**HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY**

# PROJECT REPORT

## Semi-supervised Object Detection

**TRUONG GIA BACH** 20210087
**TRAN DUONG CHINH** 20210122
**PHAN DUC HUNG** 20214903
**NGUYEN VIET MINH** 20214917
**PHAM QUANG TRUNG** 20214935

**Course: Computer Vision**

**Supervisor:** PhD. Dang Tuan Linh

Signature

**Department:** Computer Science

**School:** Information and Communication Technology

**HANOI, 6/2024**

# TABLE OF CONTENTS

# CONTRIBUTIONS

The following table shows the detail contribution of each member in our group.

| Name | ID | Contributions | Percentage |
|---|---|---|---|
| Truong Gia Bach | 20210087 | Unbiased Teacher Model, Efficient Teacher Model | 20% |
| Tran Duong Chinh | 20210122 | Data augmentation, Data preprocessing, DETR model | 20% |
| Phan Duc Hung | 20214903 | YoLov9-C,-M, v10-L Models, SSD300-VGG16 model | 20% |
| Nguyen Viet Minh | 20214917 | FasterRCNN-R50, RetinaNet-R50 models, Teacher-Student Model, Slides | 20% |
| Pham Quang Trung | 20214935 | Study problem, split dataset, data cleaning, inference, report | 20% |

# 1 Introduction

Object detection is a computer vision technique that identifies and locates objects within an image or video. This technology combines image classification and object localization to achieve its goals. Here are the key components and steps involved in object detection:

- Image Classification: This involves categorizing an image into a specific class or category. For example, identifying whether an image contains a dog, cat, or car.

- Object Localization: This refers to determining the location of an object within an image. This is usually represented by bounding boxes that enclose the object.

- Object Detection: This is the process of combining image classification and object localization to detect and classify multiple objects within a single image. Each detected object is assigned a category and a bounding box that indicates its location.

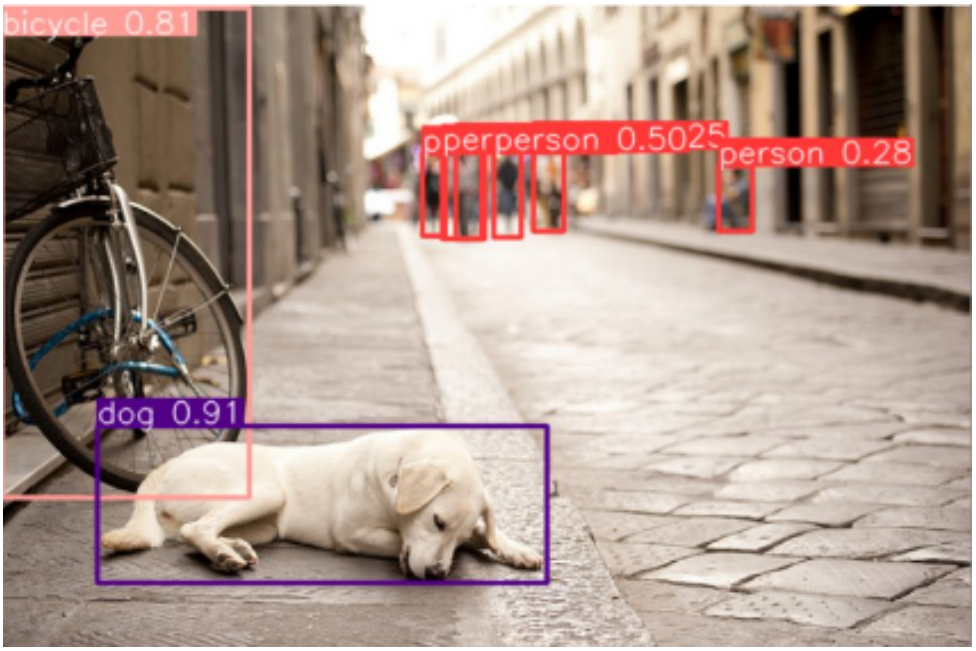



**Figure 1.1:** Object Detection

Semi-supervised object detection is an advanced approach in computer vision that aims to leverage both labeled and unlabeled data to improve the performance and efficiency of object detection models. This is particularly useful in scenarios where obtaining a large amount of labeled data is expensive or time-consuming.

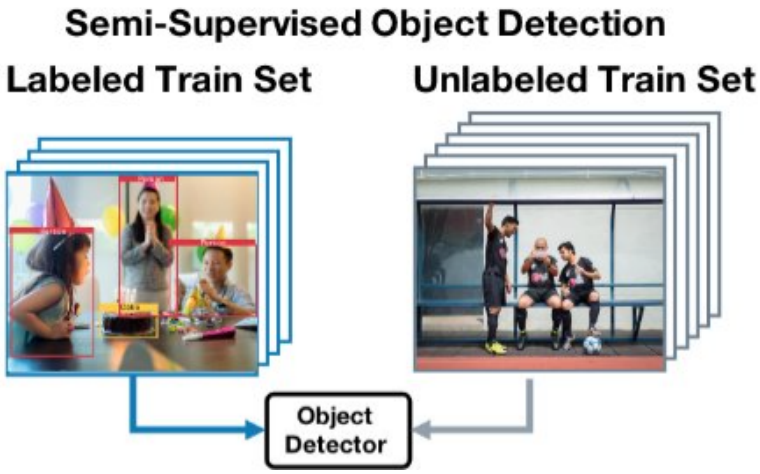



**Figure 1.2:** Semi-Supervised Object Detection

We know that through definition here that if we want our semi-supervised object detection to work best, we must maximize the performance, result of supervised task through labeled dataset. Thinking about label data as unlabel data with highest confidence score.

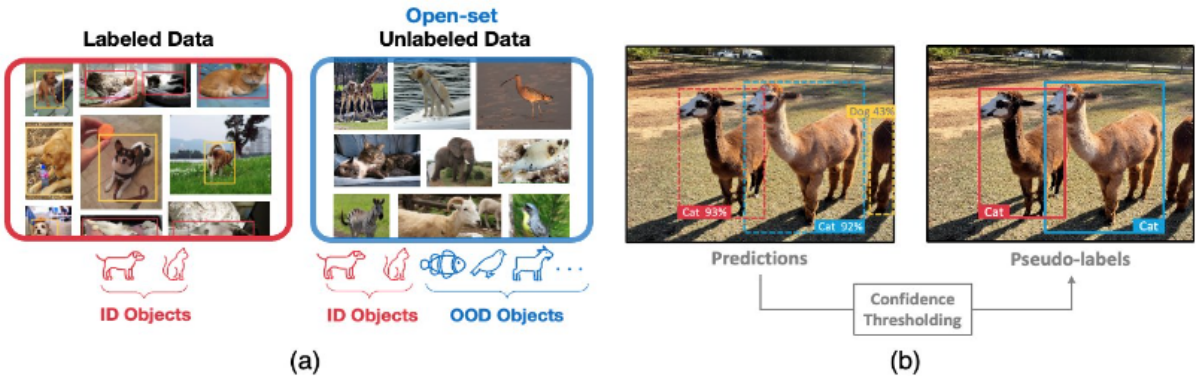



**Figure 1.3:** Semi-Supervised Object Detection

# 2 Dataset

The famous dataset COCO (Common Objects In Context) has been taken as a standard benchmark for semi-supervised object detection task[1]. We take version 2017 which contains 90 different objects and 860000 annotations for unique object[2].

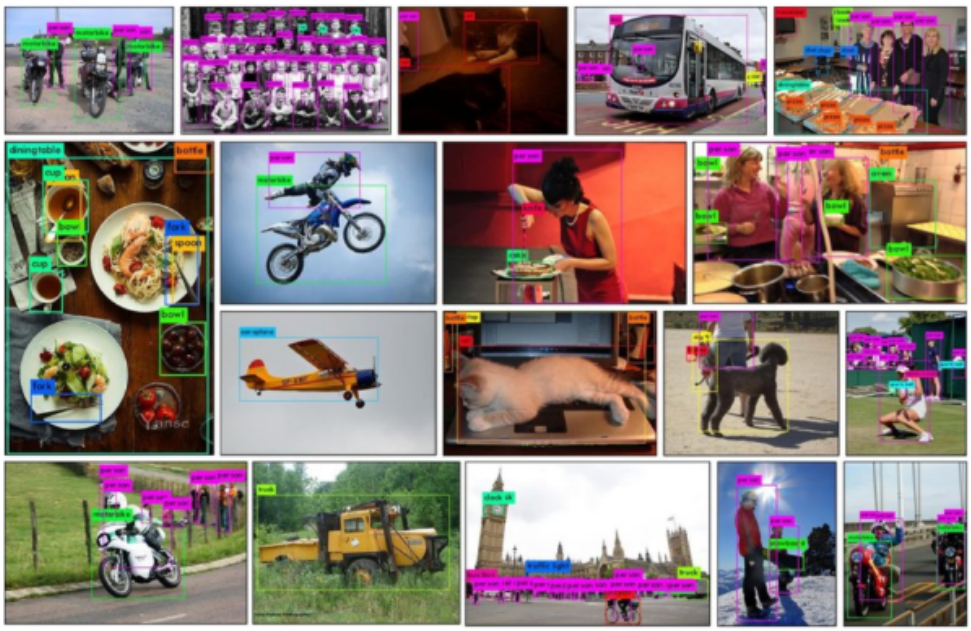


**Figure 2.1:** COCO object detection

Because of hardware and machine limitation, we split the COCO dataset only take the label from 2 to 9 which is vehicle category_id

```
{2: {'supercategory': 'vehicle', 'id': 2, 'name': 'bicycle'},
 3: {'supercategory': 'vehicle', 'id': 3, 'name': 'car'},
 4: {'supercategory': 'vehicle', 'id': 4, 'name': 'motorcycle'},
 5: {'supercategory': 'vehicle', 'id': 5, 'name': 'airplane'},
 6: {'supercategory': 'vehicle', 'id': 6, 'name': 'bus'},
 7: {'supercategory': 'vehicle', 'id': 7, 'name': 'train'},
 8: {'supercategory': 'vehicle', 'id': 8, 'name': 'truck'},
 9: {'supercategory': 'vehicle', 'id': 9, 'name': 'boat'}}
```

**Figure 2.2:** Vehicle category and id.

In Semi-Supervised Object Detection, we use 10% as labeled data, 90% as unlabeled data and 5% of unlabeled data being split for validation, evaluation step. Divide by .json file: annotation.

118287 images minimize to 27358 images

**Figure 2.3:** COCO object detection

This is a picture of 2 vehicles type category which are train and bicycle.

We also thinking about data distribution problem, for balancing the dataset when we divide by randomly pick for label, unlabel, valid .json file.

Beside of balance, we must make sure that all classes categories must be in all the 3 divided files for model training and testing problem

The result show that there is no need of manually balance, or no need of stratified sampling

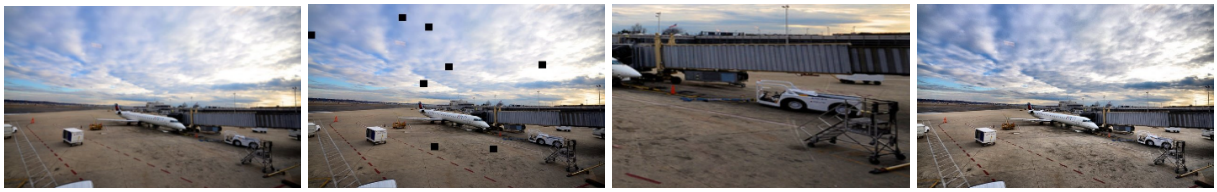
```
1  Labeled categories distribution: Counter({3: 4297, 9: 1071, 8: 1071, 4:
   842, 2: 732, 6: 544, 5: 490, 7: 485})
2  Unlabeled categories distribution: Counter({3: 37516, 9: 9271, 8: 8482,
   4: 7527, 2: 6017, 6: 5242, 5: 4465, 7: 3859})
3  Valid categories distribution: Counter({3: 2054, 8: 420, 9: 417, 2:
  364, 4: 356, 6: 283, 7: 227, 5: 180})
```

# 3 Data Augmentation and Preprocessing

For different models, we might have different data processing ways but we will summarize all here and go more details in each model in part 4

- Data augmentation: RandomHorizontalFlip, Blur, MedianBlur, ToGray, CLAHE(enhance Contrast using Adaptive Histogram Equaliztion) , RandomSizeCrop, RandomPad, RandomErasing, RandomVerticalFlip;
- Data preprocessing: Image resizing(640, 640) and normalization

**(a)** Normal **(b)** To Gray **(c)** Horizontal **(d)** Vertical



**(e)** Blur **(f)** Random Erasing **(g)** RandomsizeCrop,Pad **(h)** CLAHE

# 4 Teacher - Student Models

For semi-supervised object detection, the first thing pop out is pseudo-labeling:

Work with unlabeled data:

- Initial Training: train an object detection model on small labeled dataset,
- Predict on Unlabeled Data: use trained model to predict on the large unlabeled dataset,
- Pseudo Labeling: assign to unlabeled data by model prediction, with confidence scores, those with low confident score may be filtered out,
- Combining Data: combine original and pseudo label together,
- Retraining: retrained model to evaluate, continue train or train from initally.

Based on pseudo-labeling method, Teacher-Student Model instead of model learn by its self, it have teacher model guide student model to learn through teacher prediction. So this mean that teacher model must be do as good as possible for guiding the student

For model evaluation choose mAP (Mean Average Precision):

$$mAP = \frac{1}{|\text{classes}|} \sum_{c \in \text{classes}} \frac{|TP_c|}{|FP_c| + |TP_c|} \tag{1}$$

It's a widely used evaluation metric in object detection tasks, Precision-Recall curves are generated for each object class. Average Precision (AP) is computed for each class. mAP is the mean of AP across all classes.

For our specific problem, we will use the mAP 50-95 for the average of the mean average precision calculated at varying IoU thresholds, ranging from 0.50 to 0.95, which considering only the "easy" and practical detections.

## 4.1 Teacher model

We want to choose best teacher model based on result. So first step here we want to do is object detection with label dataset and choosing the best result one for labeling all the unlabel dataset

For candidates for model:

- We thinking about some famous model like R-CNN family and pick out Faster RCNN with R50 backbone represent for this, Retina with R50 backbone, SSD300 for minimize the calculation cost with VGG 16 backbone[3]. And then some most powerful model recently published like DETR, YoLov9, YoLov10

- We have here two-stage, one-stage method, the different is RPN and anchor-based, anchor-free method , the different predefined anchor boxes or not

### 4.1.1 FasterRCNN-R50 model

FasterRCNN-R50 model is one of the most famous method for object detection, two-stage and anchor-based which is highlighted some components:

- Region Proposal Network (RPN): This network generates region proposals, which are likely to contain objects. It works by sliding a small network over the convolutional feature map and predicting object bounds and scores at each position.
- Region of Interest (RoI) Pooling: This layer extracts fixed-size feature maps from the regions proposed by the RPN, which are then used for classification and bounding box regression.
- Object Detection Network: After RoI pooling, the features are passed through fully connected layers to classify the objects and refine their bounding boxes.

with Resnet50 backbone:

ResNet-50 is a residual network with 50 layers, used as the feature extractor in the Faster R-CNN architecture. It incorporates residual blocks, which help in training very deep networks by allowing gradients to flow through the network directly.

We choose Resnet50 while not the others because of its residual learning framework and not Resnet101 because of the speed and we make a trade between the quality and the time running

The combination of Faster R-CNN's region proposal mechanism and ResNet-50's deep, residual feature extraction leads to high accuracy in object detection tasks. Although not the fastest model, Faster R-CNN with ResNet-50 is more efficient than traditional R-CNN or Fast R-CNN, providing a good balance between speed and accuracy

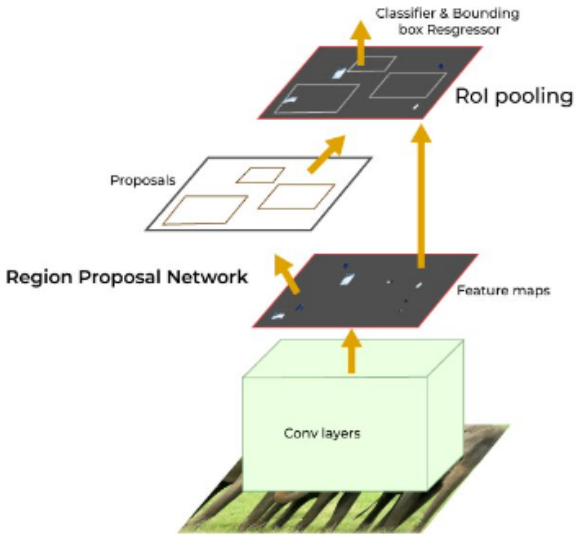



**Figure 4.1:** FasterRCNN model

### 4.1.2 RetinaNet-R50 model

RetinaNet with Resnet50 backbone is a Single-Stage Detector: Unlike Faster R-CNN, which uses a two-stage approach (region proposal and detection), RetinaNet is a single-stage detector. This means it directly predicts object classes and bounding boxes from feature maps, which can make it faster and more efficient.

Focal Loss is one of the key innovations in RetinaNet is the focal loss function. Focal loss helps to address the class imbalance problem by down-weighting the loss assigned to well-classified examples, allowing the model to focus on hard-to-classify examples.

ResNet-50 is used as the backbone network in RetinaNet, extracting rich features from the input images. The residual connections in ResNet-50 help in training deeper networks by mitigating the vanishing gradient problem.

RetinaNet uses a feature pyramid network (FPN) on top of ResNet-50 to create a rich multi-scale feature representation. This allows the model to detect objects at different scales effectively.

As a single-stage detector, RetinaNet is generally faster than two-stage detectors like Faster R-CNN, making it more suitable for real-time applications. The FPN architecture enhances RetinaNet's ability to detect small objects, which can be a challenge for some other object detection models. The focal loss function significantly improves the detection performance on datasets with a high class imbalance, which is common in real-world scenarios.

Small Discusstion about FasterRCNN and Retina:

For this 2 basic and famous models, we will use no pretrained model for seing how teacher-student method performance, if the teacher is faster RCNN or Retina by its self, it will become pseudo-labeling but we know this no way happening. The hard thing for these 2 for best visualizing result is by choosing NMS and score threshold, default config:

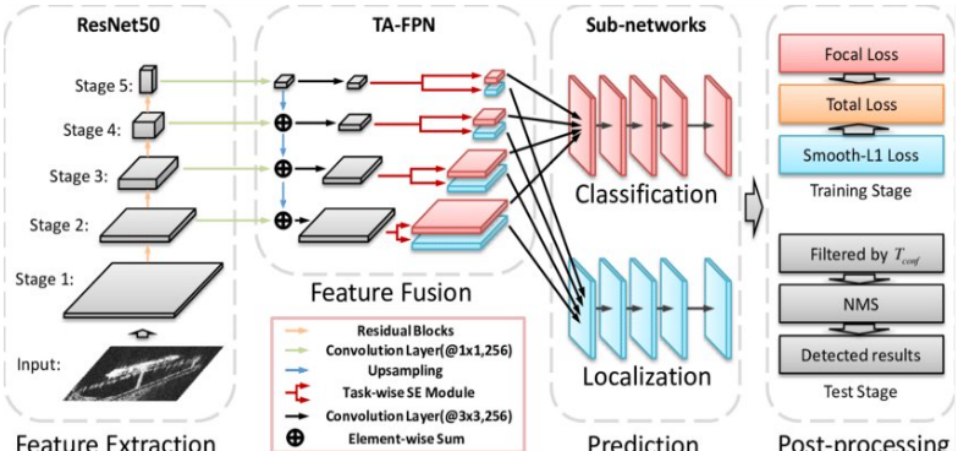



**Figure 4.2:** RetinaNet-R50 model

**(a)** Faster RCNN, mAP: 12.9%

**(b)** RetinaNet, mAP: 7.2%

The default configuration not doing so good, we need to choose NMS and score threshold for FasterRCNN and RetinaNet for best describe result of these 2 models:

Non-Maximum Suppression is a technique used to eliminate redundant or overlapping bounding boxes for the same object. When multiple detections overlap significantly and correspond to the same object, NMS ensures only the best detection is retained.

Score Thresholding is a technique used to filter out low-confidence detections in object detection models. Each detected object is assigned a confidence score (usually between 0 and 1) representing the model's certainty that the object belongs to a particular class. Thresholding involves setting a minimum confidence score. Any detections with a score below this threshold are discarded.

| **Retina** | **50, 75** | **30, 50** | **20, 35** | **15, 30** |
|---|---|---|---|---|
| mAP | <1% | 12.7% | 14% | 18.6% |

**Table 4.1:** Choosing NMS, score threshold for RetinaNet

For Faster RCNN, NMS 50, score 75 threshold is good enough so we stop searching

### 4.1.3 SSD300-VGG16 model

Single Shot MultiBox Detector (SSD) is a single-stage object detection model, meaning it performs both object localization and classification in a single forward pass through the net-

**(a)** Faster RCNN 5075

**(b)** RetinaNet 1530

work. SSD300 refers to the version of SSD that uses 300x300 input images, which balances speed and accuracy. SSD uses multiple feature maps of different resolutions to detect objects at various scales, enhancing its ability to handle objects of different sizes.

With backbone VGG16 is a deep convolutional neural network known for its simplicity and effectiveness. It consists of 16 layers, primarily convolutional layers followed by fully connected layers. The pre-trained VGG16 network is often used as the backbone for SSD, providing robust feature extraction capabilities.

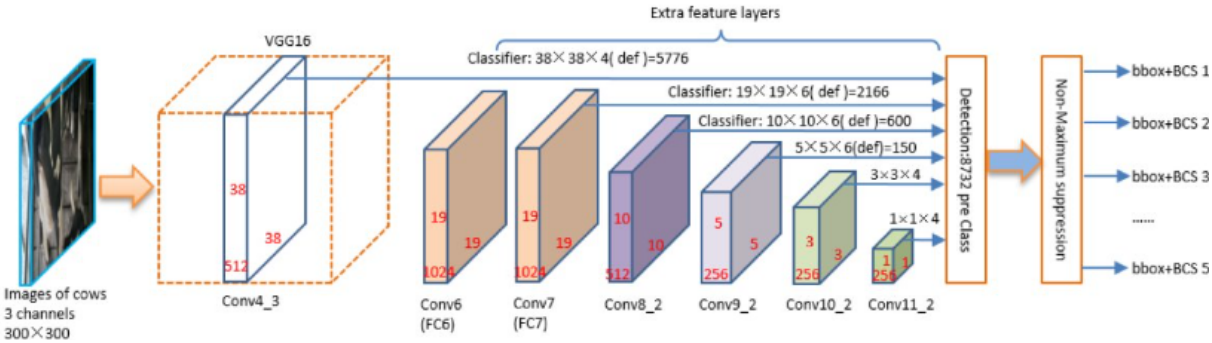



**Figure 4.5:** SSD300-VGG16 model

SSD300 is designed for real-time object detection. Its single-stage architecture allows it to be faster than two-stage detectors like Faster R-CNN. The architecture of SSD is relatively simple, making it easier to implement and understand compared to more complex models. The use of multi-scale feature maps allows SSD to detect objects at different scales effectively.

### 4.1.4 YOLOv9 model

You Only Look Once (YOLO) is a series of highly popular object detection models known for their real-time performance and high accuracy. YOLOv9[4], [5] is a state-of-the- Unterscheidung (German for "differentiation") real-time object detection model released in February 2024. It aims to outperform all existing object detection methods, including those based on convolutions or transformers.

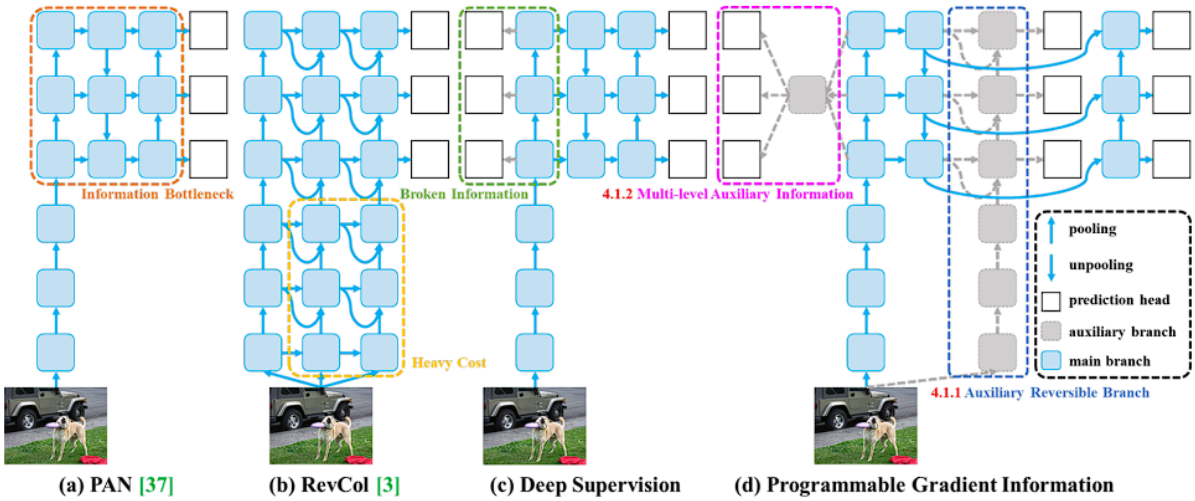



**Figure 4.6:** PGI and related network architectures and methods. (a) Path Aggregation Network (PAN), (b) Reversible Columns (RevCol), (c) conventional deep supervision, and (d) our proposed Programmable Gradient Information (PGI). PGI is mainly composed of three components: (1) main branch: architecture used for inference, (2) auxiliary reversible branch: generate reliable gradients to supply main branch for backward transmission, and (3) multi-auxiliary information: control main branch learning plannable multi-level of semantic information.

The model's success is attributed to two key innovations:

- Programmable Gradient Information (PGI): This technique helps in addressing the information bottleneck problem, allowing for better gradient flow and information retention during training.
- Generalized Efficient Layer Aggregation Network (GELAN): This novel network architecture is designed to improve both efficiency and accuracy.

To test the accuracy in dataset, we use YOLOv9-C for pretrain model to get the most powerful model with all pretrained information. For no-pretrain model, we use Yolov9-m which is smaller one to find the answer for question if small amount of label data is enough for YOLOv9 to learn.

### 4.1.5 YOLOv10 Model

YOLOv10[6] releases on May 2024, built on the Ultralytics Python package by researchers at Tsinghua University, introduces a new approach to real-time object detection, addressing both the post-processing and model architecture deficiencies found in previous YOLO versions. By eliminating non-maximum suppression (NMS) and optimizing various model components, YOLOv10 achieves state-of-the-art performance with significantly reduced computational overhead. Extensive experiments demonstrate its superior accuracy-latency trade-offs across multiple model scales.

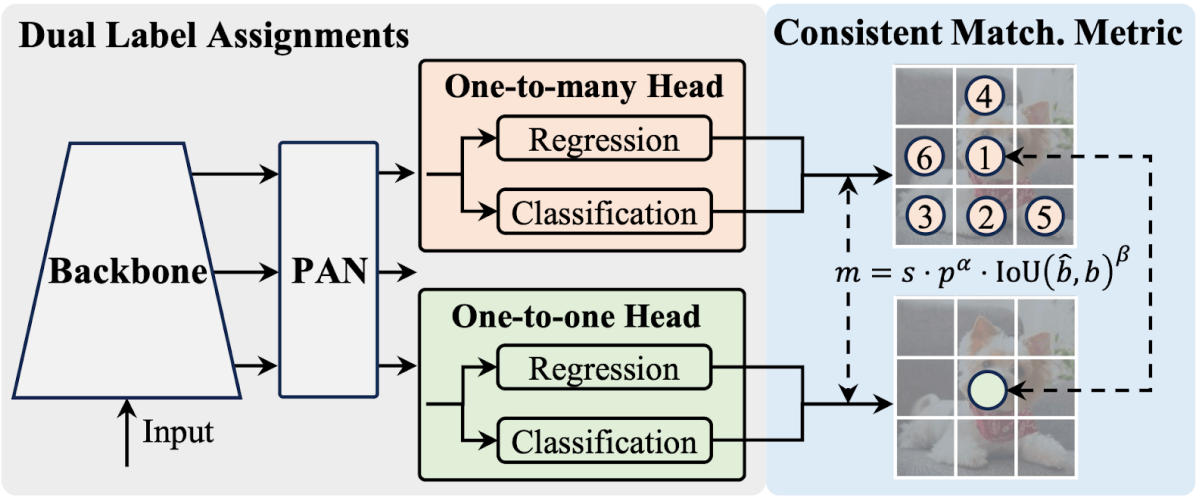
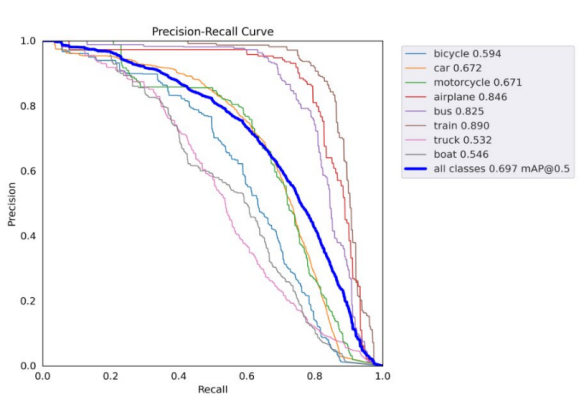



**Figure 4.7:** Consistent dual assignments for NMS-free training in Yolov10 model

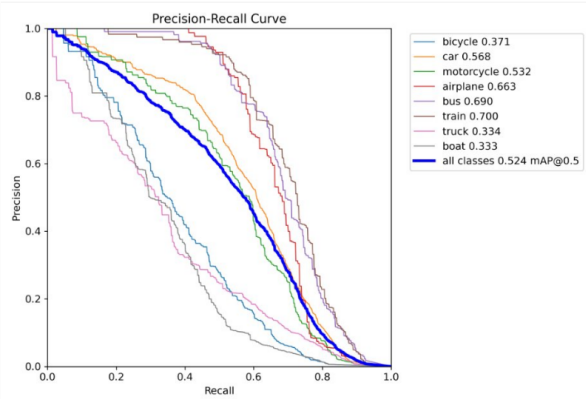
The architecture of YOLOv10 builds on the strengths of its predecessors while introducing several key innovations. The model comprises the following components:

- Backbone: This component is responsible for feature extraction and employs an enhanced version of CSPNet (Cross Stage Partial Network) to improve gradient flow and reduce computational redundancy.
- Neck: Designed to aggregate features from different scales and pass them to the head, the neck incorporates PAN (Path Aggregation Network) layers for effective multi-scale feature fusion.
- One-to-Many Head: During training, this head generates multiple predictions per object to provide rich supervisory signals and enhance learning accuracy.
- One-to-One Head: During inference, this head generates a single best prediction per object, eliminating the need for non-maximum suppression (NMS), thus reducing latency and improving efficiency.

Making a comparison between YOLOv9 model and YOLOv10 model on the dataset, we see a higher precision and recall from YOLOv9.




**(a)** YOLOv9 Precision-Recall Curve




**(b)** YOLOv10 Precision-Recall Curve

We also compare the numbers of parameters and FLoating-point Operations Per Second (FLOP) for all the model we choose, including: YOLOv9-C, YOLOv9-M and YOLOv10-L to see the efficiency. The YOLOv10-L has higher FLOP while number of parameters is smaller than YOLOv9-M.

| | **Test Size** | **Params** | **FLOPs** |
|---|---|---|---|
| YoLov9-C | 640 | 20.0M | 76.3G |
| YoLov9-M | 640 | 25.3M | 102.1G |
| YoLov10-L | 640 | 24.4M | 124.3G |

**Table 4.2:** YoLov9 and v10 comparison

### 4.1.6 DETR model

The DEtection TRansformer (DETR)[7] is an innovative object detection model that uses a transformer architecture, allowing it to capture global context and simplify the detection pipeline by removing the need for anchor boxes and non-maximum suppression. By treating object detection as a set prediction problem, it employs a bipartite matching loss to match predicted objects with ground truth. Despite its advantages, such as end-to-end learning and scalability, DETR requires significant training time and struggles with small object detection.

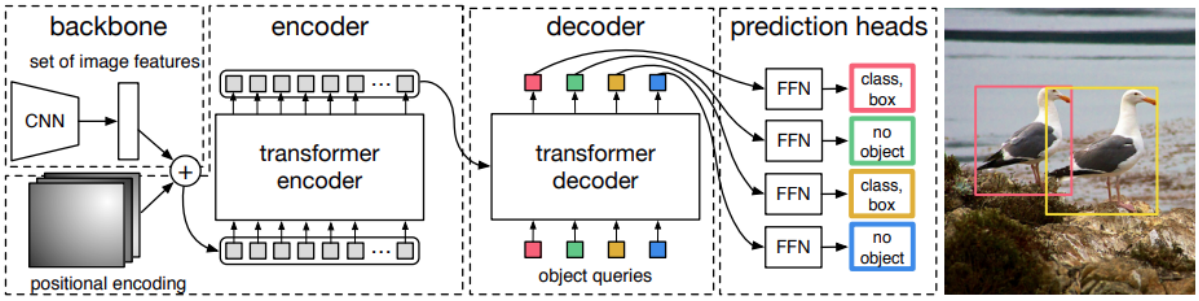



**Figure 4.9:** DETR architecture, featuring a backbone, a transformer encoder, a transformer decoder, and four prediction heads.

DETR starts by processing an input image through a CNN encoder to extract high-level spatial features, which serve as the foundation for subsequent operations. Since transformers lack inherent spatial understanding, DETR adds positional encodings to the CNN's output, helping the model understand the spatial relationships between image parts. It introduces "object queries," "keys," and "values" to drive the self-attention mechanism, with object queries being learnable representations and keys and values representing spatial and feature information, respectively.

Multi-head self-attention in DETR captures complex relationships between objects, focusing on different image regions simultaneously to understand both local and global contexts. To

align predicted bounding boxes with actual objects, DETR uses bipartite matching, associating predictions with ground-truth objects to refine the model during training.

### 4.1.7 Choosing Teacher model

| | **MAP(IOU=0.5:0.95)** | **MAP(IOU=0.5)** | Training Time |
|---|---|---|---|
| **FasterRCNN -R50 No pretrain** | 17.2% | 35.1% | 2h15m |
| **RetinaNet -R50 No pretrain** | 18.6% | 38.1% | 1h7m |
| **SSD300 -VGG16** | 25.5%<br>NoPretrain: <1% | 46.2%<br>NoPretrain: <1% | 20m<br>NoPretrain: 12m |
| **YoLov9c with pretrain, NoPretrain using m version** | 50.8%<br>NoPretrain: 6.5% | 69.7%<br>NoPretrain: 13% | 54m<br>NoPretrain: 42m |
| **YoLov10l** | 34.5% | 52.3% | 50m |
| **DETR** | 44.5% | 70.3% | 1h30m |

**Table 4.3:** Teacher model Results

For choosing teacher model, we need to choose best model one so we decide YoLov9-C to be the teacher for guiding other model

## 4.2 Student model

We want to see how student model best benefit from the teacher model, we must choose the model basically enough and not having mAP result too high, especially lower than teacher model(YoLov9) but not too low

For candidates for student model:

- We thinking Faster RCNN with R50 backbone no pretrained, Retina with R50 backbone no pretrained, SSD300 for minimize the calculation cost with VGG 16 backbone with pretrained to see the differences between pretrain and no pretrain.

- We guess pretrain one will not learn much from unlabeled data predicted by YOLOv9 because it has already pretrain on COCO dataset and see all unlabeled data.

# 5 Unbiased and Efficient Teacher Models

## 5.1 Efficient Teacher

The Efficient Teacher framework is a novel approach designed to enhance Semi-Supervised Object Detection (SSOD) for one-stage anchor-based detectors, such as YOLOv5[8]. The frame-

work addresses the key challenges in SSOD, including pseudo label inconsistency, performance inefficiencies, and the difficulty of applying SSOD to one-stage anchor-based detectors.

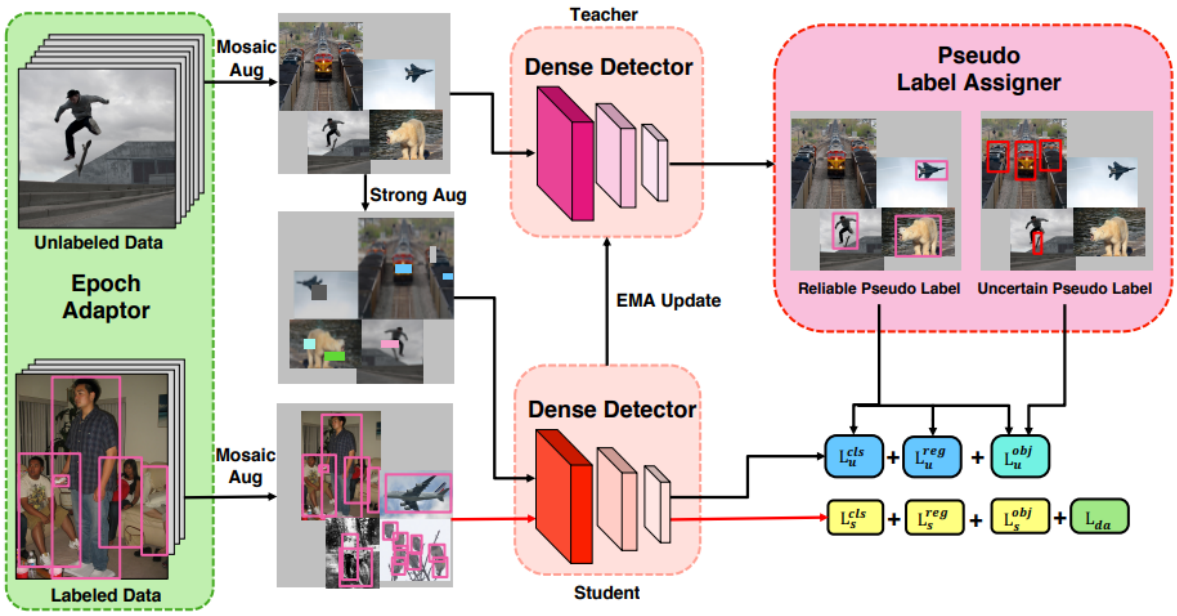



**Figure 5.1:** Efficient Teacher framework

Efficient Teacher proposes three modules to implement a scalable and effective Semi-Supervised Object Detection framework:

- Dense Detector: This component serves as a baseline model, extending RetinaNet with techniques inspired by YOLOv5, such as dense sampling. The Dense Detector improves pseudo label quality through dense input, enhancing inference efficiency and overall detection performance.
- Pseudo Label Assigner (PLA): PLA refines pseudo label assignment by distinguishing between reliable and uncertain pseudo labels. Reliable pseudo labels are used for default supervised training, while uncertain ones guide the student model with a soft loss calculation. This method mitigates the pseudo label inconsistency problem that can mislead model updates.
- Epoch Adaptor (EA): EA optimizes the training process through domain and distribution adaptation. During the Burn-In phase, it employs adversarial learning for domain adaptation. In the SSOD training phase, it dynamically estimates pseudo label thresholds based on the proportions of each class label in the labeled data, ensuring consistent feature learning across different domains.

Comparing with the common teacher-student framework introduced earlier, we see the most noticeable additional component in the Pseudo Label Assigner (PLA). Traditionally, the teacher-student model applies a Pseudo Label Filter (PLF) strategy, which filters out pseudo labels below a set threshold. This method can result in sub-optimal assignments since, during the entire process of SSOD training, the scores of pseudo labels continue to increase, which can lead to the Pseudo Label Filter treating incorrect pseudo labels as reliable ones and including

them in training, resulting in the phenomenon of failure. PLA solves this sensitivity issue by introducing a soft unsupervised loss that efficiently leverages uncertain pseudo labels below the set threshold. The loss of the Dense Detector in SSOD is defined as a pair of a single labeled image and a single unlabeled image:

$$L = L_s + \lambda L_u \tag{2}$$

where $L_s$ represents the loss function computed on a labeled image, while $L_u$ represents the loss function computed on an unlabeled image. $\lambda$ is used to balance the supervised loss and the semi-supervised loss.

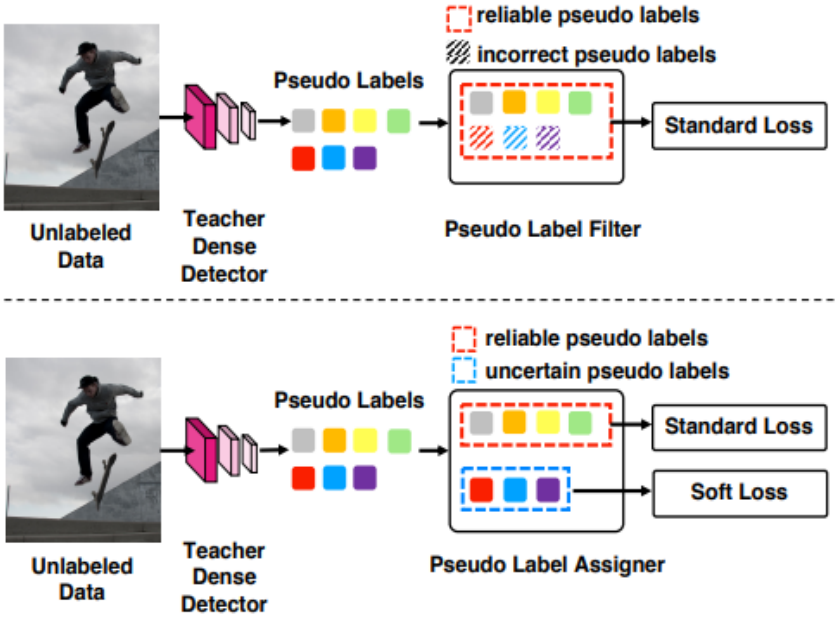



**Figure 5.2:** Compare Pseudo Label Assigner (PLA) with traditional Pseudo Label Filter (PLF)

When experimenting with the Efficient Teacher, we follow the original framework, which uses the YOLOv5 model. In the future, we hope to try more recent models, including the YOLOv9 or YOLOv10 versions, to hopefully improve the results.

## 5.2 Unbiased Teacher

The Unbiased Teacher framework is a Semi-Supervised Object Detection framework that leverages both labeled and unlabeled data to improve the performance of object detectors[9], [10]. The framework extends the traditional teacher-student model, aiming to address the inefficiencies in pseudo-labeling, especially in bounding box regression. The key stages of the Unbiased Teacher framework include:

- Burn-in Stage: This initial phase involves training an object detector using the labeled dataset. The standard supervised losses are used to build a foundational model.
- Mutual Learning Stage: In this phase, the pre-trained detector is duplicated into a Teacher model and a Student model. The Teacher model generates pseudo-labels from weakly-

augmented unlabeled images. These pseudo-labels, filtered by a confidence threshold, are then used to calculate the unsupervised loss for training the Student model. The Student model is trained with a combination of supervised and unsupervised losses, and its weights are periodically updated to the Teacher model through an Exponential Moving Average (EMA) mechanism

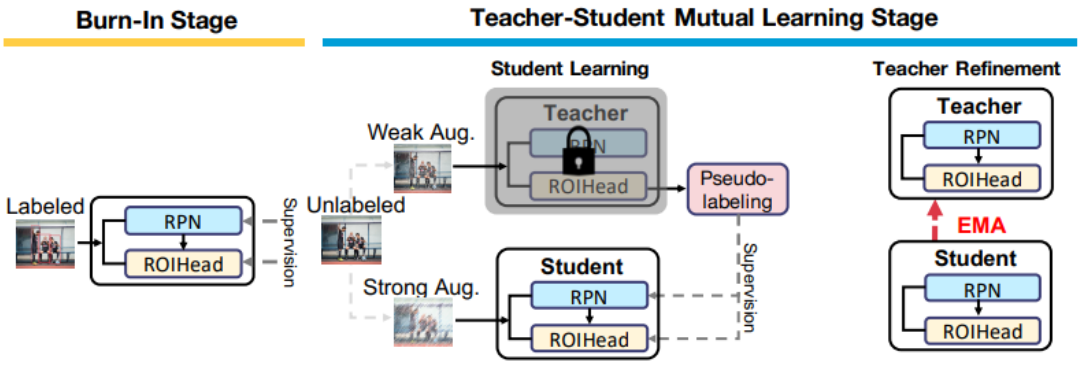



**Figure 5.3:** Unbiased Teacher framework

The Unbiased Teacher framework introduces a key innovation, the Listen2Student Mechanism, that distinguishes it from common teacher-student frameworks. This novel approach is specifically designed to enhance the quality of pseudo-labels used in training. It prevents misleading pseudo-labels in bounding box regression by comparing the uncertainties between the Teacher and the Student models. This mechanism ensures that only reliable pseudo-labels are utilized, thus improving the accuracy of the regression branch in a semi-supervised setting.

While Student Learning with Pseudo-Labeling can commonly be found in other teacher-student frameworks, to obtain more stable pseudo-labels, the Unbiased Teacher framework applies EMA to gradually update the Teacher model. The slowly progressing Teacher model can be regarded as the ensemble of the Student models in different training iterations.

$$\theta_t \leftarrow \alpha\theta_t + (1-\alpha)\theta_s.$$

This approach has been shown to be effective in many existing works, e.g., ADAM optimization, Batch Normalization, and now demonstrates its effectiveness in alleviating the pseudo-labeling bias issue for the Semi-Supervised Object Detection problem.

For our work, we experiment with the second version of the framework, Unbiased Teacher v2. We also use the FasterRCNN-R50 as it has proven its efficiency in our original teacher-student framework.

# 6 Experimental Results

| | **MAP(IOU=0.5:0.95)** | **MAP(IOU=0.5)** | Training Time |
|---|---|---|---|
| **FasterRCNN -R50 No pretrain** | 17.2% → 31.6% | 35.1% → 50.8% | 2h15m → 9h28m |
| **RetinaNet -R50 No pretrain** | 18.6% → 34.3% | 38.1% → 52.4% | 1h7m → 9h9m |
| **SSD300 -VGG16** | 25.5% → 26.7% NoPretrain: <1% → 1.5% | 46.2% → 46.6% NoPretrain: <1% → 4.1% | 20m → 1h40m NoPretrain: 12m → 1h40m |
| **YoLov9m NoPretrain** | 6.5% → 37.4% | 13% → 54.4% | 42m → 5h10m |
| **Efficient Teacher, YoLov5-L** | 8.8% | 19.4% | 4h50m |
| **Unbiased Teacher, Faster RCNN-R50 6 epochs** | 33.2% | 57.7% | 11h30m |

**Table 6.1:** Final Semi-Supervised Model Result

We can see from the result that the Teacher-Student model do quite while leverage mAP result a lot, the pretrained one show little different, this can be witnessed from begin because pretrained model have been trained on COCO dataset. So all no pretrained model is proved that our teacher model do quite good its job. Efficient Teacher can do better with YoLo newer verison while Unbiased Teacher is the best model result we have but the training time is really slow.

For trade off between time training and accuracy, we recommend Teacher-Student model RetinaNet is really balance, also run faster than FasterRCNN. If you want something as good as possible can use Unbiased Teacher with RetinaNet-R101, it may better than FasterRCNN-R50

Conclusion and improvement

- Student-Teacher Model is most important model in semi-supervised object detection problem, most of other turn around it and we can all see how good it is.
- Unbiased Teacher v2 is better than normal Teacher-Student Model though it teach by model its self which is FasterRCNN-R50, if we can apply to YoLov9 or like that, sure that the accuracy will improve.

- Further we can do is that using more Data augmentation method, GAN if possible then it may improve a little accuracy, not too much because we think newest model like YoLov9 already data augmentation good enough.
- Do YoLov9, v10 or other robust model for efficient teacher model might help it compare with others semi-supervised model.

# REFERENCES


[1] T. Lin, M. Maire, S. J. Belongie, *et al.*, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014.
arXiv: 1405.0312. [Online]. Available: http://arxiv.org/abs/1405.0312.

[2] Sep. 2020. [Online]. Available:
https://www.kaggle.com/datasets/awsaf49/coco-2017-dataset.

[3] A. Paszke, S. Gross, F. Massa, *et al.*,
"Pytorch: An imperative style, high-performance deep learning library,"
in *Advances in Neural Information Processing Systems 32*,
Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available:
http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[4] H.-S. Chang, C.-Y. Wang, R. R. Wang, G. Chou, and H.-Y. M. Liao, "YOLOR-based multi-task learning," *arXiv preprint arXiv:2309.16921*, 2023.

[5] C.-Y. Wang and H.-Y. M. Liao, "YOLOv9: Learning what you want to learn using programmable gradient information," 2024.

[6] A. Wang, H. Chen, L. Liu, *et al.*, "Yolov10: Real-time end-to-end object detection," *arXiv preprint arXiv:2405.14458*, 2024.

[7] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, *End-to-end object detection with transformers*, 2020. arXiv: 2005.12872.

[8] B. Xu, M. Chen, W. Guan, and L. Hu, "Efficient teacher: Semi-supervised object detection for yolov5," *arXiv preprint arXiv:2302.07577*, 2023.

[9] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, *Detectron2*,
https://github.com/facebookresearch/detectron2, 2019.

[10] Y.-C. Liu, C.-Y. Ma, and Z. Kira, *Unbiased teacher v2: Semi-supervised object detection for anchor-free and anchor-based detectors*, 2022. arXiv: 2206.09500.