An Effective Machine Learning Approach to Detect Satellite Signals in Passive RF Space Domain Awareness Data

Kriti Tripathi, Travis Bessell, Thomas Q. Wang, Tim Spitzer
Clearbox Systems Pty. Ltd., Australia
sda@clearboxsystems.com.au

ABSTRACT

An increase in Resident Space Objects (RSOs) requires improved custody through tracking and space object characterisation. A technology that can assist with this is Passive Radio Frequency (RF). One challenge with Passive RF technology is determining if a data collection contains a satellite signal, which is often a Human-In-The-Loop (HITL) task. This work presents a two-staged approach to automating the HITL component of identifying satellite signals in both small and large bandwidth and time data captures, using: 1. Image Classification, and 2. Object Detection. A customised Convolutional Neural Network (CNN) is designed for the image classification stage that classifies a small bandwidth Power Spectral Density (PSD) data capture as containing a satellite signal or not. Several out-of-the-box object detectors such as Faster Region-based Convolutional Neural Network (RCNN), You Only Look Once (YOLO) and Single Shot Detector (SSD) are compared for the object detection problem whereby the detector determines the frequency and time window of one or more satellite signals in a large bandwidth and long duration PSD data capture. The results show that for the image classification problem, the custom CNN model presented in this paper outperforms an existing Energy Detection technique. Furthermore, for the Object Detection problem, SSD looks to be a promising technique for autonomously detecting satellite signals in large data captures.

1. INTRODUCTION

The recent increase in space traffic due to the rapid proliferation of satellites has necessitated advances in space surveillance technology for improved tracking and characterisation of space objects. Passive Radio Frequency (RF) is a relatively new technology compared with traditional sensors that perform space surveillance, including optical telescopes and radars. Passive RF sensors can assist in tracking and characterisation of active space objects through the RF transmissions that they emit [7]. There are several advantages of employing passive RF technology over other traditional surveillance sensors which include the fact that the technology is not limited by weather or the time of day and is also covert as it does not emit any signature. One major limitation is that the technology is limited to tracking satellites with active, detectable transmissions within the sensors' field of regard. The technology can complement the traditional sensors and ultimately provide additional information for improved custody and characterisation of space objects.

Passive RF data can be represented as Power Spectral Density (PSD) data which can be used to infer the characteristics and the state of a satellite. An existing technique for determining if a capture contains a signal within the PSD data is known as Energy Detection [11]. The Energy Detection based approach is one of the most common way of sensing signal in spectrum data by comparing the data with a threshold which is typically based on the noise floor of the data. Some of the challenges with energy detection is choosing the correct threshold but also the inability to differentiate between satellite signal and interference from terrestrial or other signals.

If no or limited prior knowledge is known about the satellite, such as its approximate orbit and transmission characteristics, a wide bandwidth and long duration capture using passive RF sensors can increase the probability of intercept of a transmission from the satellite. Satellites can operate over a wide range of radio frequencies in the same or closely adjacent frequency ranges to terrestrial signals and other satellites, making the identification and attribution of individual satellite signals challenging in these types of captures.

Individual satellite signal identification in a wide bandwidth and/or long duration capture, in the presence of other satellite signals, terrestrial signals and noise, is a laborious task which typically requires human input. Automation of

satellite detection in passive RF data can boost time and resource efficiency as well as lower misclassification due to human error. Passive RF sensor networks can make use of omni-directional antennas which allow for simultaneous satellites to be observed in a data capture. Automated processing can aid in detecting these multiple satellites without requiring any prior knowledge of transmission frequencies. Two ML applications are explored in this paper to automate this processing for captures of different sizes: image classification and object detection. Image classification is a machine learning technique where categorisation, pattern recognition and labelling of images is utilised to ascertain if an image belongs to one of the target classes [5]. On the other hand, object detection is a computer vision methodology that aims to concurrently solve the challenge of object classification and localisation [12]. Object detection can be used across multiple contexts such as videos, real-time or otherwise, as well as images.

This paper begins with a brief overview of the sensor network and the collected passive RF data in section 2. Section 3 and 4 detail the background and architecture for each of the image classification and object detection contexts. The results from experimentation with the two aforementioned techniques are discussed in section 5. Conclusions from the research and the results are drawn in section 6 and potential future work is addressed in section 7. References for the paper are listed in section 9 and additional materials supporting the research and results are collated in section 10.

2. PASSIVE RF SENSOR NETWORK

For this research, real data was captured from Clearbox Systems' SpaceAware passive RF sensor network which was ingested by the Machine Learning (ML) models. SpaceAware is an Australian sensor network that currently consists of three sensor sites located at Adelaide, South Australia, Canberra, Australian Capital Territory and Mingenew, Western Australia. The location of the sensor sites and their low earth orbit coverage at an altitude of 400km is shown in Fig. 1.

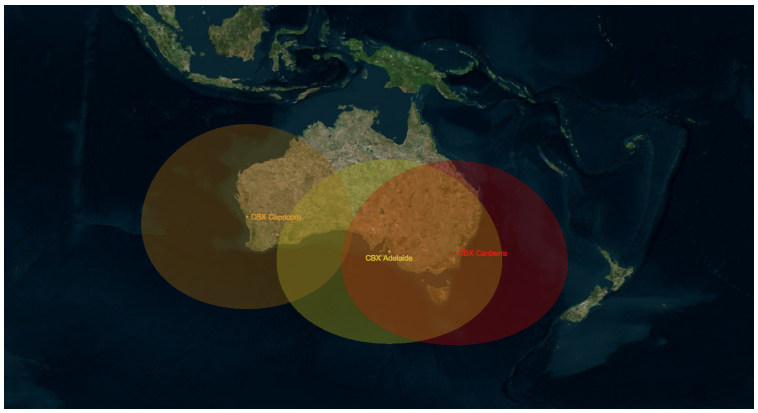


Fig. 1: Coverage of three sites within the SpaceAware RF sensor network.

Each sensor site consists of multiple RF front ends consisting of various antennas to cover the many frequency bands that satellites transmit on. Each sensor site also contains digitisation hardware and a workstation to perform the signal processing on the digitised data. Across the three sensor sites, a combination of directional and omni-directional antennas are used which cover VHF, UHF, S, X and Ku frequency bands. Each data capture that is collected from the SpaceAware network is processed to form a Power Spectral Density (PSD) plot that is subsequently saved as an image to be later used as the inputs to the ML algorithms.

2.1 Data Collection

The data captured from the SpaceAware network is initially in the form of In-phase and Quadrature (IQ) samples from the digitisation hardware. Welch's PSD method is used to convert the time series data into frequency domain so it can be represented as a PSD image with the x axis being time and y axis being frequency [3]. Fig.2 depicts an example of a capture from the SARAL satellite in the PSD format which is the format of the images that are used as training data for the Image Classification and Object Detection algorithms.

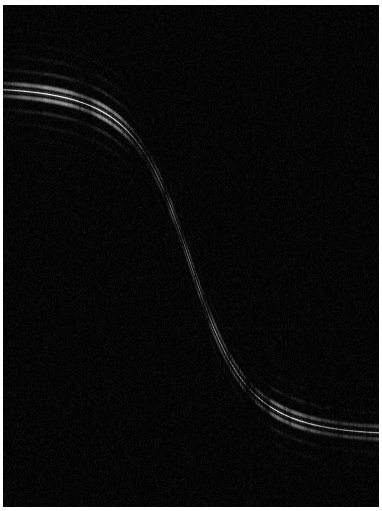


Fig. 2: Example of a LEO capture processed into PSD format.

2.2 Data Augmentation

Data augmentation is an integral part of the data pre-processing stage and can aid in dramatically increasing the size of the training data set. Additionally, data augmentation can be used to eliminate unintentional patterns that may exist in the training data. In our case, as a consequence of the data capturing process, the satellite signal is always centered in the PSD capture as shown in Fig.2. To prevent the ML algorithms from learning this behaviour, the satellite signal's position is randomised over a noisy (Gaussian) image. This is achieved by using the bounding box annotations for each image, a characteristic in an object detection problem, and randomly overlaying the pixels in the box over an image with Gaussian noise. An example augment of Fig. 2 is illustrated in Fig. 3, where the satellite signal is shifted to the upper left corner of the capture. It is important to note that the only characteristic of the capture being augmented is the position of the signal. The integrity and size of the signal, the brightness and contrast of the image, all remain unchanged.

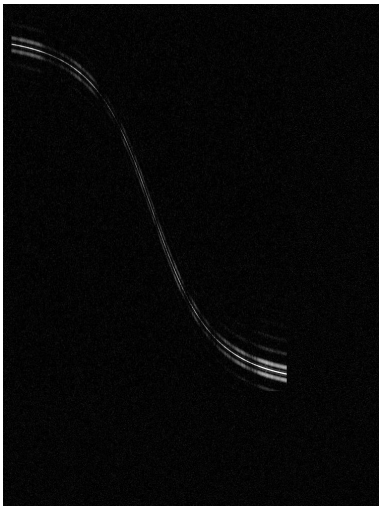


Fig. 3: An image augment of a LEO satellite signal data capture.

3. IMAGE CLASSIFICATION

3.1 Problem Background

To enable automated data processing, necessary for characterisation of satellite signals, the initial step is to determine if the data capture successfully detected any satellite signal. To this end, a check to ensure whether the PSD capture is of a satellite signal or otherwise (noise, terrestrial signal, etc.) is necessary before progressing through the remainder of the data processing pipeline. This is an archetypal image classification problem. In our case, there are two target classes: 1. Signal and 2. Other. The 'Signal' class refers to satellite signal whereas, the 'Other' class symbolises anything that could be present in a capture but is not a satellite signal. This could include, but not limited to, terrestrial signal or background noise.

3.2 Architecture

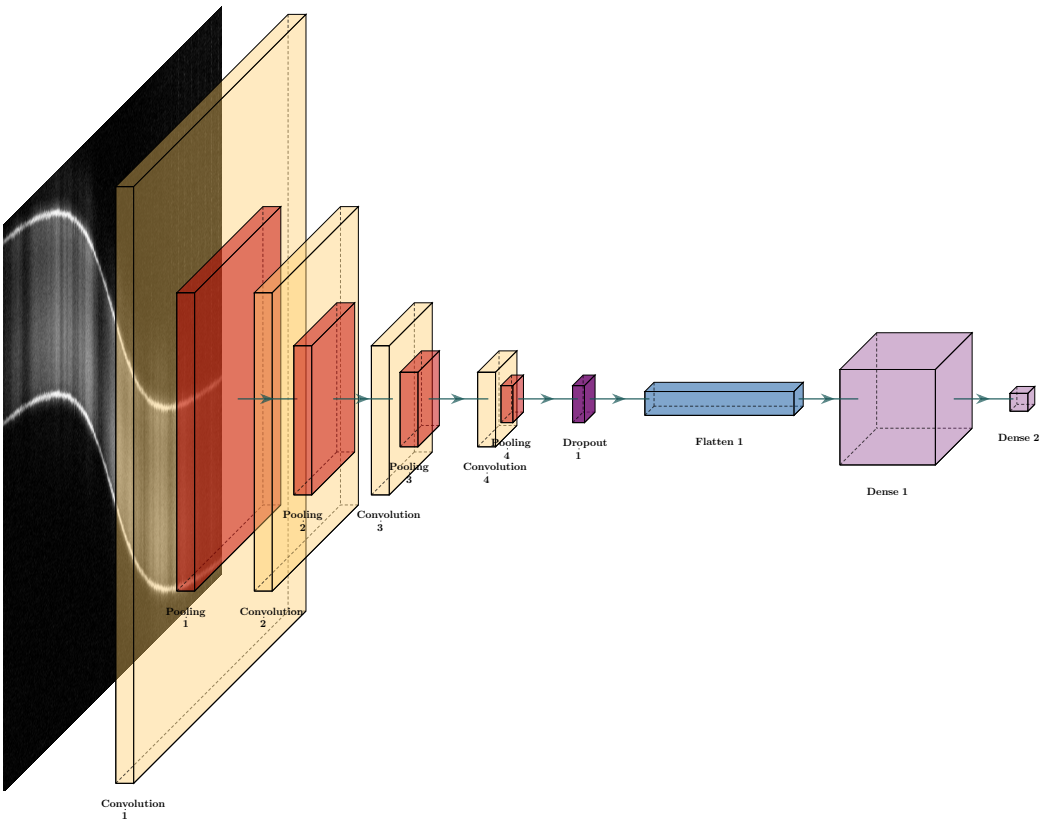


Fig. 4: Breakdown of the bespoke image classification architecture.

While there are pre-existing image classification models, it was decided to build and implement a bespoke classification model for the satellite signal classification use case. The model is built and implemented using the python libraries Tensorflow and Keras. The classifier architecture comprises of multiple convolutional and pooling layers, native to Keras, as illustrated in Fig. 4. The use of convolutional layers enables feature extraction from the original image in form of a feature map, with the number of kernels and kernel size being instrumental hyper-parameters of each layer. Dimensionality reduction of the resultant feature map is achieved through coupling each convolutional layer with a pooling layer.

A pivotal challenge in machine learning is the notion of overfitting, where the model being trained is overly tuned to the training data and is unable to successfully generalise, i.e. the patterns and features extracted by the model do not

translate well to unseen data. To account for, and mitigate the risk of overfitting, a dropout layer is included in the architecture, following the convolutional and pooling layer blocks. The dropout layer effectively "forgets" some of the features learned by the model through arbitrarily setting some input units to zero during training in accordance with a dropout rate which is a tunable parameter.

The flatten layer in the architecture merely flattens the input tensor and collapses all dimensions of the input into one axis. Finally, the last components of the classifier are two dense layers that are deeply connected. These deeply connected layers facilitate connections between each neuron in its layer to every neuron of the preceding layer. As a result, each neuron in these layers can receive inputs from every neuron in the preceding layer. These dense layers are used to perform classification on the basis of the outputs (features) from the convolutional layers.

3.3 Hyper-parameter Tuning/Optimisation

Hyper-parameter optimisation is a widely-used machine learning technique that aids in reducing the time taken to train an optimal model. Hyper-parameter optimisation refers to utilisation of optimisation strategies to determine the value for hyper-parameters for which the model's rate of training is optimal. For the satellite signal classifier, Bayesian Optimisation is used to determine these values.

Bayesian Optimisation is a global optimisation strategy that abstracts away any information about functional form by assuming that the function being optimised is a black box [9]. The Python library "BayesOpt" [6] was implemented to perform the non-linear optimisation and hyper-parameter tuning. This implementation uses a surrogate model, comprising of a distribution of functions, to represent the function to be optimised. An active learning strategy is then used to sample points of interest from the surrogate model which are evaluated for performance.

Table 1: Parameter search space for hyper-parameter tuning of the classification model. Note, the number following the filter and kernel parameter indicates the corresponding convolutional layer, and its depth, for which the parameter is defined.

Classification Parameter Search Space		
Parameter	Interval start	Interval end
Learning rate	1e-7	1e-4
Dropout Rate	0.2	0.3
Filter 1	1	8
Kernel 1	14	20
Filter 2	9	16
Kernel 2	9	13
Filter 3	17	32
Kernel 3	5	8
Filter 4	33	64
Kernel 4	1	4

The parameters tuned as a part of the hyper-parameter tuning activity are: learning rate, dropout rate and the number of filters as well as kernel sizes for each of the convolutional layers. The search spaces of each of the parameters are as described in table 1. A common practice in deep learning network engineering is to increase the number of filters with the number of convolutional layers. This is reflected in table 1 where the search spaces for filters are intentionally defined in ascending order as the depth of the layer increases. The increase in the number of filters in accordance with depth of the convolutional layers is justified by the number of abstractions obtained as the number of filters grow. For the initial layers, a smaller number of filters are used to extract information from data which is initially raw. The subsequent increase in number of filters for deeper layers accounts for more complex abstractions that are required for a better algorithmic learning.

Along the same vein, a pattern observable in the search spaces for kernel sizes in table 1 is the reduction of the kernel size as the layer depth increases. Kernel size is intrinsic for defining the size of a resultant feature map of a layer used to capture the result of the previous layer. This feature is encoded into the architecture to allow for multi-scale classification. Furthermore, the decrease in kernel size as the depth increases allows for better extraction of patterns concerning smaller objects.

4. OBJECT DETECTION

4.1 Problem Background

An extension to the image classification problem, of automating decisions about whether a PSD contains a satellite signal or not, is to allow for multiple satellites to be detected simultaneously in large data captures. A large bandwidth and time capture would benefit greatly from autonomous signal extraction which is a problem that is beyond the scope of traditional image classification. Object detection aims to both recognise if a satellite signal exists and determine its location within the PSD data.

Over the past decade, significant research has been conducted in the development of object detection techniques that conveniently fall into the following taxonomies of detectors: 1. One-stage detectors and 2. Two-stage detectors. Two stage detectors operate in two stages whereby the first stage comprises of a Region Proposal Network (RPN) which determines potential regions to be explored further. The second stage in a two-stage detector is where classification and localisation is performed. In contrast, one-stage detectors forego the RPN stage and object classification and localisation is performed directly. Popular examples of one-stage detectors are the You Only Look Once (YOLO) family as well as Single-Shot multibox Detector (SSD). Most commonly-known two-stage detectors fall within Region-based Convolutional Neural Network (R-CNN) family as well as the Feature Pyramid Network (FPN).

One-stage detectors are typically faster than their counterparts as everything is executed in one phase [10]. Meanwhile, two-stage detectors typically achieve a level of accuracy unmatched by the one-stage detectors. Traditionally, speed versus accuracy would be a trade-off considered depending on the type of object detectors being used. However, recent efforts to improve one-stage detectors' accuracy allows for both an increase in speed and also accuracy. Hence, while we experiment with a variety of detectors in this work, we focus particularly on the SSD framework.

4.2 Architecture

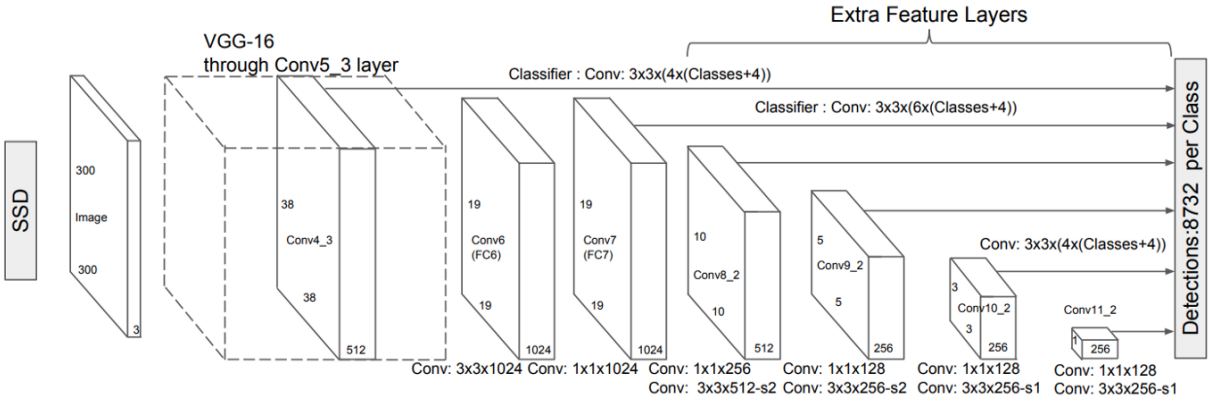


Fig. 5: Breakdown of the out-of-box SSD object detector architecture.

The SSD framework was introduced in 2016 [4] and the architecture for SSD is illustrated in Fig. 5. SSD requires inputs consisting of an image and the ground-truth boxes which are rectangular boxes highlighting instances of objects in the given image. The output produced by the detector is a set of bounding boxes which are predicted rectangular boxes drawn around the objects of interest, and their associated confidence scores.

The SSD architecture comprises of a base network, used for feature extraction, in addition to a series of convolutional layers. The characteristic feature of this detector are the extra feature layers that sit in the latter part of the detector's architecture to perform classification on the feature maps at varying spatial scales which is achieved through the use of convolutional layers decreasing in size. This multi-scale detection feature can aid in identifying small objects and in turn, such a feature allows us to reduce the false negative predictions. SSD framework is designed so that the last stage performs Non-Maximum Suppression (NMS) which is necessary in selecting the most appropriate bounding box from

a series of overlapping boxes, predicted for the same entity. Each layer in the structure uses filters to produce a fixed set of predictions that contain category (class) scores or shape offset values to the bounding box coordinates. Hence, the outputs in an object detection problem are two-fold: identification of an object (classification) and position of the object within an image (bounding-box regression).

4.2.1 Backbone Architecture

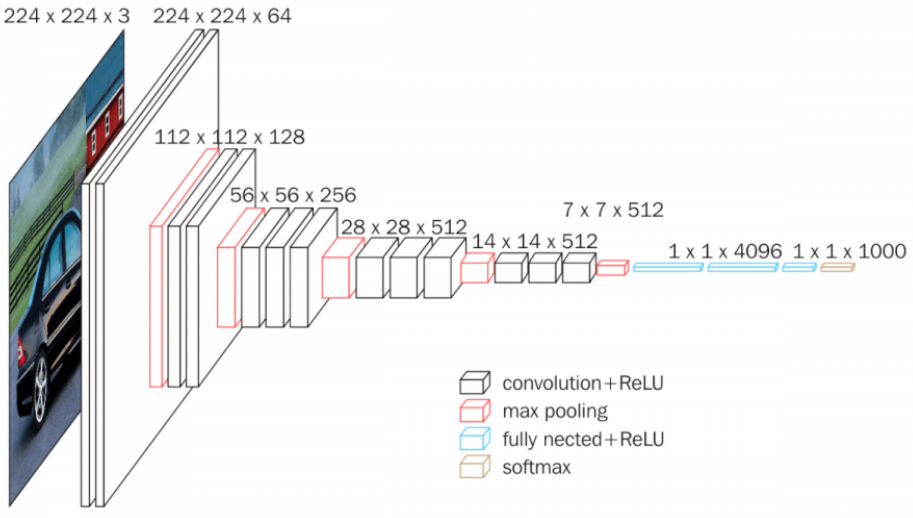


Fig. 6: VGG-16's Network Architecture.

In this work, we experiment with two base networks, VGG-16 and ResNet-50. These feature extractors have fast become two of the most popular backbones for deep learning detectors [8]. Note, VGG-16 is the VGG framework that constitutes of 16 layers with weights. As illustrated in Fig. 6, the VGG-16 architecture comprises of a series of convolutional and maxpool layers with kernel size of 3×3 and 2×2, and stride of 1×1 and 2×2. respectively. This succession of convolutional and maxpooling layers with fixed kernel size and stride throughout the architecture is characteristic of VGG. The idea behind the development of VGG is to reduce the number of tunable hyper-parameters which in turn boosts training speed as well as prevents over-fitting.

Residual Networks (ResNet) was first introduced in 2016 [2] and demonstrated that their method of redefining network layers as learning residual functions achieves an accuracy gain and allows for easier optimisation, resultant of significantly increased depth of the networks. ResNet-50 refers to the ResNet architecture with 50 layers as depicted in Table 2. There are multiple features of ResNet that make its architecture unique. The characterising idea of allowing learning of residual functions rather than unreferenced functions attempts to solve the degradation problem which becomes apparent as the depth of a network increases.

It is observed that a shallow network would have a higher training accuracy than its counterpart deeper network with additional added layers. Moreover, the training accuracy in the deeper network saturates before rapidly decreasing due to the increase in training error introduced by adding additional layers to the model. The solution adopted by ResNet to overcome the degradation problem constructs the additional layers of the deeper network as identity mappings which would allow the training error of the network to be constrained above that of the shallow network.

The ResNet network architecture follows a similar philosophy to VGG, where the convolutional layers have 3×3 filters and layers with the same output feature map size have the same number of filters. The last layer in the ResNet architecture is always a fully connected dense layer with softmax activation function. The feature that differentiates ResNet from VGG is that the former network has identity short connections which can be applied when the input and output of layers are the same dimension. Other cases where the dimensionality of inputs and outputs of layers varies are handled either using padding or a projection shortcut is used to match dimensions through the use of 1×1

Table 2: Architecture breakdown for ResNet framework with varying number of layers.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

convolutions. Refer to section 5.1 for the hyper-parameter optimisation results of the classifier.

4.3 Data Format

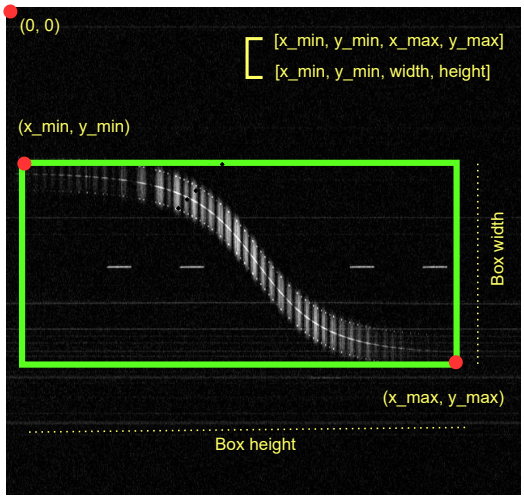


Fig. 7: Bounding box coordinate representation.

A bounding box is traditionally defined using four Euclidean coordinates, x and y coordinates of the bottom-left point of the box and either width and height of the box, or x and y coordinates of the top-right corner of the box. The two common representations of bounding box coordinates are illustrated in the top right corner of Fig. 7. Most object detection techniques adopt the bounding box notation to positionally identify an object. Additionally, to communicate the classification of an object, integer representations of class labels can be mapped. Note, the input data to our object detection algorithm is similar to that for the image classification case. The only point of contrast being that the size of the images being trained or performed inference on for the object detection problem could be significantly larger in frequency or time domain.

With the constant development and production of object detection tools, data ingestion can occur through various data

formats. However, there are some widely used formats that are typically accepted across most object detectors such as COCO and PASCAL VOC. In this work, we use the PASCAL VOC data format for data storage and representation. There are three main components within PASCAL VOC representation of the data: 1. Annotations, 2. ImageSets and 3. JPEGImages. All training and validation data involved in the learning process is stored in the JPEGImages directory the corresponding ground-truth bounding boxes and classification data stored in the Annotations directory in XML format. The annotation XML files store ground-truth detection data, including bounding box coordinates and classification labels, for all images in the dataset. The ImageSets directory details the membership of each image to either training or validation set.

4.4 Hyper-parameter Tuning/Optimisation

Table 3: Parameter search space for hyper-parameter tuning of the object detection model.

Object Detection Parameter Search Space		
Parameter	Interval start	Interval end
Learning rate	1e-6	0.5
Weight decay	0.0	0.999
Momentum	0.0	0.999
Mini-batch size	8	64

Amazon Web Services (AWS) is a cloud platform that offers tools such as SageMaker that has strong machine learning capabilities including model creation, training, deployment, inference, etc. For object detection, AWS SageMaker is used for data pre-processing, training and hyper-parameter tuning. AWS provides two-fold functionality for model tuning: 1. Standard hyper-parameter tuning job and 2. Warm start job. AWS' native object detection algorithm is an implementation of the SSD architecture with a configurable backbone and offers a host of tunable hyper-parameters. Subsequent to training AWS' object detection algorithm on the object detection (SSD) model, we use the standard hyper-parameter tuning jobs with varying backbones and a fixed Bayesian Optimisation strategy to explore the parameter space. The tuning job behaves as a foundation to warm start hyper-parameter tuning which provides additional exploration of the parameter space and progressive tuning of the hyper-parameters.

The parameters of the object detector being tuned during the training phase are: optimiser, learning rate, weight decay, momentum and mini-batch size. The search spaces for each of the parameters is detailed in table 3. Note, these are in agreement with the traditional values for parameters, as also documented by AWS. As the optimiser is a categorical parameter, rather than continuous or discreet which is the case for other hyper-parameters, the only two options explored are Adaptive Moment Estimation (ADAM) and Stochastic Gradient Descent (SGD). Additionally, experiments are conducted to determine the best suited backbone for this context between VGG-16 and ResNet-50. Refer to section 5.2 for the hyper-parameter optimisation results of the object detector.

5. RESULTS AND DISCUSSION

5.1 Image Classification

Table 4: Data distribution for image classification ingestion.

Data Distribution - Image Classification		
Data set	Size	Proportion
Training	4229	0.72
Validation	1135	0.19
Test	500	0.09
Total	**5864**	1.0

The data used for the training process in the classification phase of the research was distributed into training, validation and test set as depicted in table 4. With almost a 70-20-10% split between the three data sets, the total number of

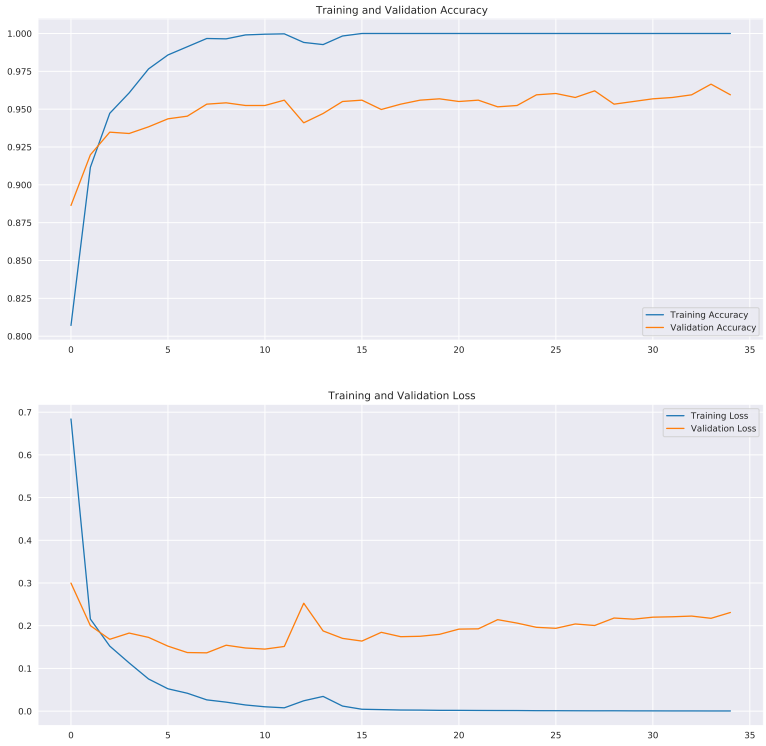


Fig. 8: Classification Training Loss and Accuracy Graphs.

samples used in this study are 5864. There are two key indicators that are a precursor to a models success: the model's ability to learn from training and validation set and the model's ability to generalise successfully to data uninvolved in the training process (test data). To measure the level of success of the classifier in achieving the former, we use training and validation accuracy and loss metrics. Keras' native accuracy metric evaluates performance by comparing the frequency of matched predictions to true labels.

The results from the hyper-parameter optimisation of the image classifier yields that the model is able to achieve a validation accuracy of 98%, when trained on the data set over a number of epochs. As illustrated in Fig. 8, the stabilisation of the training and validation loss indicates neither an under-fit nor an over-fit. The gaps in the training and validation loss graphs also indicate that a more comprehensive training data set is required as the current training samples are underrepresented when compared to the validation data set. The loss metric used in evaluating how well the model was able to learning during the training phase was binary cross entropy loss which calculates the average difference between predicted and true labels. The high accuracy and low loss achieved during the learning process for training and validation sets gives substance to the models successful learning phase.

To evaluate the second point of measuring the model's success, predictions are generated for the test data set and compared to their true label counterparts. Label predictions for the images in the set are produced by invoking the in-built Keras prediction method using the optimised classification model. Fig. 9 details the performance of the model on data never seen before. As can be inferred from the confusion matrix, the model predicted classification labels for the test data with an accuracy of 98%. Diving deeper into class-level evaluation, it is evident that all of the misclassified images belong to the signal class. This is further observable from Fig. 10 where the curve for signal (class 1) is slightly further away from a perfect classifier compared to its "other" counterpart.

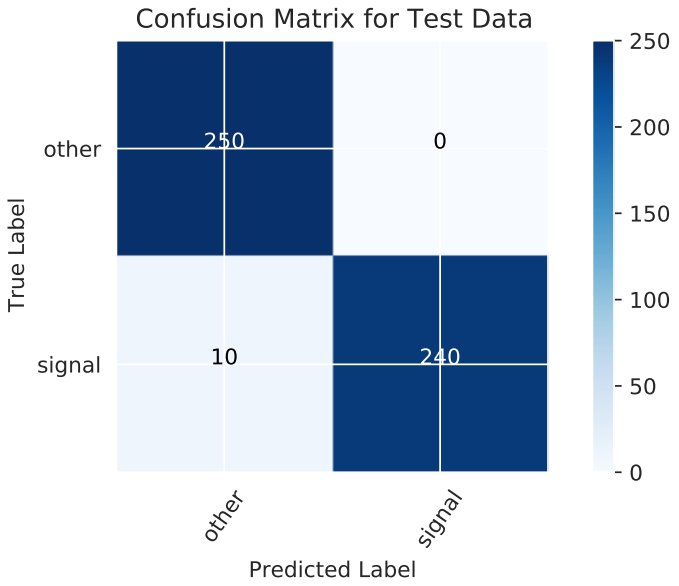


Fig. 9: Classification Test data set confusion matrix, true versus predicted labels.

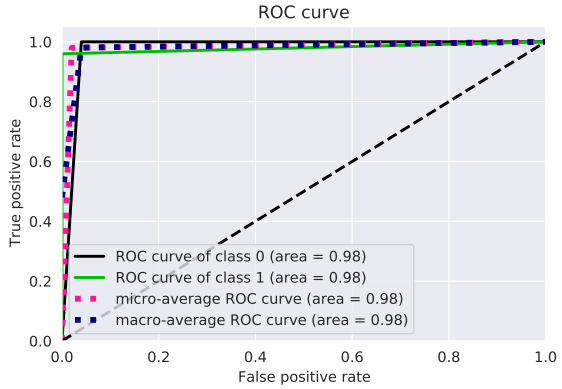


Fig. 10: Class level ROC curve for test data.

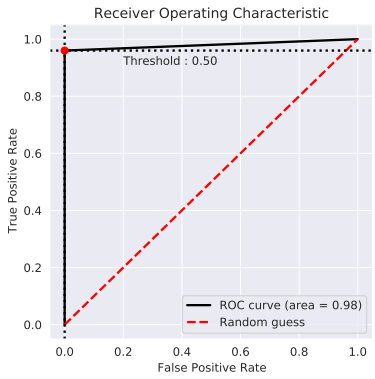


Fig. 11: ROC curve for test data overall performance.

Table 5: Metric Report for the image classifier. Note, precision, F-1 and recall scores are all values between 0 and 1.

Metric Report				
Group	Precision	Recall	F1-Score	Support
Other	0.96	1.0	0.98	250
Signal	1.00	0.96	0.98	250
Accuracy	N/A	N/A	0.98	500
Macro avg	0.98	0.98	0.98	500
Weighted avg	0.98	1.0	0.98	500

The validation accuracy is just one metric used to evaluate the model. There are various other metrics that can assist in diagnosing the performance of the model. Table 5 contains metric scores of evaluating the model on test data. Precision is used to evaluate how many of the predicted positive labels are true positive. The classifier has a precision score, for class “signal”, of 100% which solidifies that the model is successful in not predicting false positives (“other” images

classified as "signal"). Table 5 also shows a recall value of 96% for class "signal". Recall evaluates the proportion of images that are classified as signal to the total number of images that are actually signal captures. A 96% recall score suggests that there are certain images misclassified as "other" instead of the correct label "signal". In this context, we argue that false negatives is more costly than having false positives. Lastly, F1 is a metric which attempts to strike a balance between precision and recall through calculating a weighted average of the two. A F1 score of 98% is an effective indicator of a high performing model in terms of its ability to generalise to test data.

Table 6: Performance of classifier and energy detection algorithm on image samples from Adelaide and Canberra sites.

Performance Comparison			
Site	Samples	Energy Detection Accuracy	Classifier Accuracy
Adelaide	2965	2681 (90%)	2944 (**99%**)
Canberra	1919	1831 (95%)	1908 (**99%**)
Total	4884	92.4%	**99.3%**

When evaluating the classification model's performance, it is important to determine whether there is any quantifiable advantage in using it over an existing technique such as energy detection. To compare the performance of the classifier and the energy detection algorithm, both the techniques are tested on a separate set of data. Energy detection operates on the data collected from the SpaceAware sensor network by performing a coarse estimation of the noise floor and a manual threshold is set above this noise floor. Any detections observable above this threshold in the spectrum are considered as satellite signal. This method of detecting satellite signals is prone to two types of errors, missed detections from faint signals and false alarms from interference. Table 6 depicts the difference in performance between the two techniques. The results demonstrate that the classification model outperforms the energy detection technique for both sites. Hence, the collective results show that the classification model is a good candidate for determining the presence of satellite signal within a PSD capture.

5.2 Object Detection

Table 7: Data distribution for object detection ingestion.

Data Distribution - Object Detection		
Data set	Size	Proportion
Training	6030	0.79
Validation	1508	0.20
Test	50	0.01
Total	**7588**	1.0

The data involved in the end-to-end training and evaluation of the deep learning network SSD is distributed as shown in table 7. The total number of images involved in the full process is 7588. We achieve a data set of such size by using data augmentation and producing two augments per image within the training data. It is important to note that for object detection, there is only one class being detected and that is "signal" unlike for image classification where there were two class labels of "signal" and "other".

Subsequent to the hyper-parameter tuning activity via AWS' built-in hyper-parameter tuning and warm start functionality, the model is able to achieve a range of mean Average Precision (mAP) scores under varied conditions. As mAP is mean of AP over all classes, in this context mAP and AP can be used interchangeably as there is only one class to be detected (signal). The mAP scores provide further insight when considered along with the Intersection Over Union (IOU) thresholds. IOU is a vital tool in thresholding and evaluates the degree of overlap between a ground truth box and a predicted box. Moreover, a threshold can be used to eliminate boxes, below which any box predictions will be dismissed and, above which predicted boxes will be accepted as valid predictions.

Table 8 outlines that the SSD, in this context, realises substantially better performance when using the ADAM optimiser over SGD. This is also illustrated by Fig. 12 as the curves associated with ADAM optimiser outperform the yellow and orange curves associated with the SGD optimiser. Furthermore, there appears to be a marginal difference

Table 8: Results of OD with SSD network. Here mAP refers to mAP@0.5: 0.05 :0.95IOU, and the others are mAP@0.5 IOU and mAP@0.75 IOU respectively. For the breakdown of overall mAP score between IOU threshold of $0.5-0.9$, refer to Table 9 in section 10.

Performance Comparison				
Backbone	Optimiser	mAP	mAP^{50}	mAP^{75}
ResNet-50	ADAM	0.782	**0.997**	0.931
ResNet-50	SGD	0.108	0.348	0.025
VGG-16	**ADAM**	**0.843**	0.996	**0.975**
VGG-16	SGD	0.098	0.344	0.019

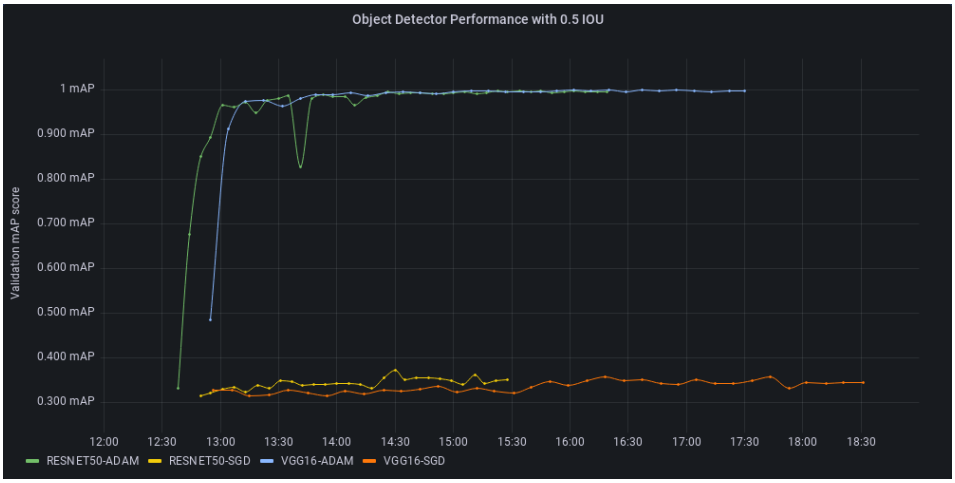


Fig. 12: Validation mAP scores over the training epochs of each backbone-optimiser configuration. Note, the IOU threshold for this experimental run is set at 0.5. The x axis in this graph is time over which the training of the four models occurred. It is important to note that the longer training time for some of the models over the others is just an artefact of simultaneously scheduling four training jobs. For a performance comparison with IOU threshold of 0.75, refer to Fig. 16 in section 10.

between the performance of the two backbones ResNet-50 and VGG-16. The best performing model on validation data, VGG-16 with ADAM backbone, has a higher mAP score when the IOU threshold is stricter where mAP^{50} is considerably lower than mAP^{75}. This behaviour is also apparent in Fig. 13 where the VGG-16 backbone with ADAM optimiser has noticeably higher mAP scores, in comparison to the other models, for stricter IOU threshold values. However, the true validation evaluation of the object detection model can be perceived using mAP which is the average of mAP scores associated with all IOU thresholds between the interval $0.5-0.95$ with a IOU step size of 0.05. Importantly, this mAP score conveys the model's overall performance averaged over a range of IOU thresholds. The VGG-16 model with a ADAM backbone is able to achieve a mAP score of 0.843. The high mAP score reinforces the success of the model's learning process of being able to detect satellite signals in a large capture.

5.2.1 Small-Medium Bandwidth/Time Inference

In order to validate the object detection model's performance on test data, it is first imperative to ensure that the model is able to successfully perform inference. In this study, we perform model inference and test it via two different methods: 1. AWS inference and 2. Bespoke offline inference. AWS allows functionality to create endpoints, and their corresponding configurations, where the model hosted can be invoked to perform inference. However, for inference on

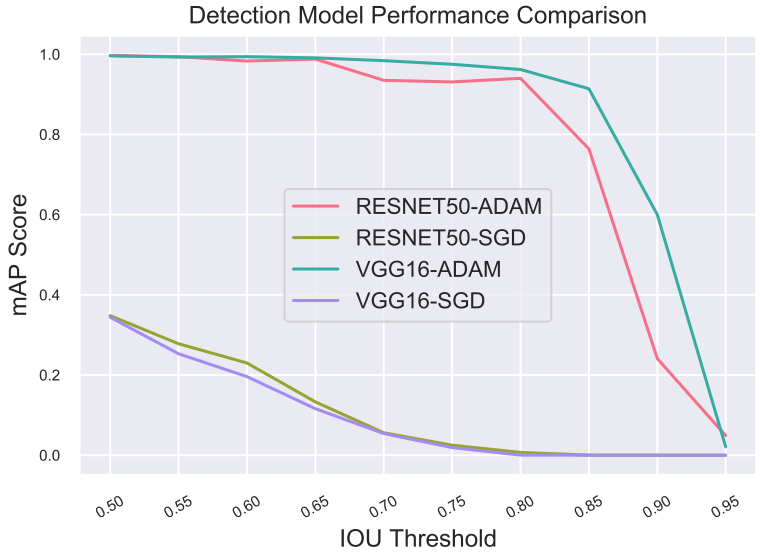
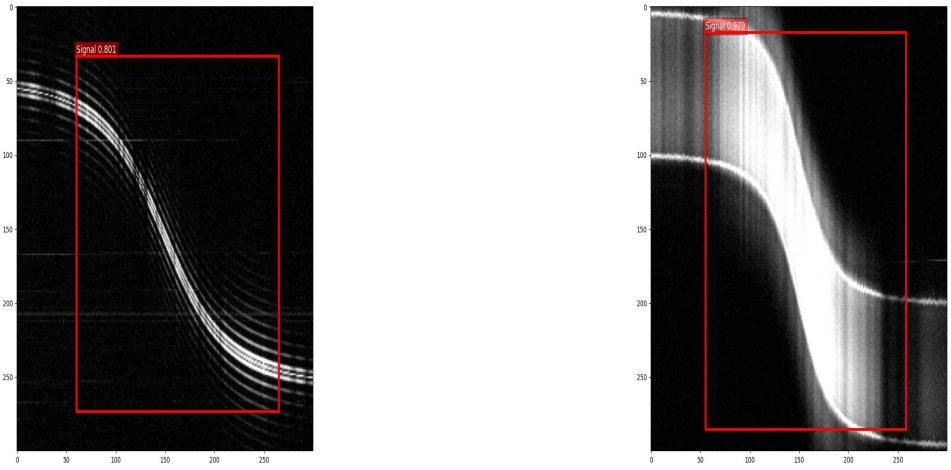


Fig. 13: Validation mAP scores over varying thresholds for each backbone-optimiser configuration. Note, the IOU thresholds for this experimental run is set between $0.5 - 0.95$ at intervals pf 0.05.

the cloud platform to successfully occur, a stable internet connection is required. Considering potential deployment of this technology to remote sites with a possible lack of internet services, it is deemed fruitful to additionally be able to perform inference offline. The offline model is developed in-house and, experimental results show that the predictions generated via AWS inference and the custom offline inference are in congruence.



(a) Small time/bandwidth inference on Satellite SARAL (b) Small time/bandwidth inference on Satellite FALCONSAT 3

Fig. 14: SSD inference results on small time/bandwidth data capture.

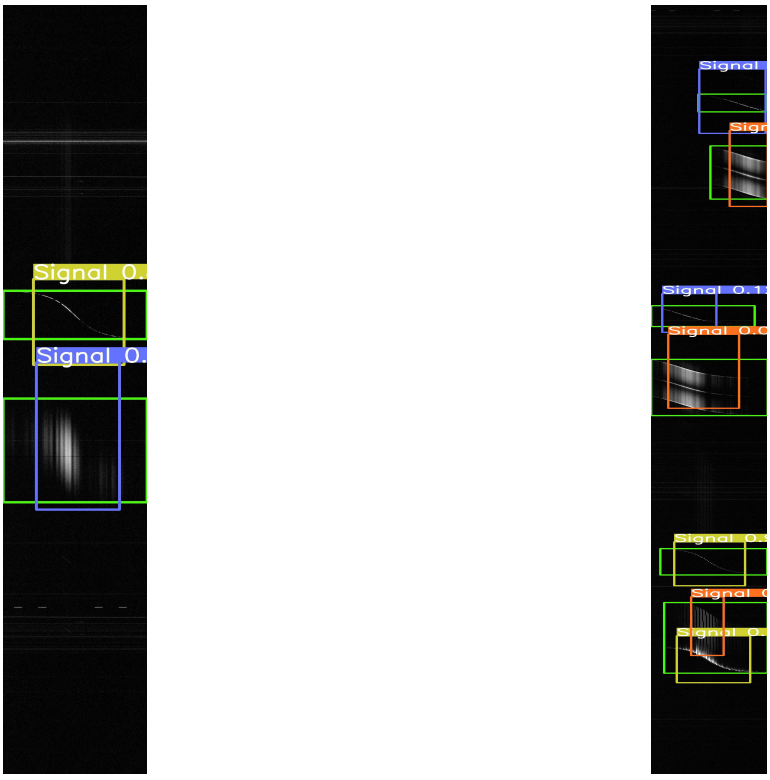
The two best performing models, ResNet-50 and VGG-16 with ADAM optimiser, are both evaluated through inference on the test set. It is found that the ResNet-50/ADAM model configuration outperforms VGG-16/ADAM when comparing test set inference. The results of performing inference using ResNet-50/ADAM backbone on a test set of data is depicted in Fig. 14. The two images in the figure illustrate satellite signals from two separate satellites. Infer-

ence results are visualised on the captures in form of bounding boxes highlighting the model's detection. In case of both the satellites, the predicted bounding boxes capture majority of the satellite signal including, arguably, the most important characteristic which is the doppler shift. It is noticeable that both bounding boxes are different in size and also communicate the confidence level alongside the class name on the top left hand corner of the predicted box. The results show that, in the case of the two aforementioned satellites, the model is more confident about its prediction for FALCONSAT 3 (right) with a confidence score of 0.98. Hence, upon evaluating the model's performance on test data, it can be concluded that the model adequately detects satellite signal in a setting where the frequency bandwidth/time capture is short in duration.

5.2.2 Large-Bandwidth/Time Inference

In the case of large bandwidth/time captures, neither AWS inference nor the bespoke inference method would successfully work. This is resultant of the sizes of the captures that the network is trained on are considerably smaller than the large data captures. The full benefit of signal extraction automation via object detection is apparent in the case of large captures whereby multiple satellite signals are able to be captured as well as being automatically extracted from an image. Hence, it is vital that the model is able to generalise its learning from short captures to long captures. However, this is a challenging task as simply resizing the larger image would introduce distorted aspect ratios of shapes that exist within the captures.

Small object detection and large scale inference has been a challenge that has continued to presented itself in practical usage. Slicing Aided Hyper Inference (SAHI) is a computer vision solution developed by [1] that tackles the issue of object detection at a larger scale as well as instance segmentation. The working concept behind SAHI is the notion of sliced inference. Sliced inference works by sliding a window of a smaller size over the original image to performance small-scale inference and predictions which are stitched back on the original image.



(a) Example 1: Large time/bandwidth inference

(b) Example 2: Large time/bandwidth inference

Fig. 15: SSD inference results on large time/bandwidth data capture using ResNet-50 and ADAM optimiser.

In a similar fashion to a small bandwidth capture, we evaluate both ResNet-50/ADAM and VGG-16/ADAM on their ability to generalise to a test data set. It is found that ResNet-50/ADAM outperforms VGG-16/ADAM for wide bandwidth captures. The results, from customising the SAHI library to integrate the ResNet-50/ADAM offline inference solution and adapt to the custom detector, are shown in Fig. 15. There are multiple components to each image: ground truth boxes indicated in green; prediction boxes graphed in various colours depending on confidence level of predictions; and classification labels alongside confidence scores present above every prediction box. The images presented in Fig. 15 are a small snapshot of the original size of the long captures designed to showcase some of the results. Both images, alongside the rest of the test results, illustrate that all prediction boxes present are localised around the ground truth boxes and therefore the detector performs well when separating satellite signal from noise. Additionally, it can be seen that, in case of the example 1, the terrestrial signal present in the top section of the image is correctly identified as not signal which is apparent from the lack of boxes in the area.

The two images also show box predictions of varying confidence levels and degree of overlap with the ground truth boxes. The apparent position of box predictions and their associated confidence thresholds suggests that the model is able to detect signal well. However, some of the lower confidence scores, and under or overlapping box predictions with ground truth boxes, indicate that further improvements can be made to elevate confidence levels and improve bounding box coordinates.

6. CONCLUSION

This study explores applications of machine learning to Space Domain Awareness (SDA) through automating signal detection in passive RF captures. In considering the challenges, ranging from manual analysis required by a human operator to determine presence of a satellite signal to successful captures being restricted to satellites with prior orbit or frequency knowledge, it becomes apparent that there is a need for a tool which allows for detection of satellite signals specifically in PSD captures that are large in frequency and time. There are two ML frameworks studied in this research, image classification and object detection, with the former solution leading into the more complex and practical second solution. The results from the case study of image classification demonstrates that the bespoke CNN model developed performs well with a high validation and test set accuracy of 98%. Furthermore, evaluating the model using metrics such as precision, recall and F1 draws the same conclusion that the trained in-house model is accurately able to classify a capture containing satellite signal.

Expanding the scope of the first solution, object detection offers the additional advantage of classifying and localising an object in an image. In this context, an SSD network is trained to detect satellite signals and highlight their location within a capture which may be large in frequency and/or time domain. The experimental results establish that the SSD configuration which yields the optimal mAP score of 0.843 requires a VGG-16 backbone and ADAM optimiser. However, when evaluating the models' performance on test data, ResNet-50 with a ADAM optimiser outperforms VGG-16 with ADAM optimiser. As generalisation on unseen data is crucial in ML, we find ResNet-50/ADAM to be the best performing SSD configuration overall. Through hyper-parameter optimisation, the detection model is able to successfully detect satellite signals in a short capture. Furthermore, an integrated solution of custom offline inference and third-party library SAHI allows for a detection model that can effectively generalise to the test data. In conclusion, the models presented in this paper successfully trained on real passive RF data and enabled automation of satellite signal detection in PSD captures.

7. FUTURE WORK

While the scope of this study has been broad enough to cover two solutions, further work needs to be done in advancing the applications detailed in this paper. Firstly, as the results demonstrated for the object detection model, revisions to the model are required to improve the bounding box predictions. One method of achieving this is through collating a more comprehensive data set for the training process and ensuring that the ground truth box annotations recorded for the data are accurate. An improvement in the box predictions could further affect an improvement in the confidence scores for the detection predictions.

Another annex to this research could be to employ similar classification tools to identify and characterise modulation types of satellites via the passive RF IQ data captured. This could be extended to include automated recognition and

clustering of satellite on the basis of their modulation types. An additional benefit of automating characterisation of satellites using the IQ data, as opposed to PSD data, would be the ability to skip the intermediate step of using mathematical transforms such as Fourier Transforms to convert the data from IQ to PSD. Other potential future work includes developing an intelligent system encoded with the ability of anomaly detection through monitoring satellite signatures and extensive historical record-keeping.

8. ACKNOWLEDGEMENT

This research was conducted as part of an Australian Cooperative Research Centre Project (CRC-P) led by Clearbox Systems in collaboration with the University of New South Wales (UNSW) Canberra, Capricorn Space and Bluerydge. The authors would like to acknowledge the support from the Australian Department of Industry, Science and Resources as well as the support from UNSW Canberra and Capricorn Space for hosting sensor sites to enable real data to be collected for this research.

9. REFERENCES

[1] Fatih Cagatay Akyon, Sinan Onur Altinuc, and Alptekin Temizel. Slicing aided hyper inference and fine-tuning for small object detection. *2022 IEEE International Conference on Image Processing (ICIP)*, pages 966–970, 2022.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[3] Mohd Noor Islam, Thomas Q. Wang, Samuel Wade, Travis Bessell, Tim Spitzer, and Jeremy Hallett. Doppler and angle of arrival estimation from digitally modulated satellite signals in passive rf space domain awareness. In *22nd Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS 2021)*, pages 694–707, Maui, Hawaii, 2021. Maui Economic Development Board.

[4] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[5] Òscar Lorente, Ian Riera, and Aditya Rana. Image classification with classic and deep learning techniques. *arXiv preprint arXiv:2105.04895*, 2021.

[6] Ruben Martinez-Cantin. Bayesopt: a bayesian optimization library for nonlinear optimization, experimental design and bandits. *J. Mach. Learn. Res.*, 15(1):3735–3739, 2014.

[7] Kameron Simon. Passive rf in support of closely spaced objects scenarios. In *ADVANCED MAUI OPTICAL AND SPACE SURVEILLANCE TECHNOLOGIES CONFERENCE. 21ST 2020. (AMOS 2020)*, pages 327–334, Maui, Hawaii, 2020. Maui Economic Development Board.

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[9] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[10] Petru Soviany and Radu Tudor Ionescu. Optimizing the trade-off between single-stage and two-stage deep object detectors using image difficulty prediction. In *2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 209–214. IEEE, 2018.

[11] Tevfik Yucek and Huseyin Arslan. A survey of spectrum sensing algorithms for cognitive radio applications. *IEEE Communications Surveys and Tutorials*, 11(1):116–130, 2009.

[12] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232, 2019.

10. APPENDIX

Table 9: Breakdown of mAP scores for each of the four model configurations with varying IOU thresholds.

mAP Score Comparison												
Backbone	Optimiser	mAP^{50}	mAP^{55}	mAP^{60}	mAP^{65}	mAP^{70}	mAP^{75}	mAP^{80}	mAP^{85}	mAP^{90}	mAP^{95}	Average
ResNet-50	ADAM	**0.997**	**0.994**	0.983	0.998	0.935	0.931	0.940	0.764	0.241	**0.049**	0.782
ResNet-50	SGD	0.348	0.278	0.230	0.133	0.056	0.025	0.007	0.0002	0.0001	0.0	0.108
VGG-16	ADAM	0.996	0.993	**0.994**	**0.991**	**0.984**	**0.975**	**0.962**	**0.914**	**0.600**	0.021	**0.843**
VGG-16	SGD	0.344	0.253	0.196	0.116	0.054	0.019	0.0002	0.0001	0.0	0.0	0.098

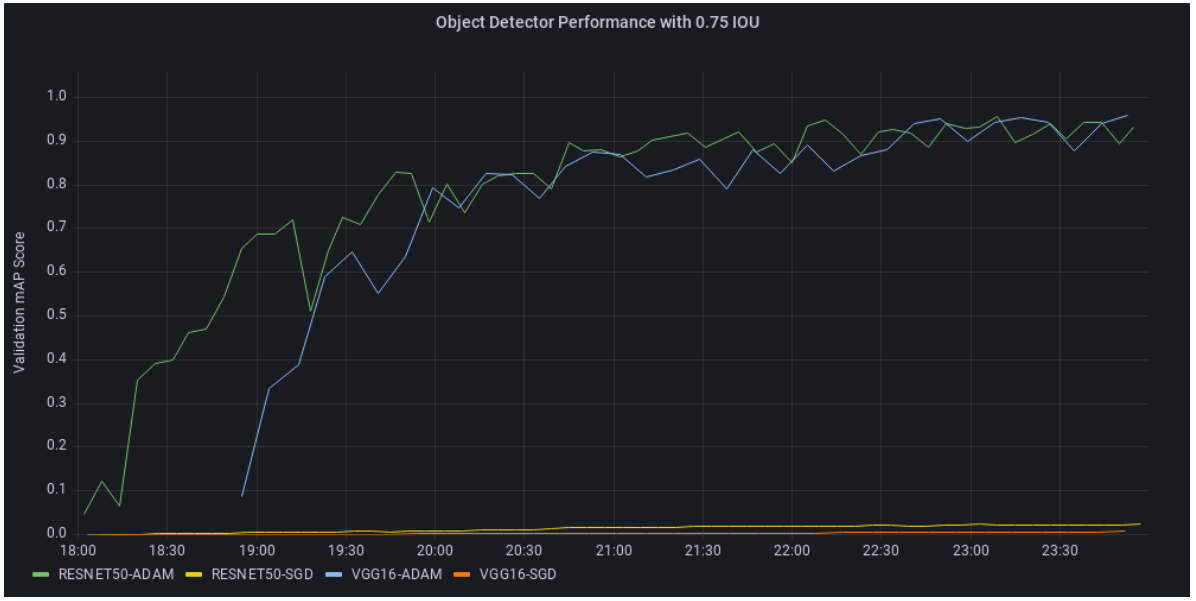


Fig. 16: Validation mAP scores over the training epochs of each backbone-optimiser configuration. Note, the IOU threshold for this experimental run is set at 0.75. It is important to note that the longer training time for some of the models over the others is just an artefact of simultaneously scheduling four training jobs.