TARGET RECOGNITION IN SAR IMAGES WITH LOW-SWAP PROCESSING HARDWARE

R.O. Lane QinetiQ, Great Malvern, UK
W.J. Holmes QinetiQ, Great Malvern, UK
T. Lamont-Smith QinetiQ, Great Malvern, UK

1 INTRODUCTION

Synthetic aperture radar (SAR) is a useful surveillance sensor due to its high resolution, long range, and day/night capability. Recent progress in machine learning (ML) means more processing tasks can be automated than previously. Target detection and classification is of particular interest. However, much modern ML research relies on powerful processing hardware, making use of cloud processing and data centres. This type of system can be used in strategic settings, where data is sent to a central point, but there is a requirement for low size, weight, and power (SWAP) solutions where on-board processing is needed to reduce dependency on tactical comms channels, which may have low bandwidth or be denied in combat situations. For example, instead of transmitting a high bandwidth raw data stream, just metadata for detected objects could be transmitted. We compare the performance of three object detection and classification algorithms applied to SAR data on low-SWAP hardware. The remainder of this paper is organised as follows. Section 2 describes algorithm and hardware selection. Section 3 describes the data collection process. Sections 4 and 5 describe the processing methodology and compare performance of the algorithms. Finally, conclusions are given in section 6.

2 ALGORITHM AND HARDWARE SELECTION

The aim of target detection and classification, also known as object detection in the ML literature, is to process an image to produce bounding box coordinates for each object, a class label, and a confidence score. Post processing can be applied, such as thresholding confidence scores to trade true detection rates against false alarms, or tracking bounding boxes between successive video frames to provide smoother estimates.

A literature review was conducted to find object detection algorithms appropriate for application on low-SWAP hardware[1]. Three algorithms were identified as being particularly suitable: You Only Look Once (YOLO)[2], RetinaNet[3], and EfficientDet[4]. These algorithms were designed to attain high accuracy while having low computational cost[5]. Various versions of YOLO have widely been used as baselines for comparison with other algorithms. This paper examines the small (s) and extra-large (x) versions of YOLOv5, and compares the performance of these models with RetinaNet and EfficientDet, using different hardware platforms when applied to SAR data.

A review of low-SWAP edge computing devices was conducted[1]. The aim was to select a particular single-board computer (SBC) to demonstrate operation of the algorithms. An SBC is a complete computer built on a single circuit board, with microprocessor(s), memory, input/output, and other features of a functional computer. SBCs have a high level of integration and low component counts. This gives them a lower SWAP than comparable multi-board computers. The Nvidia Jetson Xavier NX was selected as it has a good trade-off compared to other options: small size (70 mm x 45 mm); low mass (24 g); medium power consumption (10-15 W); high processing ability (6000 GFLOPS with 384 CUDA cores); low cost; and good support. A developer kit form factor of the board was used, as the aim of this work was to rapidly test a proof of concept, rather than develop a productionized system for deployment. It is expected that bespoke hardware solutions could further improve the power efficiency of the system. Computation of power efficiency for each algorithm was out of scope for this work.

3 SAR DATA AND SIMULATION METHODS

Comprehensive SAR data sets with annotated objects in large scenes were not available. Therefore a data set was made by combining available background scene data with target vehicle information from the public release of the moving and stationary target acquisition and recognition (MSTAR) data set[6,7]. The output was a new data set of large scenes containing multiple target vehicles. This hybrid approach of merging datasets based on real measurements has successfully been used for other sensing modalities, such as infrared[8] and measurements of electrical device current usage[9].

The background scene data was selected as follows. QinetiQ owns a number of SAR datasets collected under various UK MOD programmes. A particular set of clutter images, with aerial landscape views containing no targets, was identified for use. This dataset was collected during field trials in 2010 using the PodSAR radar mounted on a Tornado aircraft. The images primarily consist of fields, trees, hedges, bushes, and roads, with a few buildings or structures. Seven large images were identified as shown in Table 1.

Image Name	Width (pixel)	Height (pixel)	Pixel Spacing (m)	Radar Resolution (m)	Train/Test
4005 Fibua	2622	11978	0.261	0.3	Train
Larkhill 1	3200	4162	0.255	0.3	Train
Larkhill 2	3366	4024	0.255	0.3	Train
Larkhill 3	3158	4098	0.255	0.3	Validate
Stonehenge	2010	12070	0.263	0.3	Train
Stonehenge 2	1752	5580	0.263	0.3	Train
6006	23940 (9*2660)	14204	0.376 x 0.407	0.428 x 0.466	Test

Table 1: QinetiQ SAR clutter dataset; pixel spacing and radar resolution are listed as width (azimuth) x height (range).

To ensure that no information about the background can leak into decisions on whether a target is in the scene, the first six images were used in training and the final image used in testing. One of the training images was designated to provide the validation set to optimize algorithm hyper-parameters. The test image has a higher contrast than the training images. It also exhibits a meandering beam pattern as the strip-map collection took place over a distance of about 9 km. These differences ensure that reported results are not over-trained to particular image characteristics. As the test image is very wide, it was split into nine non-overlapping sub images before further processing.

The public release version of the MSTAR dataset used for this work contains target images for: one of three T72 main battle tanks (MBTs); one of three BMP2 armoured personnel carriers (APCs); or a single BTR70 APC. The BMP2 vehicles are different serial numbers of the same variant. The T72 MBTs are different variants. The images are 128x128 pixels in size with a radar resolution of 0.3048 m (1 foot)[6] and pixel spacing 0.202 m in range and cross range[10]. Training data were collected at an elevation of 15° and test data at 17°.

The background and target datasets were merged by inserting small target image chips into larger background images. Train/test set chips were respectively inserted into train/test set backgrounds to ensure the integrity of the train/test split. The background images were manually analysed to find suitable areas where targets could be inserted. Before insertion, the following processes were applied to target image chips. First, rotation by 90° so shadows point down to match the background. Second, application of an alpha-channel transparency window. This is a full-amplitude circle centred on the chip centre, with a radius of 25 pixels, and a linearly decreasing alpha value (opaqueness) in a radial direction from the circle edge to the chip edge. This reduces insertion artefacts, making the final image more realistic. Third, the amplitude of the chip was scaled so the non-target mean matches the local mean of the background. In addition to target chips, clutter chips from the MSTAR dataset were randomly inserted into background scenes. This was done to ensure the classifier could not use any subtle insertion artefacts to help with the target detection process. Shadows in the MSTAR data are more pronounced than those in the QinetiQ clutter data. To ensure the two datasets match, noise of an appropriate power level was added to the target chips prior to insertion. Bounding box annotations were fixed to be 32x32 pixels in size, centred on the inserted chip centre.

The dynamic range of radar images is inherently high, as returns from targets and buildings can be orders of magnitude brighter than those from fields and trees. Operationally, analysts adjust the brightness and contrast of screens adaptively to understand the scene. For fields, higher contrast is required, but for buildings, lower contrast is needed. In this work, image amplitudes were clipped so pixels brighter than a threshold were set to the threshold. The thresholded image was then amplitude-scaled to fit the PNG image file format dynamic range. The threshold was set to the 98th percentile for training images and 95th for test images. This partly accounts for differences in the images but also ensures the reported algorithm performance is not over-trained to a particular threshold.

The above process produces images containing background scenes and targets. The images have different dimensions from each other. The object detection algorithms in this paper scale images so they are a standard size before input. This is appropriate for normal photographs, which are usually used with such algorithms, because objects appear as different sizes depending on how close they are to the camera and the level of zoom. Examples of objects at multiple scales are then required in training. However, the physics of SAR image generation creates pixels, and hence targets, at the same size regardless of the sensor-target distance. This means the physical size of objects is encoded in the image. This is useful information for discriminating between targets and would be lost if whole images were scaled on input to the neural net. To avoid this, large images were "tiled" to produce constant size images of 768x768 pixels. In total, there are: 847 training images, each containing zero, or more, inserted target and clutter chips (718 images with targets and 129 without); 153 validation images (132 with targets, 21 without); and 315 test images (268 with targets, 47 without). The images all have a pixel spacing of 0.26 m and a radar resolution of 0.3 m. The number of examples of each class in each dataset is shown in Table 2. It is noted that the scale issue has been addressed elsewhere[11] using sliding windows instead of tiles to process large images, but that network is only applicable to fixed-size SAR images. The amount of SAR data available to train the algorithms was somewhat limited compared to open crowd-sourced photograph datasets usually used to analyse deep learning algorithms. This is due to the manual nature of the radar data background and target merging process. With further work, improvements could be made to allow a greater degree of automation and increase the data volume.

Target	Training	Validation	Test
BMP2	2301	440	372
BTR70	756	192	91
T72	2110	417	307

Table 2: Number of examples of each class in each dataset.

4 METHODOLOGY

The algorithms were trained and fine-tuned on a G4dn.xlarge elastic compute cloud (EC2) instance on Amazon Web Services (AWS). For each algorithm, the weights were initialized from published pre-trained models trained on photographs. It was surprisingly found that using pre-trained, rather than randomized, initial weights improved performance for all algorithms and datasets tested, even though the sensing modality of the pre-training photograph data is different to SAR. This is advantageous as there are many more open-source photographs than SAR images to pre-train models. The fine-tuning process adapts the model to the characteristics of SAR images. The approach contrasts to another paper[11], which only uses SAR data and no pre-training. The post-processing step of non-maximum suppression was applied for RetinaNet and EfficientDet, with a maximum intersection-over-union (IOU) value of 0.3 to prevent multiple predicted boxes being produced for a single object. This step is inherently part of the YOLOv5 implementation.

To optimize results, a simple hyperparameter search was performed to obtain good values for the batch size and learning rate while optimizing the mean average precision metric on the validation set. Batch size is a hyperparameter that defines the number of images to process simultaneously before internal neural net weights are updated during training. When a large batch size is used, a more accurate estimate for the gradient can be made to improve the weights, as the model has seen more images and annotations. However, larger batch sizes typically mean a model trains slower as fewer updates are made to the weights per epoch. In addition, larger batch sizes require more memory than smaller ones. This was the defining factor in our case, due to limited memory on the EC2 instance used to train the model. The batch size was respectively set to 8, 4, 16, or 8 images for RetinaNet, EfficientDet, YOLOv5s, or YOLOv5x. The other hyper-parameter investigated was the learning rate. The algorithms use stochastic gradient descent to estimate the error gradient. Learning rate is the amount by which weights are updated along the direction of the error gradient. The rate defines how the model converges, with typical values being a small positive numbers less than 1.0. A small learning rate reduces the risk of unstable oscillations but does present problems around becoming stuck in local minima, and vice versa for the larger learning rate. Learning rate was only varied for RetinaNet and YOLOv5, as EfficientDet does not allow this to be changed by the user.

To assess model performance, multiple metrics were computed. Precision is the number of true positives divided by the total number positives (both true and false). Recall (also known as probability of detection) is the number of true positives divided by the number ground-truth objects. A single precision and recall pair is reported for each class with the values that maximize the F1 score, which is the harmonic mean of those two metrics. False positive count is a measure of how many objects were incorrectly identified as a particular class. False alarm count is the number of false detections of a particular class in empty images. False alarm rate is the false alarm count divided by the total area of empty images. The average precision for a single class is the area under the precision-recall curve. The mean average precision (mAP) is the mean of this metric across all classes.

A trained model can be used to process unseen data to produce bounding box predictions (detections), containing the coordinates of each box, object label, and a confidence score. The confidence detection threshold was investigated. Each bounding box proposed by the model is given a probability, or “confidence”, in the annotation predicted. These confidence values must be above the threshold to be accepted. A low threshold may allow too many false positives. A high threshold may lead to a higher false negative count, where many of the ground truth objects are missed.

5 RESULTS

The training duration and number of epochs for each algorithm was: RetinaNet (60 minutes, 21 epochs), EfficientDet (314 minutes, 25 epochs with a frozen backbone then 83 epochs with all parameters allowed to update), YOLOv5s (34 minutes, 100 epochs) and YOLOv5x (179 minutes, 100 epochs). Each trained model processed test data on the EC2 instance and the low-SWAP NX board. Full results of the experiments are in a detailed technical report[12]. The primary metrics are mAP and processing time per image. Results for the EC2 instance are shown on the left of Figure 1. Both RetinaNet and EfficientDet have larger or smaller variants of model than those tested here, allowing for trade between speed and accuracy. The line associated with each data point indicated by a marker is an estimate of what different accuracies or speeds could be expected with the variant models, using published data[4].

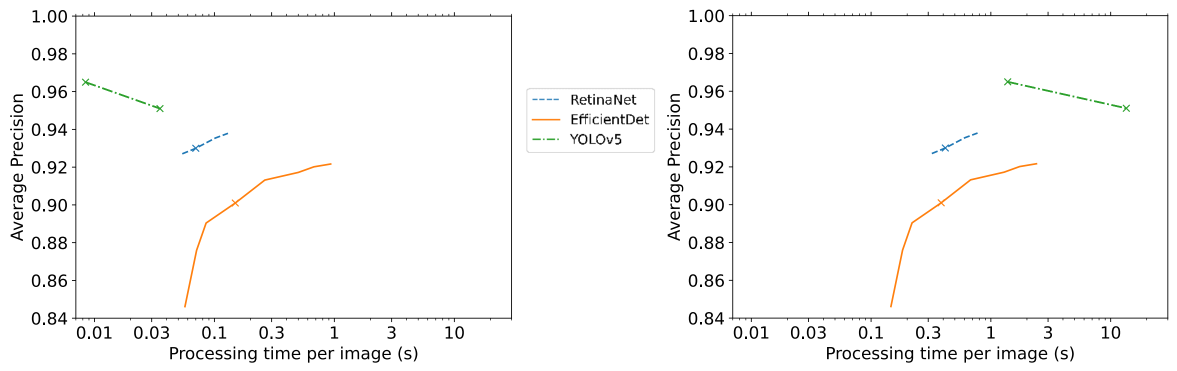
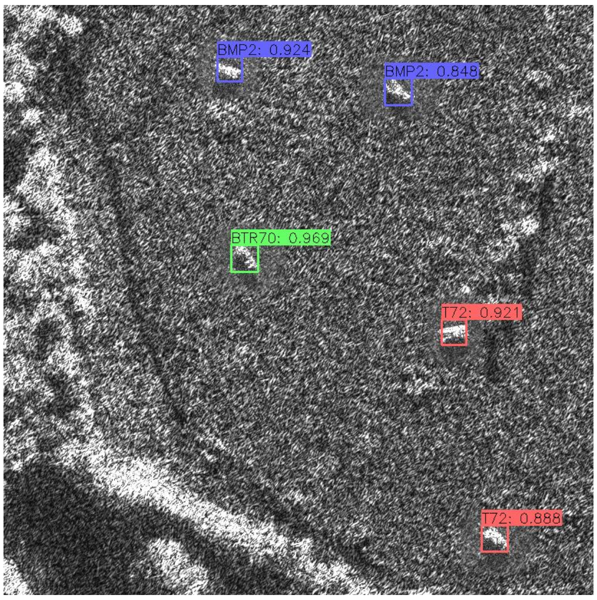


Figure 1: Accuracy-speed trade-off. Left: G4dn.xlarge instance on EC2. Right: Jetson Xavier NX.

The YOLOv5 performance curve is shown for the smallest (s) and largest (x) models. On the EC2 instance, YOLOv5 is more accurate and faster than the other algorithms. Results for the NX board are shown on the right of Figure 1. RetinaNet is more accurate than EfficientDet for the same processing speed, which mirrors the EC2 results. The speed for both algorithms is approximately two times slower on the NX board than EC2. YOLOv5 is more accurate, but slower than the other algorithms by a factor of about 4 for the small (s) model and 35 for the larger (x) model. Unlike on EC2, the PyTorch implementation of YOLOv5 was unable to make use of the OpenMP interface on the NX board, which significantly impeded the algorithm throughput. It is thought that if this issue were resolved the algorithm could be sped up.

Example RetinaNet detections are shown in Figure 2, and Table 3 shows average precision (AP) by class. While there is some variation between classes, the overall mAP of 0.93 suggests the model is able to correctly detect and classify most targets. The left of Figure 3 shows the precision-recall curve, and Table 4 gives additional metrics. The highest accuracy is achieved for the T72 class. Precision is lowest for the BTR70 class, which is consistent with the fact that this class suffered the most false positives, yet has by far the fewest instances out of the three classes. There is, however, generally not a significant problem with false detections, with only 35 false positives in 315 images and none in the 47 empty images. More information on classification errors is provided by the confusion matrix in Table 5. The BMP2 class is the most difficult to identify and has the highest proportion of missed detections (the ‘empty’ column), which is consistent with the relatively low recall performance. However, between-class errors are generally distributed across the possible combinations without any single obvious confusion being the most problematic.



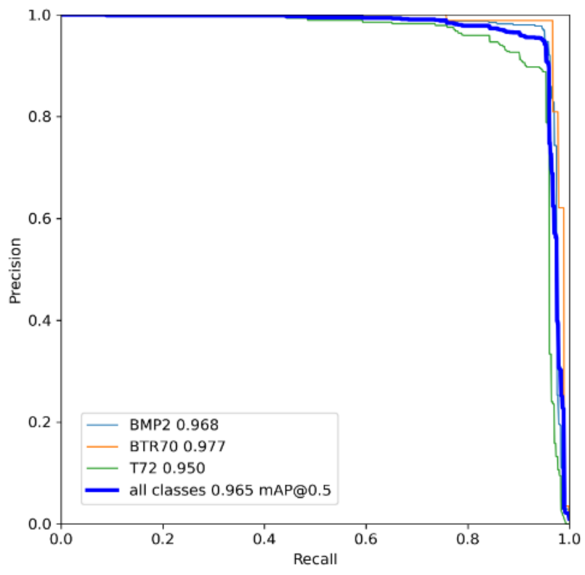


Figure 2: Left: Example RetinaNet detections. Right: Precision-recall curve for YOLOv5s.

Class Name	RetinaNet AP	EfficientDet AP
BMP2	0.895	0.893
BTR70	0.923	0.889
T72	0.973	0.921
Mean (mAP)	0.930	0.901

Table 3: Average precision of RetinaNet and EfficientDet.

Class Name	Precision	Recall	False positive count	False alarm per km^2
BMP2	0.979	0.858	7	0.00
BTR70	0.875	0.923	15	0.00
T72	0.957	0.951	13	0.00

Table 4: RetinaNet classification metrics for the test set with threshold set for F1 value of 0.576.

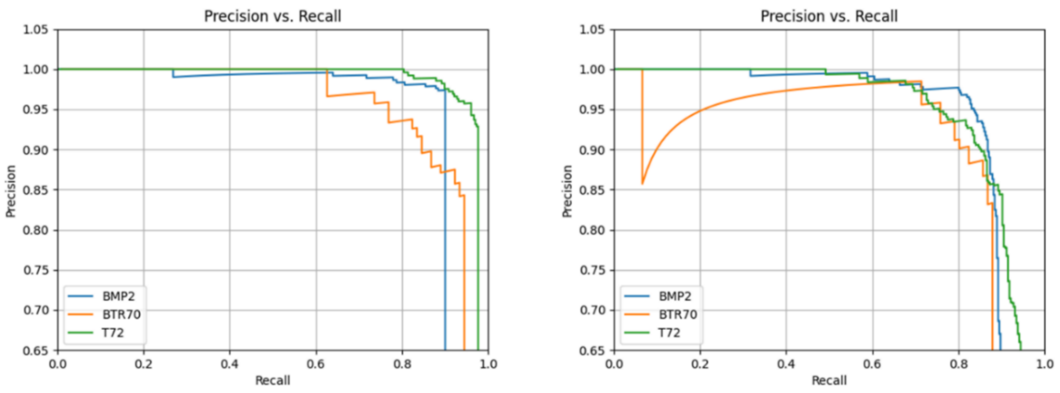


Figure 3: Precision-recall curve. Left: RetinaNet. Right: EfficientDet.

Predicted →		Empty	BMP2	BTR70	T72
Ground truth	**BMP2**	26	**319**	15	12
	BTR70	2	3	**85**	1
	T72	11	4	0	**292**

Table 5: Confusion matrix for RetinaNet on the test set with threshold set for F1 value of 0.576.

Average precision by class for EfficientDet is shown in Table 3. Good detection and classification has been obtained, with an overall mAP of 0.901, which is only a little lower than 0.93 achieved with RetinaNet. The precision-recall curve is shown on the right of Figure 3. The main part of the curve shows similar accuracy for all three classes, although the shape of the first part of the curve for the BTR70 class is somewhat atypical. This has arisen due to two high-confidence false-positive errors, which have brought down the precision at the low-recall end of the curve. The small total number of instances of this class mean that such irregularities are more likely to occur, as a single false positive tends to have a greater effect on the precision values than when there are more instances (and hence a larger number of potential true positives). Table 6 gives more detailed classification metrics. There are only a few false positive errors, except for the T72 class, which accounts for 82 out of the 99 such errors. Precision is generally better than recall, which is explained by the relatively high proportion of missed detections that can be seen from the confusion matrix shown in Table 7. The issue with missed detections, rather than between-class confusions, is the main cause of the worse recall performance than was seen with RetinaNet.

Class Name	Precision	Recall	False positive	False alarm per km^2
BMP2	0.974	0.804	8	0.00
BTR70	0.903	0.824	9	1.60
T72	0.898	0.860	82	9.05

Table 6: EfficientDet classification metrics for the test set with threshold set for F1 value of 0.272.

Predicted →		Empty	BMP2	BTR70	T72
Ground truth	**BMP2**	58	**299**	4	11
	BTR70	14	2	**75**	0
	T72	23	5	0	**279**

Table 7: Confusion matrix for EfficientDet on the test set with threshold set for F1 value of 0.272.

The metrics reported above for RetinaNet and EfficientDet were computed independently of the open-source TensorFlow implementation of those libraries. The standard YOLOv5 implementation is based on PyTorch and it would require significant work for independent metrics to be computed for this framework. Therefore self-reported results were used and not all metrics are available. It is recommended in future work that all metrics be computed and verified independently. Classification metrics for the YOLOv5s variant are shown in Table 8. Performance is good for all classes, including BTR70, for which there are many fewer examples than the other two classes. The mAP of 0.965 for YOLOv5s is higher than 0.93 for RetinaNet and 0.901 for EfficientDet. More detail of the performance can be seen in the precision-recall curve in Figure 2 and the confusion matrix in Table 9. There are most confusions for T72, with some misrecognitions as BMP2 and more false-negative errors than for the other classes. The results for YOLOv5x have a similar per-class pattern and are therefore not shown here; the overall mAP was 0.951, which is only slightly worse than 0.965 for YOLOv5s.

Class	Precision	Recall	mAP@0.5
BMP2	0.978	0.946	0.968
BTR70	0.964	0.967	0.977
T72	0.887	0.950	0.950
Mean	0.943	0.965	0.965

Table 8: YOLOv5s classification metrics. Confidence threshold set to default of 0.25, and IOU to 0.5.

Predicted →		Empty	BMP2	BTR70	T72
Ground truth	**BMP2**	14	**352**	3	3
	BTR70	2	1	**88**	0
	T72	21	24	0	**257**

Table 9: Confusion matrix for YOLOv5s on the test set.

Additional vehicle detection experiments were carried out, by merging all three vehicles into a single “target” class. These experiments investigated two concerns. First, limited SAR image fidelity means it can be difficult even for a trained person to distinguish different classes of vehicle. It is possible that better detection performance might be achieved by not attempting classification simultaneously. This hypothesis was tested by comparing detection performance of single-class models with that achieved by the multi-class models. Secondly, by measuring performance on the validation data, which includes inserted clutter chips as well as inserted targets, it was possible to assess the extent to which models were influenced by insertion artefacts, due to the way the simulated data had been created. Poorer performance on validation than test data, with more false alarms, would indicate the model erroneously detects clutter chips as targets.

Models were trained for RetinaNet and EfficientDet. The experiments used the optimal training settings from the multi-class experiments and no further parameter optimisation was performed. Both sets of models converged during training much faster for this simpler detection task than they did for the three-class problem. For RetinaNet, the selected models trained in just 6 epochs over 16 minutes, compared with 21 epochs over 60 minutes for three classes. For EfficientDet, only the first stage of training was used, with 21 epochs over 28 minutes required (compared with 25 epochs, followed by a further 83 for the three-class models over a total of 314 minutes). For both RetinaNet and EfficientDet, the training time per epoch was similar to that seen with the multi-class models, but total training time was shorter due to the smaller number of epochs needed for the models to converge.

The mAP for one-class RetinaNet (RN) and EfficientDet (ED) models was high at 0.997 and 0.974 respectively. High performance is evident from other metrics shown in Table 10. The table also compares one-class models with three-class models (by discounting any between-class confusions). For both model types the one-class models are better than the multi-class ones. Results on the validation set are better than the test set. This difference is probably due to greater similarity of the training data to the validation data. Table 11 shows a comparison of false-positive errors for the validation and test sets. For both RetinaNet and EfficientDet, the false-positive rate for the validation set is actually slightly lower than the corresponding figure for the test set. The fact that better performance was achieved on the validation data indicates that the models are not triggering solely on insertion artefacts, as clutter chips are present in the validation data but not the test data. Similar results and conclusions were obtained using YOLOv5 s and x variants.

Model type	Dataset	Precision	Recall	False positive	False alarm per km^2
RN 1-class	Validation	1.000	0.999	0	0
RN 1-class	Test	0.993	0.991	8	1.60
RN 3-class	Test	0.952	0.949	35	0
ED 1-class	Validation	0.967	0.954	34	0
ED 1-class	Test	0.905	0.964	78	0
ED 3-class	Test	0.868	0.848	99	10.64

Table 10: Accuracy of one-class and multi-class models at optimum F1 for vehicle detection.

Dataset	Total no. of images	RetinaNet FP per km^2	EfficientDet FP per km^2
Validation	153	0.000	5.55
Test	315	0.625	6.20

Table 11: Comparison of false positives (FP) in test and validation sets.

6 CONCLUSION AND ACKNOWLEDGMENT

Three object detection algorithms, RetinaNet, EfficientDet, and YOLOv5 have been analysed. In principle, they can be applied to any type of image as long as suitable training data are available. Successful application of the algorithms to SAR data has been demonstrated on low-SWAP hardware. The novelty of the work is that the algorithms are based on deep learning, which traditionally requires more powerful hardware. When tested on a powerful EC2 instance, YOLOv5 was the best algorithm in terms of both accuracy and speed. On the off-the-shelf Nvidia Jetson Xavier NX low-SWAP device, RetinaNet and EfficientDet produced operationally useful throughput with reasonable accuracy. YOLOv5 had the better accuracy but did not achieve high throughput due to implementation issues. One-class models gave better detection performance than three-class models, suggesting that better overall detection and classification performance might be achieved using a traditional two-stage process of detecting vehicles, then classifying them separately. Application of the algorithms should enhance the autonomous capability of un-crewed airborne systems to detect objects of interest, communicate this information efficiently, and navigate appropriately according to mission objectives. We thank Adam Wragge and Dylan Rivers for help with setup of the object-detection processing pipeline, Stuart Bertram for NX board setup, and Alan Blake for the supply of SAR data.

7 REFERENCES

1. A. Hadland, "AI analysis using low-SWAP hardware: literature review", QinetiQ/20/02995, Sep. 2020.
2. D. Thuan, "Evolution of YOLO algorithm and YOLOv5: the state-of-the-art object detection algorithm", Bachelor's Thesis, Oulu University of Applied Sciences, 2021.
3. T.Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, "Focal loss for dense object detection", IEEE Int. Conf. on Computer Vision, Oct. 2017.
4. M. Tan, R. Pang, Q.V. Le, "EfficientDet: scalable and efficient object detection", arXiv:1911.09070, Apr. 2020.
5. R.O. Lane, A.J. Wragge, W.J. Holmes, S.J. Bertram, T. Lamont-Smith, "Object detection in EO/IR and SAR images using low-SWAP hardware", Sensor Signal Processing for Defence, Sep. 2021.
6. T. Ross, S. Worrell, V. Velten, J. Mossing, M. Bryant, "Standard SAR ATR evaluation experiments using the MSTAR public release data set", Proc. SPIE 3370, Sep. 1998.
7. R. Schumacher, K. Rosenbach, "ATR of battlefield targets by SAR - classification results using the public MSTAR dataset compared with a dataset by QinetiQ, UK", NATO Symp. on Target Identification and Recognition Using RF Systems, Oslo, Norway, Oct. 2004.
8. C.P. Moate, S.D. Hayward, J.S. Ellis, L. Russell, R.O. Timmerman, R.O. Lane, et al., "Vehicle detection in infrared imagery using neural networks with synthetic training data", Int. Conf. on Image Analysis and Recognition, Póvoa de Varzim, Portugal, Jun. 2018.
9. R.O. Lane, "Electrical device classification using deep learning", Sensor Signal Processing for Defence, Sep. 2020.
10. C.T. Rupp, S.D. Halversen, L.J. Montagnino, C.L. Hebert, M.T. Young, M.L. Cassabaum, et al., "Analysis of spatially mismatched imagery for synthetic aperture radar ATR classification", Proc. SPIE 6967, May 2008.
11. S. Chen, H. Wang, F. Xu, "Target classification using the deep convolutional networks for SAR images", IEEE Trans. Geoscience and Remote Sensing, 54(8):4806-4817, Aug. 2016.
12. R.O. Lane, W.J. Holmes, A.J. Wragge, S.J. Bertram, T. Lamont-Smith, "AI processing final report", QinetiQ/21/00278/1.2, May 2021.