International Journal of Emerging Trends in Science and Technology
http://ijetst.in/index.php/ijetst **ISSN 2348-9480**
DOI: https://dx.doi.org/10.18535/ijetst/v8i9.01 || Vol: 09 Issue: 09 (2022)

Open access Journal

Object Detection in Unmanned Aerial Vehicle Imagery

Atiya Kailany [1] | Moayed D. Daneshyari [2] *

1. Department of Computer Science California State University, East Bay

2. Department of Computer Science California State University, East Bay

Abstract— Recently drones have emerged as a great new tool that many companies and even personal entities have decided to exploit. Looking at the potential these new drones have it would be in everyone's best interest to use these drones or Unmanned Air Vehicles to their fullest potential. One way to help achieve that is by mounting cameras onto these drones and using the data captured by the video feed received from these cameras. This project is using machine learning and computer vision to help make use of the data captured by these drones. A Retina Net model is used to train various models on images captured from the Stanford campus. The model is detecting various classes of urban artifacts such as pedestrians and cyclists.

1. INTRODUCTION

The Federal Aviation Administration (FAA) of the United States projects that the number of recreational drones in the United States will reach 1.6 million by 2025 [1], this is an annual growth of 8.5 percent over the last 5 years and it is expected to keep growing at this rate. Some might argue that drones will be taking over a large chunk of the lab industry. A study was done by Price water- house Coopers (PwC) [1] predicts that drones could replace 127 billion dollars' worth of human labor. Thus, making this industry one of the highest growing industries with a great potential for success. In formulating this project, it was worth taking into consideration what are areas or industries new and upcoming drones will dominate. PwC also provides an estimate on which industries will benefit the most from emerging drones as well as each industry's value. Computer vision has already achieved impressive results because of advances in deep learning algorithms, technology requirements, and data set availability. Object detection is the most common investigation problem performed by researchers because it has numerous applications. Object detection's goal is to recognize things from a specific category (for example, humans, dogs, vehicles, or motorbikes) in a photograph and, if any, return the area and extent of each instance of objects. It serves as the foundation for tackling sophisticated and high-level computer vision tasks like object tracking, segmentation, event detection, image captioning, scene understanding, crowd monitoring, and activity recognition.

Researchers began overcoming the difficulty of constructing general object identification algorithms capable of detecting categories of items that resemble humans. Object detection has been significantly improved. However, object detection in drone applications has not been thoroughly investigated. All applications, from surveillance to agriculture, require reliable object detection to function properly.

Drones are becoming increasingly popular in real-time applications, and it is only a matter of time before they are widely adopted in many sectors. Recently, Amazon has gained federal approval to deploy drones for delivery. Hii, Courtney Royall (2019) [3] investigated drone transportation for medical purposes and concluded that it is viable. Precision agriculture is another field of importance and is predicted to grow significantly faster than other applications, as the use of unmanned aerial vehicles (UAVs) becomes one of the most important components of managing farm tasks. Precision agriculture is a set of methods for tracking crops, gathering data, and carrying out educated crop management tasks such as optimal water supply and pesticide choices. Therefore, it is important to capitalize on this growing industry in its beginner phases and help achieve a stable drone vision system that pushes this industry forward as well as helps the lives of millions of people and industries across the globe.

2 FEATURE PYRAMID NETWORKS

Being confined to such a small area and small pixels has generated the need for a different detection method, a 2017 study at Cornell University [11] proposed a method that uses Feature Pyramid Networks. Feature pyramids are a fundamental component of recognition algorithms that detect objects at various scales. Recent deep learning object detectors, on the other hand, have eschewed pyramid representations, in part because they are computationally and memory costly. In this study, they used a deep convolutional network's inherent multi-scale, pyramidal structure to build feature pyramids at a negligible extra cost. For constructing high-level semantic feature maps at various scales, a top-down architecture with lateral linkages is designed. This architecture, known as a Feature Pyramid Network (FPN), performs significantly better as a generic feature extractor in a variety of applications. Furthermore, this method can execute at 6 frames per second on a GPU, making it a viable and accurate solution to multi-scale object detection. Detecting objects at different scales is difficult, especially for little objects.

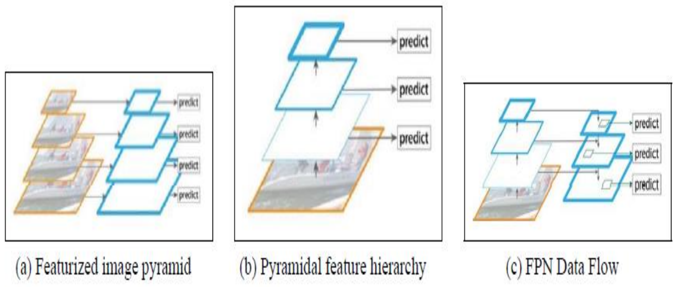


Figure 1: Feature Pyramid Network [11]

To detect objects, we can utilize a pyramid of the same image at different scales (see Figure 3). However, processing numerous scale images takes time, and the memory demand is too great to train end-to-end at the same time. As a result, we can only use it in inference to increase accuracy as much as feasible, especially for competitions where speed is not an issue. Alternatively, we can build a feature pyramid and utilize it to detect objects (See Figure 1). Closer to the image layer, however, feature maps are formed of low-level structures that are ineffective for reliable object detection.

The Feature Pyramid Network (FPN) is a feature extractor developed with accuracy and speed in mind for such pyramid concepts. It replaces detectors' feature extractors, such as Faster R-CNN, and generates many feature map layers (multi-scale feature maps) with higher quality information than the conventional feature pyramid for object identification.

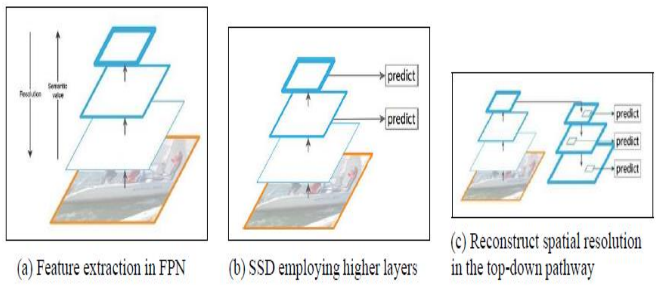

Resolution
Semantic value
predict
predict
predict
predict
predict
(a) Feature extraction in FPN
(b) SSD employing higher layers
(c) Reconstruct spatial resolution in the top-down pathway

Figure 2: Feature Pyramid Network [11]

FPN is made up of two pathways: bottom-up and top-down. For feature extraction, the bottom-up pathway is the standard convolutional network. The spatial resolution degrades as we ascend. The semantic significance of each layer increases as more high-level structures are recognized. See Figure 2.

As seen in Figure 2.c below, FPN gives a top-down approach to building higher resolution layers from a semantic-rich layer. While the reconstructed layers are semantically strong, the locations of objects after all the down-sampling and up-sampling are not precise. To assist the detector forecast the position better, we add lateral links (See Figure 3) between reconstructed layers and the relevant feature maps. It also serves as a skip connection to facilitate training, which is similar to Res Net.

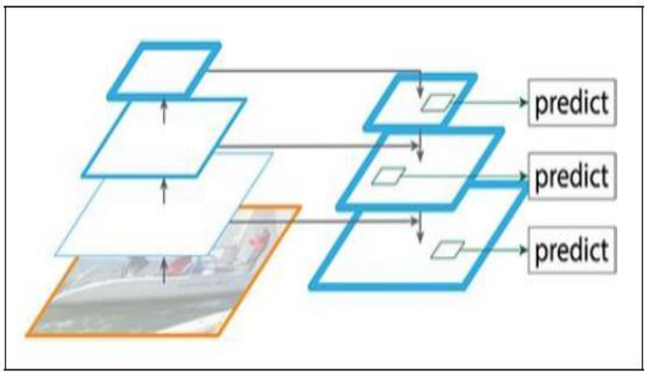


Figure 3: Skip connections [11]

3 FOCAL LOSS LAYER

To date, the most accurate object detectors are based on a two-stage technique popularized by R-CNN, [6] in which a classifier is applied to a sparse collection of potential object locations. One-stage detectors, on the other hand, that are applied across a regular, dense sampling of likely item locations have the potential to be faster and simpler but have so far behind the accuracy of two-stage detectors. To understand the inner workings and design of Focal Loss, we have to look at a few specific cases. First, we address binary cross-entropy loss for single object classification. (See Figure 4)

$$\begin{gathered} if\, y = 1 \\ CE(p, y) = -\log(p) \\ otherwise \\ CE(p, y) = -\log(1-p) \end{gathered}$$

$$\begin{gathered} if\, y = 1 \\ pt = p \\ otherwise \\ pt = 1 - p \\ CE(pt) = -\log(pt) \end{gathered}$$

Figure 4: Binary cross-entropy loss [6]

We introduce a weighting parameter for large class imbalance. Typically, this is the inverse class frequency or treated as a cross-validation hyper-parameter. It will be referred to as alpha termed balancing parameter in this context. See Figure 5.

$$CE(pt) = -\alpha_t * \log(pt)$$

Figure 5: Balancing parameter in focal loss [6]

As stated in the research, [6] readily identified negatives account for most of the loss and dominate the gradient. While alpha balances the relevance of positive and negative instances, it does not distinguish between simple and difficult cases. As a result, the authors altered the cross-entropy function and developed focused loss, as described below.

$$FL(pt) = -(1 - p_t)^{\gamma} * \alpha_t * \log(pt)$$

Figure 6: Focal loss equation [6]

In Figure 6, gamma is called the focusing param, and alpha is called the balancing param. In conclusion, Focal loss adds very little weight to well-classified examples and a large weight to miss-classified or hard classified examples.

4 MODEL ARCHITECTURE (RETINANET)

Facebook AI Research (FAIR) has released two papers in 2017 [11] and 2018 [6] respectively on their state-of-the-art object detection frameworks. These were discussed in detail in above. This subsection will simply show how these various layers come together to form this robust object detection pipeline, without going into the details of each layer. Retina Net is a unified network comprised of a backbone network and two task-specific sub-networks that communicate with one another. The backbone of the system is a convolution network that is readily available and is responsible for building a convolutional feature map over the whole input image. Using the output of the backbones as input, the first sub-net classifies the data, while the second sub-net performs convolution bounding box regression.

Backbone: FPN was constructed on top of ResNet50 or ResNet101. We may, however, use any classifier of our choosing.

The Classification Sub-net: This subnet estimates the likelihood of item existence at each spatial place for each of the A anchors and K object classes. Takes a feature map with C channels from a pyramid level as input, then applies four 3x3 convolutional layers, each with C filters and followed by ReLU activations.

Box Regression Sub-net: This sub-net is comparable to the classification net, but the parameters are not shared. If an item exists, it returns its position with relation to the anchor box. The loss function for this segment of the sub-network is smoothl1losswithsigmaequalto3.

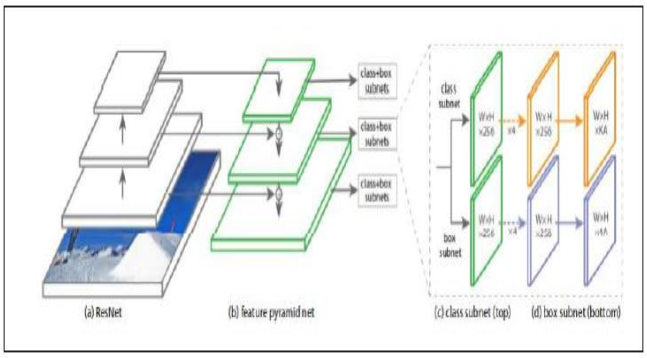


Figure 7: Focal loss equation [6]

5 APPROACH

To be able to detect the mentioned objects in the project mentioned above, we are using a Keras implementation to train the Retina Net model used in the papers above. Combing the information from both papers mentioned [6] and [11] in addition to Keras documentation and a publicly sourced implementation on Git Hub [12], a Retina Net model was trained for the purposes of this project. The development of this system features the use of computer vision and machine learning to detect if an urban artifact is present in the video feed or image the model is being run on. The model was trained on Google Colab's online environment using the Keras framework.

The system starts by capturing images and video from a designated urban area, this data is stored as a data set and is used to train the Retina Net model. Due to limitations in getting the required equipment, this step has been skipped and the SDD data set [8] has been used in its place. The data is then fed to the Object detection model with its two layers being the FPN [11] and Focal loss layer [6] to generate a class selection. In the scope of the system, this classification is returned to the drone along with the percentage accuracy. There are some limitations to this approach which will be discussed more in detail in the limitations section, however, to briefly address the issue. It has become evident that doing such processing on-board the drone will be incredibly slow in terms of the drone's decision making time frame. Thus, an approach would be to send that data to a cloud server that does the detection then return the appropriate classification accordingly. Figure 8 below shows a system diagram that illustrates the concepts of our UAV object detector model.

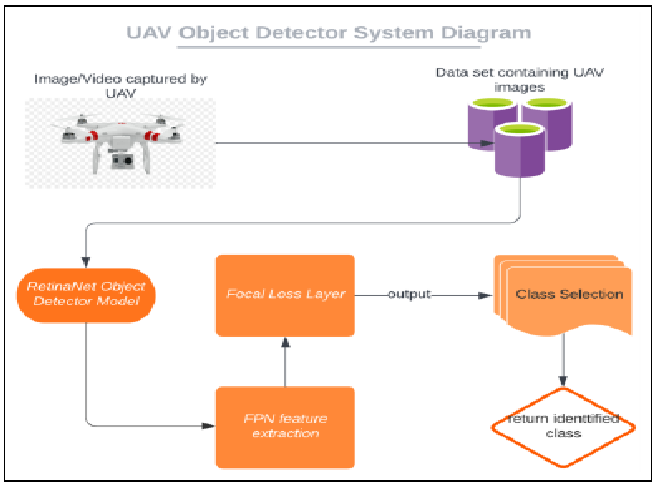

UAV Object Detector System Diagram
Image/Video captured by UAV
Data set containing UAV images
RetinaNet Object Detector Model
Focal Loss Layer
output
Class Selection
FPN feature extraction
return identified class

Figure 8: System Diagram

The Retina Net model as published from the Fizyr Git Hub repository [12] uses the following technologies and systems to build the support for the Retina Net architecture on the Keras framework. The technologies include Python, Anaconda, Keras and SDD Data set

After obtaining the required infrastructure to start training, the training process begins and is so far the lengthiest process of this whole project. The Keras implementation allows for several hyper-parameters to be modified as training is going on, and we have added Tensor board support in order to be able to adjust these hyper parameters accordingly.

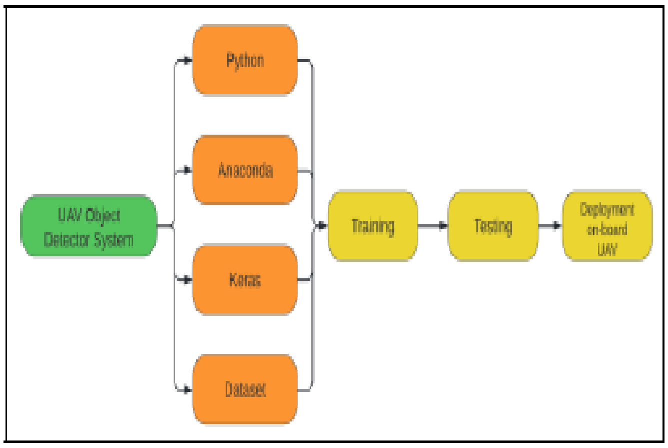


Figure 9: Development Process

6 SOFTWARE AND TOOLS

There are two major pieces of software that was developed for the UAV Object Detection system. The first is the object detection model training system and for this Python, Keras, and Open CV were used in addition to numerous python modules. Keras framework was chosen for the implementation because of its comprehensive and flexible ecosystem of “tools, libraries, and community resources” that allows easy model building and Machine Learning application deployment in Python. To standardize data for training, the system uses Open CV [13] - a library that provides highly optimized operations for image processing and manipulation.

The second major software system is a collection of three Jupyter notebooks that make use of the generated model. These notebooks will first convert the trained model into an inference model and run it on static images, video feed and finally run some analytical and accuracy testing methods to generate some statistics about the accuracy of the model.

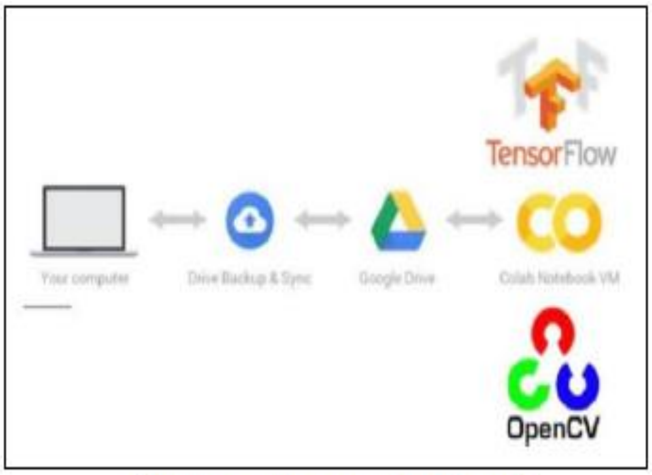


Figure 10: Software and tools used

7 DATA SET

Stanford Drone Data set (SDD) [8] is a huge data collection containing aerial photographs captured by drones over the Stanford campus. The data set is useful for challenges involving object recognition and tracking. It features around 60 aerial videos which have bounding box coordinates for each of the six classes: "Pedestrian," "Biker," "Skateboarder," "Cart," "Car," and "Bus." The data set is particularly rich in pedestrians and bikers, with these two classes accounting for around 85-95 percent of the annotations. The annotations format for the data set had to be changed in order to fit the requirement of the Retina Net model. The following were the primary steps we took; for the model, we used a sample of photographs from the enormous Stanford drone data collection. We took around 2200 training photos with 30,000+ annotations and approximately 1000 images for validation.

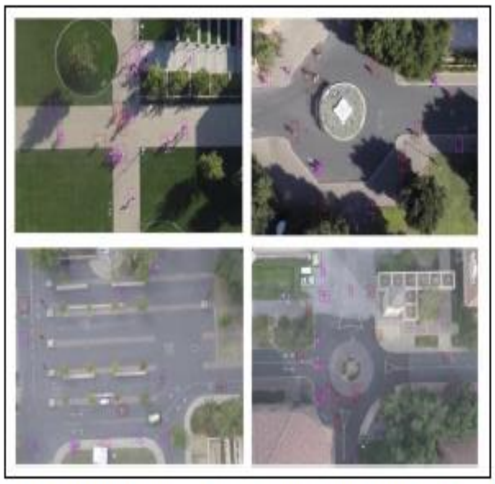
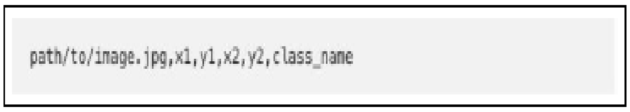


Figure 11: Sample images from stanford drone data set [8]

Generating annotations in the format required for Retina Net. Retina Net requires all annotations to be in the following format.



```
path/to/image.jpg,x1,y1,x2,y2,class_name
```

Figure 12: Required retinaNet annotation format

A challenge imposed by this data set is the abundance and dominance of the pedestrian and biker classes in the data set. It quickly became evident that differentiating between these two classes will be one of the main goals of the system.

On the other hand, there was an unavoidable hurdle in the training of this system when it came to the skater class, or at least within the scope of this project. The awful, almost identical similarity when looking from a bird’s eye view between the pedestrian and skater classes made it incredibly hard even for the human eye to distinguish the difference between these two classes.

As you can see below image of the skater and pedestrian from a high point of view look awfully similar and to be able to extract features that distinguish the two is not an easy task. This has caused the overall model accuracy to be low and has negatively affected the results of the model when running analysis.

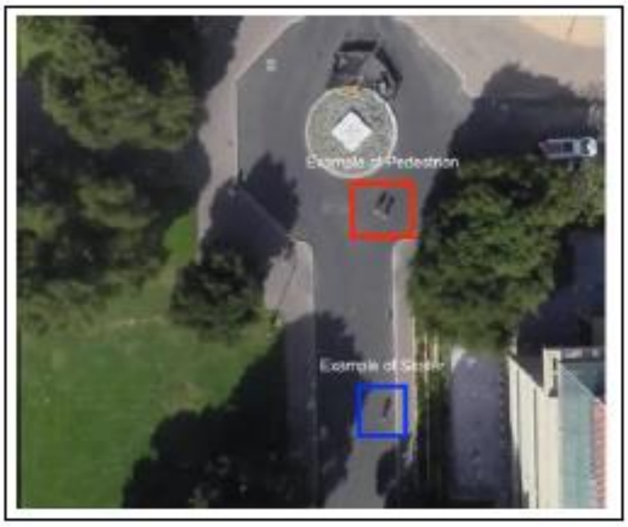


Figure 13: Pedestrian Vs. Skater

8 TRAINING

Accuracy is an excellent statistic to use when evaluating how well a model works since it is not only simple to comprehend but also sufficient in most circumstances for gauging model performance. A well-rounded set of evaluation metrics is calculated using the Tensor flow implementation of the Keras API metrics [14], including inbuilt Categorical Accuracy() from the Tensor Flow Metrics add-ons [15], to give a better understanding of how each class behaves in training.

Various methodologies for selecting model hyper parameters are now available, including empirical choice, manual search, grid search, random search, and a variety of optimization algorithms. Empirical and manual choice selections were utilized due to the time restrictions of this project.

Separating data into training and testing data is a frequent practice for predicting model performance. For the purposes of this project, 80 percent of the data will be utilized for training, while the remaining 20 percent will be used for testing. This split, sometimes known as an 80/20 split, is a typical rule of thumb.

The output of the training process includes:

• Saved Model (contains weights and bias),

• Evaluation metrics (As show in error Analysis script)

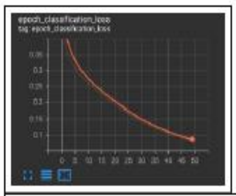
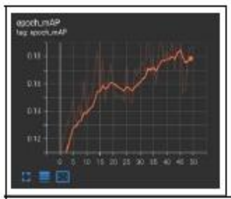
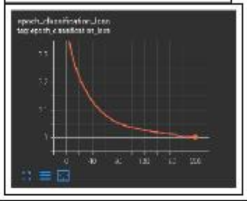
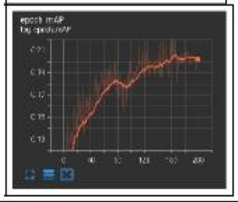
• Tensor Board logs yield various graphs including accuracy and loss per epoch.

The number of times the training algorithm is presented in the whole training data set is determined by a training parameter called epochs. We decided to train for more than 50 epochs (the number of times the model will work through the entire training data set).

As seen in Table 1, which covers changes in accuracy and loss across epochs, as the training progresses in time (number of epochs), the accuracy typically improves and the loss decreases. The number of samples properly identified divided by the total number of samples is known as accuracy. As a result, the greater the precision, the better. A number of 1.0 means that all samples have been correctly classified 100 percent of the time. Loss is a metric that measures how well the network's output matches the ground truth, and it's calculated using training and validation data.

A loss of 0.0 indicates that the output vector is identical to the ground truth. As a result, a smaller loss value is preferable. Loss refers to how far the predicted values differ from the actual values in the training data. There are a variety of loss functions that may be used to calculate loss; however, we prefer the Categorical Cross-Entropy loss function for this project.

Table 1: Tensorboard logs for models trained

	TensorBoard Logs	
Model	Epochs	mAP
Model 1		
Model 2		

Due to the high demand for memory for each epoch, we were only able to train 50 epochs due to Google Collab's usage limitation. We were only able to use small batch sizes because the environment could not allocate enough memory for higher batch sizes which caused the model to crash and stop training after a certain number of epochs.

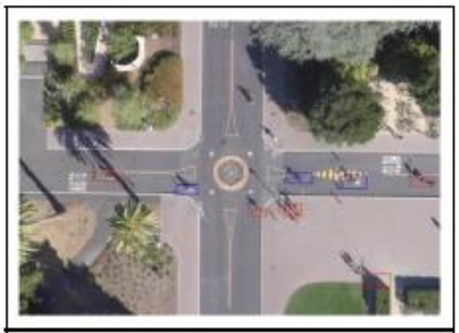
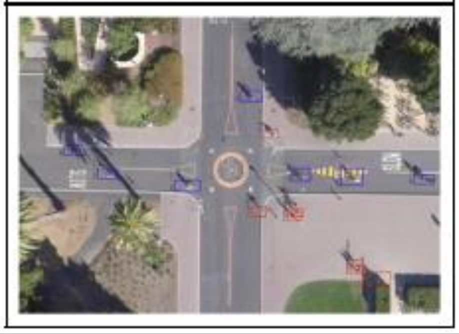
A solution to this problem would be to have smaller batch sizes which we discussed with the creators of the Fizyr implementation, and we concluded that we are limited to batch size less than 8 if you have good hardware. However, for Google Collaborator the only batch size that allowed me to allocate enough memory to train all 200 epochs was a batch size of 2. We have some failed training attempts of higher batch sizes on the repository for reference.

As can be seen from the table, model 2 was able to achieve a higher epoch's loss due to its ability to train for higher epochs. This was only possible through changing the batch size as mentioned previously in addition to the step count. We have roughly 2586 samples of training images in our data set and as a rule of thumb we could always set the steps to:

Steps = Number of Samples / batch size

This method would yield a total step of 1293 for model 2 which ended up working perfectly and allowed for maximization of the number of epochs trained due to lesser memory demand. In addition to allowing the model to receive much higher epoch's classification loss and giving the model a higher accuracy compared to model 1 as you will be able to see in the images in the upcoming sections.

Table 2: : Image inference for each trained model

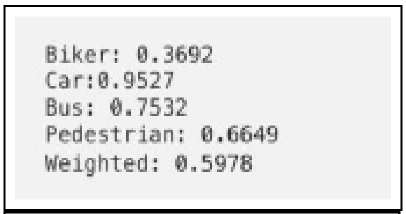
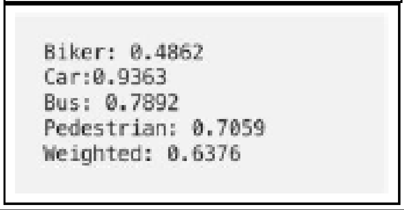
Models and corresponding inference	
Model	Inference
Model 1	
Model 2	

Looking at Table 2, the blue inference boxes signify the biker class while the red boxes signify the pedestrian class. We can clearly see that model 2 had a slightly higher accuracy compared to model 1. If we look at the biker classes, we can see that model 2 correctly identified all the biker classes present in the image while model 2 only identified a few. In addition to miss-classifying a couple of biker classes as pedestrians. Furthermore, model 2 is able to detect slightly more pedestrians on the scene compared to model 1. While model 1 did not even detect them which is partially due to the confidence level being set at 50 percent.

9 MODEL STATISTICS

The following figures were obtained for each model. Please see table 3 for models along with their corresponding accuracies.

Table 3: Models Accuracies

Models and corresponding accuracy	
Model	Accuracy
Model 1	 Biker: 0.3692 Car:0.9527 Bus: 0.7532 Pedestrian: 0.6649 Weighted: 0.5978
Model 2	 Biker: 0.4862 Car:0.9363 Bus: 0.7892 Pedestrian: 0.7059 Weighted: 0.6376

From the figures above we can see that our highest accuracies were in the easily detectable classes from a high altitude like a car and a Bus. However, when it came to the biker class we see the lowest accuracies there, this is heavily due to the model confusing the pedestrian class with the biker class. Again, this was something that we expected and as such a solution we proposed was object tracking which was addressed in detail in the previous sections.

10 CONCLUSION AND FUTURE WORK

This capstone project documents the design, implementation, and testing of object detection in unmanned aerial imagery, a model trained using Retina Net to detect urban artifacts. The system trained a RetinaNet-based RNN with a Focal

Accuracy Layer with approximately 63 percent accuracy on the testing dataset. An adequate dataset was used for the classes of pedestrian, biker, car, bus, and skateboarder. Hyper-parameter tuning and comparison of models were performed to select the best model amongst them for deployment to a few scripts for inference and error analysis.

Testing on the dataset proved to be heavily influenced by the class imbalance of having higher numbers of pedestrian and biker instances than other classes. The model could be tested on a different data set that would allow for equal distribution of each class. This is left as future work.

Special issues related to memory allocation and GPU limitation for training were discussed. While the issues for training were somewhat resolved, future work must take place for to resolve the issue of confusing the biker class with the pedestrian class. Object tracking would prove to be the needed solution to distinguish between these two classes if implemented correctly. Investigation into object tracking of such small artifacts on a heavily imbalanced data set is a research topic in and of itself.

In summary, this work serves as a proof of concept that an effective object detection model can be achieved through the use of Computer Vision and Machine Learning used on a heavily populated area from a high altitude via a UAV.

11 REFERENCES

1. "Unmanned Aircraft Systems (UAS)." https://www.faa.gov/uas/ (accessed Apr. 30, 2022).
2. "Communications-review-july-2017.pdf." Accessed: Apr. 30, 2022. [Online]. Available: https://www.pwc.com/gx/en/communications/pdf/communications-review-july-2017.pdf
3. M. S. Y. Hii, P. Courtney, and P. G. Royall, "An Evaluation of the Delivery of Medicines Using Drones," Drones, vol. 3, no. 3, Art. No. 3, Sep. 2019, doi: 10.3390/drones3030052.
4. C. Xiang, H. Shi, N. Li, M. Ding, and H. Zhou, "Pedestrian Detection Under Unmanned Aerial Vehicle an Improved Single-Stage Detector Based on RetinaNet," in 2019 12th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Oct. 2019, pp. 1–6. doi: 10.1109/CISP-BMEI48845.2019.8965666.
5. X. Wang, P. Cheng, X. Liu, and B. Uzochukwu, "Fast and Accurate, Convolutional Neural Network Based Approach for Object Detection from UAV," in IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society, Oct. 2018, pp. 3171–3175. doi: 10.1109/IECON.2018.8592805.
6. T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," ArXiv170802002 Cs, Feb. 2018, Accessed: Apr. 30, 2022. [Online]. Available: http://arxiv.org/abs/1708.02002
7. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks | IEEE Journals Magazine | IEEE Xplore." https://ieeexplore.ieee.org/document/7485869 (accessed Apr. 30, 2022
8. "Computational Vision and Geometry Lab." https://cvgl.stanford.edu/projects/uavdata/ (accessedApr.30, 2022).
9. "Multi Object Tracking with UAVs using Deep SORT and YOLOv3 Retina Net Detection Frame- work | Proceedings of the 1st ACM Workshop on Autonomous and Intelligent Mobile Systems." https://dl.acm.org/doi/abs/10.1145/3377283.3377284 (accessed Apr. 30, 2022).
10. "VisDrone – Vision Meets Drones: A Challenge." http://aiskyeye.com/ (accessed Apr. 30, 2022).
11. T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," ArXiv161203144 Cs, Apr. 2017, Accessed: Apr. 30, 2022. [Online]. Available: http://arxiv.org/abs/1612.03144
12. Keras RetinaNet. Fizyr, 2022. Accessed: Apr. 30, 2022. [Online]. Available: https://github.com/fizyr/keras-retinanet

13. "Home - OpenCV." https://opencv.org/ (accessed May 02, 2022).
14. K. Team, "Keras documentation: Metrics." https://keras.io/api/metrics/ (accessed May 03, 2022).
15. "tfa. Metrics. MultiLabelConfusionMatrix| TensorFlow Addons," TensorFlow. https://www.tensorflow.org/addons/apidocs/python/tfa/metrics/MultiLabelConfusionMatrix (accessedMay03, 2022).