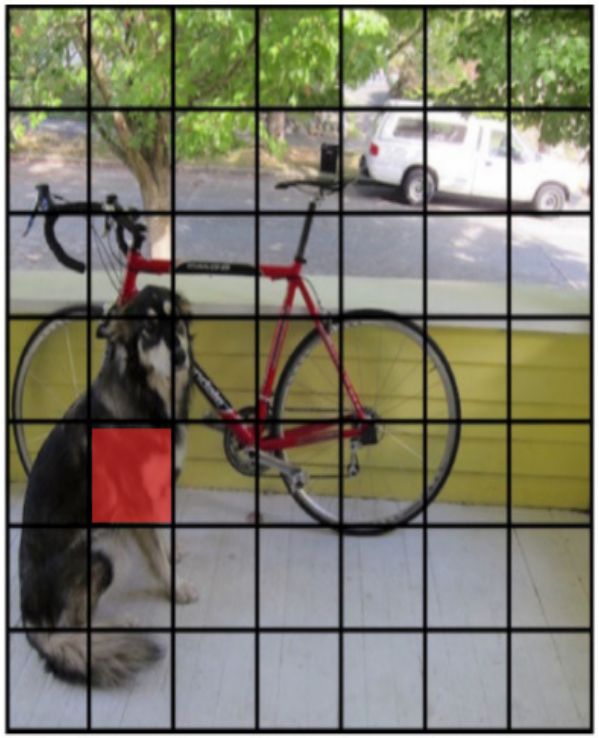


Figure: A prediction grid cell with two bounding boxes.

$$
\begin{matrix}
\\
\textit{Bounding Box1} \\
\\
\\
\\
\\
\\
\textit{Bounding Box2} \\
\\
\\
\\
\\
\\
\end{matrix}
\left\{
\begin{bmatrix}
\vdots \\
P_{dog} \\
c_1 \\
x_1 \\
y_1 \\
w_1 \\
h_1 \\
\vdots \\
P_{dog} \\
c_2 \\
x_2 \\
y_2 \\
w_2 \\
h_2
\end{bmatrix}
\right.
$$

YOLO Detecting Head

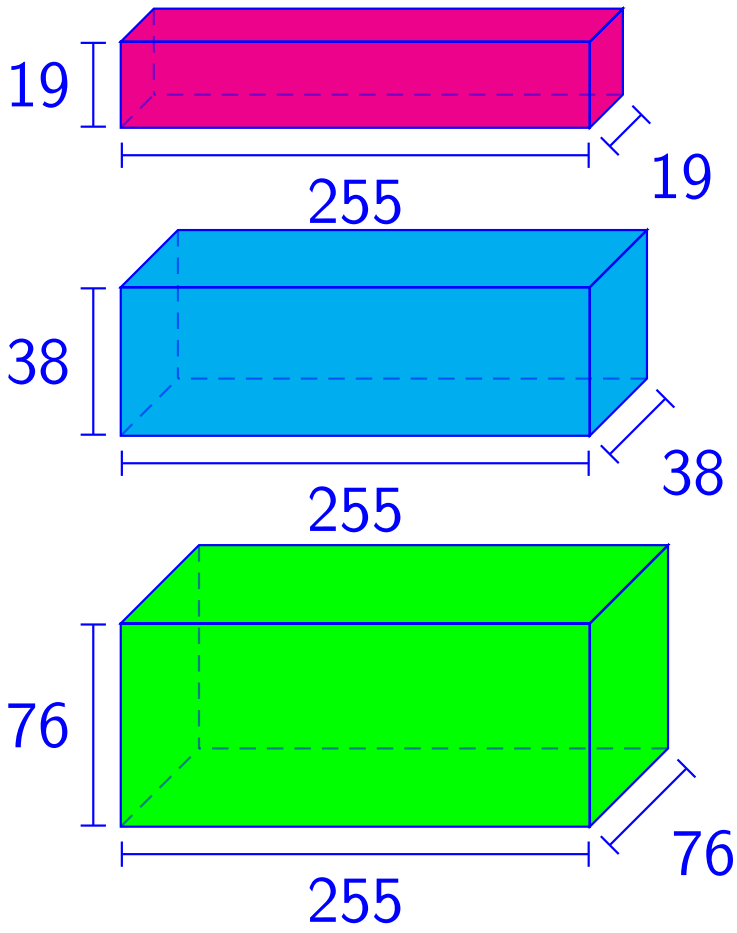
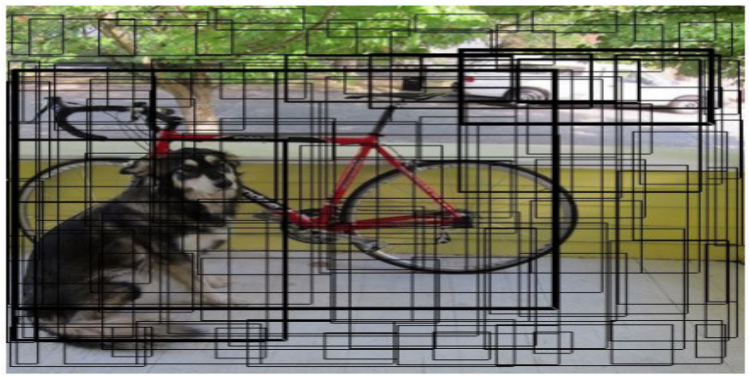


Figure: The Detecting head of YOLOv4[6].



The cumulative total across all prediction boxes is 22,743.

$$\begin{aligned} 255 &= 3 \times (80 + 1 + 4) \\ &= 3(\text{categories} \\ &\quad + \text{confidence} + \text{box}) \end{aligned}$$

[6] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: 2004.10934 [cs.CV].

Transformer-based Detector DETR

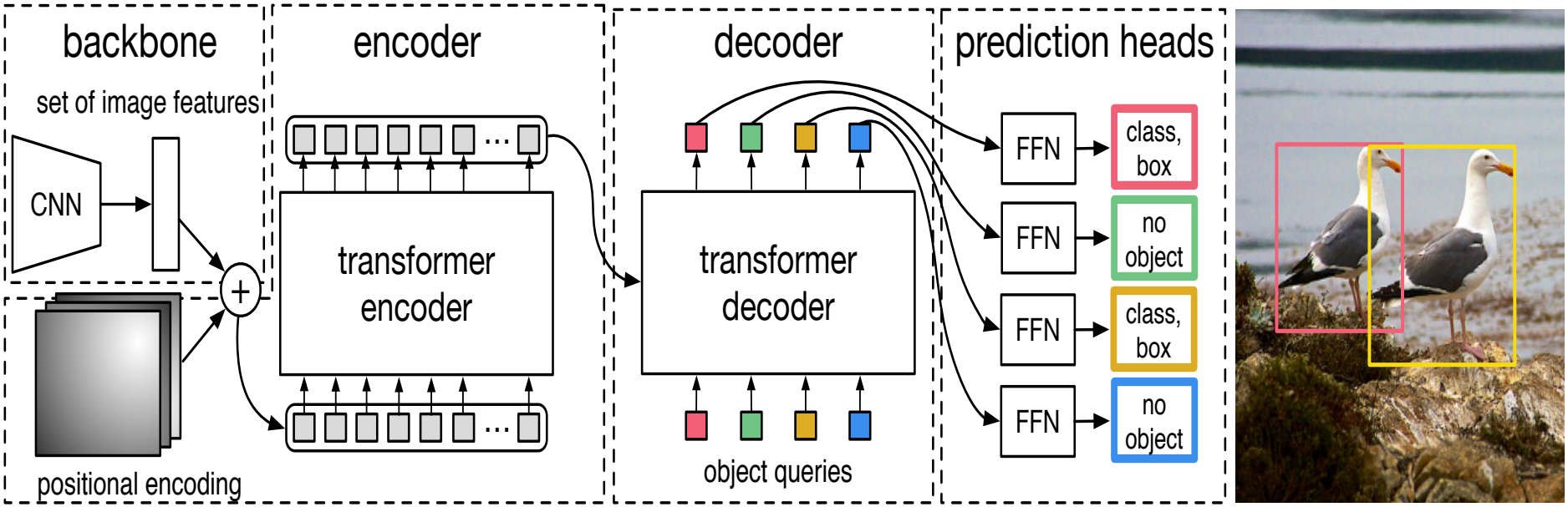


Figure: End-to-End Object Detection with Transformers (DETR) [7].

- Backbone
- Transformer
- Prediction

[7] Nicolas Carion et al. *End-to-End Object Detection with Transformers*. 2020. arXiv: 2005.12872 [cs.CV].

How does DETR predict objects?

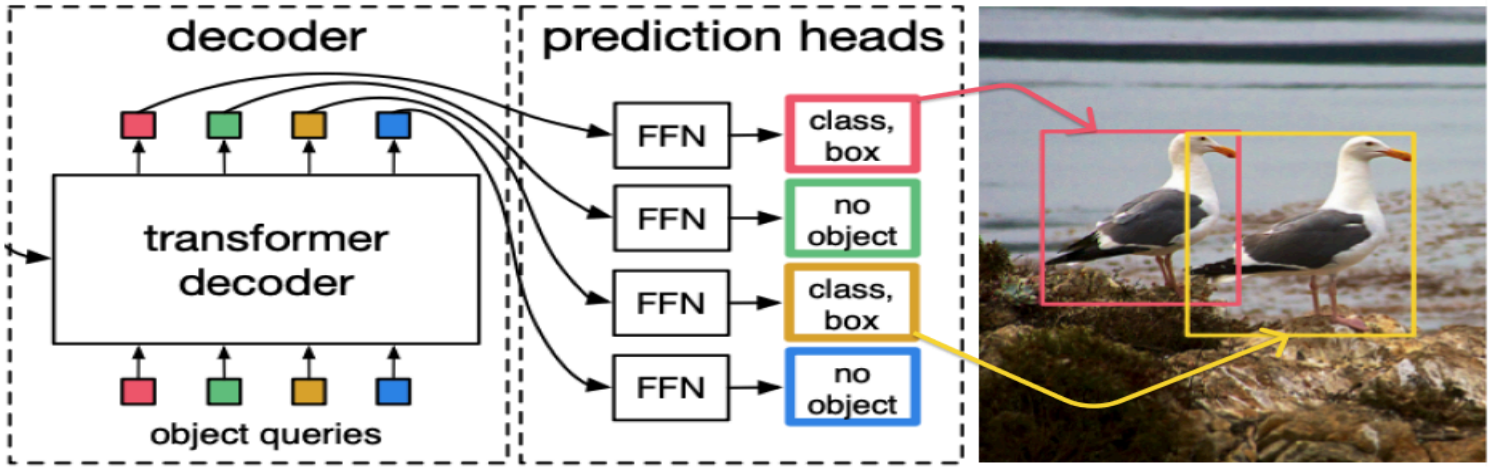


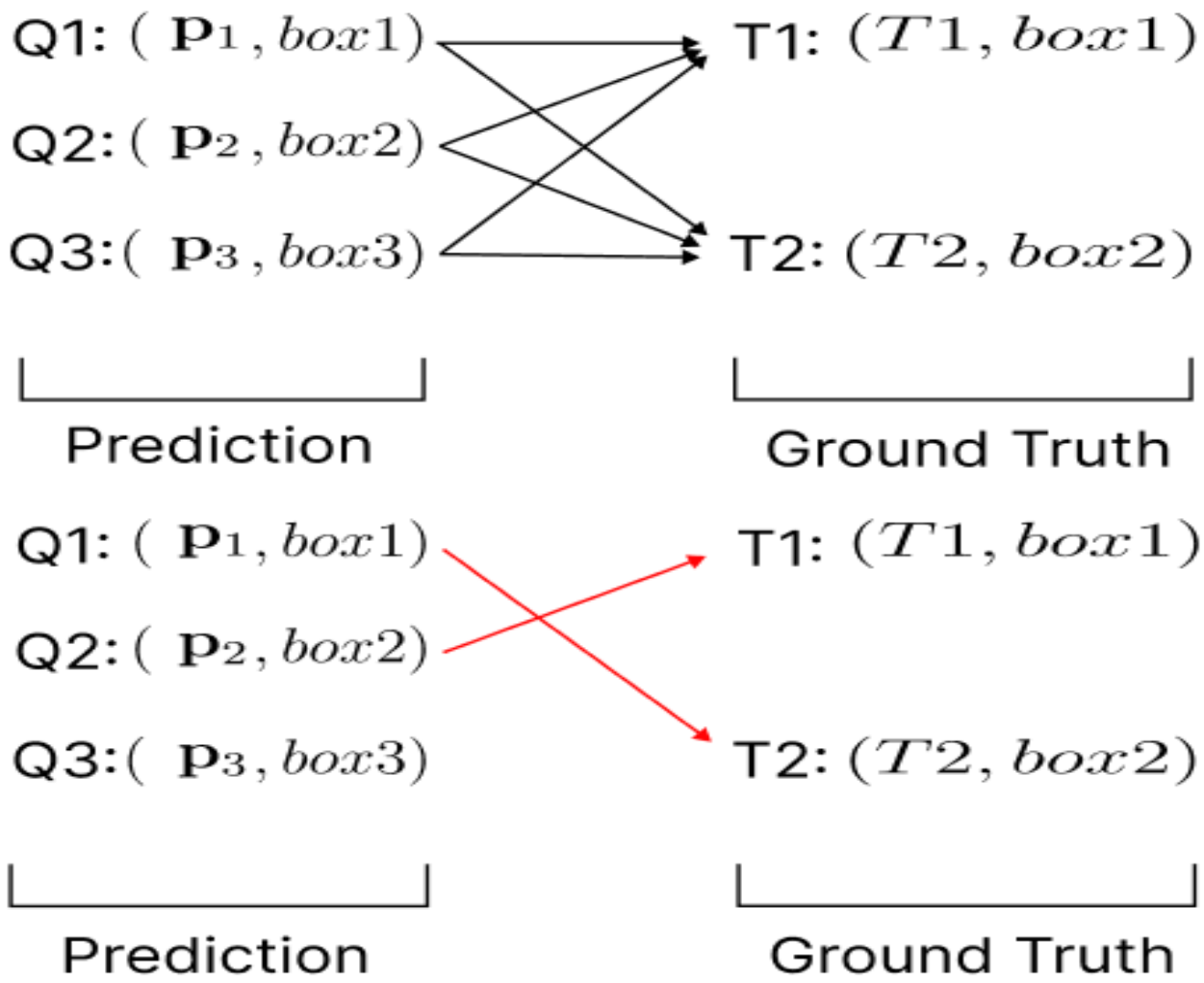
Figure: Each object query is fed into the decoder to predict the objects in the image.

Object query

A small, fixed number of learned positional embeddings[8].

[8] Carion et al., see n. 7.

Training


Q1: ($\mathbf{p}_1, box1$)
Q2: ($\mathbf{p}_2, box2$)
Q3:($\mathbf{p}_3, box3$)
T1: $(T1, box1)$
T2: $(T2, box2)$
Prediction
Ground Truth
Q1: ($\mathbf{p}_1, box1$)
Q2: ($\mathbf{p}_2, box2$)
Q3:($\mathbf{p}_3, box3$)
T1: $(T1, box1)$
T2: $(T2, box2)$
Prediction
Ground Truth

Training

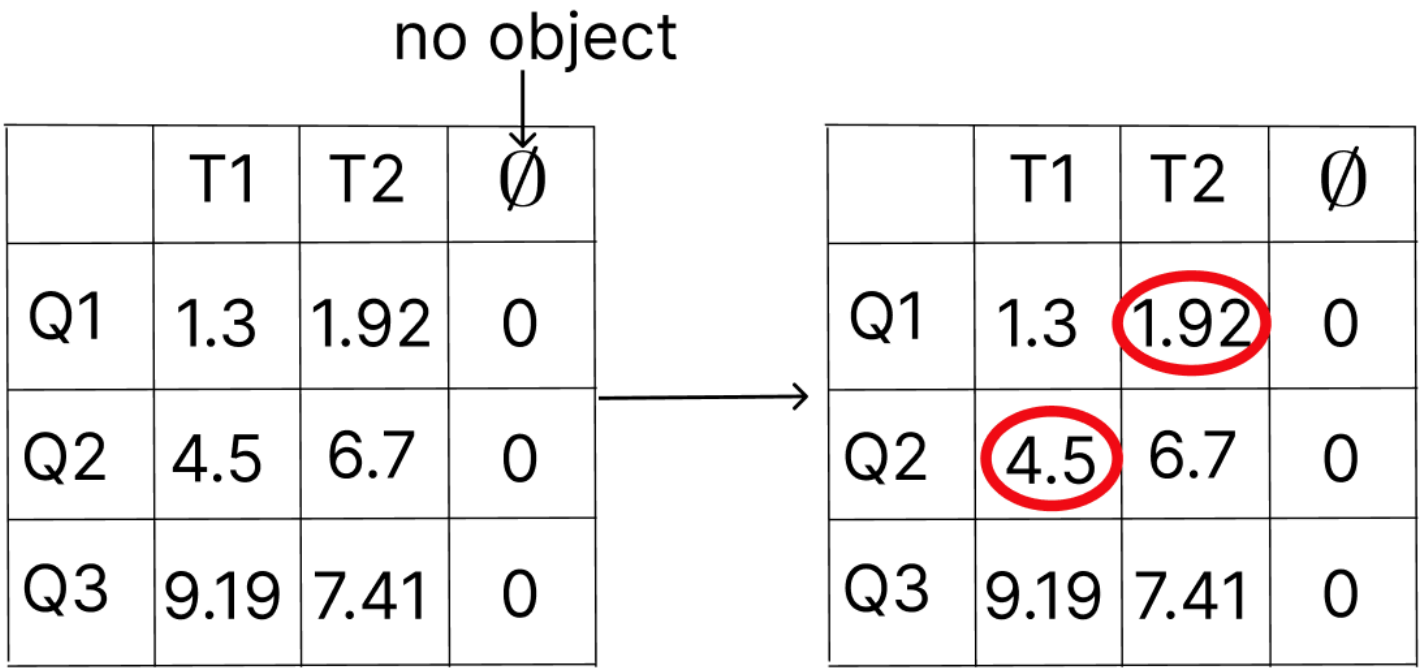


Figure: Find the pair minimize loss cost 6.42

Inference

Set threshold 0.5 as C :

$$
\begin{aligned}
Q_1 &= (\mathbf{p}_1, \text{box1}) \quad \text{where} \quad \mathbf{p}_1 = (p_{1,1}, p_{1,2}, \ldots, p_{1,n}) \\
Q_2 &= (\mathbf{p}_2, \text{box2}) \quad \text{where} \quad \mathbf{p}_2 = (p_{2,1}, p_{2,2}, \ldots, p_{2,n}) \\
Q_3 &= (\mathbf{p}_3, \text{box3}) \quad \text{where} \quad \mathbf{p}_3 = (p_{3,1}, p_{3,2}, \ldots, p_{3,n}) \\
Q &= \{Q_1, Q_2, Q_3\}
\end{aligned}
$$

The prediction is valid if the maximum probability in any vector exceeds C :

$$
\text{Prediction} = \left\{ Q_i \mid \max_j \{p_{i,j}\} > C \quad \text{for} \quad i \in \{1, 2, 3\} \right\}
$$

The comparesion of YOLO and DETR

YOLO uses a dense prediction approach, while DETR utilizes a set prediction method for object detection. $Head(S \times S \times B) \gg N$

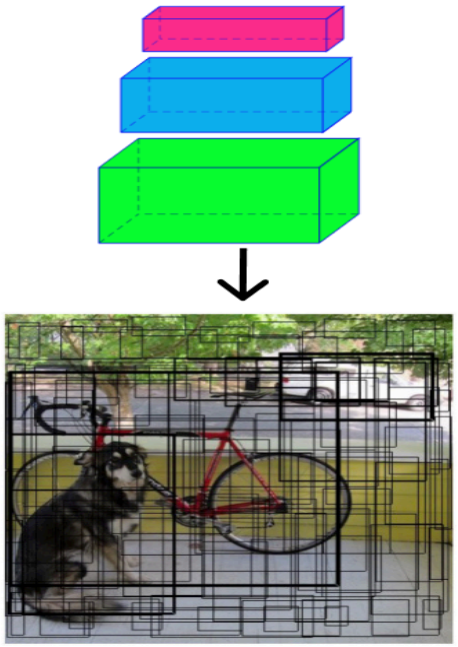


Figure: The number of YOLO Prediction $Head(S \times S \times B)$.

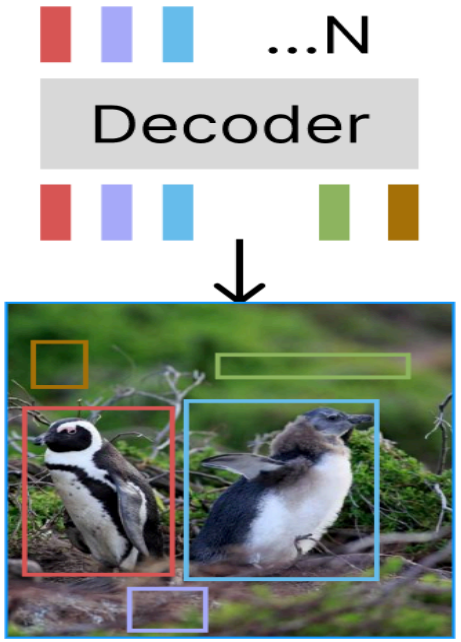


Figure: The number of DETR Prediction N.

RT-DETR

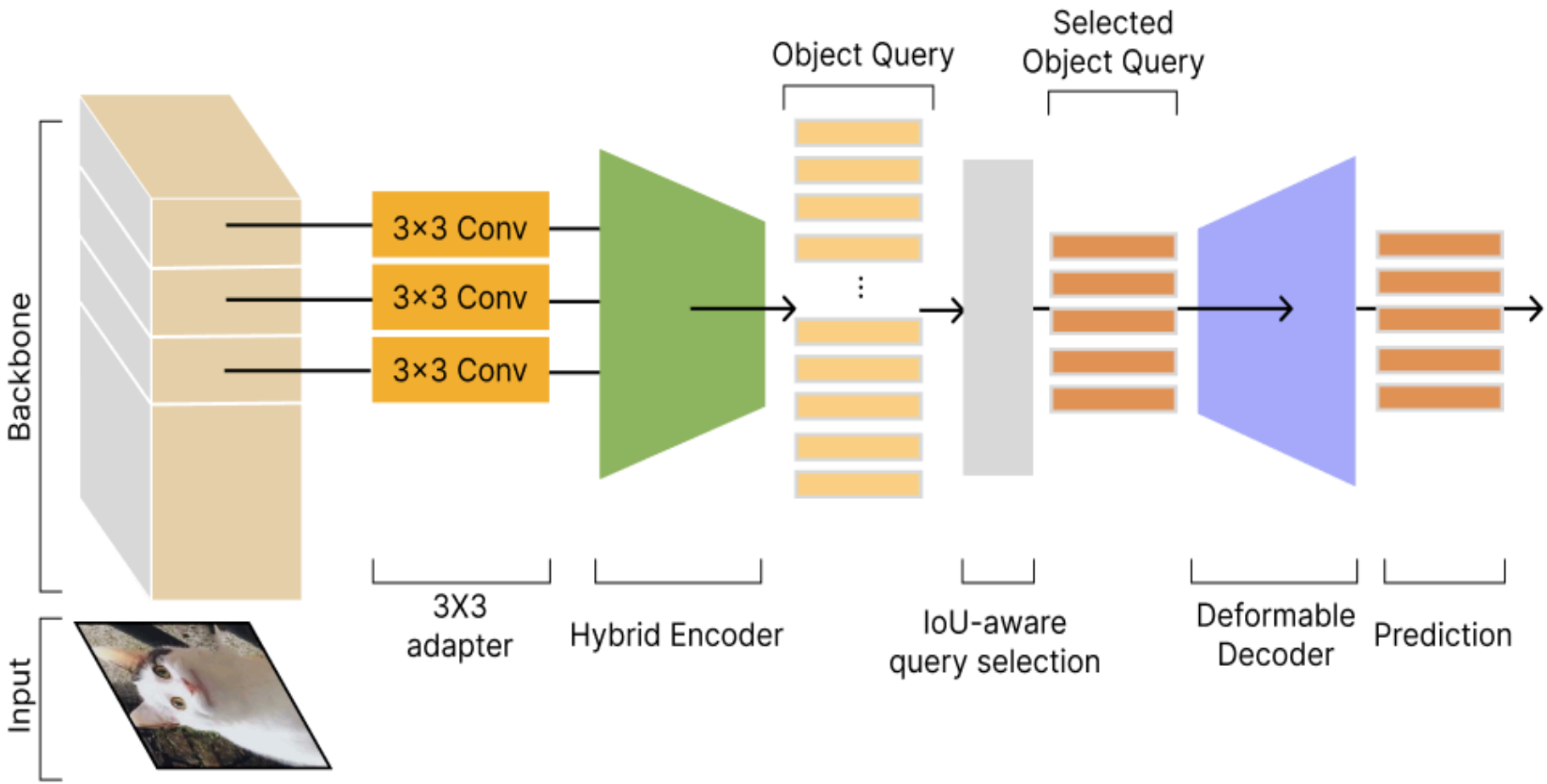


Figure: Real-Time Detection Transformer (RT-DETR)[9]

[9] Yian Zhao et al. "Detrs beat yolos on real-time object detection". In: *arXiv preprint arXiv:2304.08069* (2023).

RT-DETR Ours

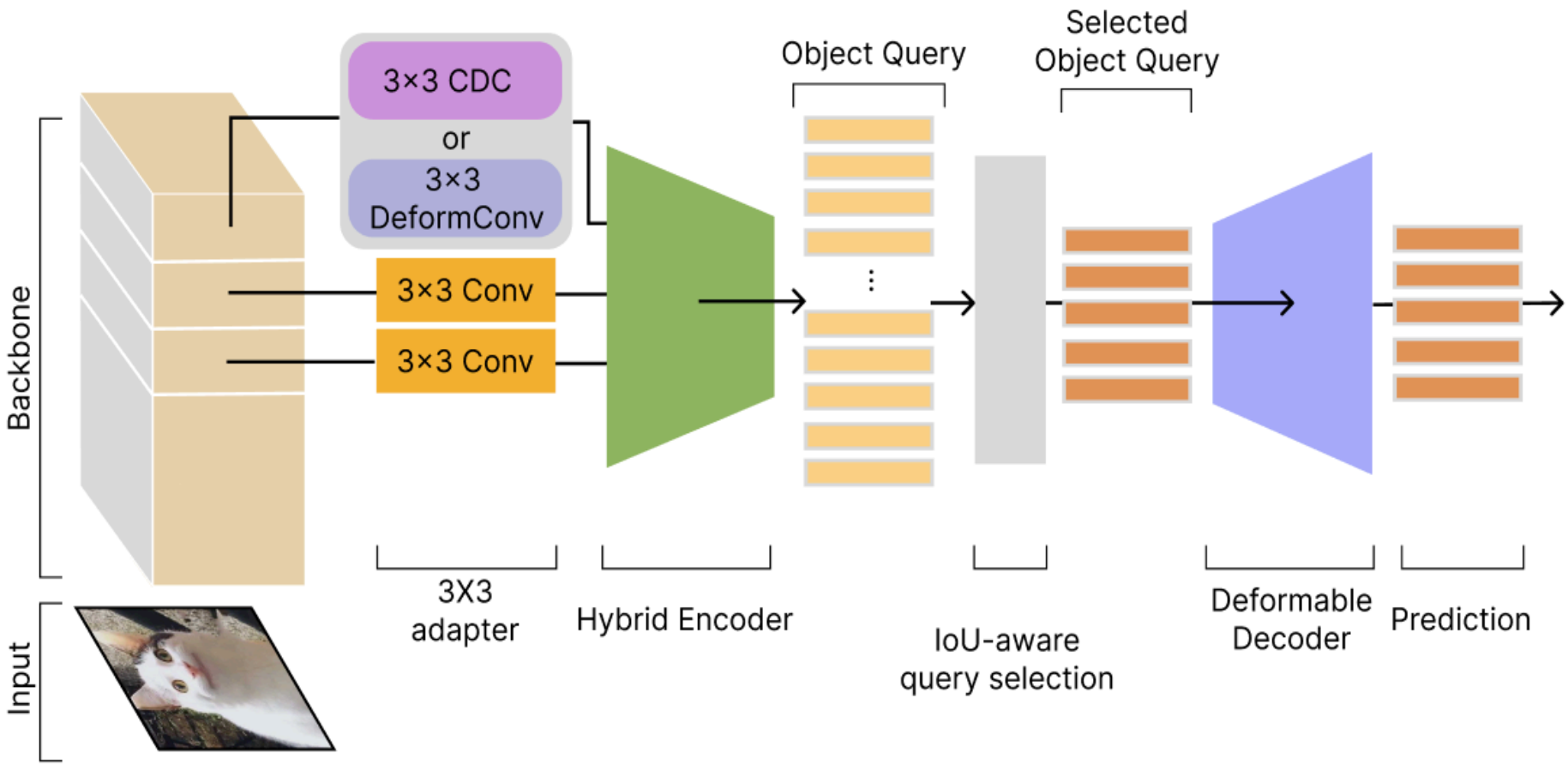
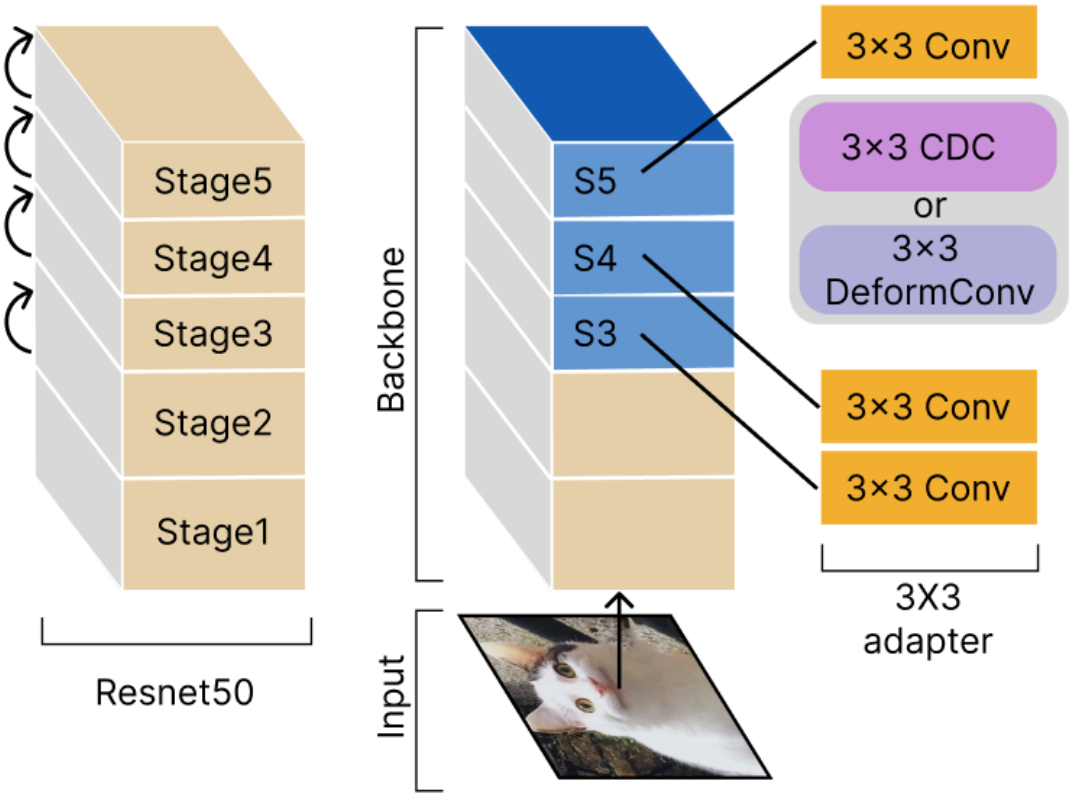


Figure: RT-DETR with Central Difference Convolution (CDC) and Deformable Convolution (DeformConv)

RT-DETR Backbone



- ResNet-50 pretrained on ImageNet.
- CDC and DeformConv at $S5$ adapter.

Figure: ResNet-50 as Backbone

Vanilla Convolution

$$y(p_0) = \sum_{p_n \in R} w(p_n) \cdot x(p_0 + p_n)$$

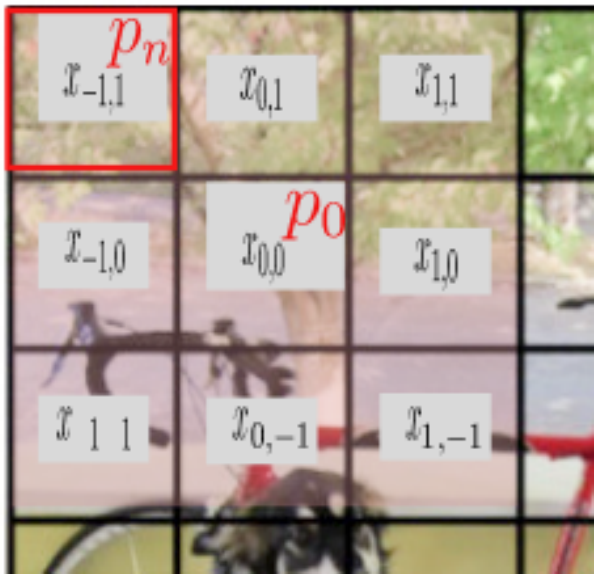
R

p_n $(-1,1)$	$(0,1)$	$(1,1)$
$(-1,0)$	p_0 $(0,0)$	$(1,0)$
$(-1,-1)$	$(0,-1)$	$(1,-1)$

w

p_n $w_{-1,1}$	$w_{0,1}$	$w_{1,1}$
$w_{-1,0}$	p_0 $w_{0,0}$	$w_{1,0}$
$w_{-1,-1}$	$w_{0,-1}$	$w_{1,-1}$

x



Central Difference Convolution (CDC)

$$y(p_0) = \sum_{p_n \in R} w(p_n) \cdot (x(p_0 + p_n) - x(p_0))$$

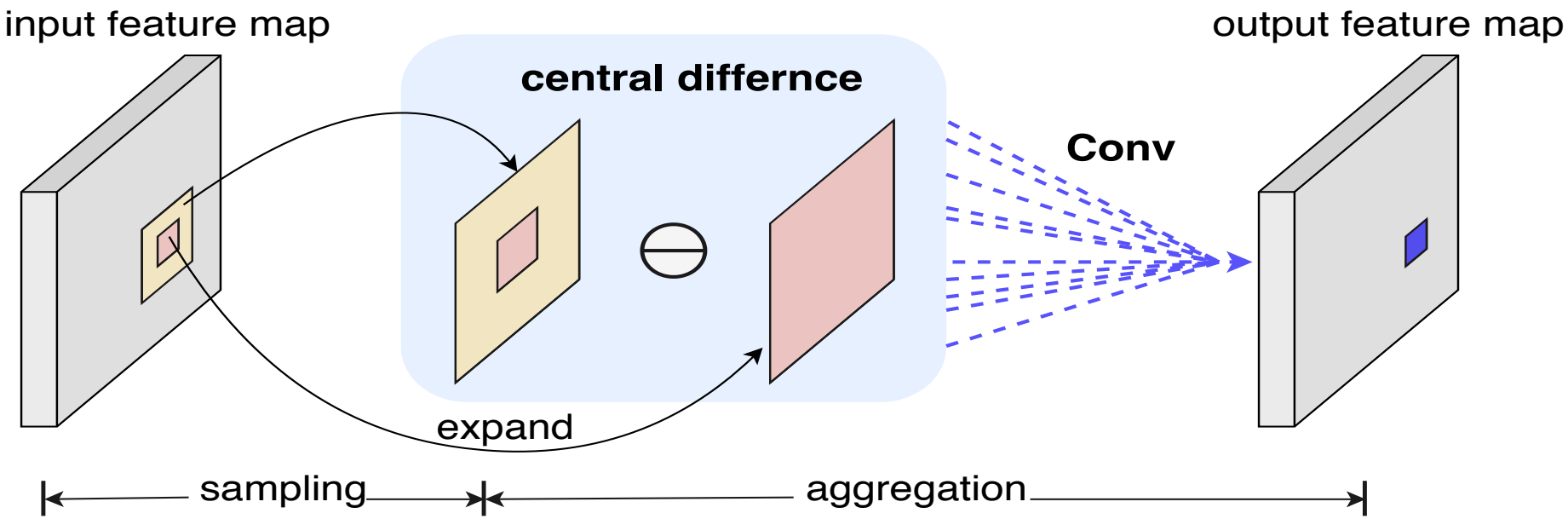


Figure: Central Difference Convolution[10]

[10] Zitong Yu et al. *Searching Central Difference Convolutional Networks for Face Anti-Spoofing*. 2020. arXiv: 2003.04092 [cs.CV].

Central Difference Convolution (CDC)

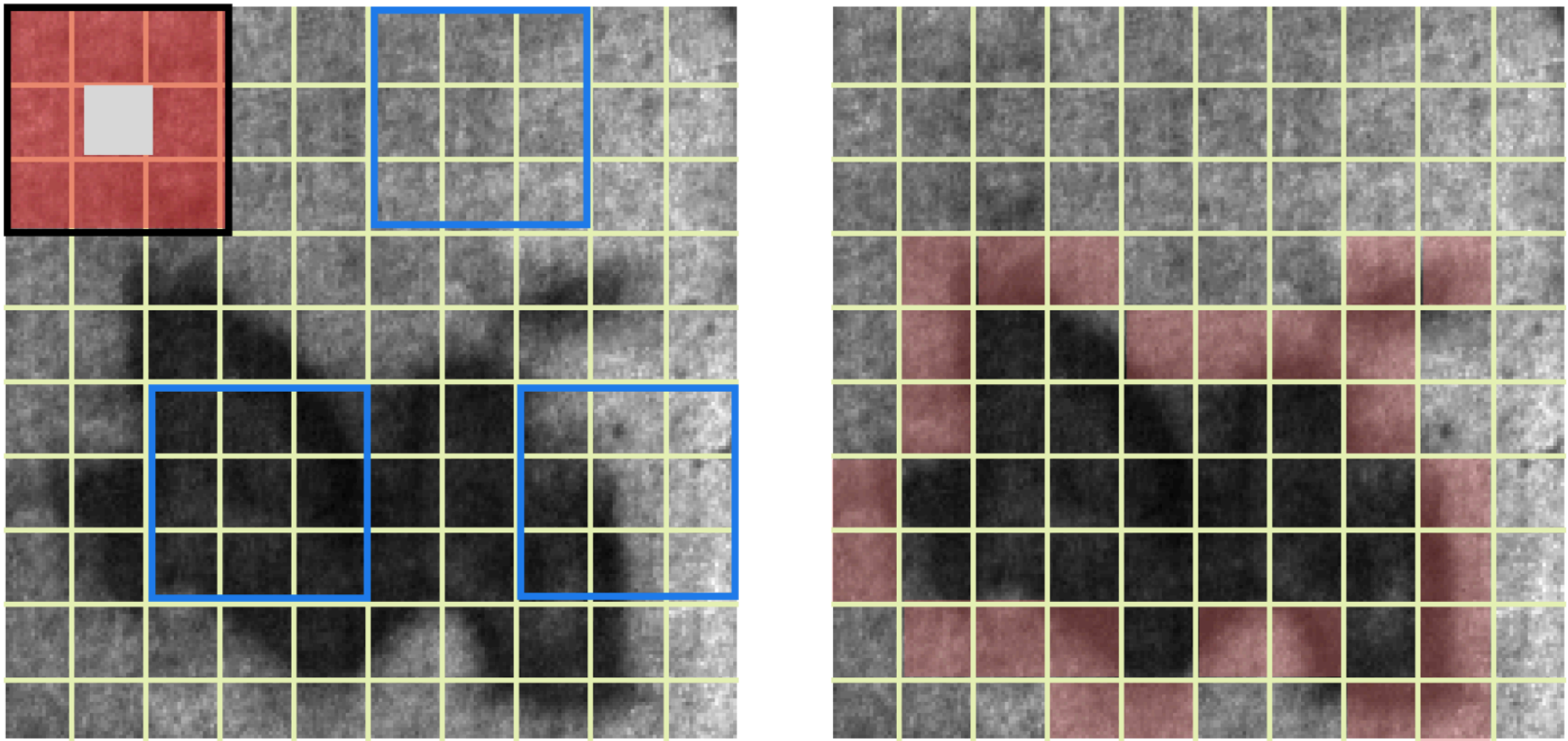


Figure: CDC kernel slid over the defect object.

Deformable Convolution (DeformConv)

$$y(p_0) = \sum_{p_n \in R} w(p_n) \cdot x(p_0 + p_n + \triangle p_n)$$

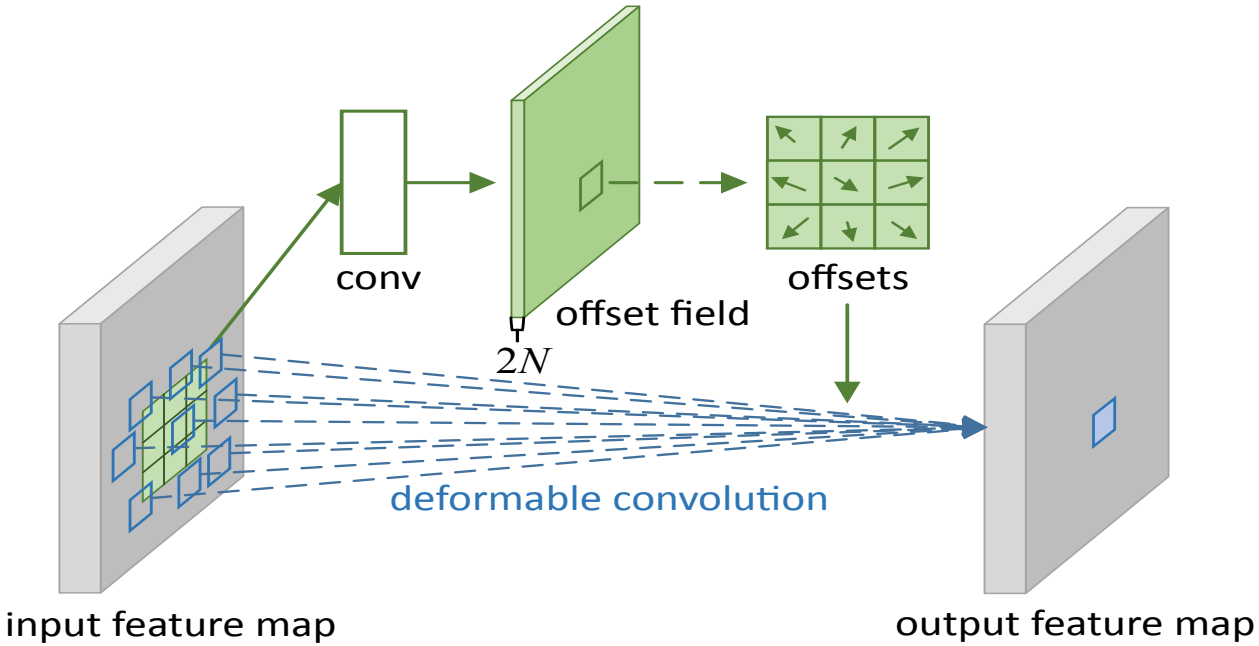


Figure: Deformable Convolution[11]

[11] Jifeng Dai et al. "Deformable convolutional networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 764–773.

Deformable Convolution (DeformConv)

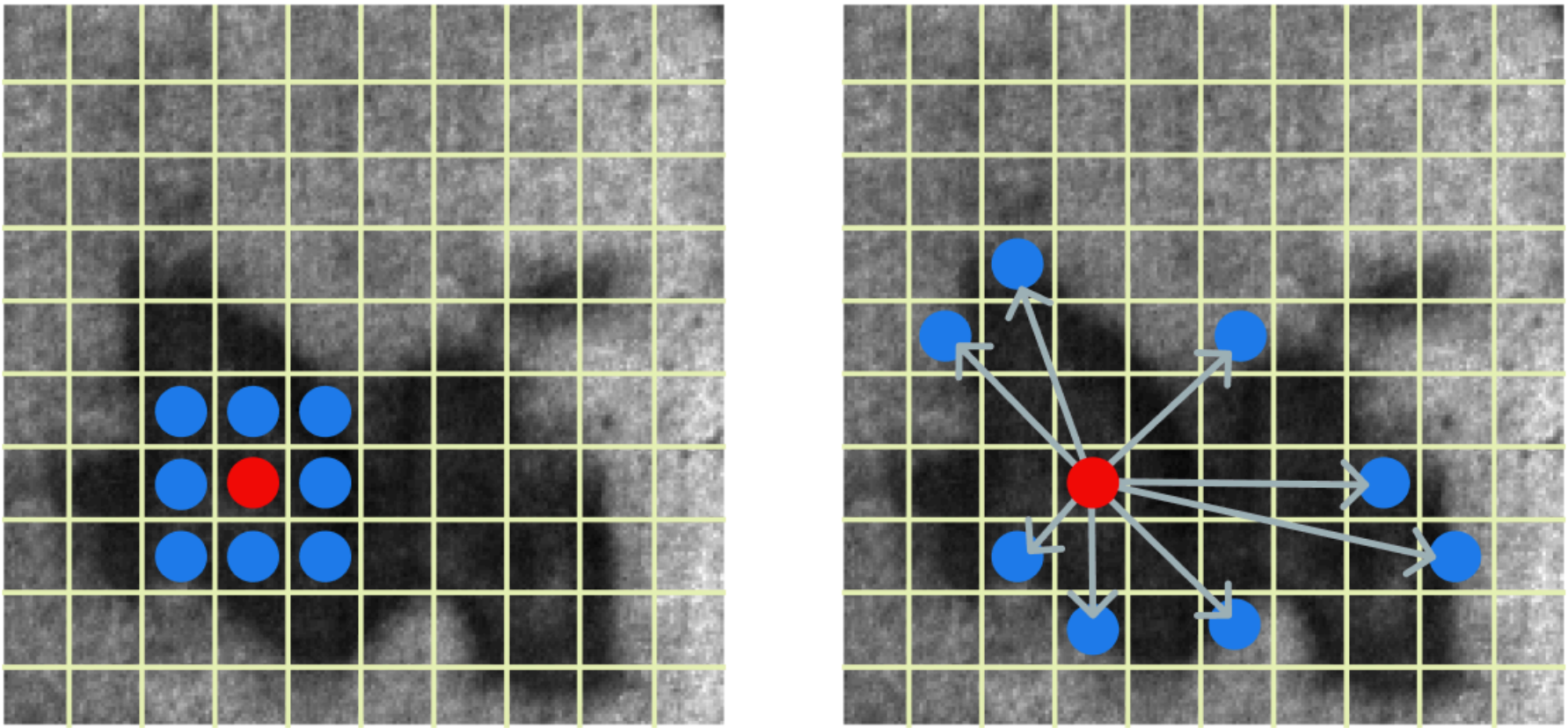


Figure: DeformConv kernel slid over the defect object.

Efficient Hybrid Encoder

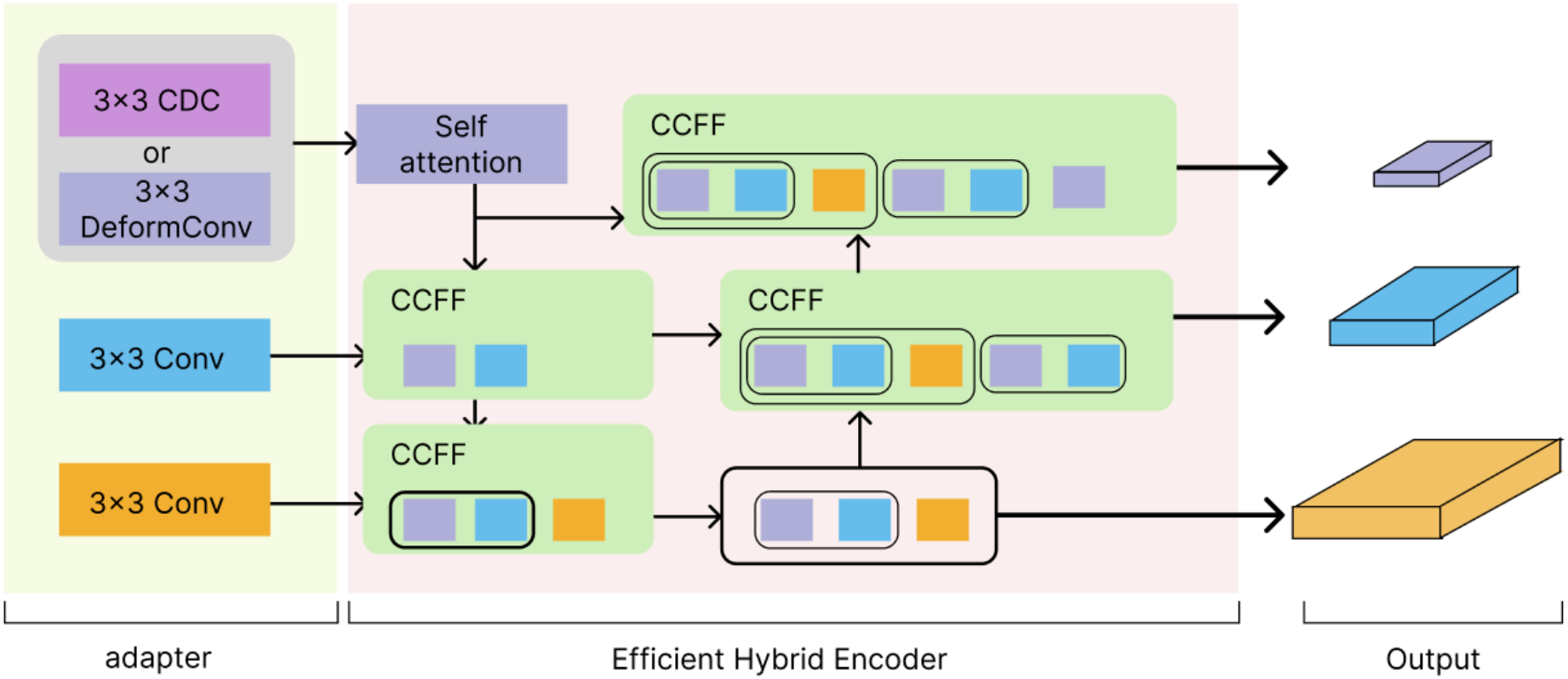


Figure: Efficient Hybrid Encoder[12]

[12] Zhao et al., see n. 9.

Efficient Hybrid Encoder-CCFF

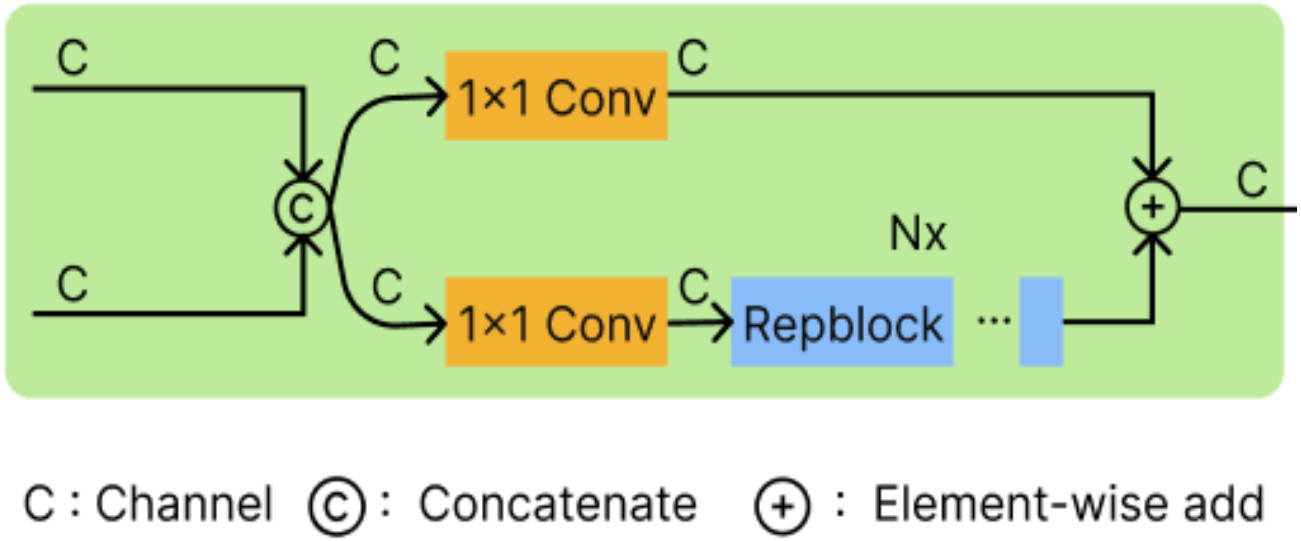


Figure: Cross-scale Feature-fusion Module (CCFF)[13]

[13] Zhao et al., see n. 9.

IoU-Aware Query Selection

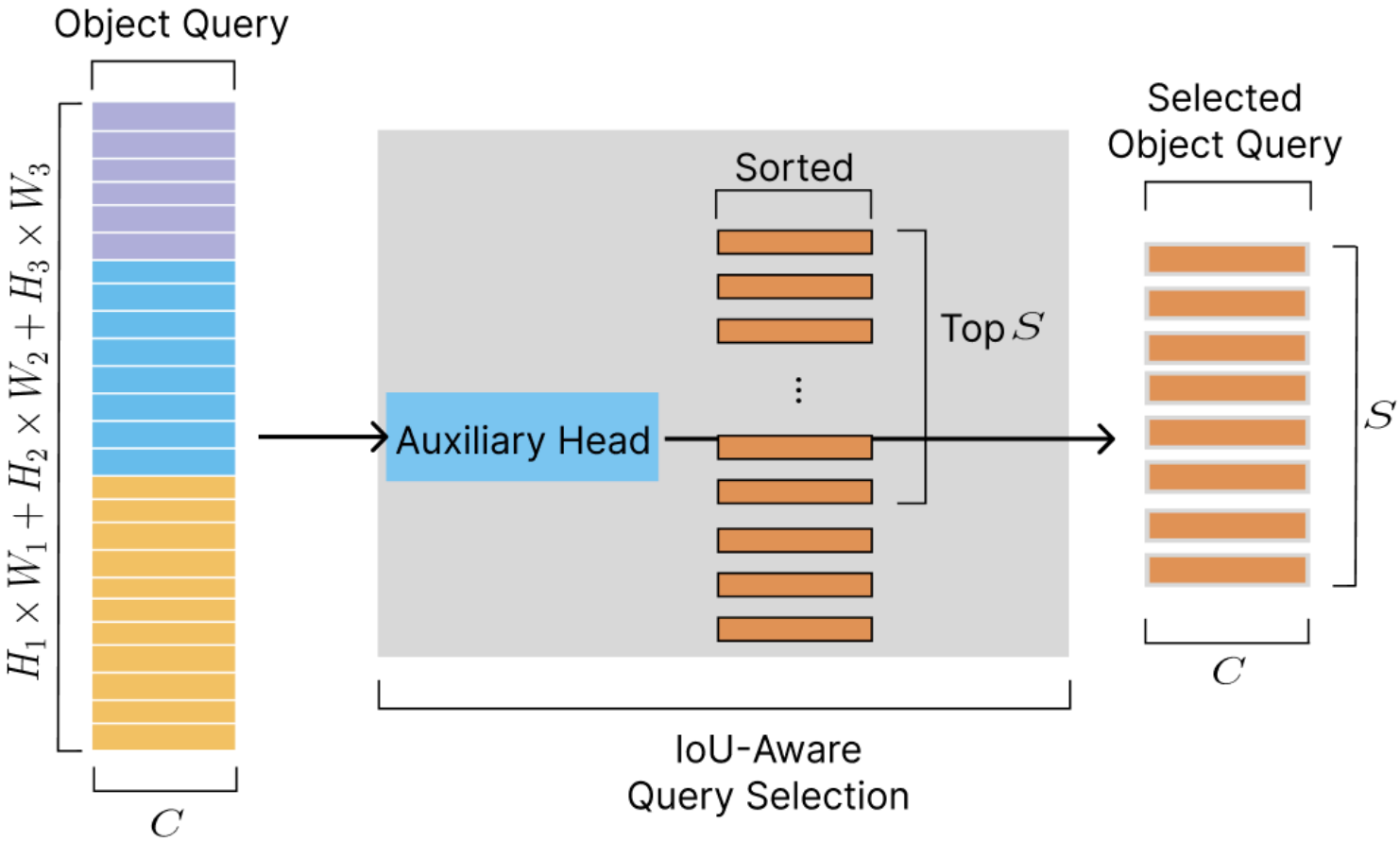


Figure: Selected Top S as Object Queries[14]

[14] Zhao et al., see n. 9.

Deformable Decoder

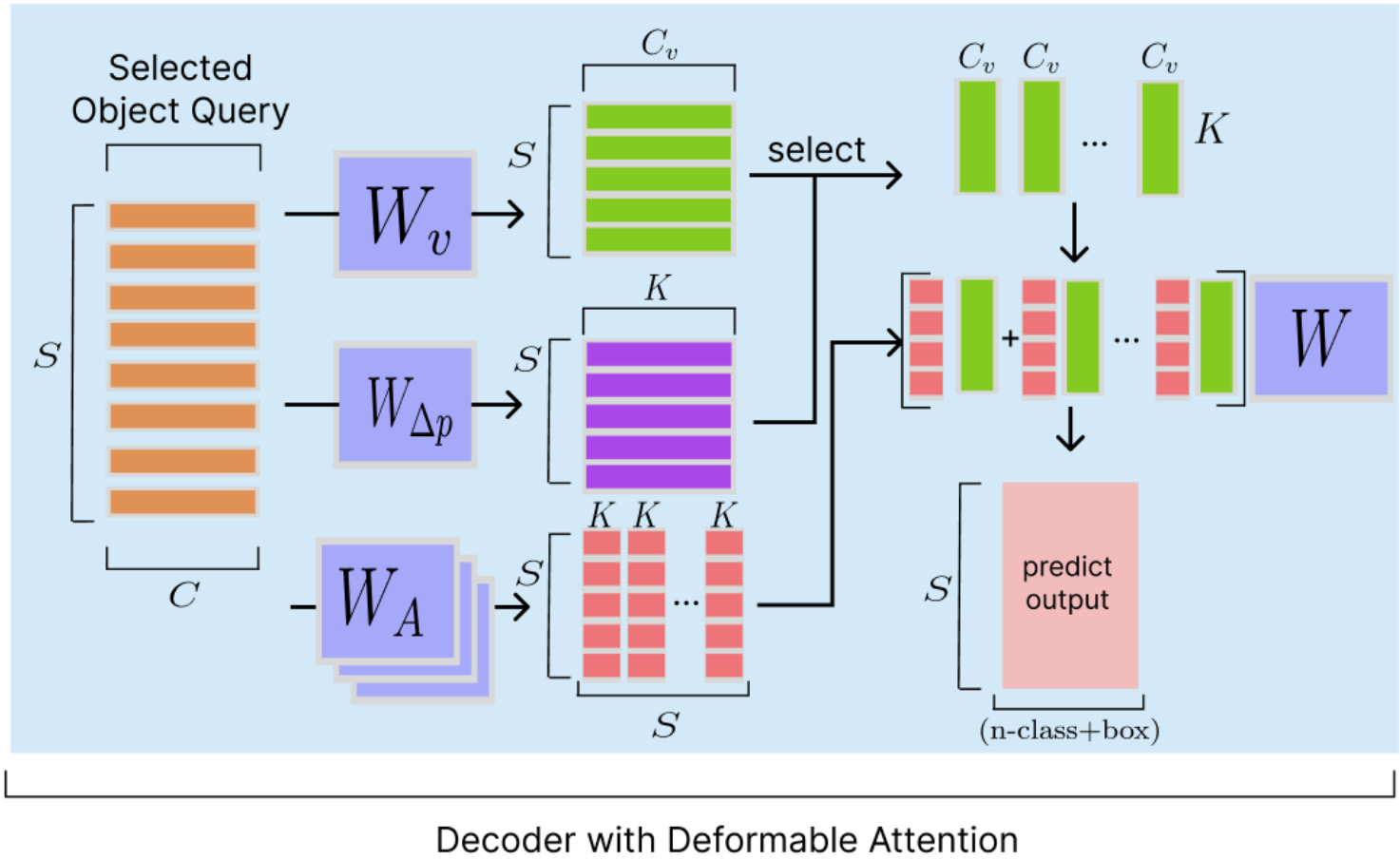


Figure: Decoder with Deformable Attention[15].

[15] Xizhou Zhu et al. "Deformable detr: Deformable transformers for end-to-end object detection". In: *arXiv preprint arXiv:2010.04159* (2020).

Loss Function

The loss function employed in this study is identical to that used in the RT-DETR model. The loss function can be decomposed into two components: box loss and classification loss[16], with a weight ratio of 5:1.

$$\begin{aligned}\mathcal{L} &= 5\mathcal{L}_{box} + \mathcal{L}_{class} \\ &= 5\mathcal{L}_1 + VFL\end{aligned}$$

[16] Haoyang Zhang et al. *VarifocalNet: An IoU-aware Dense Object Detector*. 2021. arXiv: 2008.13367 [cs.CV].

Prediction and Ground Truth

The prediction of a object query and ground truth.

$$\underbrace{\begin{matrix} \textit{classification} \\ \textit{box} \end{matrix}\left\{\begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ \hat{x} \\ \hat{y} \\ \hat{w} \\ \hat{h} \end{bmatrix}\right.}_{\text{Prediction}} = d \qquad \underbrace{\begin{matrix} \textit{classification} \\ \textit{box} \end{matrix}\left\{\begin{bmatrix} 0 \\ 1 \\ 0 \\ x \\ y \\ w \\ h \end{bmatrix}\right.}_{\text{Ground truth}} = t$$

L1 Loss Function (MAE)

The box loss of a object query.

$$d^{box} = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{w} \\ \hat{h} \end{bmatrix} \quad t^{box} = \begin{bmatrix} x \\ y \\ w \\ h \end{bmatrix}$$

$$\mathcal{L}_1 = \sum_{i=1}^{4} \frac{|d_i^{box} - t_i^{box}|}{4}$$

Varifocal Loss (VFL)

Preprocess ground truth.

$$t^{class} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \rightarrow t^{class'} = \begin{bmatrix} 0 \\ \text{IoU} \\ 0 \end{bmatrix}$$

$$\text{IoU} = \frac{\cap}{\text{Ground Truth} \cup \text{Prediction}}$$

Varifocal Loss (VFL)

The classification loss of a object query.

$$d^{class} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} \; t^{class'} = \begin{bmatrix} 0 \\ \text{IoU} \\ 0 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

$$\text{VFL}(\text{p}, \text{q}) = \begin{cases} -\text{q}(\text{q}\log(\text{p}) + (1-\text{q})\log(1-\text{p})) & \text{q} > 0, \\ -\alpha \text{p}^{\gamma} \log(1-\text{p}) & \text{q} = 0 \end{cases}$$

where the value of $\alpha = 0.2$ and $\gamma = 2$ in the experiment[17].

[17] Zhao et al., see n. 9.

Experimental Result NEU-DET

Table: The experiments are conducted on the NEU-DET dataset.

Model	AP0.5-0.95	AP0.5	AP0.75	AP-medium	AP-large
RT-DETR	0.305	0.563	0.300	0.259	0.333
RT-DETR+CDC	0.287	0.532	0.260	0.255	0.311
RT-DETR+DeformConv	0.273	0.515	0.247	0.269*	0.291
Model	AR0.5-0.95	AR0.5	AR0.75	AR-medium	AR-large
RT-DETR	0.253	0.518	0.652	0.602	0.678
RT-DETR+CDC	0.245	0.51	0.646	0.6	0.674
RT-DETR+DeformConv	0.232	0.51	0.646	0.58	0.679*

Experimental Result COCO

Table: The experiments are conducted on the COCO dataset Average Precision.

Model	AP0.5-0.95	AP0.5	AP0.75
RT-DETR	0.517	0.7	0.558
RT-DETR+CDC	0.517	0.699	0.558
RT-DETR+DeformConv	0.464	0.634	0.504
Model	AP-small	AP-medium	AP-large
RT-DETR	0.343	0.562	0.692
RT-DETR+CDC	0.331	0.563*	0.69
RT-DETR+DeformConv	0.295	0.506	0.621

Experimental Result COCO

Table: The experiments are conducted on the COCO dataset Average Recall.

Model	AR0.5-0.95	AR0.5	AR0.75
RT-DETR	0.387	0.648	0.709
RT-DETR+CDC	0.385	0.646	0.711*
RT-DETR+DeformConv	0.366	0.616	0.683
Model	**AR-small**	**AR-medium**	**AR-large**
RT-DETR	0.541	0.752	0.876
RT-DETR+CDC	0.534	0.757*	0.874
RT-DETR+DeformConv	0.484	0.73	0.866

Conclusion

- Research explores CDC and DeformConv on RT-DETR.
- In defect detection tasks, DeformConv might be the better choice due to its higher recall.
- For general object detection, CDC improved RT-DETR's ability to detect medium-sized objects.
- CDC and DeformConv modules enhance RT-DETR for specific object sizes and applications.

Appendix

- Non-Maximum Suppression (NMS)
- Deformable Attention
- Hungarian Algorithm
- CDC
- Auxiliary Head
- Up/Down Sampling

Non-Maximum Suppression

Algorithm 1: Non-Maximum Suppression

Input: $B = \{b_1, \cdots, b_N\}, S = \{s_1, \cdots, s_N\}, N_t$

B is the list of initial detection boxes

S contains corresponding detection scores

N_t is the NMS threshold

begin

$D \leftarrow \{\}$;

while $B \neq$ *empty* **do**

$m \leftarrow \arg\max S$; $M \leftarrow b_m$;

$D \leftarrow D \cup M$; $B \leftarrow B - M$;

for b_i **to** B **do**

$s_i \leftarrow s_i(1 - iou(M, b_i))$

end

end

return D, S

end

Self Attention and Deformable Attention

Let

$$I^{s\times c}, \{W_q, W_v, W_k\} \in \mathbb{R}^{c\times c_v}$$

Self Attention

$$Q = IW_q, K = IW_k, V = IWv \in \mathbb{R}^{s\times c_v}$$
$$A^{c_v\times c_v} = K^T Q$$
$$O^{s\times c_v} = VA$$

Self Attention and Deformable Attention

Deformable Attention

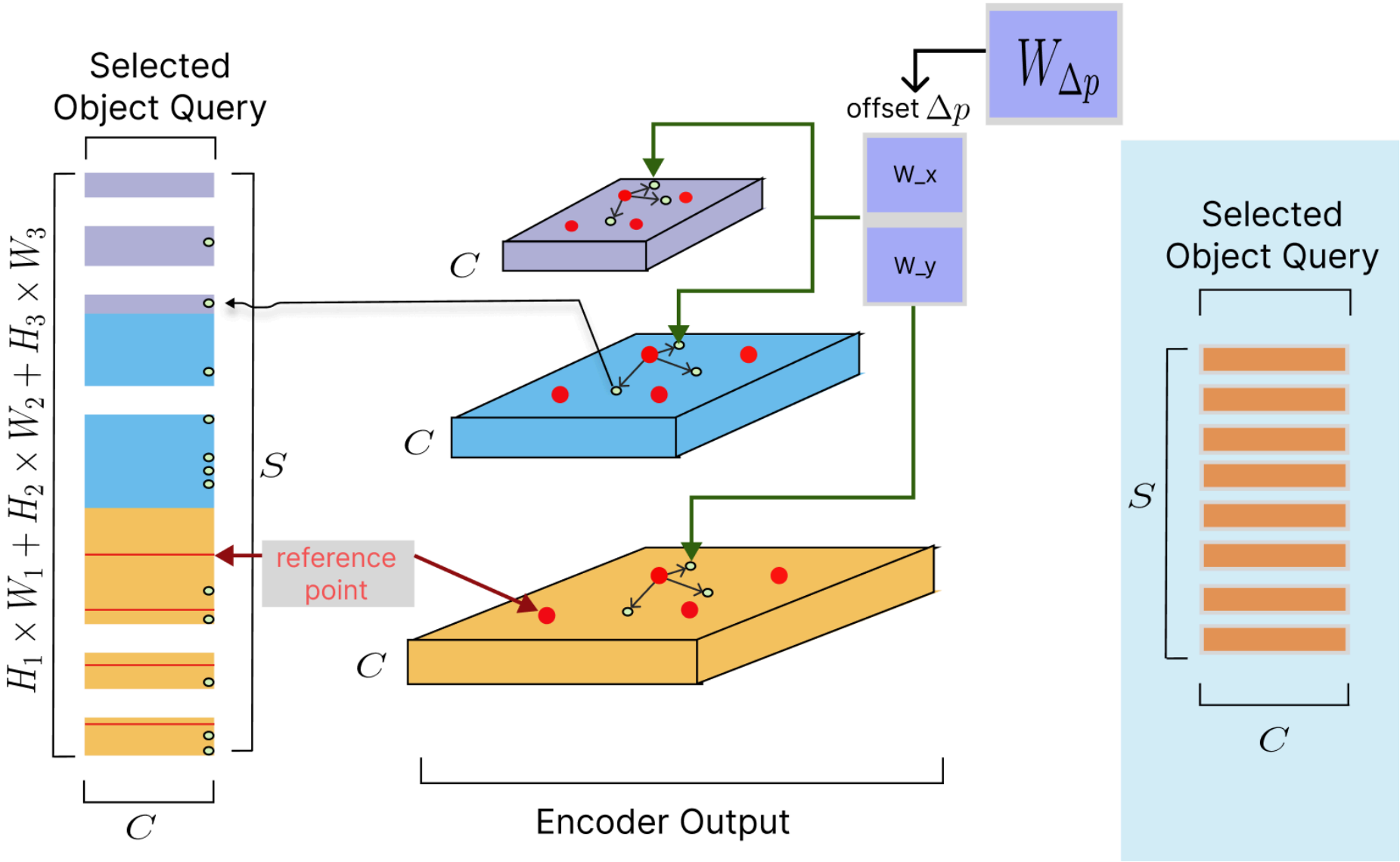
$$V^{\,s\times c_v} = IW_v^{\,c\times c_v} \quad \Delta_p = IW_{\Delta_p}^{\,c\times k} \quad A_i{}^{s\times k} = IW_{A_i}^{\,s\times k}$$

$$V \xrightarrow{\text{select by } \Delta_p} V_1,\, V_2, \cdots,\, V_s^{\,k\times c_v}$$

$$O^{\,s\times c_v} = \left[A_1\, V_1 + A_2\, V_2 + \cdots + A_s\, V_s\right] W^{\,c_v\times c_v}$$

where $i \in [1, 2, 3, \cdots, s]$

Deformable Attention More Detail



Deformable Attention More Detail

Deformable Attention

$$V^{\,s\times c_v} = IW_v^{\,c\times c_v} \quad \Delta_p = IW_{\Delta_x}^{c\times pk}, IW_{\Delta_y}^{c\times pk} \quad A_i^{\,s\times pk} = IW_{A_i}^{s\times pk}$$

$$V \xrightarrow{\text{select by } \Delta_p} V_1, V_2, \cdots, V_s^{\,pk\times c_v}$$

$$O^{\,s\times c_v} = \left[A_1 V_1 + A_2 V_2 + \cdots + A_s V_s\right] W^{\,c_v\times c_v}$$

where $i \in [1, 2, 3, \cdots, s]$

Hungarian Algorithm

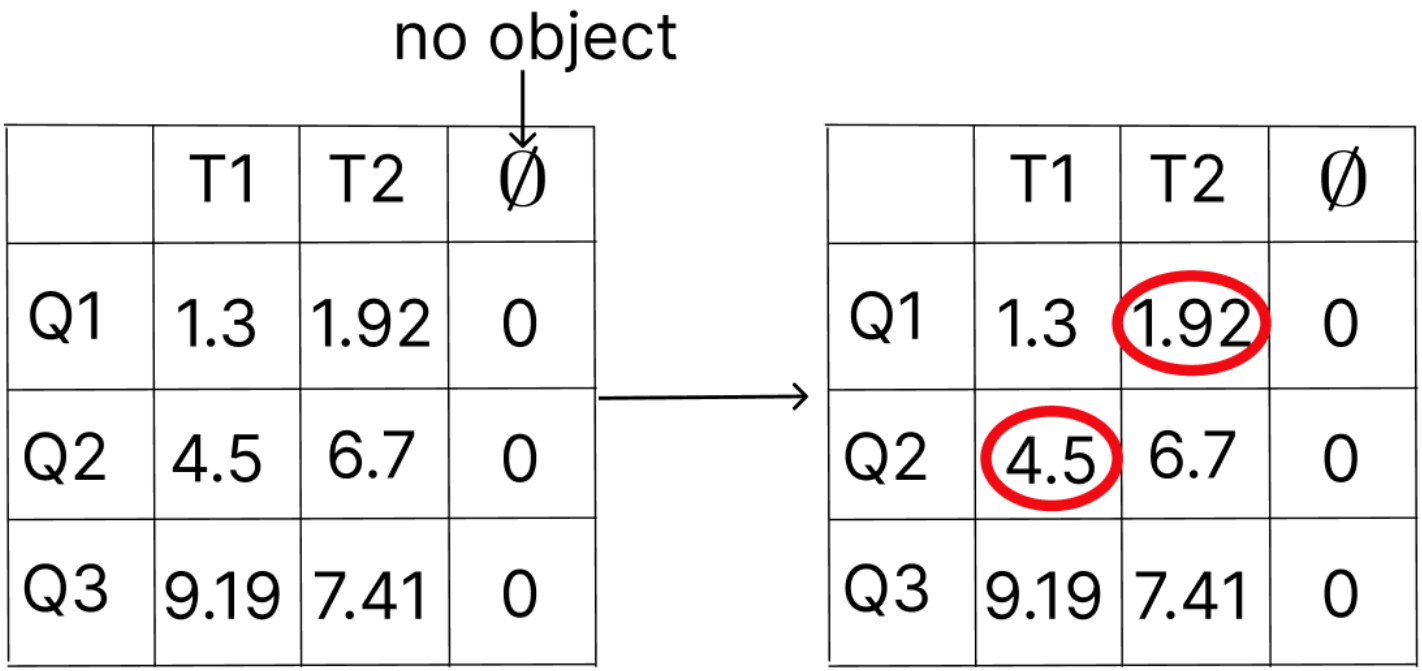


Figure: Find the pair minimize cost

Hungarian Algorithm

Preprocess matrix

	T1	T2	∅
Q1	1	2	0
Q2	4	6	0
Q3	9	7	0

subtract row minima

subtract col minima

	T1	T2	∅
Q1	0	0	0
Q2	3	4	0
Q3	8	5	0

Hungarian Algorithm

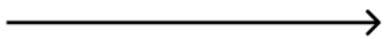
Repeat the followings:

A. Cover all the zeros with a minimum number of lines.
B. Decide whether to stop.
C. Create additional zeros.

Hungarian Algorithm

- Cover all 0 with minimum of lines
- If n lines required then stop else C

	T1	T2	Ø
Q1	0	0	0
Q2	3	4	0
Q3	8	5	0

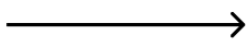


	T1	T2	Ø
Q1	~~0~~	~~0~~	~~0~~
Q2	3	4	0
Q3	8	5	0

Hungarian Algorithm

1. Find smallest element k not cover by lines
2. Subtract k from all uncovered elements
3. add k to all the elements that are covered twice.

	T1	T2	Ø
Q1	0	0	0_{+3}
Q2	3_{-3}	4_{-3}	0
Q3	8_{-3}	5_{-3}	0



	T1	T2	Ø
Q1	0	0	3
Q2	0	1	0
Q3	5	2	0

Hungarian Algorithm

	T1	T2	Ø
Q1	0	0	3
Q2	0	1	0
Q3	5	2	0

n lines required
stop

	T1	T2	Ø
Q1	0	0	3
Q2	0	1	0
Q3	5	2	0

Hungarian Algorithm

	T1	T2	Ø
Q1	0	0	3
Q2	0	1	0
Q3	5	2	0

output matrix

Q1 ⟶ T2

Q2 ⟶ T1

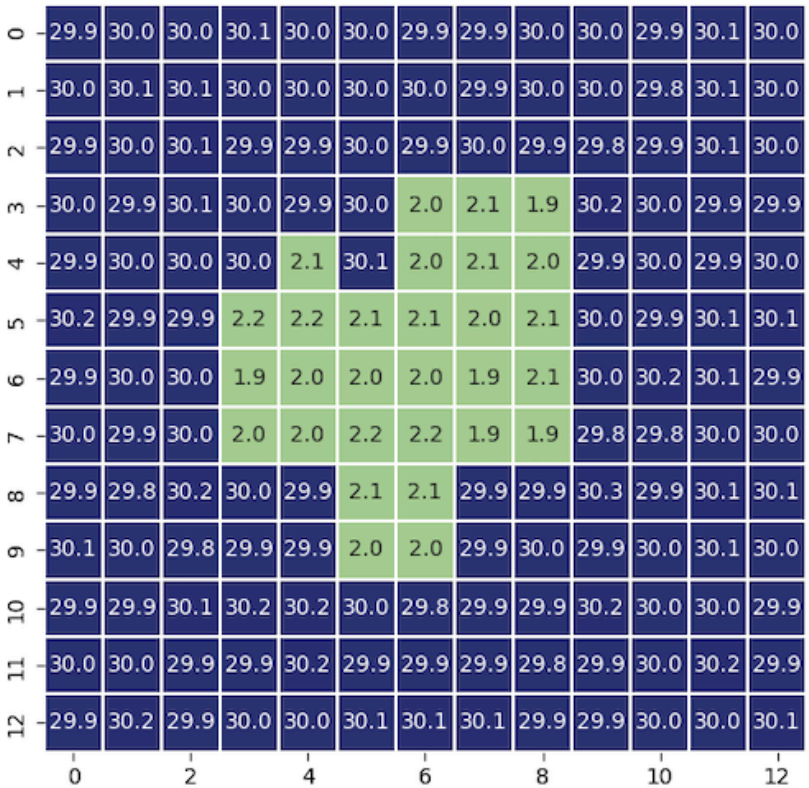
Q3 ⟶ Ø

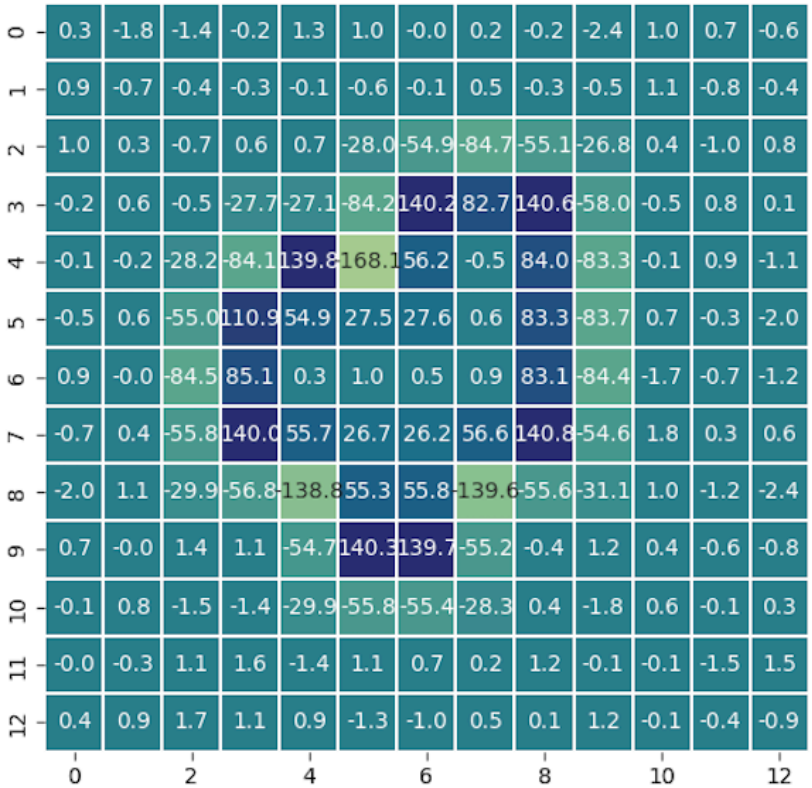
minimum cost is 4+2

	T1	T2	Ø
Q1	1	2	0
Q2	4	6	0
Q3	9	7	0

original matrix

CDC


0 29.9 30.0 30.0 30.1 30.0 30.0 29.9 29.9 30.0 30.0 29.9 30.1 30.0
1 30.0 30.1 30.1 30.0 30.0 30.0 30.0 29.9 30.0 30.0 29.8 30.1 30.0
2 29.9 30.0 30.1 29.9 29.9 30.0 29.9 30.0 29.9 29.8 29.9 30.1 30.0
3 30.0 29.9 30.1 30.0 29.9 30.0 2.0 2.1 1.9 30.2 30.0 29.9 29.9
4 29.9 30.0 30.0 30.0 2.1 30.1 2.0 2.1 2.0 29.9 30.0 29.9 30.0
5 30.2 29.9 29.9 2.2 2.2 2.1 2.1 2.0 2.1 30.0 29.9 30.1 30.1
6 29.9 30.0 30.0 1.9 2.0 2.0 2.0 1.9 2.1 30.0 30.2 30.1 29.9
7 30.0 29.9 30.0 2.0 2.0 2.2 2.2 1.9 1.9 29.8 29.8 30.0 30.0
8 29.9 29.8 30.2 30.0 29.9 2.1 2.1 29.9 29.9 30.3 29.9 30.1 30.1
9 30.1 30.0 29.8 29.9 29.9 2.0 2.0 29.9 30.0 29.9 30.0 30.1 30.0
10 29.9 29.9 30.1 30.2 30.2 30.0 29.8 29.9 29.9 30.2 30.0 30.0 29.9
11 30.0 30.0 29.9 29.9 30.2 29.9 29.9 29.9 29.8 29.9 30.0 30.2 29.9
12 29.9 30.2 29.9 30.0 30.0 30.1 30.1 30.1 29.9 29.9 30.0 30.0 30.1
0 2 4 6 8 10 12

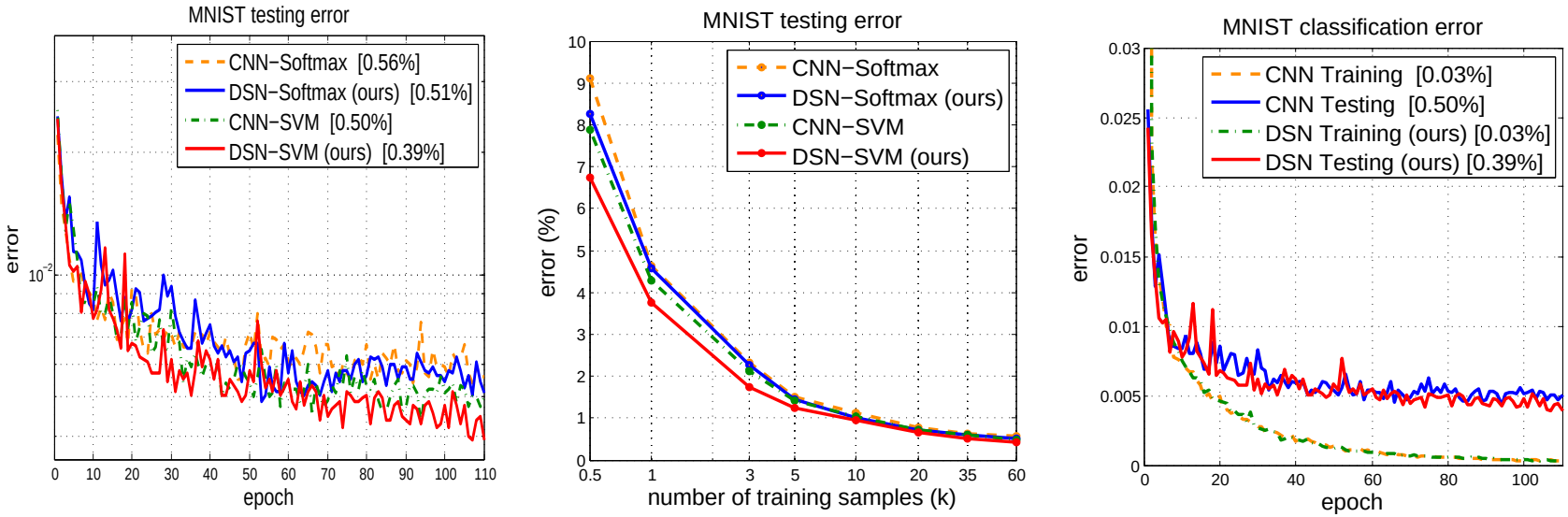

0 0.3 -1.8 -1.4 -0.2 1.3 1.0 -0.0 0.2 -0.2 -2.4 1.0 0.7 -0.6
1 0.9 -0.7 -0.4 -0.3 -0.1 -0.6 -0.1 0.5 -0.3 -0.5 1.1 -0.8 -0.4
2 1.0 0.3 -0.7 0.6 0.7 -28.0 -54.9 -84.7 -55.1 -26.8 0.4 -1.0 0.8
3 -0.2 0.6 -0.5 -27.7 -27.1 -84.2 140.2 82.7 140.6 -58.0 -0.5 0.8 0.1
4 -0.1 -0.2 -28.2 -84.1 139.8 168.1 56.2 -0.5 84.0 -83.3 -0.1 0.9 -1.1
5 -0.5 0.6 -55.0 110.9 54.9 27.5 27.6 0.6 83.3 -83.7 0.7 -0.3 -2.0
6 0.9 -0.0 -84.5 85.1 0.3 1.0 0.5 0.9 83.1 -84.4 -1.7 -0.7 -1.2
7 -0.7 0.4 -55.8 140.0 55.7 26.7 26.2 56.6 140.8 -54.6 1.8 0.3 0.6
8 -2.0 1.1 -29.9 -56.8 -138.8 55.3 55.8 -139.6 -55.6 -31.1 1.0 -1.2 -2.4
9 0.7 -0.0 1.4 1.1 -54.7 140.3 139.7 -55.2 -0.4 1.2 0.4 -0.6 -0.8
10 -0.1 0.8 -1.5 -1.4 -29.9 -55.8 -55.4 -28.3 0.4 -1.8 0.6 -0.1 0.3
11 -0.0 -0.3 1.1 1.6 -1.4 1.1 0.7 0.2 1.2 -0.1 -0.1 -1.5 1.5
12 0.4 0.9 1.7 1.1 0.9 -1.3 -1.0 0.5 0.1 1.2 -0.1 -0.4 -0.9
0 2 4 6 8 10 12

Auxiliary Loss

Figure: GoogLeNet[18]

[18] Christian Szegedy et al. *Going Deeper with Convolutions*. 2014. arXiv: 1409.4842.

Auxiliary Loss



(a) (b) (c)

Figure: Classification error on MNIST test. (a) shows test error of competing methods; (b) shows test error w.r.t. the training sample size. (c) training and testing error comparison.[20]

[20] Chen-Yu Lee et al. *Deeply-Supervised Nets*. 2014. arXiv: 1409.5185.

Up/Down Sampling

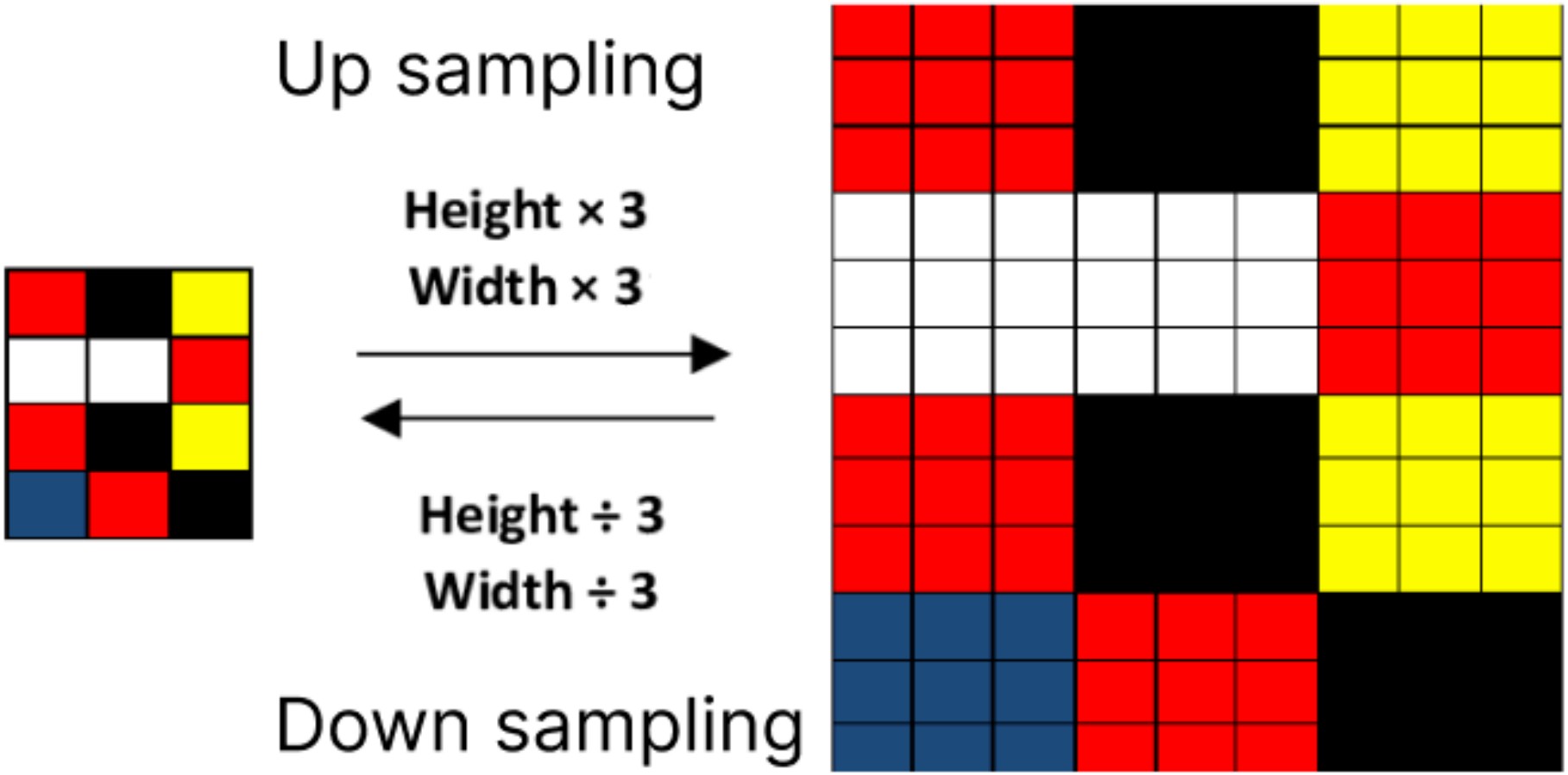


Figure: Image Upsampling and Downsampling[21]

[21] Ding Yang and Bai Jiabao. "An Optimization Method for Video Upsampling and Downsampling Using Interpolation-Dependent Image Downsampling". In: *2021 4th International Conference on Information Communication and*