Journal of Advanced College of Engineering and Management

DrishYa: OBject Detection And MonocULar Depth Estimation Using Deep Learning

Saman Tripathee[1]*, Samyam Aryal[1], Samiran Pratap Bibash[1]

Department of Computer Engineering, Advanced College of Engineering and Management, Nepal

*Email: saman.075bct054@acem.edu.np

ABSTRACT

This project focuses on object detection and depth estimation using computer vision. The aim of this project is to develop an accurate and reliable system for detecting objects and estimating their distancesfrom a camera. The system uses deep learning-based algorithms to detect objects and estimate their distances. The results of the project show that the system can accurately detect objects and estimate their distances in some scenarios. This project presents an Android app for real-time object detection and depth estimation using computer vision with YOLOv5 and U-Net architecture.

Keywords: *object detection, depth detection, deep learning, accuracy, YOLOv5, U-net*

INTRODUCTION

Object and depth detection project is a computer vision application that aims to detect the presence and location of objects in an image or a video, as well as estimate their distances from the camera. This typeof technology has a wide range of applications, such as autonomous driving, robotics, virtual reality, and surveillance systems.

The motivation for the project is to explore and develop an accurate and efficient computer vision system that can detect objects and estimate their depths in real time. The project aims to address the challenges and limitations of existing object detection techniques, such as occlusion and accuracy issues, and to develop a system that can perform real-time object detection and depth estimation. The project has practical applications in fields such as robotics, autonomous vehicles, and virtual reality, where accurate object detection and depth estimation are critical for safe and effective operation.

Existing 2D object detection systems have limitations in distinguishing between objects and their surrounding environment, which can limit their effectiveness in a range of applications. For example, in the field of theft prevention, 2D object detection systems may not be able to distinguish between real persons and their photographic images, leading to false alarms or missed detections. In medical device tracking, 2D systems may not be able to accurately identify the location and movement of surgical instruments, which can compromise patient safety. In robotics, 2D object detection systems may not provide enough information for robots to navigate through complex environments or avoid obstacles with precision, limiting their usefulness in industrial or service applications. By integrating depth estimation into object detection systems, this project aims to overcome these limitations and develop a more advanced and effective system that can address the challenges of theft prevention, medical device tracking, and robotics in real-world scenarios.

RELATED WORKS

"Real-Time Object Detection Using YOLOv4" by Alexey Bochkovskiy et al. In this paper, the authors introduce YOLOv4, a lightweight and fast object detection algorithm. They show that YOLOv4 achieves state-of-the-art performance on several benchmarks while being more efficient than previous versions of YOLO. This paper provides the theoretical foundation for using YOLOv4 in the proposed project.[1]

"U-Net: Convolutional Networks for Biomedical Image Segmentation" by Olaf Ronneberger et al. In this paper, the authors propose U-Net, a convolutional network for biomedical image segmentation. U-Net is designed to handle small datasets with limited training examples, making it ideal for the proposed project. The paper also describes the architecture of U-Net and how it is used for image segmentation.[2]

"Real-Time Monocular Depth Estimation Using Synthetic Data with Domain Adaptation via Image Style Transfer" by Amir Atapour-Abarghouei. In this paper, the authors propose a monocular depth estimation algorithm that uses domain adaptation to improve performance. The authors show that their method outperforms existing depth estimation algorithms on benchmark datasets. This paper provides a theoretical foundation for the depth estimation component of the proposed project.[3]

The author has researched in object detection and train a model for this purpose with the application of CNNs. The author has used the Keras module of python for this purpose. In the research the author has discussed the use of Mask -R CNN model for kangaroo detection in photographs.[4]

X. Ma et. al. has applied CNN - residual networks to estimate depth from a single image. The article discusses different architectures and classical methods used and derives the best solution among the methods and architectures.[5]

METHODOLOGY

Machine Learning AlgorithmsYOLOv5

Yolov5 is a state-of-the-art deep learning algorithm for object detection and classification developed by Ultralytics. The algorithm uses a combination of deep learning techniques, such as anchor boxes, feature pyramid networks, and focal loss, to improve the accuracy and robustness of object detection.[6]

Architecture

Our detection network has 24 convolutional layers followed by 2 fully connected layers. It further consists of four max pooling layers. The input image is resized into 640x640 before going through the convolutional network. A 1x1 convolution is first applied to reduce the number of channels, which is then followed by a 3x3 convolution to generate a cuboidal

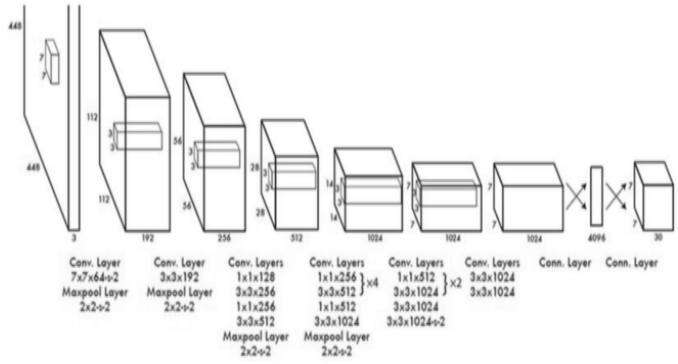


Figure 1: Yolov5 Architecture

output. The activation function under the hood is ReLU, except for the final layer, which uses a linear activation function. Some additional techniques, such as batch normalization and dropout, respectively regularize the model and prevent it from overfitting.

Data Pre-Processing and Annotation

The data augmentation technique helps to create a robust model because it creates more variation in the dataset. During the training phase, the model does not memorize the data; hence, overfitting of the

model is avoided.In this research work, data augmentation techniques are utilized, which include resizing into 640 × 640 pixels and random horizontal and vertical flips. The image can be flipped in two ways: horizontal and vertical. In order to flip the image, the following formulas are used:

To flip an image vertically, the x coordinates of the image pixel need to be changed, and this can be accomplished using:

$$x(new) = width - x(old) - 1) \quad (1)$$

While flipping an image horizontally, the y coordinates of the image pixel need to be changed, and thisis implemented using:

$$y(new) = (height - y(old) + 1) \quad (2)$$

During the image flip, the position of the bounding boxes also changes, and for implementation, the following formulas were used:

Vertical Flip forces the x coordinates to move to different locations. For bounding boxes, the new coordinate can be calculated using:

$$x(new) = \frac{width}{2} + (\frac{width}{2} - x(old)) \quad (3)$$

The Horizontal Flip deals with the y coordinate, and to manipulate the bounding boxes y coordinate, the following formula is utilized:

$$y(new) = \frac{height}{2} + (\frac{height}{2} - y(old)) \quad (4)$$

YOLOv5 uses a special type of dataset that includes information on the annotation in a specific way in the YAML file. Each row of the dataset should contain “class, x_center, y_center, width, height”.[7]

Activation Function

In the actual model, the Leaky ReLU is employed in the network’s middle layers, and the Sigmoid is used in the prediction layer, which is the last layer of the model.

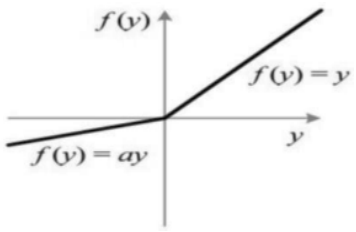


Figure 2: Leaky ReLU Activation Function

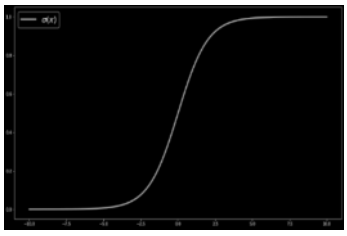


Figure 3: Sigmoid Function

Loss Function

Our YOLOv5 loss combines three different values: **the confidence score**, **the probability of the class**, and **the boundary box**. It also allows for YOLOv5 to be used in multi-label applications. The loss function can be represented by the formula:

$$Loss = l_b + l_c + l_o \quad (5)$$

Here, b, c, and o are the bounding box regression function, classification loss function, and confidence loss function, respectively.

The boundary box loss of YOLOv5 is expressed as:

$$l_b = \lambda_{coord} \sum_{i-0}^{s^2} \sum_{j=0}^{b} I_{i,j}^{0} bj(2 - w_i * h_i) \left[\left(x_i - \hat{x}_i^j\right)^2 + \left(y_i - \hat{y}_i^j\right)^2 + \left(w_i - \hat{w}_i^j\right)^2 + \left(h_i - \hat{h}_i^j\right)^2 \right] \quad (6)$$

The classification loss of YOLOv5 is expressed as:

$$l_c = \lambda_{class} \sum_{i=0}^{s^2} \sum_{j=0}^{b} I_{i,j}^{0} \sum_{CS \epsilon classes} p_i(C) \log(\hat{p}_i(C)) \quad (7)$$

Confidence loss in the model is expressed as:

$$l_o = \lambda_{no-o} \sum_{i=0}^{s^2} \sum_{j=0}^{b} I_{i,j}^{no-o} \left(C_i - \hat{C}_i\right)^2 + \lambda_o \sum_{i=0}^{s^2} \sum_{j=0}^{b} I_{i,j}^{o} \left(C_i - \hat{C}_i\right)^2 \quad (8)$$

The above formula represents λ_{coord} as the coefficient of the position vector, λ_{class} as the category loss coefficient, $\hat{x}$, $\hat{y}$ as the true central target coordinates, and $\hat{h}$, $\hat{w}$ as the targets' height and width. I^0 will be 1 if the target is inside (i, j) the anchor box; otherwise, the value is going to be 0. p_i(C) represents the category probability of the class, and $\hat{p}_i$(C) represents the categories' true value. The total number of categories CS is equal to the length of these two. [7]

U-Net

U-Net is a convolutional neural network architecture designed for semantic segmentation tasks. The network architecture is named U-Net due to its U-shape design, which consists of a contracting path and an expansive path. U-Net has been widely used in object detection and image segmentation.

Architecture

The U-Net architecture consists of two main parts: the contracting path (also called the encoder) and the expanding path (also called the decoder). The contracting path consists of a series of convolutional layers, each followed by a max pooling layer. The convolutional layers use filters of size f x f and produce k feature maps. The max pooling layers downsample the feature maps by a factor of 2. The expanding path consists of a series of transposed convolutional layers, each followed by a concatenation with the corresponding feature maps from the contracting path, and then a convolutional layer. The transposed convolutional layers use filters of size f x f and produce k/2 feature maps. The concatenation operation helps to preserve spatial information lost during the max pooling operation in the contracting path. The U-Net architecture also includes skip connections between corresponding layers in the contracting and expanding paths. These skip connections allow the

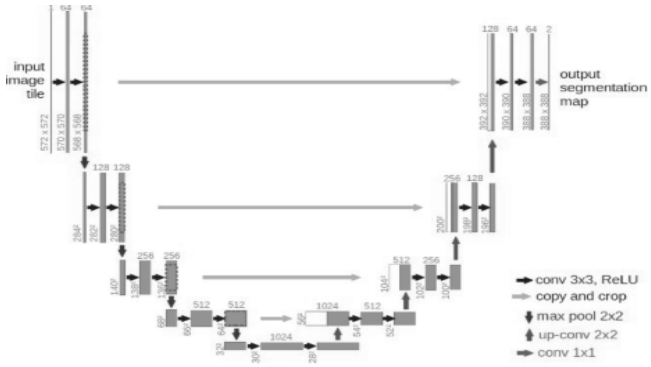


Figure 4: U-net Architecture

network to capture both high-level and low-level features, and help to produce more accurate segmentation masks.[8]

Activation Function

The tanh (hyperbolic tangent) activation function is a commonly used activation function in neural networks, which maps the input values to the range [-1, 1]. Mathematically, [9]

Loss FunctionsSSIM

$$\boldsymbol{Tanh(x)} = \frac{\mathbf{e^{x} - e^{-x}}}{\mathbf{e^{x} + e^{-x}}}$$

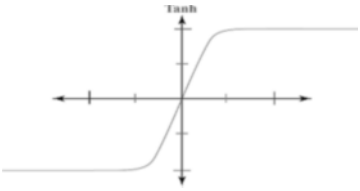


Figure 5: Tanh Activation Function

The SSIM (Structural Similarity Index) loss is a perceptual loss function that measures the structural similarity between two images. The SSIM index is a metric that takes into account the luminance, contrast, and structural information in the images.[10] Mathematically,

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (9)$$

L1 loss

The L1 loss is a commonly used loss function for monocular depth estimation, which measures the absolute difference between the predicted and ground truth depth maps. In the context of monocular depth estimation, the L1 loss is typically used to optimize the model parameters during training. Mathematically,[11]

$$L1 = |y_{actual} - y_{predicted}| \quad (10)$$

OPTIMIZATION FUNCTION

Adam Optimizer is used in our project. The Adam optimizer is an extension of the stochastic gradient descent (SGD) optimization algorithm. It uses adaptive learning rates for each parameter.[12]

DATA SOURCE

For object detection, we have used the COCO (Common Objects in Context) dataset. We've built a small- scale specific purpose model and thus have used 30 classes. [13] Our total labeled and annotated dataset consists of 30,000 images. The train-test-validation split for the model was 80% - 20% - 10%.

For depth estimation, we will use the DIODE (Dense Indoor and Outdoor DEpth)[14] dataset, using which we can estimate the depth of surrounding objects. Our total dataset for depth estimation consistedof 25,000 images. The train-test-validation split for the model was 80% - 20% - 10%.

LOSS CURVES

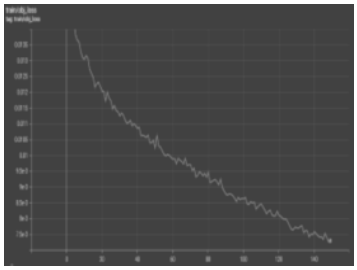


Figure 6: Training Object Loss

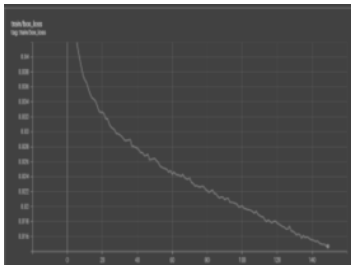
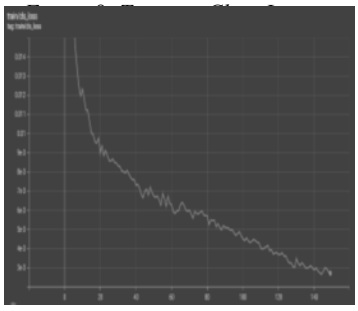


Figure 7: Training Box Loss



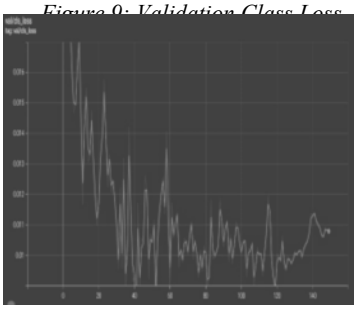


Figure 9: Validation Class Loss

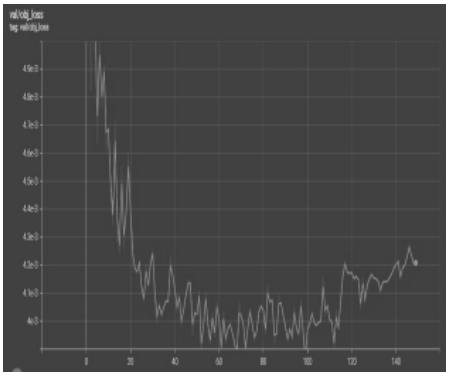


Figure 10: Validation Object Loss

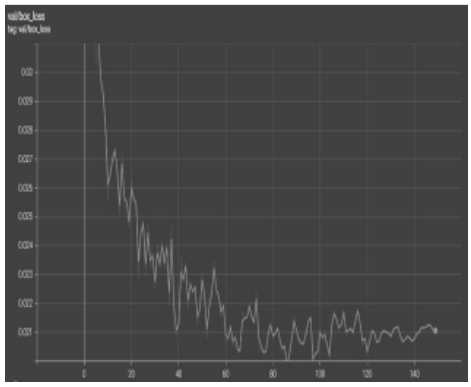


Figure 11: Validation Box Loss

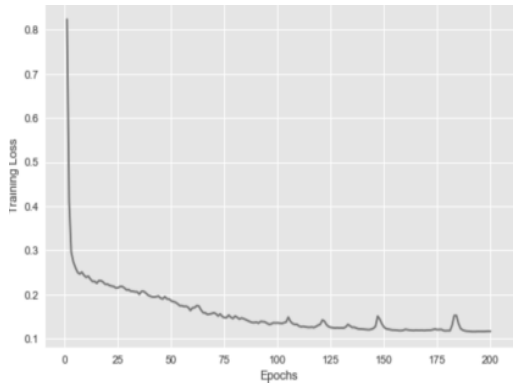


Figure 12: Train Loss for Depth Estimation Model

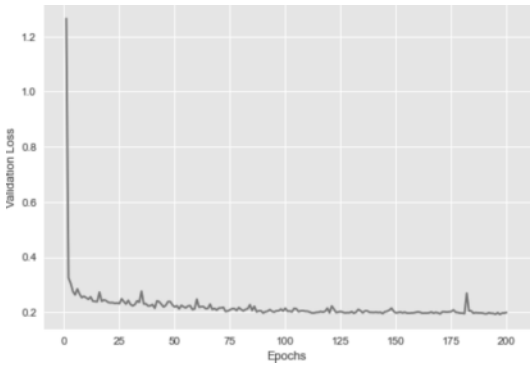


Figure 13: Validation Loss for Depth Estimation Model

PRECISION CURVES FOR OBJECT DETECTION

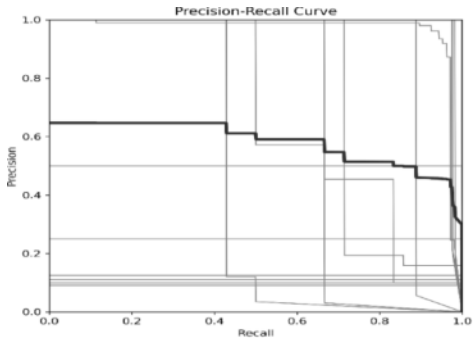


Figure 14: Precision Recall Curve for Object Detection

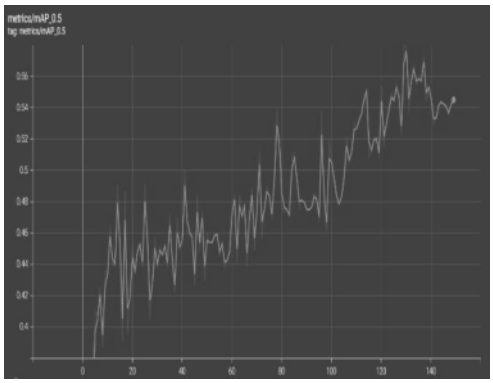


Figure 15: Mean Average Precision for Object Detection

To examine the performance of the model, mAP@IoU = 0.5, Precision, and Recall were utilized. IoU stands for Intersection over Union, and it is a basic metric used for comparing object detection models/systems. Using the TP and FP values, Precision, Recall, and mAP are calculated using equations 11, 12, 13 respectively. For calculating the mAP 0.5, the threshold value is taken as 0.5.

$$Precision = \frac{TP}{TP + FP} \quad (11)$$

$$Recall = \frac{TP}{TP + FN} \quad (12)$$

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \quad (13)$$

In mAP, N is the number of queries and AP is the Average Precision.[15]

RESULT AND ANALYSIS

Table 1: Object Detection Evaluation

Scene Number	Objects in the Scene	Detected Object	Score	Correct? (Y=Yes, N= No)	Accuracy Percentage
1	Laptop	Laptop	56%	Y	100%
	Chair	Chair	71%	Y	
	Person	Person	75%	Y	
2	Chair	Chair	83%	Y	75%
	Laptop	TV	60%	N	
	Vase	Vase	67%	Y	
3	Chair	Chair	60%	Y	100%
	Bottle	Bottle	63%	Y	
4	Chair	Chair	88%	Y	75%
	Couch	Couch	84%	Y	
	Book	Book	63%	Y	
	Table	N/A	-	N	
5	Person	Person	70%	Y	100%
	Trophy	Trophy	66%	Y	
	Bus	Bus	89%	Y	

Mean Accuracy Percentage = 90%

Table 2: Distance Detection Table

Scene Number	Detected Objects	Actual Distance	Detected Distance	Correct? (Y=Yes,N=No)	Accuracy Percentage
1.	Person	Far	Far	Y	100%
	Laptop	Far	Far	Y	
	Chair	Near	Near	Y	
2.	Table	Near	Near	Y	50%
	Couch	Near	Far	N	
3.	Motorcycle	Near	Near	Y	100%
	Person	Near	Near	Y	
4.	Chair	Near	Near	Y	75%
	Dining Table	Far	Far	Y	
	Person	Far	Near	N	
	Potted Plant	Far	Far	Y	
5.	Chair	Near	Near	Y	100%
	Dining Table	Near	Near	Y	
	Person	Far	Far	Y	
	TV	Far	Far	Y	

Mean Accuracy Percentage = 85%OUTPUT

Figure 16: Output

CONCLUSION

From the object detection table, we can see that the model was able to correctly identify most of the objects in the scenes, but there were a few cases where it made errors in detection. The accuracy percentages ranged from 75% to 100%, with an overall average of around 90%. From the distance detection table, we can see that the model was able to correctly identify distance of most of the objectsin the scenes, but there were a few cases where it made errors in detection. The accuracy percentages ranged from 50% to 100%, with an overall average of around 85%.

Overall, these tables suggest that while computer vision models have come a long way in recent years, there is still room for improvement, especially in more challenging scenarios where the objects are more complex, or the environment is more cluttered. As with any machine learning model, training data and testing conditions can also significantly impact performance. Therefore, continuous improvement in the training data, model architecture, and testing conditions can help increase the accuracy and reliability of the model over time.

SUGGESTIONS AND RECOMMENDATION

- Further dataset augmentation
- Use of Transfer learning
- Increase the depth of the model
- Use of a larger GPU
- Incorporate other sensors

REFERENCES

[1] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," Apr. 2020, Accessed: May 04, 2023. [Online]. Available: https://arxiv.org/abs/2004.10934v1

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9351, pp. 234–241, 2015, doi: 10.1007/978-3-319-24574-4_28/COVER.

[3] A. Atapour-Abarghouei and T. P. Breckon, "Real-Time Monocular Depth Estimation Using Synthetic Data with Domain Adaptation via Image Style Transfer," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2800–2810, Dec. 2018, doi: 10.1109/CVPR.2018.00296.

[4] "How to Train an Object Detection Model with Keras - MachineLearningMastery.com." https://machinelearningmastery.com/how-to-train-an-object-detection-model-with-keras/ (accessed May 04, 2023).

[5] X. Ma, Z. Geng, and Z. Bie, "Depth Estimation from Single Image Using CNN-Residual Network."

[6] G. Jocher et al., "ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation," Nov. 2022, doi: 10.5281/ZENODO.7347926.

[7] S. Paul, S. Batra, K. Mohiuddin, M. N. Miladi, D. Anand, and O. A. Nasr, "A Novel Ensemble Weight-Assisted Yolov5-Based Deep Learning Technique for the Localization and Detection of Malaria Parasites," Electronics (Switzerland), vol. 11, no. 23, Dec. 2022, doi: 10.3390/electronics11233999.

[8] "U-Net Explained | Papers With Code." https://paperswithcode.com/method/u-net (accessed May 04, 2023).

[9] "Tanh - Cuemath." https://www.cuemath.com/trigonometry/tanh/ (accessed May 04, 2023).

[10] "All about Structural Similarity Index (SSIM): Theory + Code in PyTorch | by Pranjal Datta | SRM MIC | Medium." https://medium.com/srm-mic/all-about-structural-similarity-index-ssim-theory-code-in-pytorch-6551b455541e (accessed May 04, 2023).

[11] "L1 loss function, explained." https://stephenallwright.com/l1-loss-function/ (accessed May 04, 2023).

[12] "Adam - Cornell University Computational Optimization Open Textbook - Optimization Wiki." https://optimization.cbe.cornell.edu/index.php?title=Adam (accessed May 04, 2023).

[13] "COCO - Common Objects in Context." https://cocodataset.org/#home (accessed Jun. 27, 2022).

[14] "DIODE Dataset." https://diode-dataset.org/ (accessed May 04, 2023).

[15] "(PDF) A Survey on Performance Metrics for Object-Detection Algorithms." https://www.researchgate.net/publication/343194514_A_Survey_on_Performance_Metrics_for

[16] Object-Detection_Algorithms (accessed May 04, 2023).