EVALUATION OF DRONE DETECTION TECHNOLOGIES

Phung Si Qi[1] , Tan Jia Hui, Joy[2] , Tay Jia Sheng[3], Jerry S/O Tamilchelvamani[4]
[1] Raffles Institution, 1 Raffles Institution Ln, Singapore 575954
[2] Nanyang Girls' High School, 2 Linden Dr, Singapore 288683
[3] Hwa Chong Institution, 661 Bukit Timah Road, Singapore 269734
[4] Defence Science and Technology Agency, 1 Depot Road, Singapore 109679

Abstract
From drones used in breathtaking light shows to drone models sold in toy stores, drones are ever-present in our society. While drones have their fair share of benefits, there are a host of problems as well, one of the most pressing ones being their usage in asymmetric warfare and criminal activities. It is thus crucial for authorities to detect and take down unauthorised drones in Singapore's airspace. This paper will evaluate the effectiveness of a drone detection model in various environments and its accuracy when tested with distractors.

Introduction
The Defence Science Technology Agency (DSTA) of Singapore has developed a computer vision model to detect drones for the protection of Singapore's airspace. In order to evaluate this model, we quantified its performance, identified shortcomings and re-trained new models to address those shortcomings.

Evaluation Process

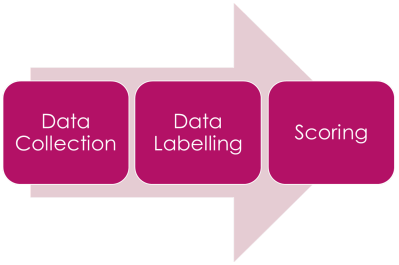


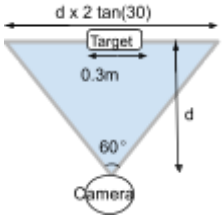
Figure 1. Flowchart of the project processes

The evaluation process is summarised in Figure 1. To assess the performance of the model, data was collected in step 1 and labelled in step 2. Subsequently, the model underwent testing using this dataset, and its performance was evaluated through scoring in step 3. The details of the three steps of Data Collection, Data Labelling and Scoring process are discussed below.

Data Collection
Test data comprising drones and distractors were collected. The drone data was deliberately made diverse, featuring drones at various distances and backgrounds. The data where the drones were far away tested the model's accuracy in detecting drones where details were limited while the data where the drones at different backgrounds tested the model's resiliency to noise. The distractor data was deliberately chosen based on objects that resembled drones or or commonly found together with drones. The distractors tested the model's propensity to be misled or fooled by similar features and biases that exist in the model due to errors made in selection and labelling of the training data. In step 1, a total of 12206 images were collected with the drone images consisting of 7755 images and the distractor images, the remaining 4451 images.

1. Drone data

After the drone images were collected, they had to be sorted based on distance. Figure 2 shows the derivation of formula (1), which was used to calculate the distance of the drone from the camera, d.



$$Fraction\ of\ target\ width\ to\ picture\ width\ (x) = \frac{0.3}{d * 2 * \tan(30°)}$$

$$Distance\ of\ drone\ from\ camera, in\ metres\ (d) = \frac{0.26}{x} \quad ------ \quad (1)$$

Figure 2. Derivation of formula (1)

The typical length of many commercial drones was found to be around 0.3m, and thus 0.3m was used as the length of the drones in the images. As the angle of the field of view for many modern video-capturing devices is 60°, it was assumed that the camera used to capture images had a field of view of 60° as well. Using these assumptions, formula (1) was obtained.

Formula (2) illustrates how x in formula (1) in Figure 2 is obtained, with the aid of an example in Figure 3. As shown in the bottom of Figure 3, the target width is 354 pixels while the image width is 1920 pixels. Hence, x = 354/1920. This value is then substituted into formula (1) to obtain the distance of the drone.

$$x = \frac{target\ width\ (px)}{picture\ width\ (px)} \quad ------ \quad (2)$$

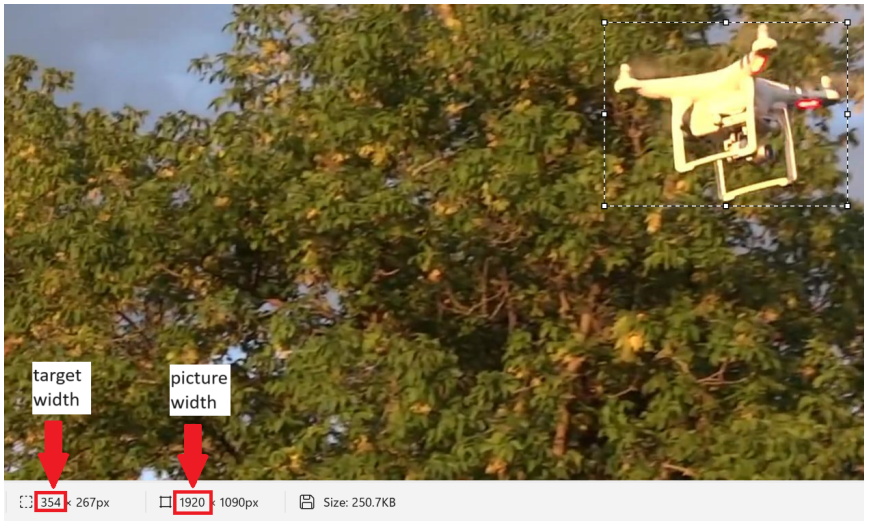


Figure 3. Illustration of how x is obtained

Based on the distances obtained, the images were sorted into three categories: a) less than 5m, b) between 5m to 10m and c) greater than 15m. Next, backgrounds were chosen based on perspective. In operations, the drones were likely to be viewed from below. In this situation, the background was likely to be the sky. Similarly, the background was either field (open space, trees) or urban (buildings, roads) when the drone was viewed from eye level or higher angles. Thus, the categories of 1) Sky, 2) Field and 3) Urban were chosen as well. This resulted in the drone data being categorised into a total of nine bins.

The volume of data for some of the bins was low due to the dearth of suitable data sets on the Internet. For example it was difficult to obtain data for Urban (d>15) and it originally had zero images.

Hence, synthetic data generation was utilised to supplement the test data. A tool based on the Unity game engine allowed 3D objects of interest to be rendered quickly against any background. Images of the scene could then be captured using a camera object.

Within the Unity workspace illustrated in Figure 4, a 3D drone model was imported and positioned against background images of Sky, Field, and Urban environments. The camera object was placed to capture drone images at specified angles and distances. Parameters such as drone size and position were also adjusted. By repeatedly randomising these factors, a large volume of images was quickly generated and sorted into folders. This tool proved much more efficient and effective in generating large volumes of varied data compared to manually sourcing for images from the Internet.

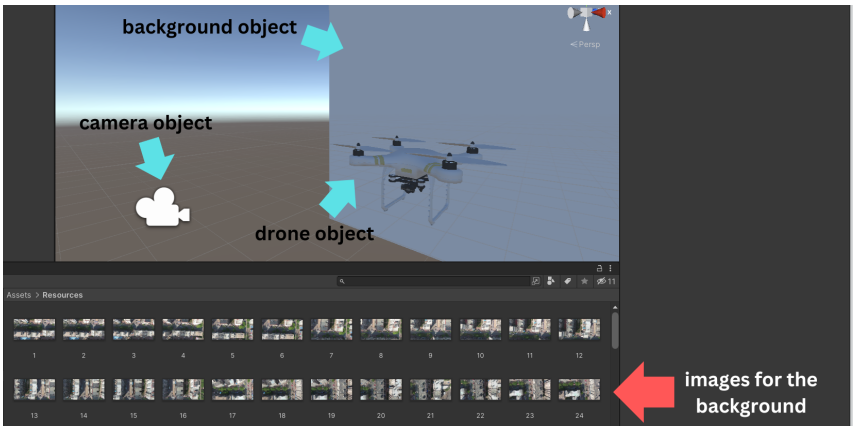


Figure 4. Unity workspace for synthetic image generation

The collected data encompassed mainly two mediums: images and videos. Managing images was straightforward, but processing videos required additional steps. To extract images from videos, the tool FFmpeg was employed, using the command shown in Figure 5 below. Although 30 frames could be extracted per second, the objects and background seldom differed by much between every frame, hence only the first frame of every second was extracted. Parameters such as the duration of the video were also set in order to select suitable data.

```
ffmpeg.exe -ss <start time>-t <end time> -i <folder path>\input_file.mp4 -qscale:v <int
to indicate img quality> -r <refresh rate> <folder path>\<image name>-%4d.jpeg
```

Figure 5. Command to extract images from videos

2. Distractor data

Images of six different types of distractors were collected, namely helicopters, airplanes, birds, buildings, cars and people. These were objects typically expected to be in the same environment as drones, resulting in potential false positives.

It was essential to ensure that the size of distractors in the images were not too small or large as the model's performance could be sensitive to this factor. Formula (2) above was applied to the distractor data to obtain x, and only images where x was in the range of 0.4 to 0.7 were selected. This criteria retained distractors that were recognizable and yet still had the potential to be mistaken as a drone.

3. Data distribution

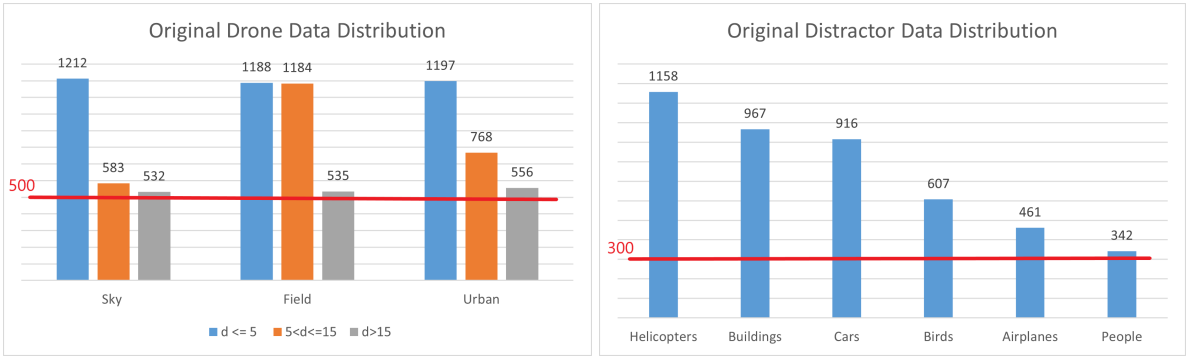


Figure 6. Drone and Distractor Data Distribution

Originally, the distribution of data across the bins was uneven, as shown by the bar charts in Figure 6, where there was more data for d<=5 and helicopters. To ensure fair evaluation and benchmarking, we sought uniform distribution across the bins. The drone data was capped at 500 per bin and the distractor data was capped at 300 per bin, as shown by the red lines in Figure 6. These numbers were chosen based on the smallest volume of data in each bin. Images were randomly chosen to avoid biases.

Data Labelling

To score the model accurately, it was imperative to acquire data regarding the size and position of drones within the images. Data labelling involved the drawing of bounding boxes around drones, generating coordinates known as groundtruths. This process was executed using the Computer Vision Annotation Tool (CVAT), depicted in Figure 7. Subsequently, the scoring process involved comparing these groundtruths with the detections generated by the DSTA model.

Figure 7. CVAT annotation workspace

For synthetically generated images, a script was used to automatically generate the groundtruths, since parameters such as drone size could be obtained from the Unity tool.

Testing and Scoring

The DSTA model was run against the drone data collected in step 1 of the evaluation process and the model's results were compared with the groundtruths generated in step 2 (Figure 1). The DSTA model was run with a confidence threshold of 0.5, as seen in Figure 8. The confidence threshold determined the minimum certainty at which the model labels a detection as a drone. A higher threshold allowed the model to be more selective in its detections, resulting in fewer false positives but potentially more false negatives.

```
python detect.py --weights drone_detect.pt --conf-thres 0.5 --img-size 640 --save-txt
--save-conf --source <images folder path> --project <output folder path>
```

Figure 8. Command to run DSTA model

While various metrics were available to measure the performance of computer vision models, the AP (Average Precision) score was used due to its popularity and ease of calculation. The AP score ranges from 0% to 100%, with a higher score indicating higher accuracy.

An object detection metrics tool [1] was used to calculate the AP score. By checking the model's results against the groundtruths, the tool calculated and reported the AP score. Figure 9 shows the command used to run the object detection metrics tool.

```
python pascalvoc.py -gt <groundtruth folder path> -det <detection folder path> -sp
<output folder path>
```

Figure 9. Command to run object detection metrics tool

For distractor data, the false positive rate of the model was calculated using the formula below. This indicated when a distractor was wrongly detected as a drone.

$$False\ positive\ rate\ (distractors) = \frac{Number\ of\ images\ with\ drones\ detected}{Total\ number\ of\ images} * 100$$

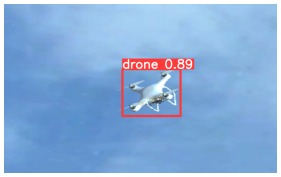
Evaluation of Performance of DSTA Model

Drone Data Results

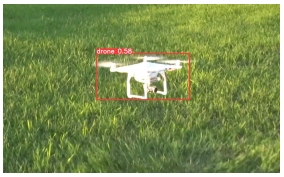
Distance (m) / Background	Sky	Field	Urban
d <= 5	78.9%	52.5%	23.6%
5 < d <= 15	35.4%	6.76%	1.16%
d > 15	3.19%	0.00%	0.12%

Table 1. AP Scores of DSTA model across 9 bins, each containing 500 images

From Table 1, at d <= 5, the AP for Sky was highest at 78.9%, followed by 52.5% for Field, and Urban had the lowest AP of 23.6%. The same trend was observed when 5 < d <= 15. This suggested that the model performed best when the drone was against a Sky background, followed by Field, then Urban. An explanation could be that the Sky background was the least noisy with least objects present in the images. In an Urban environment, objects such as buildings and roads could have reduced the clarity of the outline of the drone, thus resulting in a poorer performance.



a. d<=5 against Sky



b. d<=5 against Field

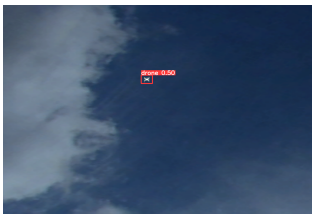
c. d<=5 against Urban

Figure 10. Sample images compared by background

Regardless of background, the AP score decreased as distance of the drone from the camera increased. For example, the AP score for d <=5 in the Sky background was highest at 78.9%, decreasing to 35.4% at 5 < d <= 15, and 3.19% when d > 15. Clearly, the further the distance, the worse the model performed. The model likely struggled with feature extraction of objects at further distances due to insufficient details.

a. d<=5 against Sky

b. 5<d<=15 against Sky



c. d>15 against Sky

Figure 11. Sample images compared by distance

At distance d>15, the AP scores were considerably low at 3.19%, 0.00% and 0.12% for Sky, Field and Urban respectively. Evidently, the model performed poorly against drones that were far away regardless of background.

a. d>15 against Sky

b. d>15 against Field

c. d>15 against Urban

Figure 12. Sample images where d>15

Distractor Data Results

Distractors	Helicopters	Planes	Birds	Cars	Buildings	People
False positive rate	88.0%	79.3%	8.00%	7.00%	3.33%	0.33%

Table 2. False positive rate by distractor type

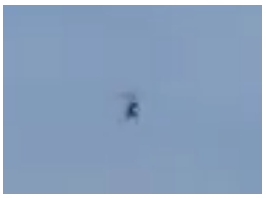
From Table 2, helicopters and planes had a false positive rate of 88.0% and 79.3% respectively, which was significantly higher than that of other distractors (between 8.00% and 0.33%). This indicated that the model was least adept at distinguishing helicopters and

airplanes, likely due to their shapes and features resembling that of drones. Further analysis of each type of distractor with notable observations is illustrated below.

a. Helicopters

Comparing across different helicopter images, the model had a higher percentage of false positives for higher quality images. A possible reason could be that a higher resolution allowed for helicopter features resembling drones, such as propellers, to be more clearly seen and hence were falsely detected as drones.

a. High quality image detected



b. Low quality image undetected

Figure 13. Sample images of quality differences in helicopter images

b. Planes

The type of plane influenced the false positive rate. Propeller planes that were similar in size to drones yielded a low percentage of false positives, while larger commercial planes yielded a high percentage. Logically, larger commercial planes would yield more true negatives as their shapes were more distinctly different from drones, yet this was not the case. It was possible that there was a lack of commercial airplanes in the training data, leading to the poor performance of the model.

a. Commercial planes detected

a. Propeller planes undetected

Figure 14. Sample images for comparison of type of plane

c. Buildings

Rather than detecting the whole building as a drone, the model detected parts of buildings as false positives. Namely, parts of buildings which had white parts scattered on them were more commonly detected, compared to parts of buildings which only had one uniform colour. The scattered white parts were likely more similar to drones due to their shapes and colour.

a. White parts of buildings detected

b.Uniform colour of buildings undetected

Figure 15. Sample images of colour differences in buildings

Benchmarks

Previously, in order to calculate AP scores for each of the 9 bins, 500 images were used for each bin. However, using such a distribution was not reflective of operational conditions, where it is more likely for the drone to be at further distances from the camera. Therefore, a new distribution was used, as shown by the green lines in Figure 15, where images of d<=5 were reduced from 500 to 111 in each bin, allowing images of 5<d<=15 and d>15 to constitute a larger percentage of the total images. (500 images constituted 15% of total images while 111 images constituted 3.33%.) This allowed for a more representative overall AP score based on the use case of the DSTA model.

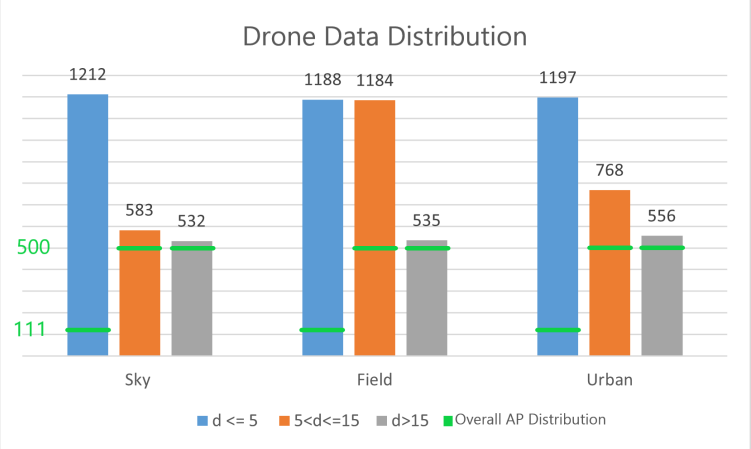


Figure 16. Drone Data Distribution

In addition to the overall AP score for drone data, 2 other benchmarks were used. They were the false negative rate (drone data) and the overall false positive rate (distractor data). The false negative rate indicated how often a drone went undetected, while the overall false positive rate indicated how often the model detected all the distractors as drones. The formulas used are shown below.

$$False\ negative\ rate\ (drones) = \frac{Number\ of\ images\ with\ drones\ undetected}{Total\ number\ of\ images\ with\ drones} * 100$$

$$Overall\ false\ positive\ rate\ (distractors) = \frac{Sum\ of\ false\ positive\ rate\ of\ all\ distractors}{6}$$

Benchmark	Results
Overall AP score (Drones)	11.5%
False negative rate (Drones)	85.6%
Overall false positive rate (Distractors)	31.0%

Table 4. Benchmark results of DSTA model

The DSTA model had a relatively low AP score of 11.5% and a high false negative rate of 85.6%. The low accuracy could be due to the training data set containing more images of

larger drones while a large proportion of test data comprising images of smaller drones. The high confidence threshold could also have influenced these results. Retraining was thus explored to improve the results.

Retraining

The DSTA model was originally trained on the nano version of the YOLOv5 model. Research shows that larger models have higher accuracy in detection [2]. This is due to more layers and parameters being present in larger models, as illustrated in Figure 16. More features of the image would be passed into the model and be more rigorously analysed. Amongst the different model sizes, the small and large models were selected for training. The rationale of choosing a small model was that it is slightly larger than the nano model with similar inference speed. As for the large model, a model with a more extensive architecture was needed, but since the XLarge model could exceed computing resources and had an undesirable training speed, the large model was used instead.

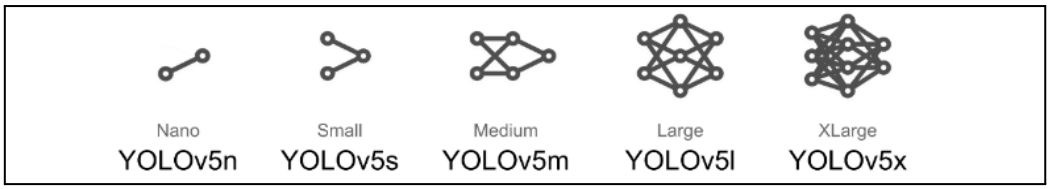


Figure 17. Visual comparison of YOLOv5 models [2]

To understand the effect and impact of different YOLOv5 architectures, the training data was kept the same when training the small and large models. The commands shown in Figure 17 were used.

```
python train.py --img 640 --batch 4 --epochs 50 --data drone_training_data.yaml --weights yolov5s.pt (small model)

python train.py --img 640 --batch 4 --epochs 50 --data drone_training_data.yaml --weights yolov5l.pt (large model)
```

Figure 18. Terminal commands used for training

After training, the new models were tested on the aforementioned benchmarks. It was hypothesised that the overall AP score will increase with model size, while the false negative rate and false positive rate will decrease.

Test/Model	Nano (DSTA)	Small	Large
Overall AP score (drones)	11.5%	32.1%	46.4%
False negative rate (drones)	85.6%	67.6%	53.3%
False positive rate (distractors)	31.0%	33.2%	34.5%

Table 5. Performance of Models against Benchmarks

Results showed that as the size of the model increased, the overall AP score increased and the false negative rate decreased. This indicated that larger models could detect drones more accurately as hypothesised. Unexpectedly, there was a slight increase in the false positive rate. This suggested a potential decline in the model's capacity to distinguish distractors from drones. This could be because the same training dataset size was used for larger models instead of using larger datasets relative to their size, potentially resulting in overfitting of data [3]. However, the increase in false positive rate was only by a mere 3% from nano to large, compared to the more significant 35% increase in AP score and 32% decrease in false negative rate. Therefore, it can be concluded that larger models are still generally more accurate in drone detection.

A drawback of larger models was that their processing speed lagged significantly behind that of smaller models. Specifically, the large model exhibited a Non-Maximum Suppression (NMS) speed per image four times slower than the original nano model. As such, the nano model may be preferred in scenarios where speed is a primary consideration. In the context of the DSTA model, the ability to rapidly detect drones is crucial for promptly responding to potential threats. In scenarios where speed is less critical, a larger model can be employed to achieve more precise detections.

Conclusion

In summary, the model performed better in Sky backgrounds than for Field and Urban backgrounds. Furthermore, it demonstrated significantly higher accuracy in detecting images of larger drones. After retraining, the model experienced a boost in AP score, showing that computer vision models with a larger architecture generally leads to more accurate drone detection, although there is a trade off of speed for accuracy.

Takeaways

Overall, there were several takeaways from this research. Firstly, establishing a consistent framework is key when comparing across different categories. For example, in order to ensure a fair comparison across distractors, only images with distractors of the same size range and background were included. Next, synthetic data can help overcome data shortages for certain categories, such as Urban environments. Lastly, it is paramount that training data size is relative to model size [4]. In order to better improve the performance of larger models, a larger training dataset should be used. Rather than reusing the same training dataset, a possible extension would be to curate new datasets relative to the different model sizes.

Acknowledgements

We would like to thank Jerry and Ling Ling for mentoring us in our research. They have provided much advice and guidance which made this project possible. We also like to thank Dylan, Daryl, Cheng Ee, Zec and Kevin for working together with us.

References

[1] Padilla, Rafael and Passos, Wesley L. and Dias, Thadeu L. B. and Netto, Sergio L. and da Silva, Eduardo A. B. (2021). Object Detection Metrics. Retrieved from https://www.mdpi.com/2079-9292/10/3/279/pdf.

[2] Noh, C.-M., Jang, J.-G., Kim, S.-S., Lee, S.-S., Shin, S.-C., & Lee, J.-C. (2023, April 21). A study on the optimization of the coil defect detection model based on Deep Learning. MDPI. Retrieved from https://www.mdpi.com/2076-3417/13/8/5200.

[3] Jallad, K. A., Aljnidi, M., & Desouki, M. S. (2020, August 31). Anomaly detection optimization using big data and deep learning to reduce false-positive - journal of big data. SpringerOpen. Retrieved from https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00346-1.

[4] Dorfman, E. (2022). How much data is required for machine learning?. PostIndustria. Retrieved from https://postindustria.com/how-much-data-is-required-for-machine-learning/.