WHAT IS YOLOV5: A DEEP LOOK INTO THE INTERNAL FEATURES OF THE POPULAR OBJECT DETECTOR

Rahima Khanam* **and Muhammad Hussain**

Department of Computer Science, Huddersfield University, Queensgate, Huddersfield HD1 3DH, UK;
*Correspondence: rahima.khanam@hud.ac.uk;

July 31, 2024

ABSTRACT

This study presents a comprehensive analysis of the YOLOv5 object detection model, examining its architecture, training methodologies, and performance. Key components, including the Cross Stage Partial backbone and Path Aggregation-Network, are explored in detail. The paper reviews the model's performance across various metrics and hardware platforms. Additionally, the study discusses the transition from Darknet to PyTorch and its impact on model development. Overall, this research provides insights into YOLOv5's capabilities and its position within the broader landscape of object detection and why it is a popular choice for constrained edge deployment scenarios.

Keywords Automation; Computer Vision; YOLO; YOLOV5; Object Detection; Real-Time Image processing; Convolutional Neural Networks(CNN); YOLO version comparison

1 Introduction

Computer vision is a rapidly evolving field that empowers machines [1] to interpret and comprehend visual information [2]. A critical component of this discipline is object detection [3], which involves accurately identifying and locating objects within images or video sequences [4]. To address this challenge, various algorithmic approaches have been developed, with significant advancements achieved in recent years[5].

One such groundbreaking technique is the You Only Look Once (YOLO) algorithm, introduced by Redmon et al. in 2015 [6]. This name reflects its distinctive approach, examining the entire image just once to identify objects and their positions. In contrast to conventional methods employing two-stage detection processes, YOLO treats object detection as a regression problem [6]. In the YOLO paradigm, a single convolutional neural network is employed to predict bounding boxes and class probabilities for an entire image [7]. This streamlined approach differs from traditional methods with more intricate pipelines.

Building upon the foundation laid by YOLO, this paper delves into the YOLOv5 architecture, a state-of-the-art object detection model that has garnered significant attention due to its exceptional performance and efficiency.

1.1 Survey Objective

This study aims to evaluate the performance of the YOLOv5 object detection model in comparison to state-of-the-art alternatives. The research focuses on assessing the trade-off between model accuracy and inference speed across the various YOLOv5 variants (n, s, m, l, x). By examining these different model sizes, this study seeks to identify the optimal balance between performance metrics for diverse application requirements.

The investigation will explore the factors contributing to YOLOv5's performance gains, with particular emphasis on:

1. The role of PyTorch training methodologies
2. The impact of architectural innovations such as the CSP backbone and PA-Net neck

3. The effectiveness of data augmentation techniques, including mosaic augmentation
4. The influence of loss calculation methods and bounding box anchor generation

Additionally, the study will analyze the implications of YOLOv5's transition from the Darknet framework to PyTorch, considering aspects such as model development, deployment, and accessibility for custom object detection tasks.

By providing a comprehensive examination of YOLOv5's capabilities and innovations, this research aims to contribute to the broader understanding of advanced object detection algorithms and their practical applications in computer vision.

2 Evolution of Yolov5

The YOLOv5 [8] repository emerged as an evolution of the YOLOv3 [9] PyTorch implementation developed by Glenn Jocher in 2020. It was released following the release of YOLOv4 [10]. Yolov5 gained popularity as a platform for transitioning YOLOv3 models from Darknet to PyTorch for production deployment. Recognizing the utility of this PyTorch-based approach, Ultralytics initiated a development trajectory focused on enhancing the YOLOv3 architecture and training methodologies within the PyTorch framework. This endeavor aimed to empower a broader community of developers to create and deploy custom object detectors.

This is YOLOv5 development timeline:

- **April 1, 2020:** Initiation of development on a series of compound-scaled PyTorch models based on YOLOv3/YOLOv4 architectures.
- **May 27, 2020:** Public release of the YOLOv5 repository, demonstrating state-of-the-art performance among existing YOLO implementations.
- **June 9, 2020:** Integration of CSP (Cross-Stage Partial) modules, contributing to improved model speed, size, and accuracy.
- **June 19, 2020:** Adoption of FP16 precision as default, resulting in smaller checkpoints and faster inference.
- **June 22, 2020:** Implementation of PANet updates, including new detection heads, reduced parameters, and enhanced mAP (mean Average Precision).

Initially, the model incorporating these advancements was designated YOLOv4, aligning with the contemporaneous release of the YOLOv4 architecture within the Darknet framework. However, to avoid potential confusion and inconsistencies, it was subsequently revised to YOLOv5. While this change sparked some debate within the research community, a comparative analysis of YOLOv4 and YOLOv5 was conducted to facilitate objective evaluation [11].

This study primarily focuses on the novel techniques and performance metrics introduced within the YOLOv5 framework, as detailed within the GitHub repository. It is essential to acknowledge the rapid evolution of the YOLOv5 model since its inception, which suggests ongoing research and development in this domain. As such, YOLOv5 should be considered a dynamic and evolving system rather than a static model.

3 Architectural footprint of Yolov5

Object detection, a primary application of YOLOv5, entails the extraction of salient features from input images. These features are subsequently processed by a predictive model to localize and classify objects within the image.

The YOLO architecture introduced the end-to-end, differentiable approach to object detection by unifying the tasks of bounding box regression and object classification into a single neural network [12]. Fundamentally, the YOLO network comprises three core components. The **backbone**, a convolutional neural network, is responsible for encoding image information into feature maps at varying scales. These feature maps are then processed by the **neck**, a series of layers designed to integrate and refine feature representations. Finally, the **head** module generates predictions for object bounding boxes and class labels based on the processed features.

While there exists considerable flexibility in the architectural design of each component, YOLOv4 and YOLOv5 have significantly advanced the field by incorporating techniques from other computer vision domains. This synergistic approach has demonstrated substantial improvements in object detection performance within the YOLO framework.

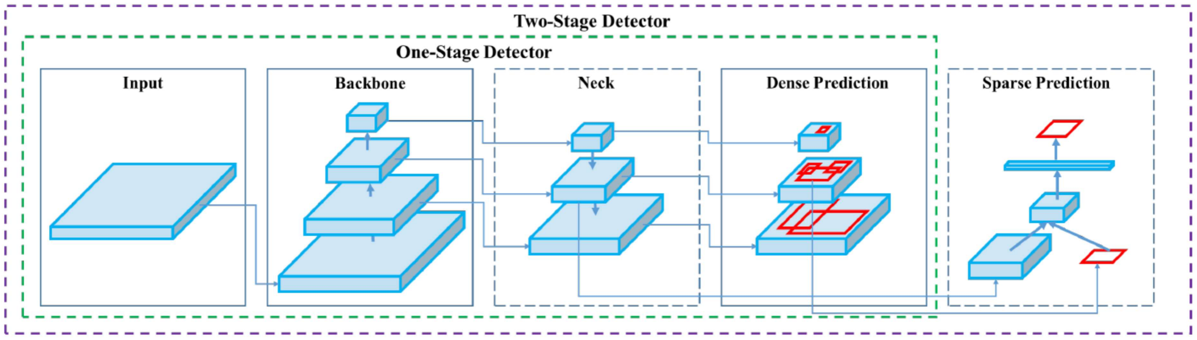

Figure 1: Process of Object Detection [13]

3.1 YOLOv5 Training Methods

The efficacy of an object detection system is contingent not only on its underlying architecture but also on the employed training methodologies. While architectural innovations often garner significant attention, the role of training procedures in achieving optimal performance is equally critical. There are two primary training techniques employed in YOLOv5:

- **Data augmentation** is a pivotal component of the YOLOv5 training pipeline. By introducing diverse transformations to the training dataset, this technique enhances model robustness and generalization capabilities. Consequently, the model becomes more resilient to variations in real-world image conditions.
- The **Loss function** is a composite metric calculated from three primary components: Generalized Intersection over Union (GIoU), objectness, and classification loss. These loss components are carefully designed to optimize mean average precision (mAP), a widely adopted evaluation metric for object detection models.

3.2 Transition to PyTorch

YOLOv5 represents a significant advancement by transitioning the YOLO architecture from the Darknet framework to PyTorch [14]. The Darknet framework, primarily implemented in C, affords researchers granular control over network operations. While this level of control is advantageous for experimentation, it often hinders the rapid integration of novel research findings due to the necessity of custom gradient calculations for each new implementation.

Porting the training procedures of Darknet to PyTorch, as achieved in YOLOv3, is a complex undertaking in itself. YOLOv5 extends this effort by further optimizing and refining these procedures within the PyTorch ecosystem.

3.3 Data augumentation

YOLOv5 incorporates data augmentation techniques within its training pipeline to enhance model robustness and generalization. During each training epoch, images are subjected to a series of augmentations through an online data loader, including:

- **Scaling:** Adjustments to image size.
- **Color space manipulation:** Modifications to color channels.
- **Mosaic augmentation:** A novel technique that combines four images into four randomly sized tiles.

.

Introduced in the YOLOv3 PyTorch repository and subsequently integrated into YOLOv5, mosaic data augmentation has proven particularly effective in addressing the challenge of small object detection, a common limitation in datasets by the Common Objects in Context (COCO) object detection benchmark. By combining multiple images into a single training example, this technique exposes the model to a wider variety of object scales and spatial arrangements, thereby enhancing its ability to accurately detect smaller objects.

3.4 Bounding Box Anchors

The YOLOv3 PyTorch repository introduced a novel approach to anchor box generation, employing K-means clustering and genetic algorithms to derive anchor box dimensions directly from the distribution of bounding boxes within a given dataset. This methodology is particularly critical for custom object detection tasks, as the scale and aspect ratios of objects often diverge significantly from those commonly found in standard datasets like COCO.

The YOLOv5 architecture predicts bounding box coordinates as offsets relative to a predefined set of anchor box dimensions. These anchor dimensions are essential for initializing the prediction process and can significantly influence the model's performance.

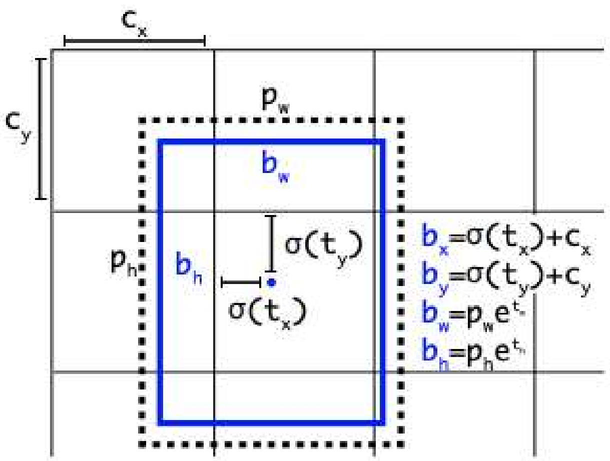

Figure 2: Bounding box prediction based on an anchor box [15]

3.5 Loss Calculation

The YOLOv5 loss function is a composite of three components: Binary Cross-Entropy (BCE) for class prediction and objectness, and Complete Intersection over Union (CIoU) for localization. The overall loss is computed as a weighted sum of these individual losses:

$$\text{Loss} = \lambda_1 \cdot L_{\text{cls}} + \lambda_2 \cdot L_{\text{obj}} + \lambda_3 \cdot L_{\text{loc}} \tag{1}$$

where L_{cls}, L_{obj}, and L_{loc} represent the BCE loss for class prediction, BCE loss for objectness, and CIoU loss for localization, respectively. The coefficients λ_1, λ_2, and λ_3 are hyperparameters that balance the contributions of each loss component to the overall optimization process.

3.6 16 Bit Floating Point Precision

The PyTorch framework offers the capability to reduce computational precision from 32-bit to 16-bit floating-point numbers during both training and inference. When applied to YOLOv5, this technique has demonstrated potential for significant acceleration in inference speed. However, it is essential to note that these performance gains is only contingent upon specific GPU architectures, notably the V100 and T4 models for Yolov5. While the limitations of hardware compatibility exist at present, ongoing developments by NVIDIA indicate a prospective expansion of this efficiency enhancement to a wider range of hardware platforms.

3.7 CSP Backbone

Both YOLOv4 and YOLOv5 incorporate the CSP (Cross-Stage Partial) bottleneck module for feature extraction. This architectural innovation, introduced by WongKinYiu et al., addresses the issue of redundant gradient information prevalent in larger convolutional neural network backbones. By decoupling feature maps into two main parts and recombining them, the CSP module effectively reduces computational cost and model complexity without compromising performance. This efficiency enhancement is particularly advantageous for the YOLO family, where rapid inference and compact model size are paramount.

CSP models draw inspiration from the DenseNet architecture as shown in Figure 3, addressing the challenges inherent in deep CNNs, including the vanishing gradient problem. By fostering direct connections between layers, DenseNet aimed to enhance feature propagation, promote feature reuse, and reduce the overall number of network parameters.

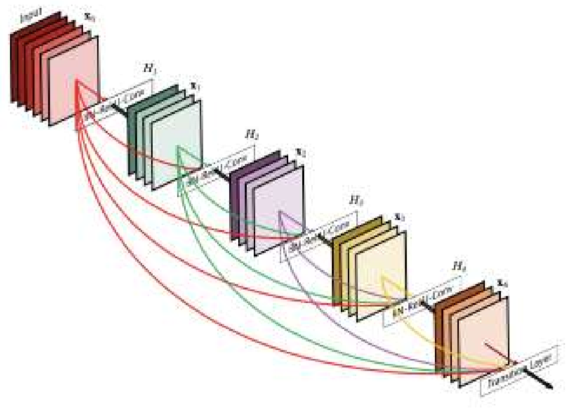

Figure 3: Interconnected dense layers in DenseNet [16]

The CSPResNext50 and CSPDarknet53 architectures incorporate modifications to the DenseNet structure as illustrated in Figure 4. Specifically, the feature maps generated by the base layer are divided into two branches. One branch is processed through a dense block, while the other is forwarded directly to the subsequent stage. This architectural modification aims to mitigate computational bottlenecks inherent to DenseNet and enhance learning by preserving an unaltered representation of the original feature map.

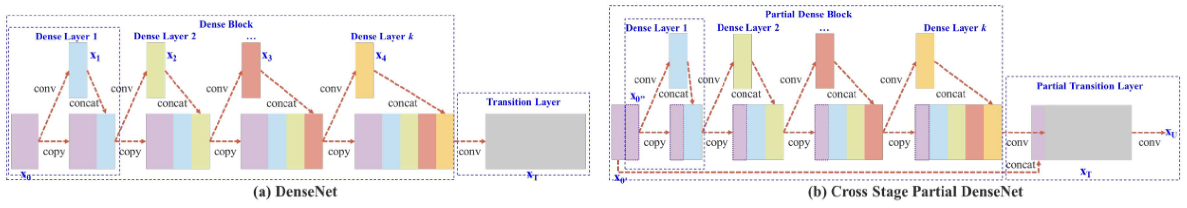

Figure 4: (a) DenseNet and (b) CSPDenseNet [17]

3.8 PA-Net Neck

Both YOLOv4 and YOLOv5 employ the PA-Net architecture for feature aggregation [18]. As illustrated in Figure 5, each P_i represents a distinct feature layer extracted from the CSP backbone. This architectural choice is inspired by the EfficientDet object detection framework, where the BiFPN module was deemed optimal for feature integration. While the BiFPN approach serves as a benchmark, it is plausible that alternative implementations within the YOLO framework could yield further performance improvements.

It is evident that YOLOv5 builds upon the foundational research of YOLOv4 in selecting the most suitable neck architecture. YOLOv4 comprehensively evaluated a range of options including FPN, PAN, NAS-FPN, BiFPN, ASFF, and SFAM.

4 YOLOv5 Models

The YOLOv5 architecture encompasses five distinct models, ranging from the computationally efficient YOLOv5n to the high-precision YOLOv5x.

- **YOLOv5n:** Designed for resource-constrained environments, YOLOv5n is the smallest and fastest model in the series. Its compact size, less than 2.5 MB in INT8 format and approximately 4 MB in FP32 format, makes it suitable for deployment on edge devices and IoT platforms. Compatibility with OpenCV DNN further enhances its utility for mobile applications.
- **YOLOv5s:** Representing the baseline model, YOLOv5s incorporates approximately 7.2 million parameters. Its performance characteristics make it suitable for CPU-based inference tasks.

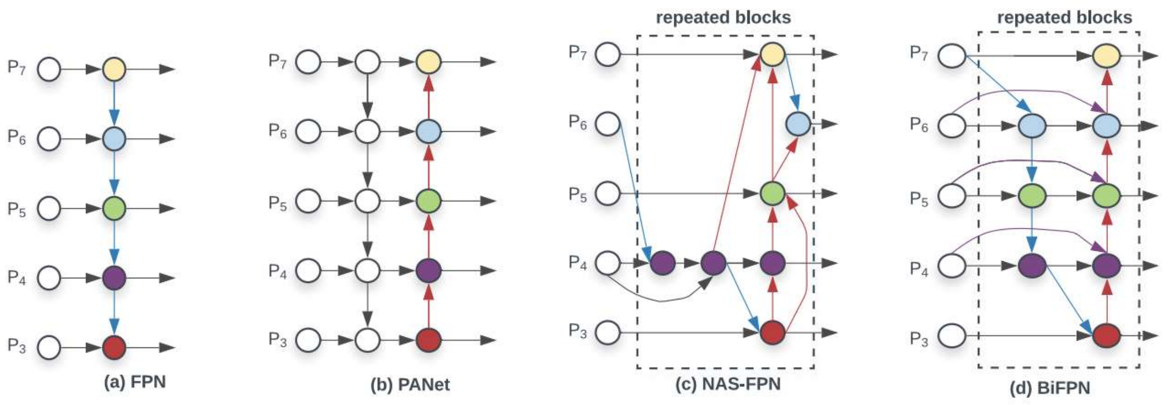

Figure 5: Variations of FPN architectures [19]

- **YOLOv5m:** Positioned as a mid-sized model with 21.2 million parameters, YOLOv5m offers a balanced combination of speed and accuracy. This model is often considered a versatile choice for a wide range of object detection applications and datasets.
- **YOLOv5l:** With 46.5 million parameters, YOLOv5l is designed for scenarios requiring higher precision, particularly in detecting smaller objects within images.
- **YOLOv5x:** As the largest and most complex model in the series, YOLOv5x achieves the highest mAP among its counterparts. However, this performance comes at the cost of increased computational requirements due to its 86.7 million parameters.

A comprehensive overview of the model variants, encompassing inference speed across CPU and GPU platforms as well as the number of parameters for a 640-pixel image size, is presented in Table 1.

Table 1: YOLOv5 Model Performance

Model	Params (Million)	Accuracy (mAP 0.5)	CPU Time (ms)	GPU Time (ms)
YOLOv5n	1.9	45.7	45	6.3
YOLOv5s	7.2	56.8	98	6.4
YOLOv5m	21.2	64.1	224	8.2
YOLOv5l	46.5	67.3	430	10.1
YOLOv5x	86.7	68.9	766	12.1

5 YOLOv5 Annotation Format

YOLOv5 employs a PyTorch TXT annotation format that closely resembles the YOLO Darknet TXT standard, with the addition of a YAML file specifying model configuration and class labels. To facilitate compatibility with YOLOv5, annotation data generated from various tools may require conversion. Platforms such as Roboflow offer support for multiple annotation formats and can export data directly into the YOLOv5-compatible format. Popular annotation tools including VOTT, LabelImg, and CVAT can also be utilized, with appropriate data conversion steps.

6 YOLOv5 Labelling tools

For efficient data management and annotation, Ultralytics, the developers of YOLOv5, recommends Roboflow as a compatible labeling tool [20]. Table 2 provides an overview of third-party integration platforms compatible with the YOLOv5. For each platform, the table outlines its primary functionalities when integrated with YOLOv5.

Table 2: YOLOv5 Integrations

Integration Platform	Functionality
Deci [21]	Automated compilation and quantization of YOLOv5 for enhanced inference performance.
ClearML [22]	Tracking, visualization, and remote training of YOLOv5 models using an open-source platform.
Roboflow [18]	Labeling and exporting custom datasets directly compatible with YOLOv5 training.
Weights and Biases [23]	Tracking and visualization of YOLOv5 training runs in the cloud.

7 Discussion

YOLOv5 represents a significant advancement in the field of object detection, building upon the foundations laid by its predecessors while introducing novel improvements are listed below:

1. **Architectural Advancements:** YOLOv5 represents a significant advancement in object detection, building upon previous YOLO iterations. The incorporation of the CSP backbone and PA-Net neck enhances computational efficiency without compromising accuracy. The CSP module effectively addresses redundant gradient information, while the PA-Net architecture optimizes feature aggregation.
2. **Model Versatility:** The range of YOLOv5 models (n, s, m, l, x) provides flexibility for diverse hardware and application requirements. The smallest model, YOLOv5n, expands object detection capabilities to edge devices and IoT platforms.
3. **Training Methodology Innovations:** YOLOv5's emphasis on data augmentation, particularly mosaic augmentation, improves small object detection and reduces dataset size requirements. The adoption of 16-bit floating-point precision demonstrates a forward-looking approach to optimization.
4. **Performance and Impact:** YOLOv5's high mAP scores and low inference times position it as a strong contender for real-time object detection. The transition to PyTorch has democratized access to the model, fostering broader research and development.

8 Conclusion

YOLOv5 has emerged as a significant advancement in object detection, demonstrating a compelling balance of speed, accuracy, and user-friendliness. While the core architecture builds upon established principles, the model's implementation within the PyTorch framework represents a substantial leap forward, enhancing both development efficiency and deployment capabilities. The availability of multiple model variants tailored to diverse computational constraints expands YOLOv5's applicability across various domains from renewable energy [24] to quality inspection in Manufacturing [25].

The model's accessibility, coupled with its compatibility with existing tools and platforms, positions YOLOv5 as a versatile solution for both research and practical applications. As the field of computer vision continues to evolve rapidly, YOLOv5 serves as a strong foundation for future advancements in real-time object detection and related tasks.

References

[1] M. A. Ansari, A. Crampton, and S. Parkinson. A layer-wise surface deformation defect detection by convolutional neural networks in laser powder-bed fusion images. *Materials*, 15(20):7166, Oct. 2022.

[2] Muhammad Hussain. Yolov1 to v8: Unveiling each variant–a comprehensive review of yolo. *IEEE Access*, 12:42816–42833, 2024.

[3] Arsalan Zahid, Muhammad Hussain, Richard Hill, and Hussain Al-Aqrabi. Lightweight convolutional network for automated photovoltaic defect detection. In *2023 9th International Conference on Information Technology Trends (ITT)*, pages 133–138. IEEE, 2023.

[4] Muhammad Hussain. When, where, and which?: Navigating the intersection of computer vision and generative ai for strategic business integration. *IEEE Access*, 11:127202–127215, 2023.

[5] Muhammad Hussain and Rahima Khanam. In-depth review of yolov1 to yolov10 variants for enhanced photovoltaic defect detection. In *Solar*, volume 4, pages 351–386. MDPI, 2024.

[6] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[7] Muhammad Hussain. Yolo-v5 variant selection algorithm coupled with representative augmentations for modelling production-based variance in automated lightweight pallet racking inspection. *Big Data and Cognitive Computing*, 7(2):120, 2023.

[8] Ultralytics. Yolov5, 2024. Accessed: 2024-07-23.

[9] Glenn Jocher. ultralytics/yolov3: Yolov3 implementation in pytorch, 2024. Accessed: 2024-07-23.

[10] Z. Yao, Y. Cao, S. Zheng, G. Huang, and S. Lin. Cross-iteration batch normalization. In *openaccess.thecvf.com*, 2021.

[11] Roboflow Blog. Yolov4 vs. yolov5: Which one should you use?, 2023. Accessed: 2024-07-24.

[12] Rahima Khanam, Muhammad Hussain, Richard Hill, and Paul Allen. A comprehensive review of convolutional neural networks for defect detection in industrial applications. *IEEE Access*, 2024.

[13] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[14] PyTorch. Pytorch brand guidelines. Accessed: 2024-07-25.

[15] Roboflow Blog. What is an anchor box? `https://blog.roboflow.com/what-is-an-anchor-box/`. Accessed: 2024-07-25.

[16] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[17] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 390–391, 2020.

[18] J. Solawetz. YOLOv5 New Version - Improvements and Evaluation. *Roboflow Blog*, Jun. 2020.

[19] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[20] Brad Dwyer. Roboflow and ultralytics partner to streamline yolov5 mlops, 2021. Accessed: 2024-07-29.

[21] Deci AI. Deci ai documentation, 2024. Accessed: 2024-07-29.

[22] ClearML. Yolov5 clearml integration documentation, 2021. Accessed: 2024-07-29.

[23] Weights and Biases. Yolov5 weights and biases integration guide, 2021. Accessed: 2024-07-29.

[24] Muhammad Hussain, Mahmoud Dhimish, Violeta Holmes, and Peter Mather. Deployment of ai-based rbf network for photovoltaics fault detection procedure. *AIMS Electronics and Electrical Engineering*, 4(1):1–18, 2019.

[25] Tahir Hussain, Muhammad Hussain, Hussain Al-Aqrabi, Tariq Alsboui, and Richard Hill. A review on defect detection of electroluminescence-based photovoltaic cell surface images using computer vision. *Energies*, 16(10):4012, 2023.