Grounded Language-Image Pre-training

Liunian Harold Li, Pengchuan Zhang, Haotian Zhang , Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, Kai-Wei Chang, Jianfeng Gao

CVPR 2022

Presenter: Yi-Ting Li (李易庭)

Date: Feb. 15, 2023

Intelligent Knowledge Management Laboratory
Department of Computer Science and Information Engineering
National Cheng Kung University, Tainan, Taiwan, R.O.C

Outline

Task Definition

1. Object Detection
2. Phrase Grounding

Related Work

1. DERT - Facebook
2. MDETR - Facebook

Methodology

1. Reformulating object detection as phrase grounding
2. Language-Aware Deep Fusion
3. Pre-training with Scalable Semantic-Rich Data

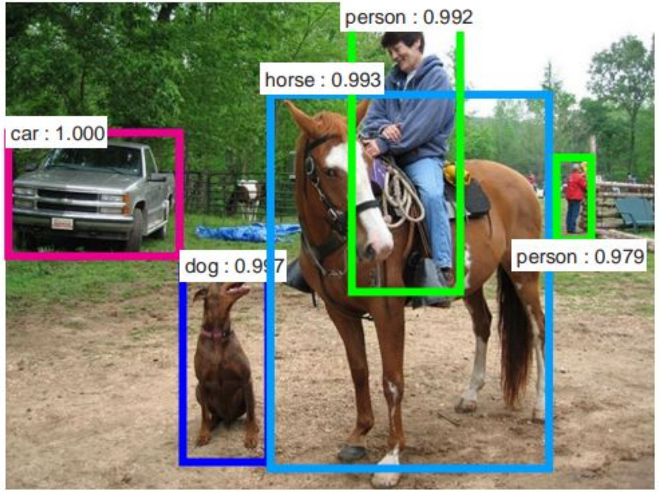
Transfer learning result

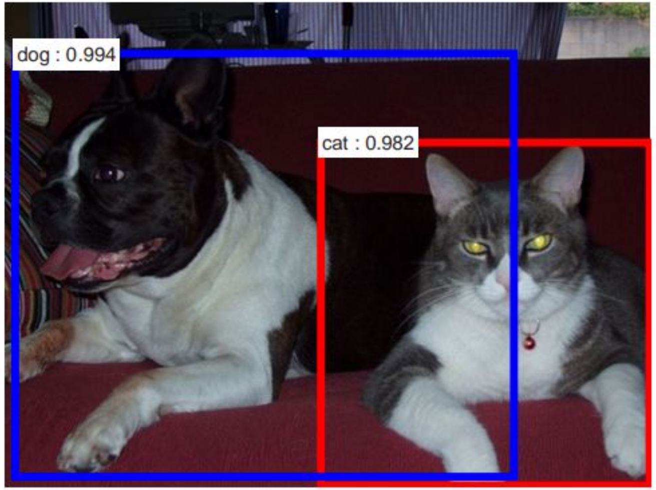
1. Zero-shot
2. Prompt tuning
3. Linear probing

SWOT

Task Definition - Object Detection

Object detection is the task of detecting instances of **objects** of a **certain class** within an image.





[1] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." NIPS. 2015.

Task Definition - Phrase Grounding

Given an **image** and a **corresponding caption**, the Phrase Grounding task aims to ground each **entity** mentioned by a noun phrase in the caption to a region in the image.

A tattooed woman with a green dress and yellow back-pack holding a water bottle is walking across the street.

[2] Fukui, Akira, et al. "Multimodal Compact Bilinear Pooling for Visual Question Answering and Visual Grounding." EMNLP. 2016.

Task Definition - Problem and Solution

Problem

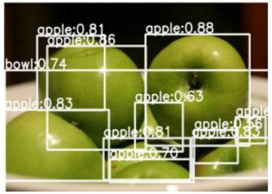
1. While training an object detection model, the model will given an **image** and **bounding box** with **text label**. Therefore the annotation fee will be costly.

2. The object detection model can only detect instances of objects of a particular class, so for the labels **not in** the training dataset, the model can't figure them out.

Solution - Treating the object detection tasks as phrase grounding

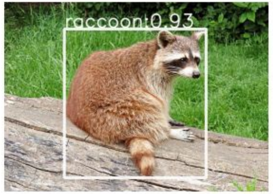
1. We could easily collect image-caption pairs from the Internet.

2. By giving prompts(caption) to control the detection target.



Prompt : person. bicycle. car. motorcycle...

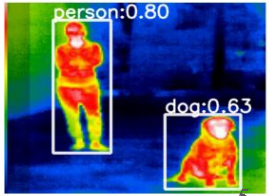
Prompt : aerosol can... lollipop... pendulum...



Prompt : raccoon

Prompt : pistol

Prompt : there are some holes on the road



Prompt : person. dog.

Task Definition - Image Caption Pair

In GLIP:

Like CLIP, unknown

Current model:

Use BLIP

[3] Li, Junnan, et al. "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation." ICML, 2022.

Related Work

DETR

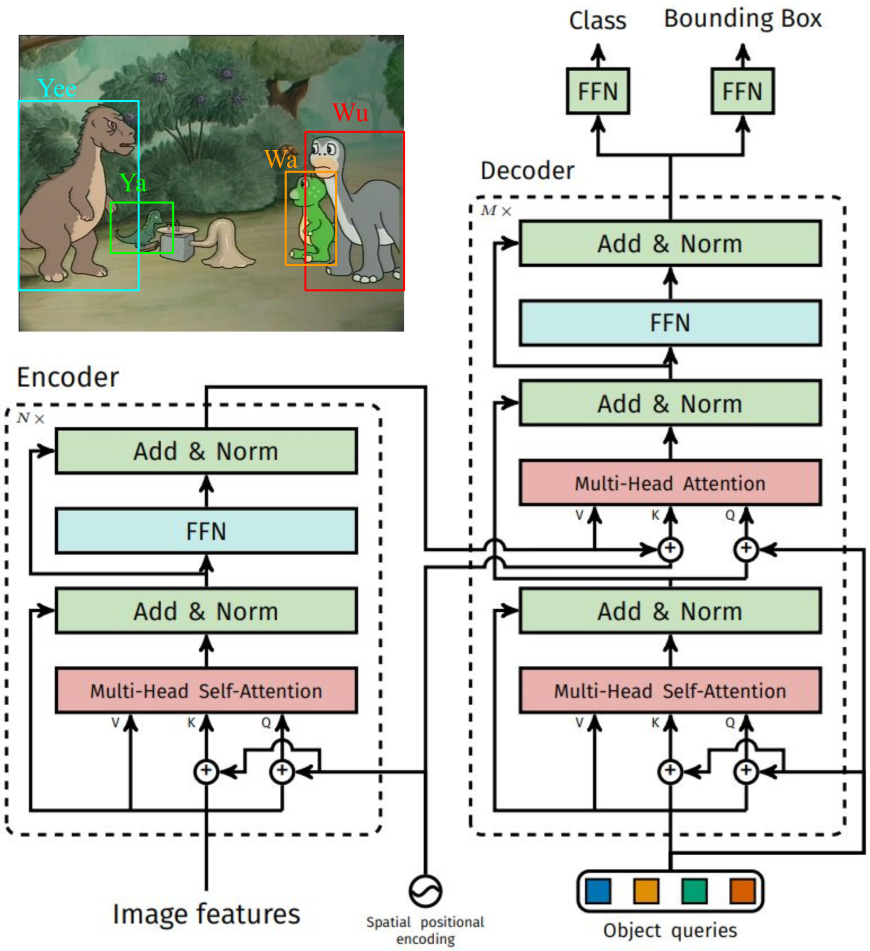
Traditional Object Detection Problem

Need so many **hand-design** component:

1. Proposals(2-stage, e.g., R-CNN family)
2. Anchor or Window centers(1-stage, e.g., Yolo family)
3. NMS (Non-maximum suppression)

Contribution

1. **Without** hand-design components anymore.
2. The **object detection set(cls, box) prediction** loss
3. Parallel Decoding (Transformer)



[4] Carion, Nicolas, et al. "End-to-end object detection with transformers." ECCV. 2020.

DETR

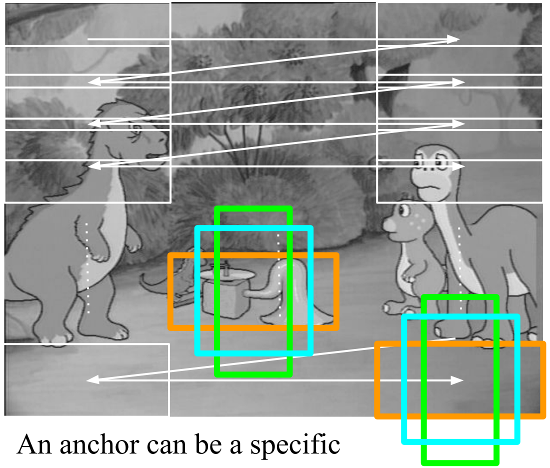
DETR implement with PyTorch

```python
import torch
from torch import nn
from torchvision.models import resnet50

class DETR(nn.Module):

    def __init__(self, num_classes, hidden_dim, nheads,
                 num_encoder_layers, num_decoder_layers):
        super().__init__()
        # We take only convolutional layers from ResNet-50 model
        self.backbone = nn.Sequential(*list(resnet50(pretrained=True).children())[:-2])
        self.conv = nn.Conv2d(2048, hidden_dim, 1)
        self.transformer = nn.Transformer(hidden_dim, nheads,
                                          num_encoder_layers, num_decoder_layers)
        self.linear_class = nn.Linear(hidden_dim, num_classes + 1)
        self.linear_bbox = nn.Linear(hidden_dim, 4)
        self.query_pos = nn.Parameter(torch.rand(100, hidden_dim))
        self.row_embed = nn.Parameter(torch.rand(50, hidden_dim // 2))
        self.col_embed = nn.Parameter(torch.rand(50, hidden_dim // 2))

    def forward(self, inputs):
        x = self.backbone(inputs)
        h = self.conv(x)
        H, W = h.shape[-2:]
        pos = torch.cat([
            self.col_embed[:W].unsqueeze(0).repeat(H, 1, 1),
            self.row_embed[:H].unsqueeze(1).repeat(1, W, 1),
        ], dim=-1).flatten(0, 1).unsqueeze(1)
        h = self.transformer(pos + h.flatten(2).permute(2, 0, 1),
                             self.query_pos.unsqueeze(1))
        return self.linear_class(h), self.linear_bbox(h).sigmoid()
```

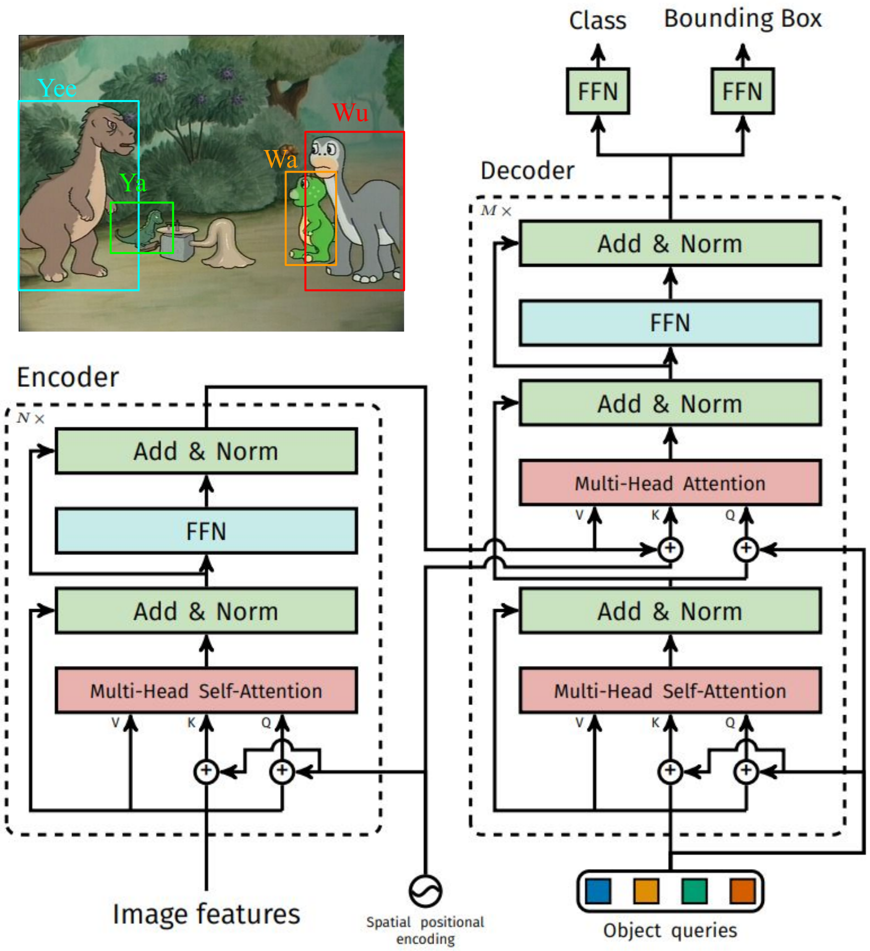
DETR

Aiden: What's the **meaning** of Object Queries?

Answer: Like a set of anchors in traditional object detection models but learnable.



An anchor can be a specific width and height rectangle

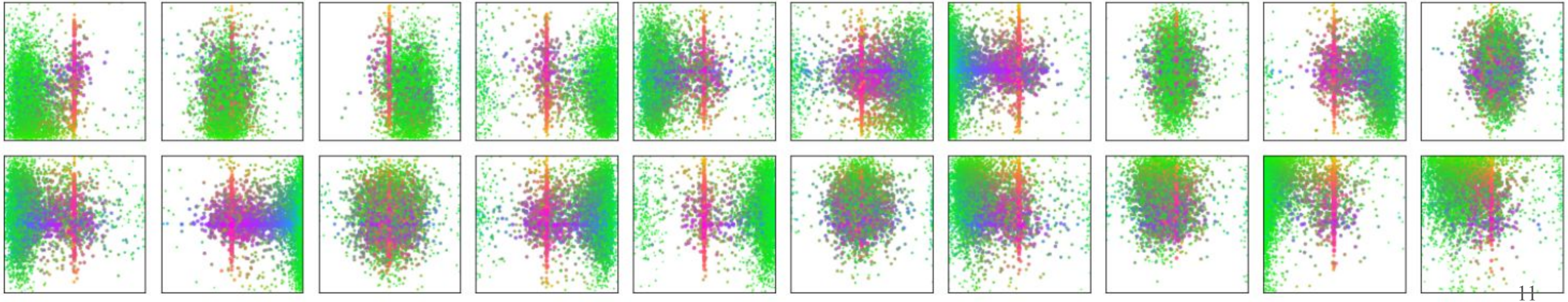


[5] What is anchor boxes?

DETR

Each box **prediction** is represented as a **point** with the coordinates of its center in the 1-by-1 square normalized by each image size.

1. Green: small bboxes
2. Red: large horizontal bboxes
3. Blue: large vertical bboxes



DETR

Question: How to compute the loss? The number of output pairs always **larger** than the number of ground truth.

Answer: Optimal **bipartite matching**

Step1) Construct the "Cost Matrix" by match loss function

$$\boxed{-\mathbb{1}_{\{c_i \neq \varnothing\}} \hat{p}_{\sigma(i)}(c_i)} + \boxed{\mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\sigma(i)})}$$

Step2) Apply **Hungarian Algorithm** to find the perfect one-to-one matching

Step3) Compute the loss value

$$\sum_{i=1}^{N} \left[\boxed{-\log \hat{p}_{\hat{\sigma}(i)}(c_i)} + \boxed{\mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}}(i))} \right]$$

Ground Truth

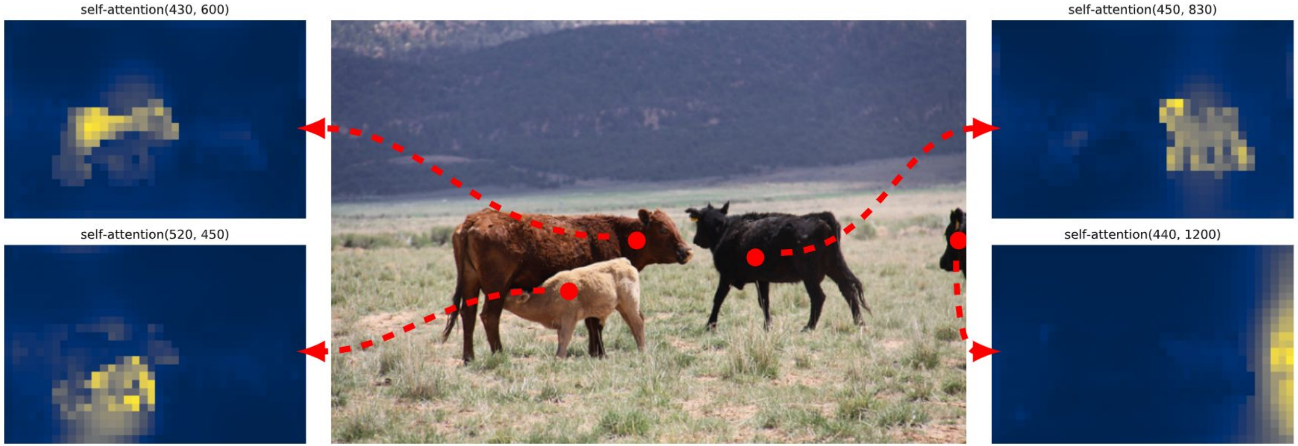
Predict	L_1	L_2	L_3	L_4
O_1	1	2	3	4
O_2	2	2	4	1
O_3	3	2	1	4
O_4	4	4	2	3
O_5	3	1	2	2

Cost Matrix

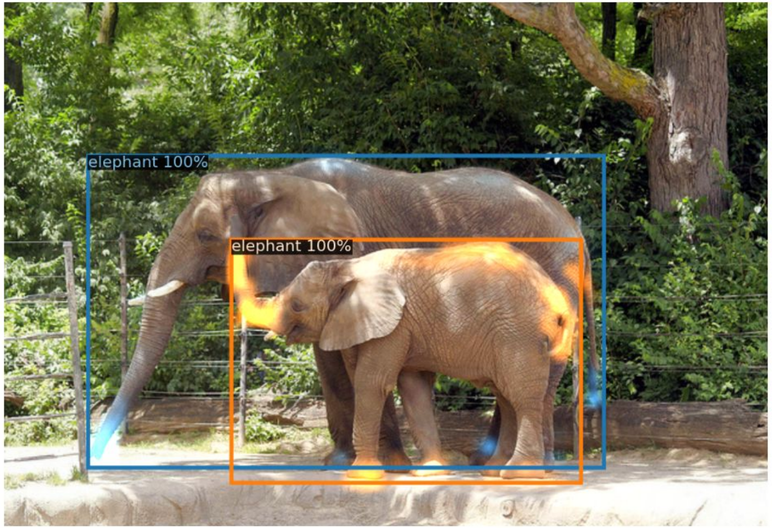
[7] Rezatofighi, Hamid, et al. "Generalized intersection over union: A metric and a loss for bounding box regression." CVPR. 2019.
[8] Hungarian Algorithm scipy.optimize.linear_sum_assignment

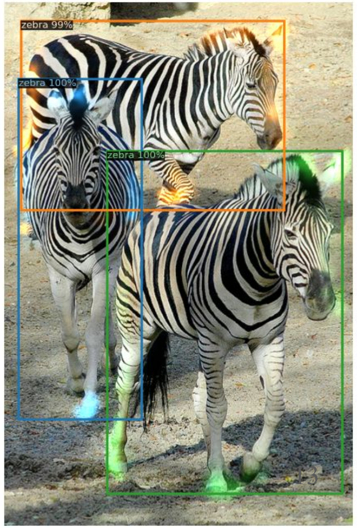
DETR

The encoder self-attention



The decoder attention

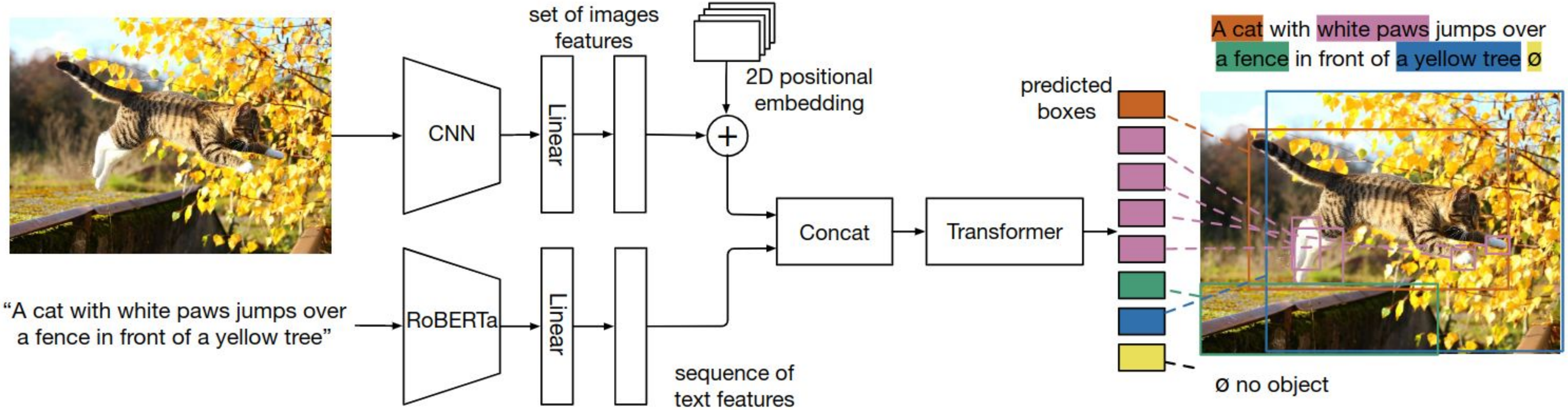




MDETR

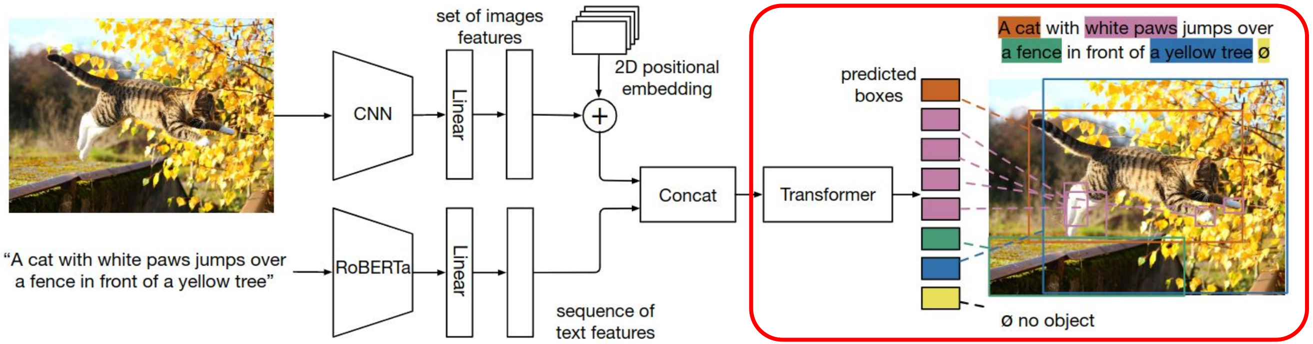
英嘉學長:Why CNN is needed? Why don’t just use Transformer?

Answer: Reason for input. The input for Transformer will be too long, if we flatten the image as the input directly; the pre-trained CNNs(like ResNet) pay more attention to the local feature(like contour)



[9] Kamath, Aishwarya, et al. "Mdetr-modulated detection for end-to-end multi-modal understanding." ICCV. 2021.
[10] Dosovitskiy, Alexey, et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." ICLR. 2020.

MDETR



Soft token prediction

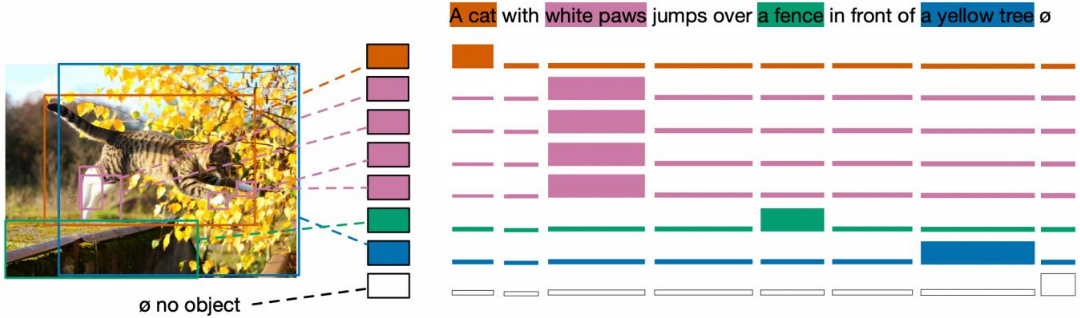
1. RoBERTa
2. The model is trained to predict a **uniform distribution** over all token positions that correspond to the object.

Aiden: How to design the loss function? Answer: as a Multi-label classification (by CE).

Contrastive alignment(InfoNCE)

Positive Pair: , A cat

Negative Pair: , with white … ∅

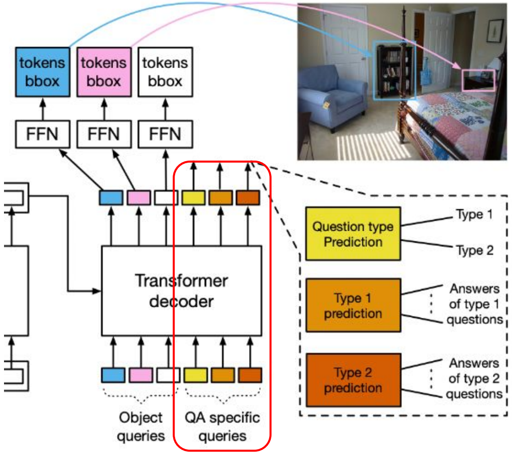


[11] Kamath, Aishwarya, et al. "Mdetr-modulated detection for end-to-end multi-modal understanding." ICCV. 2021.

MDETR

Extend model for VQA (GQA dataset)

Provide QA specific queries REL, OBJ, GLOBAL, CAT and ATTR in addition to the object queries as input to the decoder



Question: "What is on the table?"
Answer:

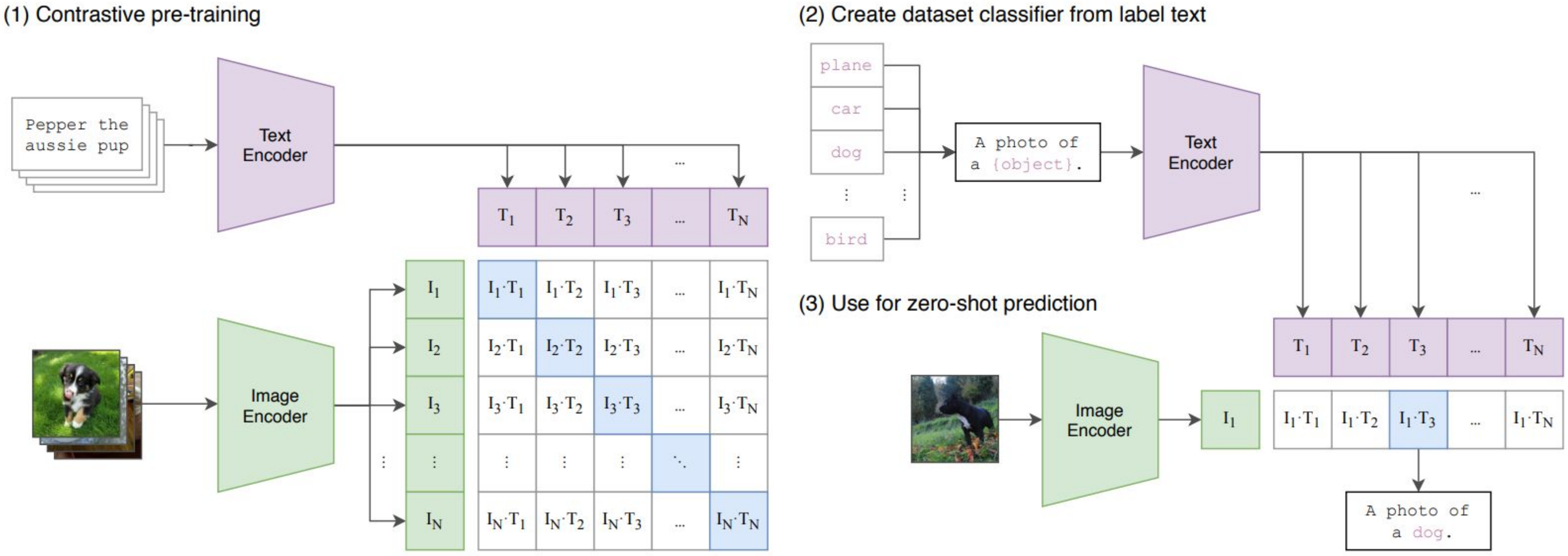
- Question type prediction: **OBJ**
- Type 1 prediction: **Laptop**

[12] Hudson, Drew A., and Christopher D. Manning. "Gqa: A new dataset for real-world visual reasoning and compositional question answering." CVPR. 2019.

MDETR

Type	Open/Binary	Semantic	Structural	Form	Example
queryGlobal	open	query	global	select: `scene`/query: `type`	How is the weather in the image?
verifyGlobal	binary	verify	global	select: `scene`/verify `type: attr`	Is it cloudy today?
chooseGlobal	open	query	global	select: `scene`/choose `type: a`|`b`	Is it sunny or cloudy?
queryAttr	open	query	attribute	select: `obj`/.../query: `type`	What color is the apple?
verifyAttr	binary	verify	attribute	select: `obj`/.../verify `type: attr`	Is the apple red?
verifyAttrs	binary	logical	attribute	select: `obj`/.../verify `t1: a1`/verify `t2: a2`/and	Is the apple red and shiny?
chooseAttr	open	choose	attribute	select: `obj`/.../choose `type: a`|`b`	Is the apple green or red?
exist	binary	verify	object	select: `obj`/.../exist	Is there an apple in the picture?
existRel	binary	verify	relation	select: `subj`/.../relate (`rel`): `obj`/exist	Is there an apple on the black table?
logicOr	binary	logical	object	select: `obj1`/.../exist/select: `obj2`/.../exist/or	Do you see either an apple or a banana there?
logicAnd	binary	logical	obj/attr	select: `obj1`/.../exist/select: `obj2`/.../exist/and	Do you see both green apples and bananas there?
queryObject	open	query	category	select: `category`/.../query: name	What kind of fruit is on the table?
chooseObject	open	choose	category	select: `category`/.../choose: `a`|`b`	What kind of fruit is it, an apple or a banana?

Related Work - CLIP

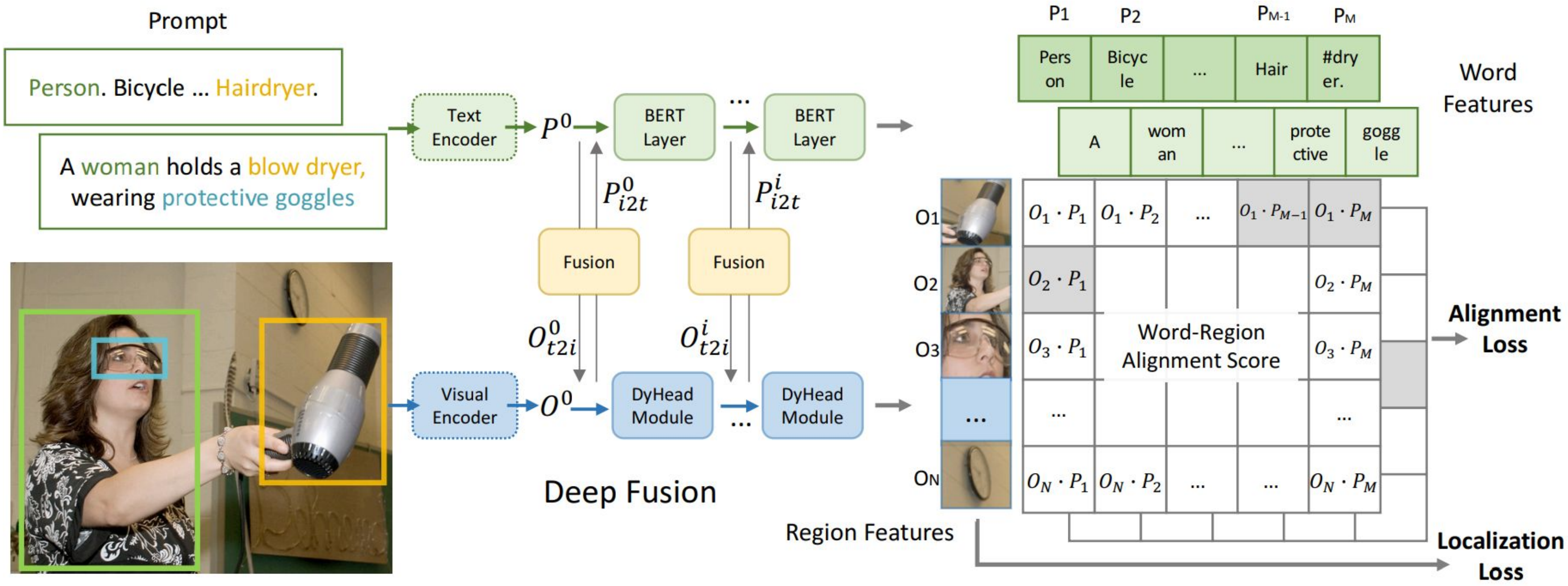


[14] Radford, Alec, et al. "Learning transferable visual models from natural language supervision." ICLR. 2021.

Methodology

Reformulating object detection as phrase grounding

Object Detection as Phrase Grounding



Reformulating object detection as phrase grounding

From Object Detection $\mathcal{L} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{loc}}$

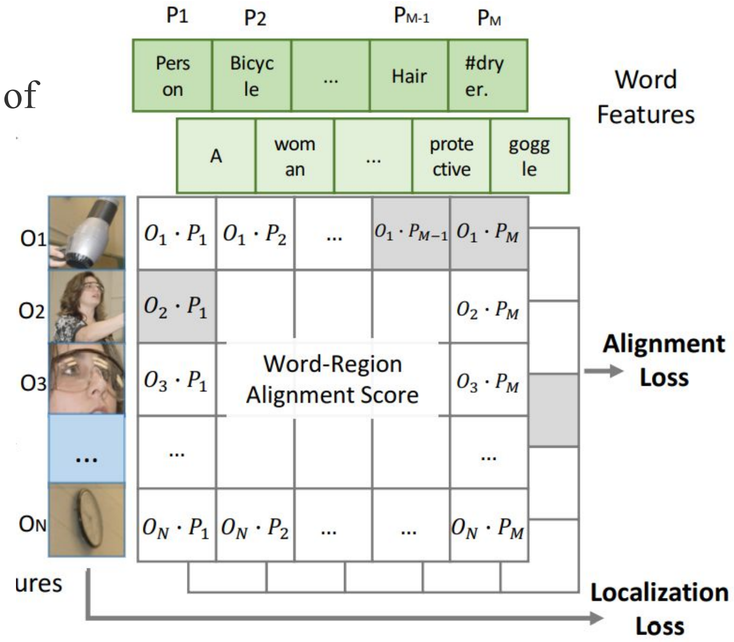
To Phrase Grounding $\mathcal{L} = S_{\text{ground}} + \mathcal{L}_{\text{loc}}$

Problem: the number of logits will always **larger** than the number of phrases in the text prompt due to following reasons:

1. some phrases contain multiple words, e.g. "traffic light".
2. some are the added tokens, e.g., "Detect:", ",".
3. some single-word phrases are splitted into multiple (sub)-word tokens, e.g., "toothbrush" to "tooth#" and "#brush".
4. [NoObj] token is added at the end of the tokenized sequence.

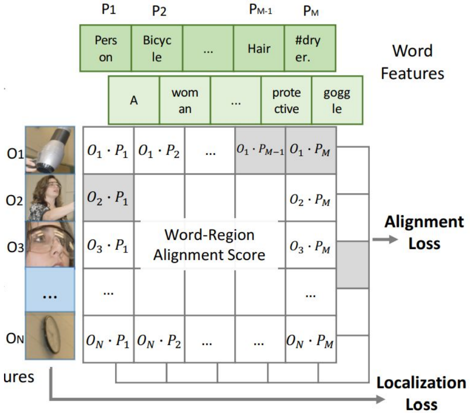
Solution:

If one phrase is positive match make **all** sub-words positive match. During inference, **average** token probabilities as the phrase probability.



Reformulating object detection as phrase grounding

Object Detection $\mathcal{L} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{loc}}$



$$
\begin{aligned}
O &= \mathrm{Enc}_I(\mathrm{Img}) &(1)\\
S_{\text{cls}} &= OW^T &(2)\\
\mathcal{L}_{\text{cls}} &= loss(S_{\text{cls}}; T) &(3)
\end{aligned}
$$

$$
\begin{aligned}
O \in \mathbb{R}^{N\times d} &= \text{object/region/box features of the input image}\\
W \in \mathbb{R}^{c\times d} &= \text{weight matrix of the box classifier}\\
S_{\text{cls}} \in \mathbb{R}^{N\times c} &= \text{logits of the classification}\\
T \in \{0,1\}^{N\times c} &= \text{target matching between regions and classes}\\
loss(S;T) &= \text{cross-entrypy loss for two-stage detectors}\\
c &= \text{number of classes}
\end{aligned}
$$

To Phrase Grounding $\mathcal{L} = S_{\text{ground}} + \mathcal{L}_{\text{loc}}$

$$
\begin{aligned}
O &= \mathrm{Enc}_I(\mathrm{Img}) &(4)\\
P &= \mathrm{Enc}_L(\mathrm{Prompt}) &(5)\\
S_{\text{ground}} &= OP^T &(6)
\end{aligned}
$$

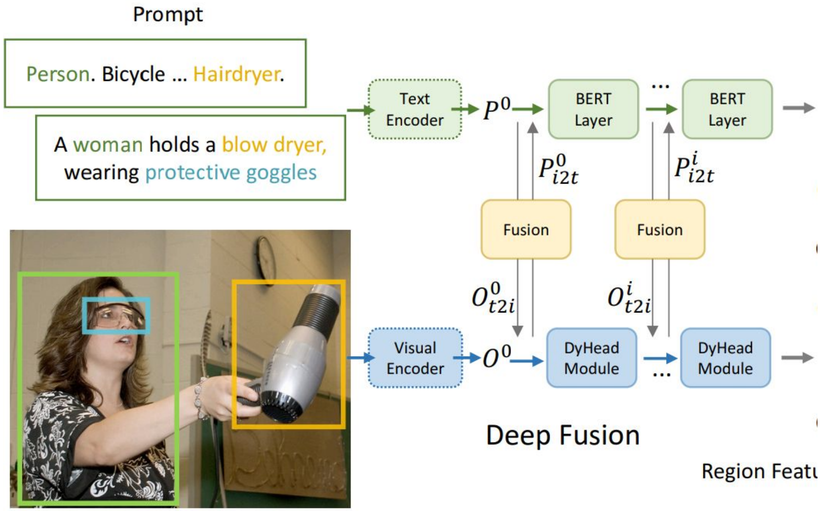
$$
P \in \mathbb{R}^{M\times d} = \text{word/token features from language encoder}
$$

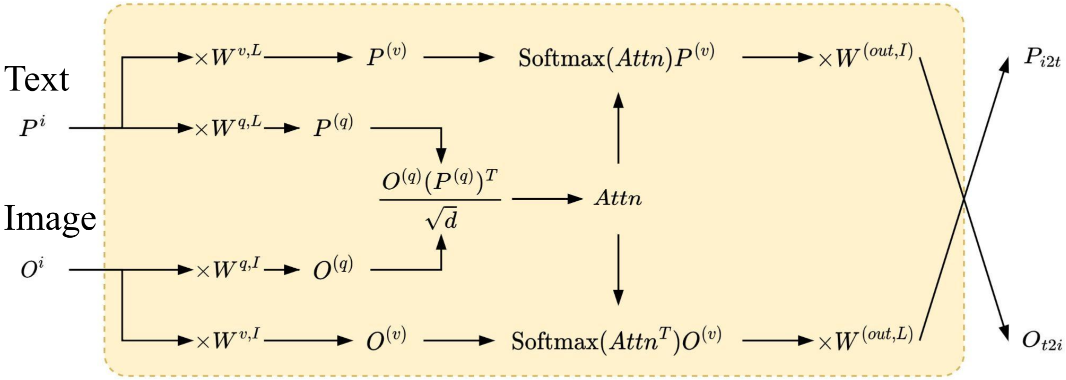
1. some phrases contain <u>multiple words</u>, e.g. “traffic light”
2. some are the <u>added tokens</u>, e.g., “Detect:”, “,”
3. some single-word phrases are splitted into multiple <u>(sub)-word</u> tokens, e.g., “toothbrush” to “tooth#” and “#brush”
4. [<u>NoObj</u>] token is added at the end of the tokenized sequence.

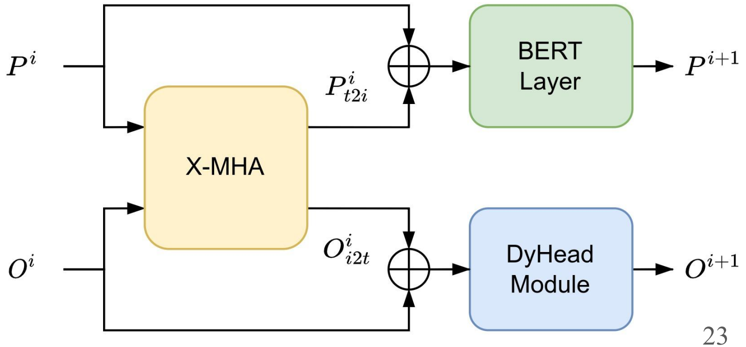
Language-Aware Deep Fusion

Simply fuse Bert-Layer with DyHead-Layer by Cross-Modal Multi-head Attention (X-MHA)

$$O^{(q)} = OW^{(q,I)},\ P^{(q)} = PW^{(q,L)},\ Attn = O^{(q)}(P^{(q)})^\top/\sqrt{d},$$
$$P^{(v)} = PW^{(v,L)},\quad O_{\text{t2i}} = \text{SoftMax}(Attn)P^{(v)}W^{(out,I)},$$
$$O^{(v)} = OW^{(v,I)},\quad P_{\text{i2t}} = \text{SoftMax}(Attn^\top)O^{(v)}W^{(out,L)},$$







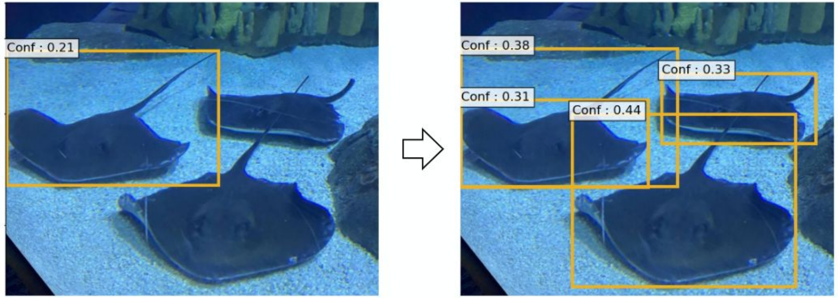
Pre-training with Scalable Semantic-Rich Data

1. Combine all the object detection dataset to train a GLIP model as the teacher model.
2. Scrape the image-text pair from the Internet as augmentation.
3. The inference output from teacher model as pseudo-labels to train the student model.

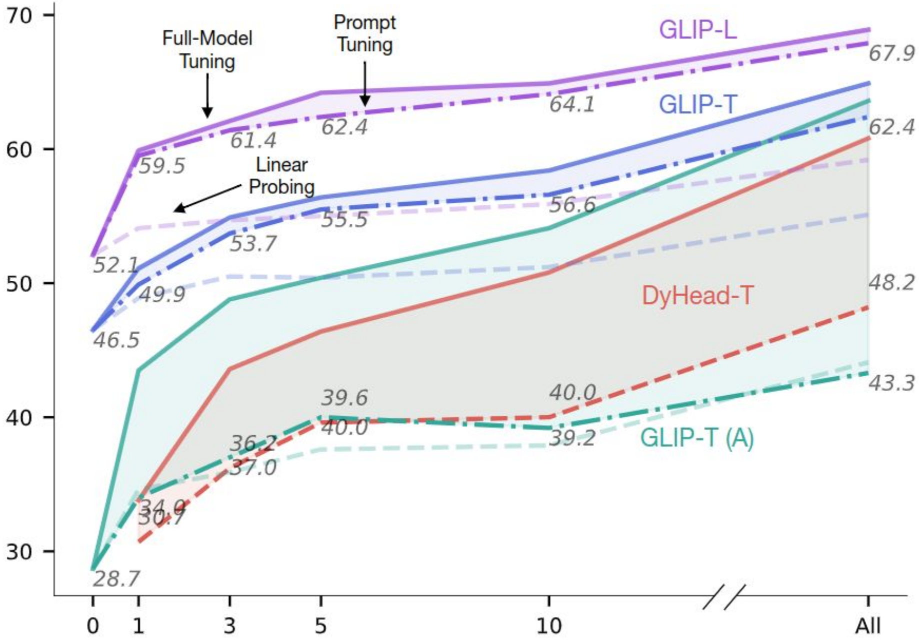
	Model	Backbone	Pre-Train Data	Zero-Shot 2017val	Fine-Tune 2017val / test-dev
	Faster RCNN	RN50-FPN	-	-	40.2 / -
	Faster RCNN	RN101-FPN	-	-	42.0 / -
	DyHead-T [10]	Swin-T	-	-	49.7 / -
	DyHead-L [10]	Swin-L	-	-	58.4 / 58.7
	DyHead-L [10]	Swin-L	O365,ImageNet21K	-	60.3 / 60.6
	SoftTeacher [65]	Swin-L	O365,SS-COCO	-	60.7 / 61.3
	DyHead-T	Swin-T	O365	43.6	53.3 / -
w/o deep fusion	GLIP-T (A)	Swin-T	O365	42.9	52.9 / -
w/ deep fusion	GLIP-T (B)	Swin-T	O365	44.9	53.8 / -
Teacher model	GLIP-T (C)	Swin-T	O365,GoldG	**46.7**	55.1 / -
	GLIP-T	Swin-T	O365,GoldG,Cap4M	46.3	54.9 / -
	GLIP-T	Swin-T	O365,GoldG,CC3M,SBU	46.6	**55.2** / -
	GLIP-L	Swin-L	FourODs,GoldG,Cap24M	**49.8**	**60.8** / 61.0
	GLIP-L	Swin-L	FourODs,GoldG+,COCO	-	- / **61.5**

Prompt Tuning

By only tuning the prompt embeddings, GLIP-T and GLIP-L can achieve performance close to full-model tuning, allowing for efficient deployment.



Prompt: … stingray … *Prompt*: … stingray, which is flat and round…



SWOT

Strengths

1. Deployment efficiency
2. Without hand-designed components to detect objects.
3. Fusion BERT with DyHead to get better cross-modal features.

Weaknesses

1. Memory costly and inference slowly

Model	Fusion	Inference (P100) Speed	Inference (P100) Memory	Train (V100) Speed	Train (V100) Memory
GLIP-T	✗	4.84 FPS	1.0 GB	2.79 FPS	11.5 GB
	✓	2.52 FPS	2.4 GB	1.62 FPS	16.0 GB
GLIP-L	✗	0.54 FPS	4.8 GB	1.27 FPS	19.7 GB
	✓	0.32 FPS	7.7 GB	0.88 FPS	23.4 GB

Opportunities

1. We can design a prompt for VQA to get the answer bounding box.

Threats

1. The model may can not understand the specific prompt pattern like: <text_1>