Object Detection Algorithms - A Review

Marek Letavay
Asseco Central Europe, a.s.
Galvaniho 19, 821 04, Bratislava,
Slovak Republic
marek.letavay@asseco-ce.com

Michael Bažant
Faculty of Electrical Engineering and Informatics, University of Pardubice
nám. Čs. Legií 565, 530 02, Pardubice,
Czech Republic
michael.bazant@upce.cz

Pavel Tuček
Faculty of Electrical Engineering and Informatics, University of Pardubice
nám. Čs. Legií 565, 530 02, Pardubice,
Czech Republic
pavel.tucek@upce.cz

***Abstract* — Over several decades, traffic engineers have persistently focused on implementing various measures to enhance the safety of drivers and pedestrians amidst the growing traffic on streets worldwide. This pressing concern stems from the continuous growth in vehicles, pedestrians, and road density. However, with the remarkable advancements in computational power and the simultaneous reduction in the cost of technical equipment, the integration of object detection has emerged as a viable solution, even in real-time applications within the realm of traffic engineering. However, it is essential to note that the impact of computer technology extends far beyond mobility-related issues. Its wide-ranging influence has permeated diverse fields of interest, including medicine, where it aids in diagnostics and treatments, face recognition systems used for security, industrial automation processes, and even artistic endeavors, where it has opened new frontiers of creativity. This indicates object detection technology's vast potential and versatility, with its ability to revolutionize numerous sectors and bring about meaningful societal advancements. This review outlines the chronological progression of the essential YOLO algorithm, highlighting its significant advancements and developments over the years.**

Keywords—Object Detection, Computer Vision, Vision Detection Algorithms, YOLO

I. Introduction

Object detection algorithms are essential in computer vision, allowing machines to identify and locate objects in images or videos. These algorithms utilize various techniques, including deep learning and machine learning, to achieve accurate results. They typically involve generating object proposals or predicting bounding boxes and class labels for detected objects. Popular object detection algorithms include Faster R-CNN, SSD, and YOLO, each with its strengths and trade-offs in speed and accuracy. These algorithms have diverse applications, ranging from autonomous driving and surveillance systems to object recognition in healthcare and retail industries. Continued advancements in object detection algorithms contribute to developing advanced computer vision systems. Before the introduction of YOLO (You Only Look Once) in 2015, various object detection algorithms already existed.

R-CNN (Regions with Convolutional Neural Networks) proposed a two-step approach, where it first generated region proposals using selective search and then classified these regions using convolutional neural networks (CNNs). Although effective, it was computationally expensive. Followed by Fast R-CNN and Faster R-CNN, it improved upon R-CNN by sharing the convolutional features across multiple region proposals, making the process more efficient. It achieved better speed and accuracy than its predecessors. Another approach was SSD (Single Shot MultiBox Detector), which eliminated the need for region proposal generation by directly predicting bounding boxes and class probabilities at different scales and aspect ratios within the network. This led to real-time object detection capabilities. Before YOLO, these object detection algorithms marked significant advancements in the field. However, their reliance on region proposal methods introduced computational complexity. YOLO brought a revolutionary change by treating object detection as a regression problem in a single network pass. This innovative approach transformed the field, offering a real-time, end-to-end solution with remarkable speed and accuracy. We will use [80] to benchmark all related algorithms and methods.

II. Traditional Computer Vision Methods. The Beginning of Object Detection Algorithmization

Object detection algorithms are a crucial component of computer vision systems that enable machines to identify and locate objects within digital images or video frames. These algorithms play a vital role in various applications, such as autonomous vehicles, surveillance systems, robotics, and image analysis. There are several approaches to object detection, with each algorithm having its strengths and limitations. Two main types of object detection algorithms exist traditional ones based on image processing (used before 2014) and those based on convolutional neural networks (used after 2014). Object detection using traditional computer vision methods emerged in the late 1990s. These methods combine classic feature detection with a machine learning algorithm such as KNN or SVM for classification.

A. Viola-Jones

This method was first introduced in 2001 by [1], [2]. For the first time, this object recognition framework allows the real-time detection of human faces without any constraints. It is based on sliding windows to go through all possible locations and scales in the predefined image to see if any window represents a human face. The sliding windows at the top search for a 'haar-like' feature. The Viola-Jones methodology and algorithm have dramatically improved its detection speed by using several unique techniques: "integral image," "feature selection," and "detection cascades." The Viola-Jones method is known for its fast processing speed, making it suitable for real-time applications. Although primarily designed for face detection, it has also been adapted for detecting other objects with slight modifications to the training process and feature selection.

B. HOG

The Histogram of Oriented Gradients (HOG) feature descriptor was introduced in 2005 by [3]. It is an improvement of the scale-invariant feature transform and shape contexts of its time. The HOG algorithm divides an image into small cells and computes each cell's gradient magnitude and orientation. The gradient magnitude represents the strength of the intensity variations, while the

gradient orientation indicates the direction of these variations (cells are similar to a sliding window).

In case of different size detection problems, the HOG detector rescales the input image multiple times while keeping the size of a detection window unchanged. The HOG descriptor is often used with machine learning algorithms, such as support vector machines (SVM), for object detection and classification tasks. It has been particularly successful in pedestrian detection, where the HOG features help distinguish human-like shapes from the background, [4], [5], [6].

C. Deformable part-based model (DPM)

The Deformable Part-based Model (DPM) is a popular object detection framework introduced by [7]. It is a variant of the traditional part-based models and has been widely used for detecting objects with articulated structures or deformable parts. The deformable parts model (DPM) algorithm, proposed by [7], is a significant advancement in traditional object detection algorithms. It operates on the "divide and conquer" principle and involves correctly training the algorithm to decompose an object. During inference, the algorithm detects different object parts. The Deformable Part-based Model provides a flexible and practical approach to object detection, particularly for objects with deformable parts or articulated structures. Its ability to model spatial relationships and capture appearance variations has made it a valuable tool in computer vision. For example, the problem of detecting a "car" can be considered detecting its window, body, and wheels, [8].

III. Deep Learning Methods

Modern object detection algorithms often consist of advanced architectures for precise object localization and instance segmentation. Moreover, with introducing deep learning, where pre-trained models trained on large datasets are fine-tuned for object detection tasks. Deep learning methods represent a subfield of machine learning that focuses on training artificial neural networks with multiple layers, known as deep neural networks, to learn and extract hierarchical representations of data. These methods have revolutionized many areas of artificial intelligence and achieved remarkable breakthroughs in tasks such as image recognition, natural language processing, and speech recognition. Unlike traditional machine learning algorithms that rely on handcrafted features, deep learning methods can automatically learn and discover relevant features from raw data. This is accomplished through large-scale labeled datasets and backpropagation, a gradient-based optimization technique that adjusts the network's parameters to minimize the difference between predicted and actual outputs. Deep Learning methodologies can do end-to-end object detection using Convolutional Neural Networks. Within the last several years, authors have introduced a variety of applications throughout different fields of interest like face detection and human head detection, [58], pedestrian safety, [59], and other relevant fields, [57]. The algorithms for object detection are divided onto one-stage object detection and two-stage object detection.

The essential difference between one-stage and two-stage detectors is in high inference speeds for one-stage detectors and high localization and recognition accuracy for two-stage detectors.

A RoI (Region of Interest) Pooling layer can further divide the two stages of a two-stage detector. The best representative of two-stage object detectors can be Faster R-CNN. The critical innovation of Faster R-CNN lies in introducing a Region Proposal Network (RPN), which operates within the network architecture and generates region proposals directly in the first stage. This approach makes external proposal generation methods, such as selective search, no longer necessary. During the second stage, features are gathered through RoI pooling operation from each candidate box to complete classification and bounding-box regression tasks, [9]. A one-stage detector can predict bounding boxes in a single step without relying on region proposals. Instead, it utilizes a grid box and anchors to locate the image's detection region and constrain the object's shape, [10].

Object detection can be performed using either a one-stage or two-stage detector. A comprehensive comparison can be seen in Fig. 1. One-stage detectors directly classify objects and determine their bounding boxes without using pre-generated region proposals and a comprehensive comparison can be seen in Fig.2. Two-stage detectors, on the other hand, use region proposals to predict objects. This involves generating region proposals using the selective search or a Region Proposal Network (RPN), classifying each proposal, and refining the proposals through bounding-box regression or binary-mask prediction. The inference time for one-stage detectors is T1st, while that of two-stage detectors is T1st + mT2nd, where m is the number of proposals with a high confidence score. Although two-stage detectors are slower, they generally achieve better accuracy than one-stage detectors. A comprehensive comparison of two-stage detectors can be seen in Fig. 15.

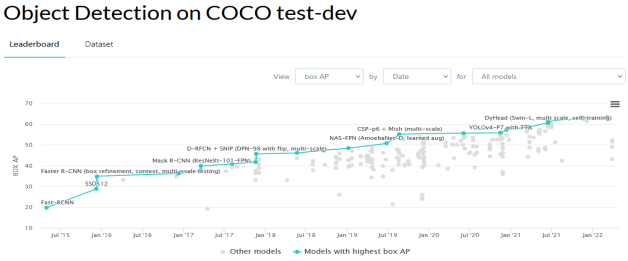

Fig. 1. Object detection methods on COCO test-dev comparison. Retrieved from [88]

A. One-Stage Detectors

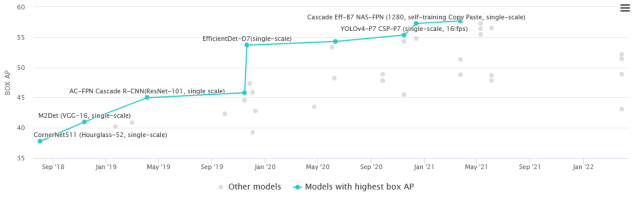

Fig. 2. One-stage detector comparison on COCO test-dev. Retrieved from [89]

1. YOLO: You Only Look Once

The YOLO algorithm, which stands for "You Only Look Once," was created by R. Joseph and his team in 2015 for detecting objects. It was the first one-stage detector in the deep learning era, [11]. The name indicates a departure from the previous detection paradigm of "proposal detection + verification." The key idea behind YOLO is to treat object detection as a regression problem, where the algorithm directly predicts bounding boxes and class probabilities in a

single neural network pass. Unlike other methods that use region proposal techniques, YOLO eliminates the need for a separate region proposal step, making it significantly faster. In other words, it uses a different approach using a single neural network for the full image. The YOLO algorithm has found extensive applications in real-time object detection scenarios, including surveillance systems, autonomous vehicles, robotics, and video analysis. Its ability to achieve high accuracy with remarkable speed has made it a popular choice among researchers and practitioners in the computer vision community. Despite its significant improvement in detection speed, YOLO suffers a drop in localization accuracy compared with two-stage detectors, especially for some small objects. (VOC07 mAP=63.4% and VOC12 mAP=57.9%.)

2. SSD: Single Shot MultiBox Detector

SSD was proposed by [12]. It was the second one-stage detector during the evolution of deep learning methods. Single-shot means that the SSD algorithm is a one-stage method, and MultiBox indicates that the SSD is a multi-frame prediction. During training, SSD optimizes the network parameters by minimizing a loss function that combines the localization loss (the difference between predicted and ground truth bounding box coordinates) and the classification loss (the difference between predicted and ground truth class labels). The algorithm learns to adjust the default anchor boxes to match the ground truth objects better. One of the strengths of SSD is its ability to efficiently detect objects at multiple scales and aspect ratios within a single shot, eliminating the need for multiple passes or region proposal methods. This makes it faster than many other object detection algorithms while maintaining good detection accuracy. Compared with Yolo, SSD uses CNN to perform detection directly instead of performing detection after the fully connected layer as Yolo does. The main contribution of SSD is the introduction of multi-reference and multi-resolution detection techniques.

The use of SSD technology offers a combination of quick detection and high accuracy rates. It has been shown to achieve VOC07 mAP at 76.8% and VOC12 mAP at 74.9%, as well as COCO mAP@.5 at 46.5%. Additionally, a fast version of SSD can run at 59 frames per second while maintaining a mAP@[.5,.95] of 26.8%.

3. RetinaNet

RetinaNet is an object detection algorithm introduced by [13]. It is designed to address the challenge of detecting objects at multiple scales and dealing with the training data's imbalance between foreground and background examples. The innovation of RetinaNet is the introduction of a novel feature pyramid network (FPN) architecture combined with a focal loss. The FPN enables the network to leverage features from different layers of a backbone network to detect objects at various scales, thereby improving accuracy across different object sizes. The statement that the one-stage method is fast but not as accurate as the two-stage because the positive and negative samples are not balanced was shown as valid. RetinaNet optimizes its network parameters during training by minimizing the focal loss, which combines the classification and regression losses for the anchor boxes. This loss function enables the network to prioritize training challenging examples and effectively handle the class imbalance. The focal loss is a type of cross-entropy loss that can be adjusted dynamically. As the certainty of the correct category increases, the zoom factor decreases to zero. This zoom factor can help lessen the impact of easy examples during training, allowing the model to focus on more complex examples, [14]

Using focal loss, one-stage detectors can achieve similar accuracy to two-stage detectors without sacrificing detection speed. For instance, COCO mAP@.5 has reached 59.1% with mAP@[.5, .95] of 39.1%.

4. EfficientDet: Scalable and Efficient Object Detection

EfficientDet is an object detection algorithm, [15]. It is designed to address the challenges of achieving high accuracy and computational efficiency in object detection tasks. EfficientDet builds upon the principles of the EfficientNet architecture, which focuses on developing efficient and scalable neural networks. It employs a compound scaling method that uniformly scales the network width, depth, and resolution to optimize accuracy and efficiency.

They have designed a weighted bi-directional feature pyramid network (BiFPN), which gives an easy and fast multi-scale feature fusion with a combination of compound scaling methods that uniformly scales the resolution, depth, and width for all backbone, feature network, and box/class prediction networks at the same time. EfficientDet consistently achieves much better efficiency than prior methods. In particular, with single-model and single-scale, EfficientDet-D7 achieves state-of-the-art 55.1 AP on COCO test-dev with 77M parameters and 410B FLOPs1, being 4x – 9x smaller and using 13x – 42x fewer FLOPs than previous detectors. EfficientDet architecture is visualized in Fig. 4.

BiFPN plays the feature network role, which takes level 3-7 features {P3, P4, P5, P6, P7} from the backbone network and repeatedly applies top-down and bottom-up bidirectional feature fusion. These fused features are fed to a class and box network to produce object class and bounding box predictions, respectively, in contrast with previous works, [16], [17], [18]. In a recent study, [19], significant progress was made in image classification by increasing the size of the network in terms of width, depth, and input resolution. The new BiFPN, class/box network, and resolution have much more scaling dimensions than image classification models. The proposed heuristic-based scaling approach is shown below. A comparison of different network design can be seen in Fig. 3.

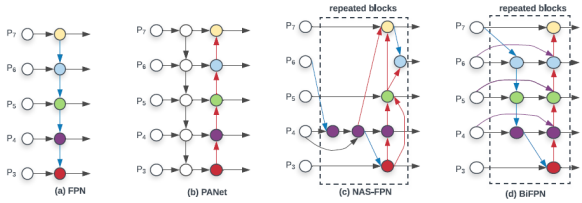

Fig.3: Feature network design a)FPN, b) PaNet, c) NAS-FPN, and d) BiFPN. Retrieved from [13]

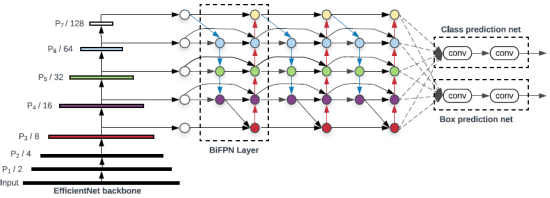

Fig.4: EfficientDet architecture. Retrieved from [13]

5. SpineNet-190

SpineNet-190 is a deep neural network architecture designed for image recognition tasks, particularly object detection and segmentation. It was introduced by [20]. The key idea behind SpineNet-190 is to create a highly efficient and effective backbone network by leveraging the "spinal cord" architecture concept. This architecture comprises a hierarchy of repeated "spinal blocks" that capture multi-scale features across different resolutions. The number "190" in SpineNet-190 refers to the total number of layers in the network. These layers allow SpineNet-190 to capture detailed features from input images, enhancing its ability to recognize objects with fine-grained details. In previous works, [21], [22], [23], [24], it was common to pre-train the backbones of object detection and segmentation models on ImageNet classification. In [25], (against [26], [27]) the study showed that ImageNet pre-training needs to improve accuracy on the COCO dataset. SpineNet-190 first used ImageNet to help COCO object detection with self-training based on discarding the labels on ImageNet. Pseudo-labels were generated on ImageNet. The pseudo-labeled ImageNet and 4labeled COCO data were combined to train a new model. By comparing [28], [29], [30], [31], in [20], one question remains. To what degree does self-training work better than pre-training?

Data Augmentation: Data augmentation is a crucial technique in training deep neural networks, including SpineNet-190, to enhance model performance and improve generalization. SpineNet-190 uses four different augmentation policies of increasing strength that work for both detection and segmentation. Standard flip (S1) and crop (S2) augmentation, [32], AutoAugment (S3), [33], [34], and RandAugment (S4), [35].

Pre-training: To evaluate the effectiveness of pre-training, they have studied ImageNet pre-trained checkpoints of varying quality. All checkpoints use the same model architecture but have different accuracies on ImageNet EfficientNet-B7 architecture, [36], which was used as a strong baseline for pre-training. For the EfficientNet-B7 architecture, with two checkpoints: 1) EfficientNet-B7 checkpoint trained with AutoAugment 84.5% top-1 accuracy on ImageNet; 2) EfficientNet-B7 checkpoint trained with the Noisy Student method, with additional 300M unlabeled images and 86.9% top-1 accuracy. Rand Init denoted training from random initialization. The impact of pre-training when varying the augmentation strength is shown in Fig. 5.

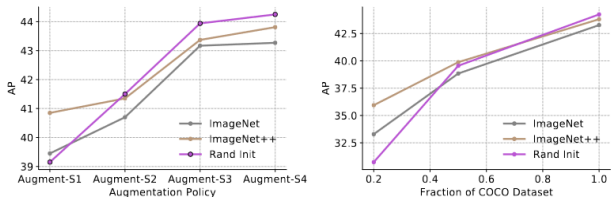

Fig. 5: The effects of data augmentation and dataset size on pre-training. Retrieved from [20]

Left: Supervised object detection performance under various ImageNet pre-trained checkpoint qualities and data augmentation strengths on COCO. Right: Supervised object detection performance under various COCO dataset sizes and ImageNet pre-trained checkpoint qualities.

6. Yolov4-P7 CSP-P7

There are two types of one-stage object detectors: anchor-based, [41], [43], and anchor-free, [37], [38], [39], [40], [36]. Anchor-free detectors, such as CenterNet, [46], are popular because they do not require complicated post-processing like Non-Maximum Suppression. However, for more accurate real-time detection, anchor-based detectors like EfficientDet, [44], YOLOv4, [45], and PP-YOLO, [42], are preferred. Scaled-YOLOv4 is a variation of YOLOv4, [47], that balances accuracy and computational efficiency well, introduced by [37]. A Structured overview of YOLOv4 is visualized in Fig. 6a and Fig. 6b.

The key idea behind Scaled-YOLOv4 is to scale up the YOLOv4 architecture by increasing the size of the network while maintaining its efficiency. This is achieved by using larger input image sizes and deeper network architectures. One common technique for scaling models is to adjust the depth (number of convolutional layers) and width (number of convolutional filters) of the backbone in a CNN. Different variations of the ResNet series are often used on various devices, with ResNet-152 and ResNet-101 being popular on cloud server GPUs, ResNet-50 and ResNet-34 on personal computer GPUs, and ResNet-18 and ResNet-10 on low-end embedded systems. Scaled-YOLOv4 is a model that balances speed and accuracy and can perform real-time object detection on multiple platforms, including 16 FPS, 30 FPS, and 60 FPS movies and embedded systems. YOLOv4-large, designed for cloud GPUs, prioritizes high accuracy for object detection. Experiments were conducted on YOLOv4-P5, YOLOv4-P6, and YOLOv4-P7 models, with YOLOv4-P6 reaching real-time performance at 30 FPS video when the width scaling factor is one and YOLOv4-P7 achieving real-time performance at 16 FPS video when the width scaling factor is 1.25.

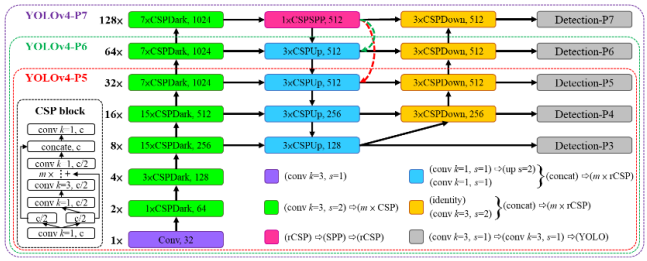

Fig. 6a: Structured overview of YOLOv4. Retrieved from [37]

Method	Backbone	Size	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
EfficientDet-D0	EfficientNet-B0	512	97*	**34.6%**	**53.0%**	**37.1%**	**12.4%**	**39.0%**	**52.7%**
YOLOv4-CSP	**CD53s**	512	97/93*	**46.2%**	**64.8%**	**50.2%**	**24.6%**	**50.4%**	**61.9%**
EfficientDet-D1	EfficientNet-B1	640	74*	40.5%	59.1%	43.7%	18.3%	45.0%	57.5%
YOLOv4-CSP	**CD53s**	640	73/70*	**47.5%**	**66.2%**	**51.7%**	**28.2%**	**51.2%**	**59.8%**
YOLOv3-SPP	D53	608	73	36.2%	60.6%	38.2%	20.6%	37.4%	46.1%
YOLOv3-SPP ours	D53	608	73	42.9%	62.4%	46.6%	25.9%	45.7%	52.4%
PP-YOLO	R50-vd-DCN	608	73	45.2%	65.2%	49.9%	26.3%	47.8%	57.2%
YOLOv4	CD53	608	62	43.5%	65.7%	47.3%	26.7%	46.7%	53.3%
YOLOv4 ours	CD53	608	62	45.5%	64.1%	49.5%	27.0%	49.0%	56.7%
EfficientDet-D2	EfficientNet-B2	768	57*	43.0%	62.3%	46.2%	22.5%	47.0%	58.4%
RetinaNet	S49s	640	53	41.5%	60.5%	44.6%	23.3%	45.0%	58.0%
ASFF	D53	608*	46	42.4%	63.0%	47.4%	25.5%	45.7%	52.3%
YOLOv4-P5	**CSP-P5**	896	43/41*	**51.8%**	**70.3%**	**56.6%**	**33.4%**	**55.7%**	**63.4%**
RetinaNet	S49	640	42	44.3%	63.8%	47.6%	25.9%	47.7%	61.1%
EfficientDet-D3	EfficientNet-B3	896	36*	47.5%	66.2%	51.5%	27.9%	51.4%	62.0%
YOLOv4-P6	**CSP-P6**	1280	32/30*	**54.5%**	**72.6%**	**59.8%**	**36.8%**	**58.3%**	**65.9%**
ASFF	D53	800*	29	43.9%	64.1%	49.2%	27.0%	46.6%	53.4%
SM-NAS: E2	-	800*600	25	40.0%	58.2%	43.4%	21.1%	42.4%	51.7%
EfficientDet-D4	EfficientNet-B4	1024	23*	49.7%	68.4%	53.9%	30.7%	53.2%	63.2%
SM-NAS: E3	-	800*600	20	42.8%	61.2%	46.5%	23.5%	45.5%	55.6%
RetinaNet	S96	1024	19	48.6%	68.4%	52.5%	32.0%	52.3%	62.0%
ATSS	R101	800*	18	43.6%	62.1%	47.4%	26.1%	47.0%	53.6%
YOLOv4-P7	**CSP-P7**	1536	17/16*	**55.5%**	**73.4%**	**60.8%**	**38.4%**	**59.4%**	**67.7%**
RDSNet	R101	600	17	36.0%	55.2%	38.7%	17.4%	39.6%	49.7%
CenterMask	R101-FPN	-	15	44.0%	-	-	25.8%	46.8%	54.9%
EfficientDet-D5	EfficientNet-B5	1280	14*	51.5%	70.5%	56.7%	33.9%	54.7%	64.1%
ATSS	R101-DCN	800*	14	46.3%	64.7%	50.4%	27.7%	49.8%	58.4%
SABL	R101	-	13	43.2%	62.0%	46.6%	25.7%	47.4%	53.9%
CenterMask	V99-FPN	-	13	46.5%	-	-	28.7%	48.9%	57.2%
EfficientDet-D6	EfficientNet-B6	1408	11*	52.6%	71.5%	57.2%	34.9%	56.0%	65.4%
RDSNet	R101	800	11	38.1%	58.5%	40.8%	21.2%	41.5%	48.2%
RetinaNet	S143	1280	10	50.7%	70.4%	54.9%	33.6%	53.9%	62.1%
SM-NAS: E5	-	1333*800	9.3	45.9%	64.6%	49.6%	27.1%	49.0%	58.0%
EfficientDet-D7	EfficientNet-B6	1536	8.2*	53.7%	72.4%	58.4%	35.8%	57.0%	66.3%
ATSS	X-32x8d-101-DCN	800*	7.0	47.7%	66.6%	52.1%	29.3%	50.8%	59.7%
ATSS	X-64x4d-101-DCN	800*	6.9	47.7%	66.5%	51.9%	29.7%	50.8%	59.4%
EfficientDet-D7x	EfficientNet-B7	1536	6.5*	55.1%	**74.3%**	59.9%	37.2%	57.9%	**68.0%**
TSD	R101	-	5.3*	43.2%	64.0%	46.9%	24.0%	46.3%	55.8%

Fig. 6b: Structured comparison of mentioned methods. Retrieved from [37]

7. Cascade Egg-B7 NAS-FPN

Cascade EGG-B7 NAS-FPN is the next level of improvement designed by [48]. Some previous works on Copy-Paste fully utilized modeling the surrounding visual context for pasting the objects. It was shown in [48], that the mechanism of pasting objects randomly could provide solid gains on top of strong baselines. Copy-Paste can also be used as an additive with semi-supervised methods that leverage extra data through pseudo-labeling (e.g., self-training). They have achieved 49.1 mask AP and 57.3 box AP, an improvement of +0.6 mask AP and +1.5 box AP over the previous state-of-the-art on COCO instance segmentation. It was also shown that Copy-Paste can significantly improve the LVIS benchmark. For reference, see Fig. 7. The baseline model outperforms the LVIS 2020 Challenge winning entry by +3.6 mask AP on rare categories.

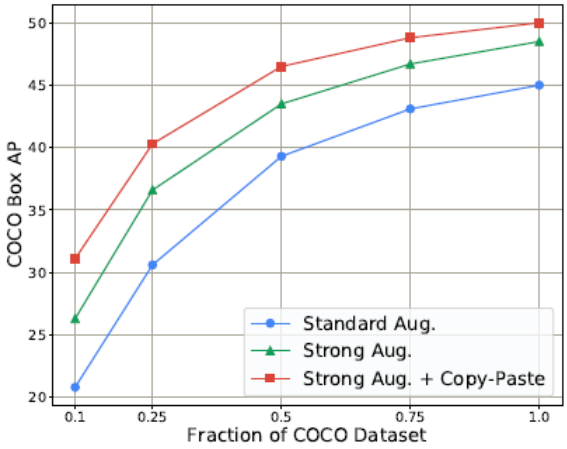

Fig. 7: Data-efficiency on the COCO benchmark: Combining the Copy-Paste augmentation with Strong Aug. (large-scale jittering) allows us to train models up to 2 more data efficient than Standard Aug. (standard scale jittering). The augmentations are highly effective and provide gains of +10 AP in the low data regime (10% of data) while still being effective in the high data regime with a gain of +5 AP. Results are for Mask RCNN EfficientNet-B7 FPN trained on an image size of 640 x 640. Retrieved from [48]

One effective data augmentation method is Copy-Paste augmentation, which involves pasting objects from one image to another. By doing this, a vast number of new training data can be created with multiple possibilities, such as choosing the source images from which objects are copied, selecting the specific object instances to copy, and deciding where to paste them on the target image. This variety of options can offer great opportunities for exploring the most effective ways to use this technique.

Fig. 8: Random Copy-Paste for creating a new image for training. Retrieved from [48]

The authors have discovered that a simple Copy-Paste strategy can significantly boost image recognition performance in various settings. A visualization can be seen in Fig. 8. This involves randomly selecting objects and pasting them onto a target image at random locations. The strategy has proven effective with various backbone architectures, scale jittering, training schedules, and image sizes.

Using an EfficientNet-B7, [51], backbone, and NAS-FPN architecture, [52], the strategy achieved a 57.3 Box AP and 49.1 Mask AP on COCO test-dev without test-time augmentations. This surpasses previous state-of-the-art instance segmentation models, including SpineNet, [49], and DetectoRS ResNeXt-101-64x4d with test time augmentation, [55]. The strategy also outperforms state-of-the-art bounding box detection results of EfficientDet-D7x-1536, [56], and YOLOv4-P7-1536, [61], despite using a smaller image size of 1280 instead of 1536.

The Copy-Paste augmentation strategy is easy to integrate into any instance segmentation codebase, can effectively utilize unlabelled images, and does not create training or inference overheads. Experiments with Mask-RCNN demonstrate that implementing Copy-Paste into its training can easily improve results by up to +1.0 AP for 48 epochs as seen in Fig. 9.

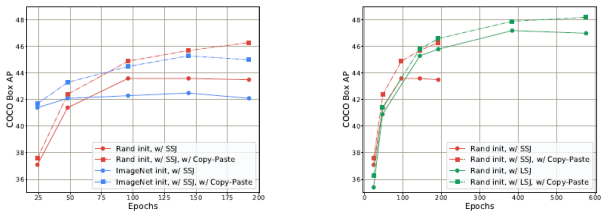

Fig. 9: Copy-Paste provides gains that are robust to training configurations. We train Mask R-CNN (ResNet-50 FPN) on 10241024 image size for varying numbers of epochs. Left Figure: Copy-Paste with and without initializing the backbone by ImageNet pre-training. Right Figure: Copy-Paste with standard and large-scale jittering. Retrieved from [48]

Regarding training Mask R-CNN, the literature suggests that a typical schedule lasts only 24 to 36 epochs. However, it is possible to see improvements in performance by using Copy-Paste even with just 2 or 3 epochs, [53], [54], [50]. As one increases the number of epochs, the gains from Copy-Paste become even more significant. A comprehensive comparison can be seen in Fig. 10a and Fig. 10b. For a

precise evaluation, it is necessary to use the same codebase, training data, settings, data pre-processing and augmentations, architectural regularization, [60], etc.

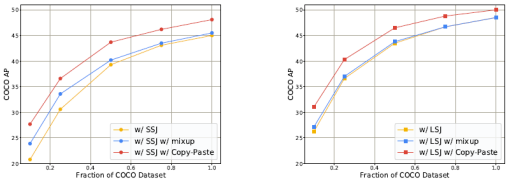

Fig. 10a: Improvement from mixup and Copy-Paste data augmentation. Retrieved from [48]

Model	FLOPs	# Params	AP_{val}	$AP_{test\text{-}dev}$	Mask AP_{val}	Mask $AP_{test\text{-}dev}$
SpineNet-190 (1536)	2076B	176M	52.2	52.5	46.1	46.3
DetectoRS ResNeXt-101-64x4d	—	—	—	55.7†	—	48.5 †
SpineNet-190 (1280)	1885B	164M	52.6	52.8	—	—
SpineNet-190 (1280) w/ self-training	1885B	164M	54.2	54.3	—	—
EfficientDet-D7x (1536)	410B	77M	54.4	55.1	—	—
YOLOv4-P7 (1536)	—	—	—	55.8†	—	—
Cascade Eff-B7 NAS-FPN (1280)	1440B	185M	54.5	54.8	46.8	46.9
w/ Copy-Paste	1440B	185M	(+1.4) **55.9**	(+1.2) **56.0**	(+0.4) **47.2**	(+0.5) **47.4**
w/ self-training Copy-Paste	1440B	185M	(+2.5) **57.0**	(+2.5) **57.3**	(+2.1) **48.9**	(+2.2) **49.1**

Fig. 10b: Results of applying Copy-Paste on top of a strong 54.8 box AP COCO model. Retrieved from [48]

8. Swin-L

Swin-L is a variant of the Swin Transformer architecture for computer vision tasks, including object detection and image classification. Swin-L stands for "Swin-Large," indicating that it is the larger version of the Swin Transformer architecture. It was introduced by [62]. The Swin Transformer architecture aims to overcome the limitations of traditional convolutional neural networks (CNNs) by adopting a self-attention mechanism inspired by the Transformer model commonly used in natural language processing. It achieves this by dividing the image into smaller patches and applying self-attention operations to capture long-range dependencies and contextual information. The Swin Transformer is a versatile hierarchical architecture that can model at different scales with linear computational complexity to image size. This makes it suitable for a wide range of vision tasks, including image classification (achieving 87.3 top-1 accuracy on ImageNet-1K) and dense prediction tasks like object detection (with 58.7 boxes AP and 51.1 masks AP on COCO test-dev) and semantic segmentation (achieving 53.5 mIoU on ADE20K). The Swin Transformer outperforms the previous state-of-the-art by a significant margin of +2.7 box AP and +2.6 mask AP on COCO, and +3.2 mIoU on ADE20K, proving the potential of Transformer-based models as vision backbones. Additionally, the hierarchical design and shifted window approach benefit all MLP architectures.

Current Transformer-based models, [66], [63], have a fixed scale for tokens, which could be better for vision applications. Additionally, images have a much higher pixel resolution than text passages have word resolution. This poses a challenge for vision tasks, such as semantic segmentation, that require precise pixel-level prediction. The computational complexity of self-attention in the Transformer model would be unmanageable for high-resolution images. A new general-purpose Transformer backbone, called Swin Transformer, was introduced to address these challenges. This model constructs hierarchical feature maps with linear computational complexity to image size, making it suitable for vision applications. As shown in Fig. 11(a), Swin Transformer creates a hierarchical representation by beginning with small-sized patches (outlined in grey) and progressively merging neighboring patches in deeper Transformer layers.

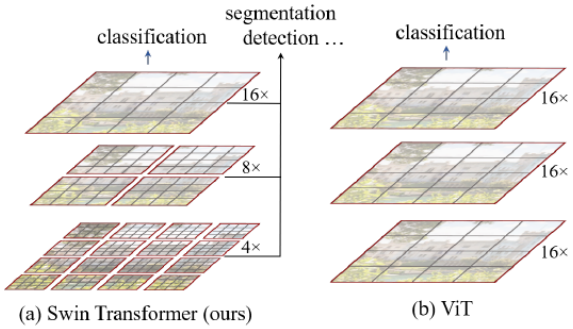

Fig. 11: Hierarchical feature map by merging image parches. B) Feature maps of a single low resolution. Retrieved from [62]

The Swin Transformer is a promising backbone for vision tasks due to its ability to achieve linear computational complexity. This is achieved by calculating self-attention within non-overlapping windows that cover the image. The number of patches in each window remains fixed, making the complexity linear to the image size. This contrasts with previous Transformer-based architectures, [63], that produce single-resolution feature maps with quadratic complexity. The Swin Transformer design includes a key element of shifting the window partition between consecutive self-attention layers (see Fig. 12), which provides connections among them, significantly enhancing modeling power. This approach is also efficient in terms of real-world latency. All query patches within a window share the same key set, facilitating memory access in hardware. This approach has much lower latency than earlier sliding window-based self-attention approaches that suffer from low latency on general hardware due to different key sets for query pixels. Overall, the shifted window approach is similar in modeling power to the sliding window method but with much lower latency.

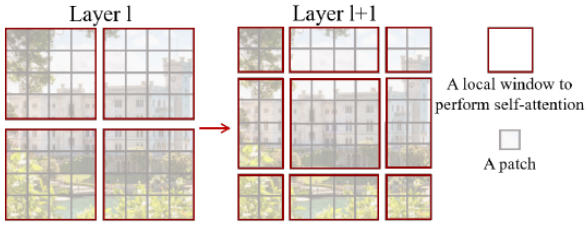

Fig. 12: Illustration of shifted window approach. Retrieved from [62]

Swin Transformer architecture (see Fig. 13.) was observed to achieve the best speed-accuracy trade-off among these image classification methods.

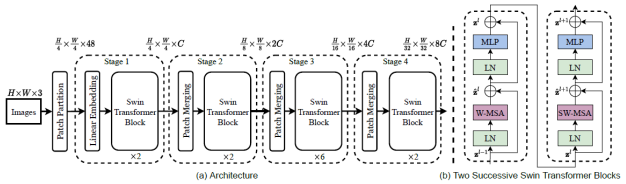

Fig. 13: Architecture of a Swin Transformer and two successive Swin Transformer Blocks. Retrieved from [62]

The Swin Transformer block is a modified version of the Transformer block that replaces the multi-head self-attention (MSA) module with a module based on shifted windows while keeping the other layers the same. The Swin

Transformer block includes a shifted window-based MSA module, a 2-layer MLP with GELU non-linearity in between, and a LayerNorm (LN) layer before each MSA module and each MLP. Additionally, a residual connection is applied after each module. Compared to previous models, the best Swin Transformer model achieves better performance with 58.7 box AP and 51.1 mask AP on COCO test-dev, surpassing the previous best results by +2.7 box AP (Copy-paste, [64], without external data) and +2.6 mask AP (DetectoRS, [65]). A comparison can be also seen in Fig. 14.

(a) Various frameworks

Method	Backbone	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	#param.	FLOPs	FPS
Cascade Mask R-CNN	R-50	46.3	64.3	50.5	82M	739G	18.0
	Swin-T	**50.5**	**69.3**	**54.9**	86M	745G	15.3
ATSS	R-50	43.5	61.9	47.0	32M	205G	28.3
	Swin-T	**47.2**	**66.5**	**51.3**	36M	215G	22.3
RepPointsV2	R-50	46.5	64.6	50.3	42M	274G	13.6
	Swin-T	**50.0**	**68.5**	**54.2**	45M	283G	12.0
Sparse R-CNN	R-50	44.5	63.4	48.2	106M	166G	21.0
	Swin-T	**47.9**	**67.3**	**52.3**	110M	172G	18.4

(b) Various backbones w. Cascade Mask R-CNN

	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	param	FLOPs	FPS
DeiT-S†	48.0	67.2	51.7	41.4	64.2	44.3	80M	889G	10.4
R50	46.3	64.3	50.5	40.1	61.7	43.4	82M	739G	18.0
Swin-T	**50.5**	**69.3**	**54.9**	**43.7**	**66.6**	**47.1**	86M	745G	15.3
X101-32	48.1	66.5	52.4	41.6	63.9	45.2	101M	819G	12.8
Swin-S	**51.8**	**70.4**	**56.3**	**44.7**	**67.9**	**48.5**	107M	838G	12.0
X101-64	48.3	66.4	52.3	41.7	64.0	45.1	140M	972G	10.4
Swin-B	**51.9**	**70.9**	**56.5**	**45.0**	**68.4**	**48.7**	145M	982G	11.6

(c) System-level Comparison

Method	mini-val AP^{box}	mini-val AP^{mask}	test-dev AP^{box}	test-dev AP^{mask}	#param.	FLOPs
RepPointsV2*	-	-	52.1	-	-	-
GCNet*	51.8	44.7	52.3	45.4	-	1041G
RelationNet++*	-	-	52.7	-	-	-
SpineNet-190	52.6	-	52.8	-	164M	1885G
ResNeSt-200*	52.5	-	53.3	47.1	-	-
EfficientDet-D7	54.4	-	55.1	-	77M	410G
DetectoRS*	-	-	55.7	48.5	-	-
YOLOv4 P7*	-	-	55.8	-	-	-
Copy-paste	55.9	47.2	56.0	47.4	185M	1440G
X101-64 (HTC++)	52.3	46.0	-	-	155M	1033G
Swin-B (HTC++)	56.4	49.1	-	-	160M	1043G
Swin-L (HTC++)	57.1	49.5	57.7	50.2	284M	1470G
Swin-L (HTC++)*	**58.0**	**50.4**	**58.7**	**51.1**	284M	-

Fig. 14: Results on COCO object detection and instance segmentation. Retrieved from [62]

B. Two-stages detectors

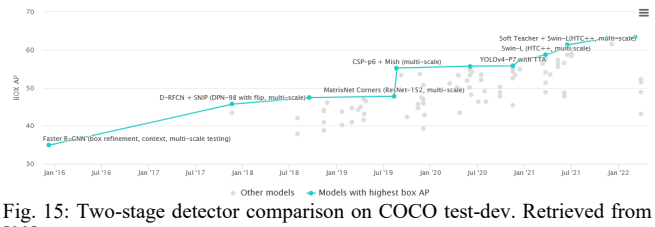

Fig. 15: Two-stage detector comparison on COCO test-dev. Retrieved from [92]

1. R-CNN: Region-based CNN detector

R-CNN (Region-based Convolutional Neural Network) is an object detection algorithm that revolutionized the field by introducing the concept of region proposals. It was proposed by [67]. R-CNN operates in a two-stage framework. In the first stage, it generates a set of region proposals likely to contain objects. These proposals are obtained using a selective search, [68], algorithm, or a similar method. The regions are then warped to a fixed size and fed into a convolutional neural network (CNN) to extract features. In the second stage, each proposal is rescaled to a fixed-size image and fed into a CNN model trained on ImageNet to extract features. The extracted features are classified and refined using support vector machines (SVMs). This step involves training SVMs to classify the proposed regions into different object categories and regress the bounding box coordinates to localize the objects accurately. Finally, linear SVM classifiers are used to predict the presence of an object within each region and to recognize object categories.

The RCNN model has shown a remarkable improvement in performance on VOC07, resulting in a substantial increase in mean Average Precision (mAP) from 33.7% (DPM-v5, [69]) to 58.5%. However, one of its notable disadvantages is that it requires redundant feature computations on numerous overlapped proposals (over 2000 boxes from one image), which leads to a slower detection speed (14s per image with GPU).

2. Fast R-CNN: faster version of R-CNN

Back in 2015, [71], introduced the Fast RCNN detector, which was an improvement of R-CNN, [70]. This detector allowed for training both a detector and a bounding box regressor under the same network configurations. On the VOC07 dataset, Fast RCNN improved the mAP from 58.5% to 70.0% and was over 200 times faster in detection speed compared to R-CNN

3. Faster R-CNN:

Faster R-CNN is an object detection algorithm proposed by [72], which builds upon the R-CNN and Fast R-CNN frameworks. It addresses the speed bottleneck of the original R-CNN by introducing a region proposal network (RPN) that shares convolutional features with the object detection network. In Faster R-CNN, the RPN generates region proposals by sliding a small network over the convolutional feature map and predicting objectness scores and bounding box offsets. These region proposals are then used for subsequent object detection. The RPN is trained end-to-end with the rest of the network, allowing it to learn to propose regions likely to contain objects. The proposed regions are then classified and refined using RoI (Region of Interest) pooling and fully connected layers. Authors in [72], present an RPN, a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. The RPN is trained end-to-end to generate high-quality region proposals, which Fast R-CNN uses for detection. Further merge of RPN and Fast R-CNN into a single network by sharing their convolutional features tells the unified network where to look.

The Faster RCNN is a deep learning detector that operates in almost real-time and is the first of its kind to be end-to-end. It has an accuracy of 42.7% in COCO, 21.9% in COCO mAP@[.5,.95], 73.2% in VOC07 mAP, and 70.4% in VOC12 mAP. Additionally, it can process up to 17 frames per second with ZFNet. The Region Proposal Network (RPN) is the primary feature of Faster-RCNN, allowing for cost-effective region proposals. This system integrates most of the individual blocks found in an object detection system,

such as proposal detection, feature extraction, and bounding box regression, into a unified, end-to-end learning framework. Although Faster RCNN surpasses Fast RCNN in terms of speed, computation redundancy remains present in the subsequent detection stage.

4. D-RFCN +SNIP

During the evaluation of the performance on different network architectures for classifying small objects on ImageNet, it was shown that CNNs are not robust to changes in scale. Based on this analysis, it was proposed in [73], to train and test detectors on the same scales as an image pyramid. Small and large objects are challenging for precise recognition at smaller and larger scales. The presented study in [73], proposed a novel training scheme called Scale Normalization for Image Pyramids (SNIP), which selectively back-propagates the gradients of object instances of different sizes as a function of the image scale.

On the COCO dataset, the single model performance is 45.7%, and an ensemble of 3 networks obtains an mAP of 48.3%. Off-the-shelf ImageNet-1000 pre-trained models were used and only trained with bounding box supervision. Contrary to the fact that deep neural networks can learn to cope with significant variations in scale given enough training data, it was shown that SNIP offers significant improvements (3.5%) over traditional object detection training paradigms. The presented ensemble with a Deformable-RFCN backbone obtained an mAP of 69.7% at 50% overlap, an improvement of 7.4% over the state-of-the-art on the COCO dataset.

Learning scale-invariant representations is crucial for accurately recognizing and localizing objects in object detection, pose estimation, and instance segmentation tasks. Numerous approaches have been suggested and explored when detecting objects at various scales. The deeper layers of modern CNNs have enormous strides (32 pixels) that lead to a very coarse representation of the input image, which makes small object detection very challenging. To tackle this issue, contemporary object detectors utilize dilated convolutions, which effectively enhance the feature map's resolution. For reference see Fig. 16.

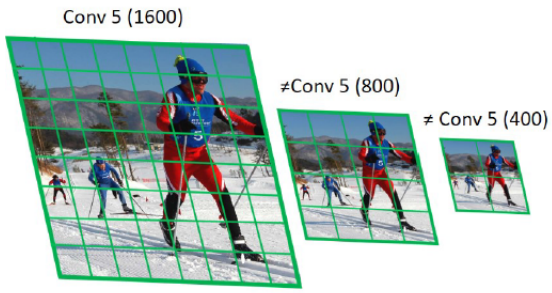

Fig. 16: The same layer convolutional features at different scales of the image are different and map to different semantic regions in the image at different scales. Retrieved from [73]

As also stated in [73], FPN, Mask-RCNN, RetinaNet, and other methods using a pyramidal representation and combining features of shallow layers with deeper layers have access to higher-level semantic information. Networks are pre-trained on images of standard resolution of 224x224. Therefore, the high-level semantic features (at conv5) generated even by feature pyramid networks will not help classify small objects (a similar argument can be made for large objects in high-resolution images).

5. YOLOv4-p7 with TTA

The results of test-time augmentation (TTA) experiments of YOLOv4-large models presented in [37], are shown below in Fig. 17. YOLOv4-P5, YOLOv4-P6, and YOLOv4-P7 get 1.1%, 0.7%, and 0.5% higher AP after TTA is applied.

Model	AP	AP_{50}	AP_{75}
YOLOv4-P5 with TTA	52.9%	70.7%	58.3%
YOLOv4-P6 with TTA	55.2%	72.9%	60.5%
YOLOv4-P7 with TTA	56.0%	73.3%	61.4%

Fig. 17: Results of YOLOv4 with Test-Time Augmentation. Retrieved from [37]

6. Soft teacher + Swin-L

Soft teaching is a technique that has been utilized in various deep-learning tasks, including object detection. It involves using the knowledge distilled from a pre-trained teacher model to guide the training process of a student model. By incorporating the knowledge of the teacher model, such as its predictions or intermediate feature representations, the student model can benefit from the teacher's expertise and achieve better performance. In the paper by [74], they propose a framework that gradually improves the quality of pseudo-labels during training. To achieve this, they introduce two techniques: a soft teacher mechanism that weighs the classification loss of each unlabelled bounding box based on the classification score produced by the teacher network and a box jittering approach that selects reliable pseudo boxes for the learning of box regression on the COCO benchmark. This innovation outperforms previous methods by a large margin, even at low labeling ratios of 1%, 5%, and 10%. By using the 123K unlabelled images of COCO, they were able to improve a 40.9 mAP baseline detector trained using the full COCO training set by +3.6 mAP, reaching 44.5 mAP. On the Swin Transformer-based object detector (58.9 mAP on test-dev), they achieved a significant improvement in detection accuracy by +1.5 mAP, reaching 60.4 mAP, and an improvement in instance segmentation accuracy by +1.2 mAP, reaching 52.4 mAP. By incorporating the Object365 pre-trained model, they were able to further improve the detection accuracy to 61.3 mAP and the instance segmentation accuracy to 53.0 mAP. A comprehensive comparison of the proportion of used labeled data is graphically represented in Fig. 18.

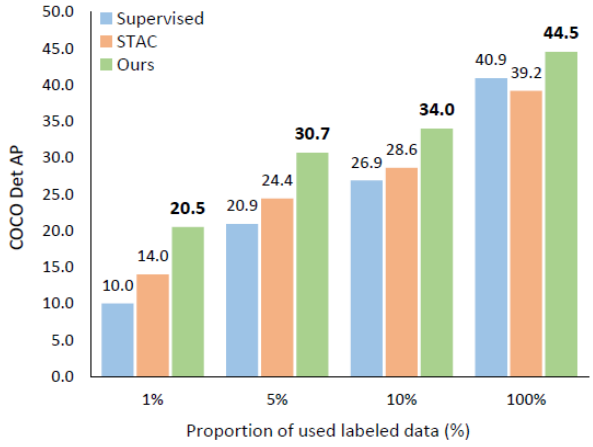

Fig. 18: End-to-end pseudo-label based semi-supervised object detection method. Retrieved from [74]

Nevertheless, acquiring labels can pose a challenge as it involves a time-consuming and costly annotation process. As a result, learning approaches have emerged that capitalize on unlabelled data while training deep neural models. Techniques such as self-supervised and semi-supervised learning have gained traction as they offer avenues to leverage unlabelled data and mitigate the label acquisition bottleneck. Some methods, as described in references, [75], [76], use a multi-stage training process. In this process, the first stage trains a detector using labeled data. After that, an unannotated data pseudo-labeling process takes place, followed by a re-training step using the pseudo-labeled data. Although these approaches can achieve decent accuracy, the quality of the pseudo labels generated by the initial detector, which is trained using a small amount of labeled data and may not be accurate, limits the final performance.

An additional advantage of this end-to-end framework is its ability to leverage the teacher model more effectively in guiding the training of the student model. This contrasts approaches, [75], [76], that solely provide "pseudo boxes with hard category labels" for the student model. To address this, a soft teacher approach is proposed. In this approach, the teacher model directly evaluates all the box candidates generated by the student model rather than relying on "pseudo boxes" to assign category labels and regression vectors to these student-generated boxes. This direct assessment of the box candidates allows for the utilization of more comprehensive supervision information during the training of the student model. To address this issue, authors in [74], proposed using reliability measures to weight the loss of each "background" box candidate. This approach measure performs significantly better than previous complex foreground/background assignment methods. From here on, this is called a "soft teacher."

An alternative method involves choosing reliable bounding boxes for teaching the student's localization branch, using a technique called box jittering. This approach involves jittering a pseudo-foreground box candidate multiple times, then regressing these jittered boxes based on the teacher model's location branch. The variance of these regressed boxes serves as a measure of reliability, and the box candidate with high reliability is used to train the student's localization branch.

In addition, on a state-of-the-art Swin-Transformer-based detector, [77], which obtains 58.9 mAP for object detection and 51.2 mAP for instance, segmentation on COCO test-dev2017, this proposed approach can boost accuracy by 1.5 mAP and 1.2 mAP respectively, resulting in 60.4 mAP and 52.4 mAP. Furthermore, by incorporating the Object365, [78], pre-trained model, the detection accuracy can achieve 61.3 mAP, and the instance segmentation accuracy can reach 53.0 mAP, setting a new benchmark for this evaluation.

This approach achieved simultaneous improvements in the detector and pseudo labels by involving a student model for detection training and a continuously updated teacher model through the exponential moving average strategy. Two straightforward techniques, "soft teacher" and "box jittering," were introduced within the end-to-end training process to enhance the effective utilization of the teacher model. The soft teacher method enabled efficient leverage of the teacher model's knowledge, while box jittering improved training efficiency. Together, these techniques facilitated the overall effectiveness of the approach.

7. DINO

In various object detection methods, anchor boxes are predefined bounding boxes that are placed at various positions and scales across an image. These anchor boxes serve as reference points for detecting and localizing objects of interest. Inaccurate anchor boxes can lead to suboptimal detection performance. DINO [79] has addressed this issue by introducing a denoising mechanism for anchor boxes. It involves a neural network to predict more accurate and adaptive anchor boxes directly from the input image instead of relying on manually defined anchor boxes. By dynamically adjusting the anchor boxes during training, DINO enhances the model's ability to detect objects with different scales and aspect ratios. Combining the DETR framework and improved denoising anchor boxes in DINO enables end-to-end object detection with enhanced accuracy and adaptability. This approach eliminates manual anchor box tuning, leading to more efficient and effective object detection models. DINO achieves 48.3AP in 12 epochs and 51.0AP in 36 epochs on COCO with a ResNet-50 backbone and multi-scale features, yielding a significant improvement of +4.9AP and +2.4AP, respectively, compared to DN-DETR, the previous best DETR-like model. DINO scales well in both model size and data size. Without bells and whistles, after pre-training on the Objects365 dataset with a SwinL backbone, DINO obtains the best results on COCO val2017 (63.2AP) and test-dev (63.3AP). DINO significantly reduces its model size and pre-training data size while achieving better results than other models on the leaderboard.

Compared to other models on the leaderboard, [80], the model size is 1/15 compared to SwinV2-G. Compared to Florence, the pre-training detection dataset was reduced to 1/5 and the backbone pre-training dataset to 1/60 while achieving better results. Authors in [79], have designed a new end-to-end DETR-like object detector with several novel techniques, including contrastive DN training, and mixed query selection, and look forward twice for different parts of the DINO model. They have validated the effectiveness of different design choices in DINO. As a result, DINO achieves 48.3AP in 12 epochs and 51.0AP in 36 epochs with ResNet-50 and multi-scale features, significantly outperforming the previous best DETR-like models. In particular, DINO trained in 12 epochs shows a more significant improvement on small objects, yielding an improvement of +7.4AP.

As shown in [79], DINO achieves the best results of 63.2AP and 63.3AP on COCO val2017 and test-dev, demonstrating its robust scalability to larger models and data sizes. Note that all the previous SOTA models do not use Transformer decoder-based detection heads (HTC++, [81], and DyHead, [82]). It is the first time an end-to-end Transformer detector has been established as a SOTA model on the leaderboard, [80]. Compared with the previous SOTA models, a much smaller model size was used (1/15 parameters compared with SwinV2-G, [83]), backbone pre-training data size (1/60 images compared with Florence), and detection pre-training data size (1/5 images compared with Florence), while achieving better results. In addition, our reported performance without test time augmentation (TTA) is a neat result without bells and whistles. These results effectively

show the superior detection performance of DINO compared with traditional detectors.

IV. Conclusion and Latest Evolution

Object detection methods and algorithms are a constantly evolving field of study that generates new insights. Notable publications, such as [93], [94], provide compelling evidence of ongoing advancements and breakthroughs. These algorithms are increasingly practical in everyday life, with improvements in accuracy, efficiency, and applicability in various real-world scenarios. Recent research developments have introduced innovative versions of YOLO, including YOLO v8 and YOLO-NAS. These latest iterations incorporate novel techniques and enhancements to improve object detection performance, enabling more precise and efficient identification of objects in images or videos. The introduction of YOLO v8 and YOLO-NAS demonstrates the ongoing commitment to advancing object detection capabilities, with researchers and practitioners exploring new avenues and pushing the boundaries of what is achievable. These latest versions showcase the rapid progress and the continuous quest for innovation in the field of object detection.

A. YOLOv8

Ultralytics has released in 2023 YOLOv8 as the highest-performing YOLO model ever released, (see Fig. 19 and Fig. 20), setting new standards in SOTA real-time detection and segmentation, opening up a new world of use cases for simple and effective AI solutions, [84]

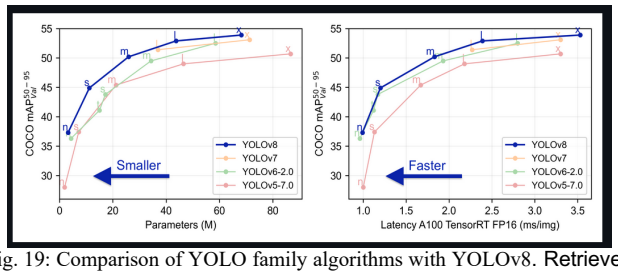

Fig. 19: Comparison of YOLO family algorithms with YOLOv8. Retrieved from [84]

The YOLO family's latest version features enhanced capabilities such as instance segmentation, pose/key points estimation, and classification. It uses anchor-free detection and new convolutional layers to make predictions more accurate.

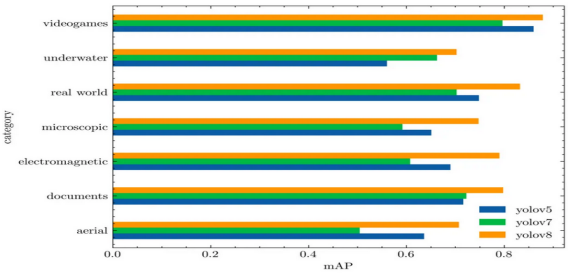

Fig. 20: YOLOv8 on RF100 comparison [90]

In addition, YOLOv8 uses mosaic augmentation during training, [85], and Fig. 21, however, because it has been found that this augmentation can be detrimental if used during the whole training process, it is disabled for the last ten epochs. YOLOv8 provided five scaled versions: YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large) and YOLOv8x (extra large). Evaluated on MS COCO dataset test-dev 2017, YOLOv8x achieved an AP of 53.9% with an image size of 640 pixels (compared to 50.7% of YOLOv5 on the same input size) with a speed of 280 FPS on an NVIDIA A100 and TensorRT, [86].

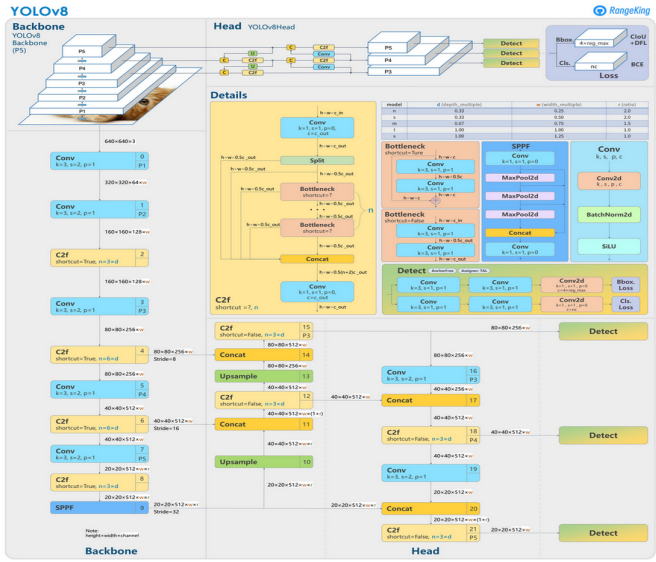

Fig. 21: YOLOv8 Architecture. Retrieved from [91]

Deep learning research often strongly emphasizes model architecture, but the training routine plays a crucial role in the success of models like YOLOv5 and YOLOv8. YOLOv8, for instance, incorporates online image augmentation during training to introduce variations in the images seen by the model at each epoch. One unique augmentation technique is mosaic augmentation, which involves merging four images. This challenges the model to recognize objects in new positions, amidst occlusion, and against different backgrounds.

However, it has been observed that this augmentation can negatively impact performance if used throughout the entire training process. Therefore, deactivating it for the last ten training epochs is beneficial. This adjustment reflects the careful attention to refining the YOLO models over time, evident in the YOLOv5 repository and YOLOv8 research.

B. YOLO NAS

YOLO-NAS, short for You Only Look Once Neural Architecture Search, is an extension of the widely used YOLO (You Only Look Once) object detection framework incorporating neural architecture search (NAS) methods. NAS aims to automate designing and optimizing neural network architectures to improve their performance. In the case of YOLO-NAS, NAS techniques are explicitly applied to the YOLO architecture to discover more effective network designs for object detection tasks. By conducting an automated search, YOLO-NAS explores various architectural configurations, such as different layer types, sizes, and connectivity patterns. The aim is to identify optimal network structures that maximize detection accuracy while minimizing computational complexity. By

leveraging NAS, YOLO-NAS aims to enhance the performance and efficiency of YOLO models by discovering novel architectural variations that may surpass manually designed configurations. This approach enables the creation of customized and optimized network architectures specifically tailored for object detection tasks. YOLO-NAS exemplifies the integration of NAS techniques within the YOLO framework, advancing object detection capabilities. The new YOLO-NAS delivers state-of-the-art (SOTA) performance with unparalleled accuracy-speed performance, outperforming other models such as YOLOv5, YOLOv6, YOLOv7, and YOLOv8. A comprehensive comparison can be seen in Fig. 22.

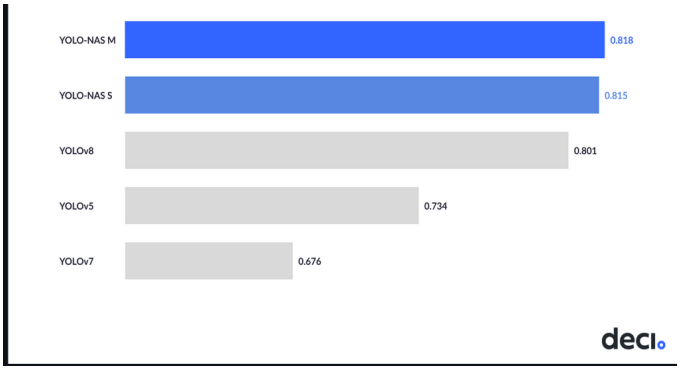

Fig. 22: Average mAP on Roboflow-100 for YOLO-NAS vs other models. Retrieved from [87].

According to Deci, YOLO-NAS is around 0.5 mAP points more accurate and 10–20% faster than equivalent variants of YOLOv8 and YOLOv7, [87].

ACKNOWLEDGMENT

This article was supported thanks to the generous support under the Operational Program Integrated Infrastructure for the project: "Research Center for Data Analysis and Protection - II. stage," Project no. 313021W479, co-financed by the European Regional Development Fund ".

REFERENCES

[1] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, vol. 1. IEEE,2001, pp. I–I.

[2] P. Viola and M. J. Jones, “Robust real-time face detection,” International journal of computer vision, vol. 57, no. 2, pp. 137–154, 2004.

[3] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 1. IEEE, 2005, pp. 886–893.

[4] P. Felzenszwalb, D. McAllester, and D. Ramanan, “A discriminatively trained, multiscale, deformable part model,” in Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008, pp. 1–8.

[5] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester, “Cascade object detection with deformable part models,” in Computer vision and pattern recognition (CVPR), 2010, IEEE conference on. IEEE, 2010, pp. 2241–2248.

[6] T. Malisiewicz, A. Gupta, and A. A. Efros, “Ensemble of exemplar-svms for object detection and beyond,” in Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, 2011, pp. 89–96.

[7] P. Felzenszwalb, D. McAllester, and D. Ramanan, “A discriminatively trained, multiscale, deformable part model,” in Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008, pp. 1–8.

[8] Jiang Zhao, Shilong Ji, Zhihao Cai, Yiwen Zeng and Yingxun Wang. Moving Object Detection and Tracking by Event Frame from Neuromorphic Vision Sensors, 2022, Biomimetics, 7, 31.

[9] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Advances in Neural Information Processing Systems 28 (NIPS 2015), C. Cortes and N. Lawrence and D. Lee and M. Sugiyama and R. Garnett. ISBN: 9781510825024

[10] Lohia, Aditya; Kadam, Kalyani Dhananjay; Joshi, Rahul Raghvendra; and Bongale, Dr. Anupkumar M., "Bibliometric Analysis of One-stage and Two-stage Object Detection" (2021). Library Philosophy and Practice (e-journal). 4910.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.

[12] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. and Berg, A.C. (2016) SSD: Single Shot Multibox Detector. European Conference on Computer Vision, Amsterdam, 8-16 October 2016, 21-37.

[13] Lin, T.-Y., Goyal, P., Girshick, R., He, K. and Dollár, P. (2017) Focal Loss for Dense Object Detection. Proceedings of the IEEE International Conference on Computer Vision, Venice, 22-29 October 2017, 2999-3007.

[14] He, K., Zhang, X., Ren, S. and Sun, J. (2016) Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, 27-30 June 2016, 770-778.

[15] M. Tan, R. Pang and Q. Le, "EfficientDet: Scalable and Efficient Object Detection," in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020 pp. 10778-10787.

[16] Saining Xie, Ross Girshick, Piotr Doll´ar, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. CVPR, pages 5987–5995, 2017

[17] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. AAAI, 2019.

[18] Golnaz Ghiasi, Tsung-Yi Lin, Ruoming Pang, and Quoc V. Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. CVPR, 2019.

[19] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. ICML, 2019.

[20] Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin D. Cubuk, and Quoc V. Le. 2020. Rethinking pre-training and self-training. In Proceedings of the 34th International Conference on

Neural Information Processing Systems (NIPS'20). Curran Associates Inc., Red Hook, NY, USA, Article 323, 3833–3845.

[21] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014.

[22] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In ICML,2014.

[23] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015.

[24] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille.Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. PAMI, 2017.

[25] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In ICCV, 2019.

[26] Zhiqiang Shen, Zhuang Liu, Jianguo Li, Yu-Gang Jiang, Yurong Chen, and Xiangyang Xue. Object detection from scratch with deep supervision. IEEE transactions on pattern analysis and machine intelligence, 42(2):398–412, 2019.

[27] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. In Advances in Neural Information Processing Systems, 2018.

[28] I Zeki Yalniz, Hervé Jégou, Kan Chen, Manohar Paluri, and Dhruv Mahajan. Billion-scale semi-supervised learning for image classification. arXiv preprint arXiv:1905.00546, 2019.

[29] Qizhe Xie, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Self-training with noisy student improves imagenet classification. In CVPR, 2020.

[30] Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. Revisiting self-training for neural sequence generation. In ICLR, 2020.

[31] Jacob Kahn, Ann Lee, and Awni Hannun. Self-training for end-to-end speech recognition. In ICASSP, 2019.

[32] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In ICCV, 2017.

[33] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. AutoAugment: Learning augmentation policies from data. In CVPR, 2019.

[34] Barret Zoph, Ekin D. Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V. Le. Learning data augmentation strategies for object detection. In ECCV, 2020.

[35] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. RandAugment: Practical automated data augmentation with a reduced search space. In Advances in Neural Information Processing Systems, 2020.

[36] Mingxing Tan and Quoc V Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In ICML, 2019.

[37] C. Wang, A. Bochkovskiy and H. Liao, "Scaled-YOLOv4: Scaling Cross Stage Partial Network," in 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 2021 pp. 13024-13033.

[38] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qing-ming Huang, and Qi Tian. CenterNet: Keypoint triplets for object detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 6569–6578, 2019.

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016

[39] Hei Law and Jia Deng. CornerNet: Detecting objects as paired keypoints. In Proceedings of the European Conference on Computer Vision (ECCV), pages 734–750, 2018.

[40] Hei Law, Yun Teng, Olga Russakovsky, and Jia Deng. CornerNet-Lite: Efficient keypoint based object detection. arXiv preprint arXiv:1904.08900, 2019

[41] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Doll´ar. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 2980–2988, 2017

[42] Xiang Long, Kaipeng Deng, Guanzhong Wang, Yang Zhang, Qingqing Dang, Yuan Gao, Hui Shen, Jianguo Ren, Shumin Han, Errui Ding, et al. PP-YOLO: An effective and efficient implementation of object detector. arXiv preprint arXiv:2007.12099, 2020.

[43] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767, 2018

[44] Mingxing Tan, Ruoming Pang, and Quoc V Le. EfficientDet:Scalable and efficient object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[45] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: Fully convolutional one-stage object detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 9627–9636, 2019

[46] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. In arXiv preprint arXiv:1904.07850, 2019.

[47] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934, 2020

[48] Ghiasi, Golnaz & Cui, Yin & Srinivas, Aravind & Qian, Rui & Lin, Tsung-Yi & Cubuk, Ekin & Le, Quoc & Zoph, Barret. (2021). Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation. 2917-2927. 10.1109/CVPR46437.2021.00294.

[49] Xianzhi Du, Tsung-Yi Lin, Pengchong Jin, Golnaz Ghiasi,Mingxing Tan, Yin Cui, Quoc V Le, and Xiaodan Song. Spinenet: Learning scale-permuted backbone for recognition and localization. In CVPR, 2020

[50] Hao-Shu Fang, Jianhua Sun, Runzhong Wang, Minghao Gou, Yong-Lu Li, and Cewu Lu. Instaboost: Boosting instance segmentation via probability map guided copy-pasting. In ICCV, 2019

[51] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In ICML, 2019

[52] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In CVPR, 2019

[53] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In ICCV, 2019.
[54] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In ICCV, 2017
[55] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. arXiv preprint arXiv:2006.02334, 2020
[56] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In CVPR, 2020
[57] Pathak AR, Pandey M, Rautaray S: Application of deep learning for object detection. Proc Comput Sci 132:1706–1717, 2018
[58] Mohamed C, Petr D, Dominik S, Zdenek N: New end-to-end strategy based on DeepLabv3+ semantic segmentation for human head detection. Sensors 21:5848, 2021
[59] M. Letavay, M. Bažant, P. Tuček and M. Prokša, "SWOT Analysis for Object Detection in Traffic Engineering Based on YOLO Implementation – Case Study," 2022 14th International Conference on Advanced Semiconductor Devices and Microsystems (ASDAM), Smolenice, Slovakia, pp. 1-4, 2022
[60] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using drop-connect. In ICML, 2013
[61] Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin D. Cubuk, and Quoc V. Le. Rethinking pre-training and self-training. In NeurIPS, 2020
[62] Z. Liu, et al., "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," in 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 2021 pp. 9992-10002.
[63] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In International Conference on Learning Representations, 2021
[64] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung- Yi Lin, Ekin D Cubuk, Quoc V Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. arXiv preprint arXiv:2012.07177, 2020
[65] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. arXiv preprint arXiv:2006.02334, 2020
[66] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems, pages 5998–6008, 2017
[67] Girshick, Ross & Donahue, Jeff & Darrell, Trevor & Malik, Jitendra. (2013). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition.
[68] K. E. Van de Sande, J. R. Uijlings, T. Gevers, and A. W. Smeulders, "Segmentation as selective search for object recognition," in Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, 2011, pp. 1879–1886.
[69] P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan: Object Detection with Discriminatively Trained Part Based Models, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 32, No. 9, 2010
[70] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580– 587.
[71] R. Girshick, "Fast r-cnn," in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1440–1448.
[72] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: towards real-time object detection with region proposal networks. In Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'15). MIT Press, Cambridge, MA, USA, 91–99.
[73] B. Singh and L. Davis, "An Analysis of Scale Invariance in Object Detection - SNIP," in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 2018 pp. 3578-3587.
[74] M. Xu, et al., "End-to-End Semi-Supervised Object Detection with Soft Teacher," in 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 2021 pp. 3040-3049.
[75] Sohn, Kihyuk et al. "A Simple Semi-Supervised Learning Framework for Object Detection." ArXiv abs/2005.04757 (2020): n. pag.
[76] Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin D Cubuk, and Quoc V Le. Rethinking pre-training and self-training. NIPS, 2020.
[77] Z. Liu *et al*., "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Montreal, QC, Canada, 2021, pp. 9992-10002, doi: 10.1109/ICCV48922.2021.00986.
[78] A. Shrivastava, A. Gupta, and R. Girshick, "Training Region-Based Object Detectors with Online Hard Example Mining," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2016
[79] Li, Feng & Liu, Shilong & Zhang, Lei & Su, Hang & Zhu, Jun & Ni, Lionel & Shum, Heung-Yeung. (2022). DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection.
[80] R. Stojnic, R. Taylor, M. Kardas, et al., (2023), *Papers with Code - coco test-dev benchmark (object detection)*. URL https://paperswithcode.com/sota/object-detection-on-coco (accessed date February 2023).
[81] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4974–4983, 2019.
[82] Xiyang Dai, Yinpeng Chen, Bin Xiao, Dongdong Chen, Mengchen Liu, Lu Yuan, and Lei Zhang. Dynamic head: Unifying object detection heads with

attentions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7373–7382, 2021.
[83] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. arXiv preprint arXiv:2111.09883, 2021.
[84] G. Jocher, (2023), *Introducing YOLOv8: Faster, Simpler, More Accurate,* URL https://ultralytics.com, (accessed date February 2023).
[85] Terven, J. and Cordova-Esparza, D., "A Comprehensive Review of YOLO: From YOLOv1 and Beyond", arXiv e-prints, 2023. doi:10.48550/arXiv.2304.00501.
[86] G. Jocher, A. Chaurasia, and J. Qiu, *YOLO by Ultralytics.*, URL https://github.com/ultralytics/ultralytics, 2023. Accessed date February 2023.
[87] R. El-Yaniv, J. Elial, Z. Geifman, et al., (2023), *The Deep Learning Development Platform*, URL https://deci.ai/blog/yolo-nas-foundation-model-object-detection/, (accessed date February 2023)
[88] R. Stojnic, R. Taylor, M. Kardas, et al., (2023), *Papers with Code - Object Detection on COCO test-dev,* URL https://paperswithcode.com/sota/object-detection-on-coco (accessed date February 2023).
[89] R. Stojnic, R. Taylor, M. Kardas, et al., (2023), *Papers with Code - Object Detection on COCO test-dev Single Scale,* URL https://paperswithcode.com/sota/object-detection-on-coco?tag_filter=12 (accessed date February 2023).
[90] Ch. Bhalerao, (2023). *Medium: YOLO v8! The real state-of-the-art? URL* https://medium.com/mlearning-ai/yolo-v8-the-real-state-of-the-art-eda6c86a1b90, (accessed date February 2023).
[91] J. Solawetz, Francesco, (2023), *Roboflow: What is YOLOv8,* URL https://blog.roboflow.com/whats-new-in-yolov8/, (accessed date February 2023).
[92] R. Stojnic, R. Taylor, M. Kardas, et al., (2023), *Papers with Code - Object Detection on COCO test-dev Multi Scale,* URL https://paperswithcode.com/sota/object-detection-on-coco?tag_filter=10 (accessed date February 2023).
[93] Oungsub Kim, Yohan Han, Jongpil Jeong, "Real-time Inspection System Based on Moire Pattern and YOLOv7 for Coated High-reflective Injection Molding Product," WSEAS Transactions on Computer Research, vol. 10, pp. 120-125, 2022.
[94] Doohwan Kim, Yo-Han Han, Jongpil Jeong, "Design and Implementation of Real-time Anomaly Detection System based on YOLOv4," WSEAS Transactions on Electronics, vol. 13, pp. 130-136, 2022.