A Comparison of YOLO Models for License Plate Detection

Arjan Chakravarthy [1]

Abstract

License plate detection and recognition (LPDR) has become an increasingly popular field of study under image recognition. The ability to autonomously create a bounding box around a license plate using machine learning models for image recognition is an important component of intelligent transportation systems. The YOLO model has become a popular CNN model for image detection because of its speed and accuracy. This work focuses on the use of the YOLO model for object localization as applied to license plate detection. The research compares the accuracy and speed between the YOLOv3 and YOLOv4 models along with various configuration parameters and finds that the YOLOv4 model is most accurate while the YOLOv3 with reduced image resolution is the fastest in prediction time.

Keywords : **Image recognition, LPDR, YOLO model**

I. INTRODUCTION

Visual recognition is a common application of machine learning models given its widespread application [1]. Visual recognition can be segmented into a series of computer vision tasks, namely, image classification, object localization, object detection, and object segmentation. Image classification involves identifying whether a single object class exists within an image. Object localization allows for the identification of multiple object classes within an image and bounding boxes to identify the location of each object. Object detection combines image classification and object localization to detect multiple object classes and their locations in an image. Finally, object segmentation provides for the identification of the exact pixels associated with an object class as opposed to a bounding box. In this paper, I focused on object detection for license plates [2].

A. Image Detection Models

In the field of object detection, the methods used have evolved over time with each having their benefits. Histogram of Oriented Gradients (HOG), Region Based Convolutional Neural Network (RCNN), Fast RCNN, Faster RCNN, and YOLO are just a few examples of such models. Each model employs a different method for detecting an object but all except for HOG utilize a convolutional neural network (CNN). A CNN is an artificial neural network usually applied to analyze images. In simple terms, a CNN is designed to process pixels from images. If asked to classify an image, a CNN takes a group of pixels from the image and compares it with the features from the training dataset in order to identify and locate an object class. Instead of seeing whether each pixel aligns between an image and the training image (which can often easily lead to incorrect classifications), the CNN is effective because of the patterns that it notices across pixels. The YOLO model differs from the family of RCNN models in that it processes the images differently [3].

B. The YOLO Model

YOLO (You Only Look Once) has become the most

Manuscript Received : October 3, 2021 ; Revised : November 15, 2021 ; Accepted : November 20, 2021. Date of Publication : December 5, 2021

A. Chakravarthy[1] is *Student Researcher (Computer Science),* at American International School 1106, Jawaharlal Nehru Road, Tharamani, Chennai, Tamil Nadu 600113. India. Email : arjanchak@gmail.com ; ORCID iD : https://orcid.org/0000-0002-6691-154X

DOI : https//doi.org/10.17010/ijcs/2021/v6/i6/167643

prominent model for object detection and localization. The YOLO model works by deconstructing an entire image into smaller, more manageable chunks and finding the center of each object to be classified. Based on the center, the model creates countless bounding boxes. Each bounding box is then assigned a predicted probability as to whether an image class exists in that given region. These predicted probabilities are calculated by the model based on the data used to train the neural network. The final bounding box is determined by applying the Intersection Over Union (IOU) threshold across all candidate bounding boxes. The model then decides on the final bounding box based on the highest predicted probability [4], [5].

II. DATASET AND FEATURES

Data from Larxel's Kaggle Dataset of 433 images of license plates was used for the present study [6]. The dataset consisted of a variety of license plates in different settings from different countries and with different lighting conditions. This provided a wide range of real-world images with obfuscations and dim lighting, all of which would measure the model's ability to accurately detect the license plate.

Along with each image is an XML file with label information including bounding box and object class among other information. YOLO models simply require five numbers associated with an image stored in a space-separated .txt file. The first number corresponds to the object class, the second and third correspond to the x and y coordinates respectively of the center of the bounding box, and the fourth and fifth numbers represent the height and width respectively of the bounding box. I quickly created a program to extract the necessary values from the XML file and calculated the center of the bounding box. Since the width and height were already delineated in the XML file, I only had to detect and copy these values to the .txt file.

III. METHODS

After creating the .txt files for all 433 images, I proceeded to download AlexeyAB's darknet repository which contained all of the different YOLO models.

I started with the YOLOv3 model to get baseline metrics across our validation dataset. This would help us understand where future models were situated in relation to the YOLOv3, the model that popularized the use of YOLO. The baseline run was performed on the validation dataset with the image size set to 416 x 416 pixels. I then reduced the pixel size to 320 x 320 for the second run. Although the length and width of the images can be different, they have to be a multiple of 32 since YOLO down-samples the input by 32. I preferred to keep the length and width values the same to ensure uniformity across all images. The premise behind reducing the image sizes was to explore how the accuracy and time varied between the larger and smaller image sizes [7].

I then ran the YOLOv4 model on the validation dataset to compare its results to the YOLOv3 model. I also tried the two different image sizes on the YOLOv4 model to see which combination would yield the highest accuracy and which would yield the fastest detection time.

Finally, I experimented with a third factor that could affect the accuracy of the model: converting the images to grayscale. RGB images can influence a model's accuracy by unnecessarily adding a layer of complexity. The difference between the model accuracy of RGB and grayscale images relies heavily on the dataset. In other words, converting images to grayscale does not always improve accuracy or prediction time but it can prove to be effective with some datasets [8]. Instead of testing grayscale conversion on both YOLOv3 and YOLOv4, I tested only on the YOLOv4 model. I anticipated the YOLOv4 model would be more accurate than the YOLOv3 model (as has been discovered by countless other studies [9]), so I wanted to understand if the accuracy or detection time would improve further with grayscale images.

To determine the detection time of the different models, I created a video of the images from the dataset with a frame rate of 25 frames per second (FPS). The model is then run on the video to determine the detection time in frames per second.

Finally, I created precision-recall graphs for the YOLOv4 model with 416 x 416 images with varied IOU thresholds. The IOU threshold is usually preset to 0.5, meaning the overlap between the predicted and expected bounding boxes must be above 0.5. In order to create a precision-recall curve, the predictions are arranged in descending order based on predicted probability. The IOU threshold then determines whether a given prediction is counted as true positive, false positive, or false negative [10].

IV. RESULTS AND DISCUSSION

A. Performance Metrics

The YOLO model is unique because it is both accurate and fast and it is able to work in real-time [11]. Machine learning models are measured on several different metrics. Some of the key metrics I will be considering here are:

- Precision
- Recall
- F_1 score
- Mean Average Precision (mAP)
- Detection speed

Precision and recall are defined as follows:

Precision = True Positive / (*True Positive + False Positive*) (1)

Recall = True Positive / (*True Positive + False Negative*) (2)

In practical terms, precision is defined as the ratio of correctly predicted positive detections (true positives) to the total number of positive detections for a given class (true positives + false positives). Recall is defined as the ratio of the correctly predicted positive detections (true positives) to the total number of instances of a given class (true positives + false negatives). Precision and recall tend to be inversely related in most real-world scenarios. As efforts are taken to improve the precision of a model, these efforts potentially eliminate true positive detections resulting in a lower recall.

For this reason, it is important to optimize model configuration parameters to identify the point where precision and recall can be maximized. The F_1 score helps in this regard as it is the harmonic mean of precision and recall as shown in the eq. (3), expressed both in terms of precision and recall as well as true positives (TP), false positives (FP), and false negatives (FN).

$$F_1 = (2 * precision * recall) / (\text{precision} + \text{recall}) = TP / (TP + ½ (FP + FN) \quad (3)$$

Perfect scores in precision and recall would result in a perfect F_1 score.

In object detection models, another metric used to measure the correctness and reliability of a model is the mean average precision (mAP). The average precision (AP) is given by the integral of the precision-recall curve for a given IOU (Intersection Over Union) threshold. Conceptually, the AP of a model is maximized when both precision and recall are maximized, and this agrees with eq. (4) where $p(r)$ is a function of the precision p as it varies with the recall r integrated from 0 to 1 (the range of the precision) [12].

$$AP = \int_0^1 (p(r)\ dr) \quad (4)$$

The mAP is determined by averaging the APs across several IOU thresholds [13].

Beyond the metrics that help measure the reliability of a model, it is equally critical to evaluate the runtime of the model. Each image detection model faces a trade-off

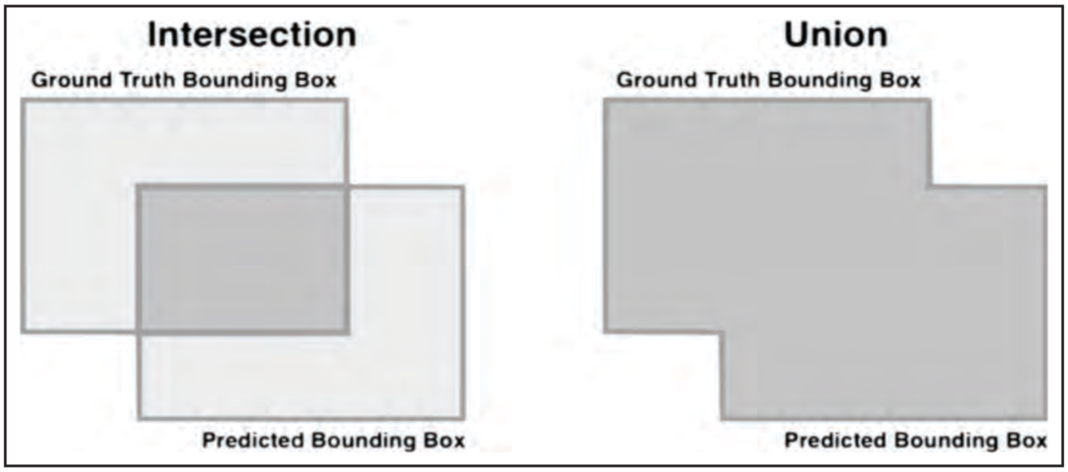

Fig. 1. Intersection Over Union (IOU)

between accuracy and speed. Finding the right balance between the two is crucial to optimizing the overall efficiency of a model. Thus, I also consider the time it takes for each model to predict a given object class and compare how the mAP may be compromised with faster runtimes.

While not considered a performance-related metric, the IOU threshold is an important configuration parameter in object detection models to optimize performance. The IOU is calculated simply by dividing the area of overlap of the predicted and labeled bounding boxes by the area of union of these same two bounding boxes. Fig. 1 is an example.

IOU = Area of Intersection / Area of Union (5)

IOU thresholds are generally set at 0.5 but can vary based on the application. In cases where the model is being used to count the presence of a given object class, a smaller IOU threshold can be suitable. However, in our application, it is important to capture a sufficient portion of the license plate so that the detected image can then be used with Optical Character Recognition (OCR) to read the license plate.

B. Results and Discussion

The results from the various model runs are given in Table I.

As shown in other studies, the YOLOv4 model performed with higher accuracy than the YOLOv3 model for both image sizes. With YOLOv4 416 x 416, the mAP was 90.35 compared to an mAP of 84.24 with YOLOv3 416 x 416. Similarly, with image size set to 320 x 320, YOLOv4outperformed YOLOv3, 89.27 to 81.98 respectively.

The image size has some impact on the accuracy, with larger image sizes producing more accurate predictions. The YOLOv3 models with images of size 416 x 416 resulted in an improved mAP of 84.24 compared with the model with images of size 320 x 320 which only had an mAP of 81.98. For YOLOv4, the mAP of the model that ran 320 x 320 was 89.27, while it was 90.35 for the model with image sizes of 416 x 416. Since both models had higher mAP values when the size of the images was increased, I concluded that increasing size did correlate with improved accuracy. I concluded that with the lower resolution, the model would not be able to create precise bounding boxes as detecting the boundaries for the license plates is more difficult. With higher resolution images, the model would more clearly distinguish the license plate from other objects in the image.

It is also interesting to note that the metrics for each model are consistent across different trials. Since the precision, recall, and F_1 Score are closely related in what they measure (especially the F_1 Score as it depends on the precision and recall) when one of the three is greater for one model, it is likely that the others are also greater. This trend does falter with the IOU values which are calculated differently from the other performance metrics and thus, a higher IOU value does not necessarily warrant a higher mAP or F_1 Score.

With regards to detection time, the YOLOv4 model proved to be slower for both image sizes. The YOLOv4 detection times for both the 416 x 416 and 320 x 320 image sizes were 71.5 and 41.7 fps respectively, while the corresponding times for the YOLOv3 model were 87.3 and 65.5 fps. This was unexpected as the YOLOv4 is

TABLE I.

RESULTS FROM YOLO TRIALS

	mAP	IOU	Precision	Recall	*F*1-Score	FPS
YOLOv3 320 x 320	81.98	66.19	0.86	0.83	0.85	87.3
YOLOv3 416 x 416	84.24	65.45	0.87	0.87	0.87	65.5
YOLOv4 320 x 320	89.27	69.87	0.88	0.89	0.88	71.5
YOLOv4 416 x 416	90.35	71.01	0.91	0.89	0.90	54.7
YOLOv4 320 x 320 Grayscale	89.92	68.35	0.88	0.9	0.89	61.8
YOLOv4 416 x 416 Grayscale	90.99	68.75	0.89	0.95	0.92	51.5

designed for better performance times when compared to YOLOv3. The reasons for this decrease can be a topic for future research.

With both YOLOv3 and YOLOv4, detection times increased with smaller image sizes. This was to be expected as images of lower resolution decrease the time needed to detect a license plate within the image.

The YOLOv4 416 x 416 grayscale model had the highest accuracy among the models. While there was a marginal increase in the mAP between the grayscale and RGB versions of the YOLOv4 416 x 416 models, there was a larger increase in the F_1 scores with the YOLOv4 416 x 416 grayscale model having the highest F_1 at 0.92. However, it was evident that this high accuracy came at the expense of time. The detection time for both trials of the grayscale model was slower than their RGB counterparts. Intuitively, grayscale models should be faster as the monochromatic images should reduce processing time but this could vary with the dataset, and especially seeing that accuracy increased, it does follow that the trade-off would be a decrease in detection time. Also, the difference in time detection between the YOLOv4 416 x 416 grayscale and RGB models was only 3.2 FPS (51.5 *vs* 54.7 respectively). The YOLOv4 grayscale for the model based on smaller image resolution showed a notable reduction in detection time from its RGB counterpart. The 320 x 320 grayscale had an FPS of 61.8, while the model with the RGB images had an FPS of 71.5.

C. Precision-Recall Curve

Precision-recall curves were created for the YOLOv4 416 x 416 model. The graph below depicts the precision-recall curves for IOU threshold between 0.4 and 0.9. These curves are plotted based on 0.1 increments in recall.

Each IOU curve is determined by plotting a series of precision and recall points for varying confidence thresholds. As shown in Fig. 2, the precision tends to decrease as the recall increases. This is intuitive and to be expected. With a high confidence threshold, there will be fewer and more accurate positive detections leading to a high precision but low recall. As the confidence threshold is dropped, the recall increases as the precision drops.

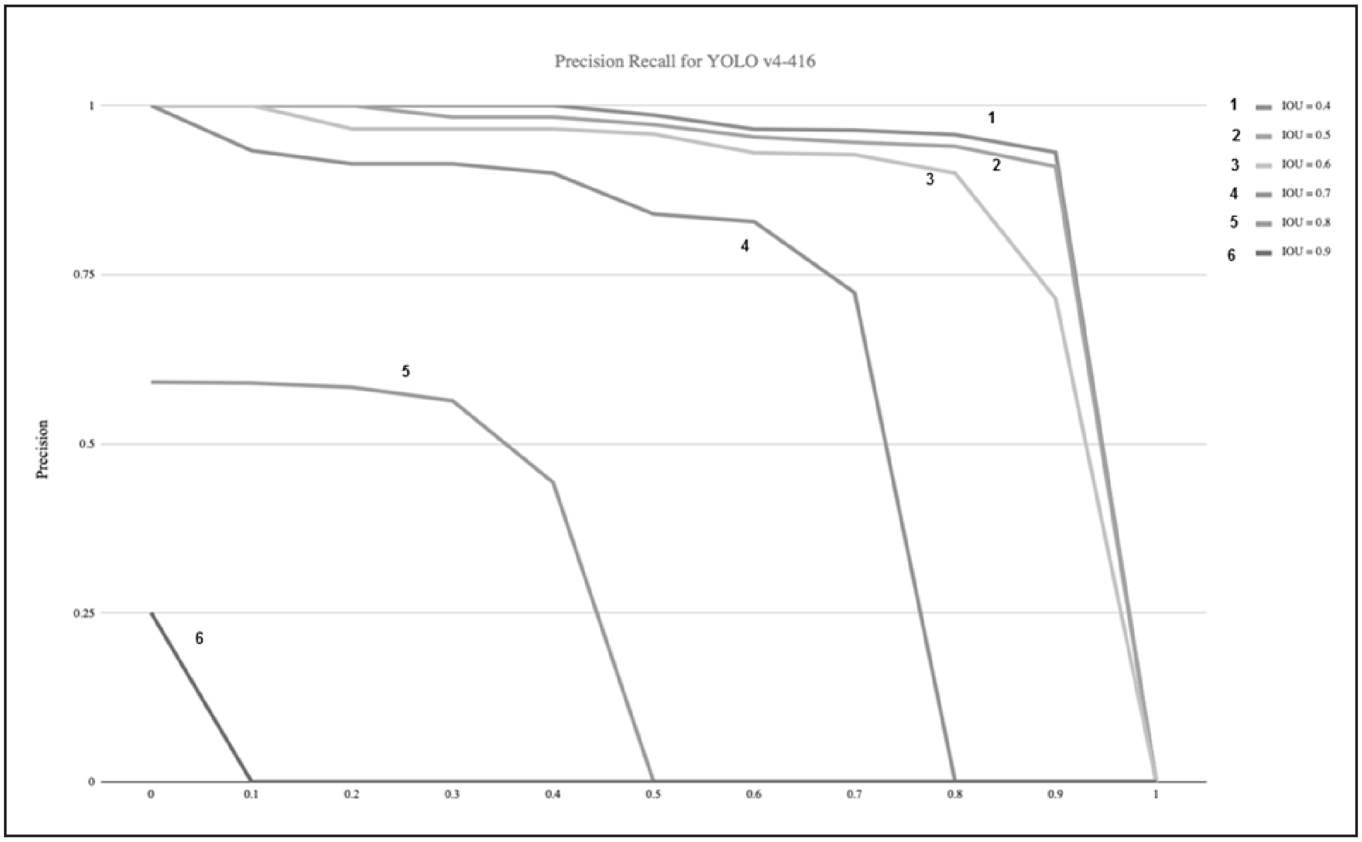

Fig. 2. Precision-Recall Curves

This process is repeated for several IOU thresholds. As the IOU threshold increases to 0.9, the point where precision and recall are maximized moves closer to the origin. At higher IOU thresholds, only predicted bounding boxes with extremely high overlap are considered. The number of true positives decreases at an IOU threshold of 0.9 as the model does not classify license plates that would otherwise be classified at lower thresholds. However, the number of false positives also decreases as the model sets a higher threshold for which images should be classified as having license plates.

The recall also decreases at higher IOU thresholds for a similar reason. Since the recall also depends on the number of true positives, it will also decrease at higher IOU thresholds as the model will miss a lot of the license plates that should be classified correctly. Additionally, the false negatives will also be high at such a high IOU threshold as the model will not classify license plates as such when the IOU threshold is lower.

With regards to lower thresholds, the precision and recall are both higher as there are more true positives. Many license plates that would not otherwise be detected at higher IOU thresholds are now detected. However, one disadvantage of setting a low IOU threshold is the increase in false positives. The model at lower thresholds may allow bounding boxes that would otherwise be rejected at higher thresholds. The recall also increases because the number of false negatives will decrease with lower thresholds.

Although at first, it may seem that setting lower IOU thresholds are ideal as they maximize the precision and recall, it is important to consider the application for which the model is being used. Lower IOU thresholds can be used if the purpose of the model is simply to count the number of objects in a certain image. However, if the purpose is to accurately detect an object with a precise bounding box, higher thresholds (perhaps 0.5 or 0.6) are ideal as they produce higher quality bounding boxes. Given that our intent is object detection and localization where it is important to determine the exact location and full bounding box for a license plate, the highest IOU threshold that does not significantly compromise on the precision and recall is ideal.

Based on the graph, an IOU of between 0.5 and 0.6 would work well as it is the highest IOU threshold that provides for both a precision and recall of greater than 0.9.

V. CONCLUSION

Through this analysis, I was able to compare the different YOLO model versions as well as the impact of different image sizes in running the model and converting the dataset into grayscale.

As part of future research, it would be interesting to explore why YOLOv4 performed with a slower detection time on our dataset. This was contrary to what was expected based on earlier studies. In addition, it would be interesting to understand why grayscale images allowed for higher accuracy on this license plate detection and why the detection time was lower for the model based on grayscale images as compared to RGB images.

Future work can also consider additional image preprocessing steps to evaluate their impact on model performance. One such preprocessing step could be varying the contrast of the images [14]. This can make the colors more vibrant and thus, edge detection for the model easier. Another step that can be taken is to de-noise the images. Many datasets have images that contain a lot of noise or extraneous color and brightness which can often impede the accuracy or detection time. These preprocessing steps are avenues for further research and exploration of YOLO models applied to license plate detection.

AUTHOR'S CONTRIBUTION

Arjan Chakravarthy is the sole author and has performed the entirety of the work described in this paper. He prepared the training and test datasets, ran the various models against the data, and analyzed the output.

CONFLICT OF INTEREST

The author certifies that he has no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in the manuscript.

FUNDING ACKNOWLEDGEMENT

The author has not received any financial support for the research, authorship, and/or for the publication of the article.

REFERENCES

[1] S. Pal and S. K. Behera, "Traffic sign recognition for self driving vehicles using matlab and tensorflow," *Indian J. Comput. Sci.,* vol. 5, no. 6, Nov.-Dec. 2020, doi: 10.17010/ijcs/2020/v5/i6/157502

[2] O. Russakovsky et al., "ImageNet large scale visual recognition challenge." 2015. [Online]. Available: arXiv: 1409.0575 [cs.CV]

[3] P. Dwivedi. "YOLOv5 compared to faster RCNN. Who Wins?" *Towards data science.com*. (accessed : May 7, 2021). https://towardsdatascience.com/yolov5-compared-to-faster-rcnn-who-wins-a771cd6c9fb4. .

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. "You only look once: Unified, real-time object detection," in *2016 IEEE Conf. Comput. Vision Pattern Recognition (CVPR),* 2016, pp. 779-788. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf

[5] G. Karimi. "Introduction to Yolo algorithm for object detection," *Section.io*. https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/ (accessed May 7, 2021).

[6] "Car License Plate Detection, Version 1", Kaggle, Sep. 2020. https://www.kaggle.com/andrewmvd/car-plate-detection/activity (accessed May 10, 2021).

[7]] D. Radečić. "How to detect license plates with Python and YOLO." Towards Data Science.com. https://towardsdatascience.com/how-to-detect-license-plates-with-python-and-yolo-8842aa6d25f7 (accessed May 6, 2021).

[8] P. Canuma. "Image pre-processing." Medium.com. https://prince-canuma.medium.com/image-pre-processing-c1aec0be3edf (accessed May 9, 2021).

[9] J. Solawetz. "YOLOv4 - Ten tactics to build a better model." robloflow. https://blog.roboflow.com/yolov4-tactics/ (accessed May 12, 2021).

[10] A. Rosebrock. "Intersection over Union (IOU) for Object Detection." PyImageSearch.com, https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/ (accessed May 18, 2021).

[11] R. Rajkumar, A. Kaushal, and A. Saha, "Accelerating machine learning research using transfer learning," *Indian J. Comput. Sci.,* vol. 3, no. 2, Mar.-Apr. 2018, doi: 10.17010/ijcs/2018/v3/i2/123212

[12] T. C. Arlen. "Understanding the MAP evaluation metric for object detection." Medium.com. https://medium.com/@timothycarlen/understanding-the-map-evaluation-metric-for-object-detection-a07fe6962cf3 (accessed March 7, 2021).

[13] J. Hui, "Map (Mean Average Precision) for Object Detection." Medium.com. [Online]. Available: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173 (accessed May 14, 2021).

[14] J. Nelson, "When to use contrast as a preprocessing step," roboflow.com. https://blog.roboflow.com/when-to-use-contrast-as-a-preprocessing-step/ (accessed May 19, 2021).

APPENDIX

The code for the work described in this paper can be found at:

https://drive.google.com/drive/folders/1i2e_xMApt-1yIGnbQMdrMg2uOBniUWTf?usp=sharing

About the Author

Arjan Chakravarthy has a background in electronics and software development with a focus on IoT applications. His most recent work focuses on image processing and machine learning as applied to image recognition and object localization.