SSC24-IV-07

Machine Learning Models for Optimisation of Satellite Laser Communication Terminals and Optical Space Networks

Georgia Harvey, Cameron Anderson, Murray Ireland
Craft Prospect Ltd
Suite 12 Fairfield, 1048 Govan Road, Glasgow, United Kingdom, G51 4XS; +44 (0) 7421 994 712
georgia@craftprospect.com

Rob Hunter, Afonso Nunes, Rob Stansfield
CGI IT UK Ltd
20 Fenchurch Street, 14th Floor, London, United Kingdom, EC3M 3BY, +44 (0) 2076 379 111
rob.hunter@cgi.com

ABSTRACT

Laser communication systems in optical SatCom systems are mainly used in point-to-point networks, possibly deploying standard routing solutions which perform sub-optimally when deployed in transport and access satellite networks. Optical signal acquisition is also affected by the changing satellite environment and atmospheric conditions.

In response, Craft Prospect and partners focused on upgrading optical systems by using machine learning (ML) methods to improve laser communication terminals and to simplify interconnectivity of networked laser SatCom systems. In this context we target the European OPS-SAT Versatile Optical Laboratory for Telecoms (VOLT) mission led by Craft Prospect and the European High Throughput Optical Network (HydRON) project which benefits from using machine learning systems.

Based on initial research and development work on enhancement of satellite free space laser communication systems with machine learning, two use cases and their requirements were identified. Firstly, leveraging autonomic networking principles to develop distributed ML agents for space network nodes which monitor local network state and can autonomously make rerouting decisions on impaired links in a localised way to improve the average network throughput. This enables links between optical ground stations (OGS) and the space network segments to quickly switch in a smooth and responsive way without having to have multiple paths open; searches carried out for preset alternate paths in long/overloaded flow table lists; or an over-reliance on a software defined networking (SDN) controller. Secondly, for space segment laser terminals, development of a ML model to boost detection accuracy in coarse acquisition in reduced signal-to-noise ratio (SNR) cases with strong background light conditions and varying beacon intensity. This is relevant for the scenario where a LEO satellite receives an uplink signal from OGS with strong atmospheric fluctuations or background Earth reflections and the detector is saturated. Similarly, in inter-satellite links (ISL) where satellites are undergoing relative movement and the signal intensity rate of change is rapid, the acquisition sensor quickly arrives in a region where SNR is low, or the sensor is saturated.

In this contribution, we present the developed ML models for these use cases and describe training techniques and datasets. We discuss performance results from tests carried out with a network simulator with the results compared to an SDN controller solution to demonstrate the benefits. For small network topologies, the ML solution resulted in the throughput at the network endpoint being 16.7% higher following link degradation than the SDN controller solution. The spatial acquisition detection errors and accuracy under different SNR conditions using the ML solution was compared to a benchmark Centre of Gravity detection method typically used in laser terminal beacon acquisition. Results showed that the beacon detections from the ML solution were closer in distance to the true spot than the conventional method in low SNR conditions by over an order of magnitude, in addition to having an overall accuracy of 97.77%.

With this we show how to enable autonomic routing in optical data, and laser terminals with improved acquisition rates in networks for OPS-SAT VOLT and for multitudes of future interconnected laser SatCom missions.

INTRODUCTION

Laser communication systems in optical payloads are often used in non-optimised satellite communication (SatCom) systems. In response to this Craft Prospect and partners are focused on upgrading optical systems by using machine learning (ML) methods to improve laser communication terminals and to simplify interconnectivity of networked laser SatCom systems. In this context we target the European OPS-SAT Versatile Optical Laboratory for Telecoms (VOLT) mission led by Craft Prospect and the European Space Agency (ESA) High Throughput Optical Network (HydRON) project which benefits from using machine learning systems.

The space and network segments were the focus of examination, and this led to the down selection of two main opportunities:

1. ML for Spatial Acquisition
2. ML for Network Management

Spatial Acquisition Use Case

This use case is based on using ML algorithms to improve the spatial acquisition segment of the acquisition, pointing, and tracking process (APT). The APT process is an essential part of communication between Laser Communication Terminals (LCTs), and so is a key part of HydRON. Accurate beam pointing must be established and maintained to have data transmission with minimal losses. The method of detection chosen for analysis for this activity was CMOS and InGaAs camera sensors. This decision was based on previous CPL work and expertise with APT systems.

Detection is conventionally done using traditional computer vision (CV) algorithms, such as the Centre of Gravity (CoG) algorithm, which identifies the incoming laser beacon spot on the detector from its high pixel intensity value as compared to the background details on the detector. Accuracy and time of the conventional solution can be affected by effects on the sensor such as background noise and dead/hot pixel anomalies. As such, this use case involved detection of the incoming beacon in low Signal to Noise Ratio (SNR) conditions to investigate if ML can be used to boost the SNR during detection. Two main scenarios were selected for analysis where the baseline CoG accuracy is reduced:

In Scenario 1, a low Earth orbit (LEO) satellite points towards the Earth optical ground station (OGS) and is receiving uplink. Strong background light can be present on the LCT sensor from background Earth reflections (due to Earth albedo) which reduces the uplink signal dynamic range on the acquisition sensor and furthermore the signal fluctuates (approximately 20dB) in power at the receiving LCT due to atmospheric turbulence. Here SNR boosting is of interest for improved acquisition.

In Scenario 2, an inter-satellite link (ISL) is being established and the two satellites with LCT are undergoing movement towards each other or away from each other and the intensity of the beam increases and decreases over their fly-by. Here although no strong background noise is expected as the LCTs are pointing into dark space but due to the received changing beacon power, SNR boosting is of interest for improved acquisition due to the strong signal fluctuation rate. In this scenario the acquisition sensor quickly ends up in a region where SNR in the image is low or the sensor is saturated.

This allows ML model detection accuracy to be compared with SNR showing where the use of ML is most effective. The results from the ML model detection were fed into a conventional CoG algorithm to get an enhanced output as the bounding box generated by the model allows for the Region of Interest (ROI) to be decreased and the centre of the spot to be better found for tracking purposes. The combined approach was compared with a pure CoG detection-based result.

Network Healing Use Case

The use case focused on network management was performed in partnership with CGI, using their experience with concepts and simulations of networks.

In HydRON, the network architecture is built on optical links between the spacecraft and the OGS, and on optical ISL[1]. In HydRON study, several routing concepts were analysed, and preference was given to centralised routing solutions due to the ability to maintain network routing semantics as promptly as possible when facing the changing network topologies. However, as the name suggests, centralised routing solutions fully rely on the central controller decisions to act and react to network impairments. Thus, we see room for improved fault management and decrease of routing reaction times if routing devices can have a certain level or discretionary routing decision, time and scope limited.

Optical links result in the high throughput requirements as set out by HydRON, however optical links between satellites and OGS can also be affected by atmospheric conditions such as turbulence or adverse weather conditions, which affects the requirements on optical links to have a clear line of sight. Additionally, ISLs could be affected by factors such as changing distances between satellites, capabilities of different LCTs and varying solar background conditions. These scenarios could cause performance degradation on network links

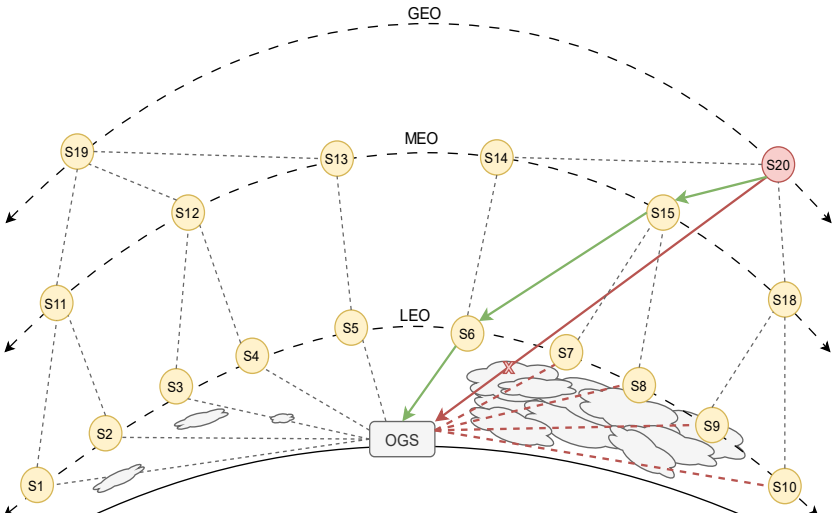

Figure 1: Hypothetical illustration of real-world application of network healing use case.

and lower throughput. The conventional solution is to handle small degradation via buffering at the network nodes until normal operation can resume. For more persistent outage, the SDN centralised controller monitors network links and if it detects faulty links through regular link probing, then it is configured to re-route the traffic on some other path through the network to reach the required end point. However, with a centralised controller, this means there is a communication overhead between nodes and the controller, increasing the time for the throughput reduction to be resolved.

The ML solution to improve the data-plane resilience to outages is through implementing autonomic network functionality[8] by deploying a software application on each switch node to monitor its links, and then analyse the traffic to make a prediction on the link quality using ML. This software application is known as an Agent for this work. The Agent then determines if local re-routing actions are required and what the best path to the end point is based on its knowledge of the network topology and latencies across the network. The Agent applications can send notifications of path updates to other Agents, so knowledge of a link degradation is shared with the nodes neighbouring a degraded link, or within a limited local radius. The aim is that by temporarily bypassing the communication overhead with a controller and having the nodes themselves perform localised re-routing, it will allow the throughput to recover from degradation more effectively. It also offers redundancy if connection is lost with the controller node. The performance of the ML Agents will be compared to that of the SDN controllers for the same scenarios.

It is envisioned that this ML enabled network healing functionality could fit into real use cases such as an OGS with up to 20 space nodes in view wherein some of the optical links are degraded with reduced throughput due to cloud cover, atmospheric turbulence, or some temporary obstruction like a passing aircraft. Local re-routing capability would enable the satellite nodes to quickly detect the degradation and perform re-routing to maintain overall end user quality. Figure 1 is a hypothetical illustration of this real use case; with 20 space nodes organised by LEO, medium Earth orbits (MEO) and geostationary orbits (GEO), and dashed lines between nodes highlighting potential links. In this example, space node S20 needs to downlink to the OGS, but cloud cover is impairing the link. Network healing ML Agents would allow S20 to detect the degradation on its link, and for the other nodes in the network to report on their link qualities, allowing S20 route its traffic through other satellites on the optimal path determined by the ML. Similarly, there could be the case of multiple OGS in view of the space segment with the path to one OGS obstructed, wherein the local healing solution could enable routing to another OGS, reducing handover.

MACHINE LEARNING MODELS

Spatial Acquisition Model

The YOLOv3 Tiny model was used to perform object detection on camera sensor images to distinguish the incoming beacon from the noise conditions. YOLO – or You Only Look Once – is a fully convolutional, single-stage object detection model designed for real time object detection. For the tiny version of YOLOv3, the model is comprised of a feature extracting backbone of 7 convolutional layers using 3x3 kernels interspersed with normalisation, pooling and leaky ReLU layers. This architecture is seen in Figure 2.

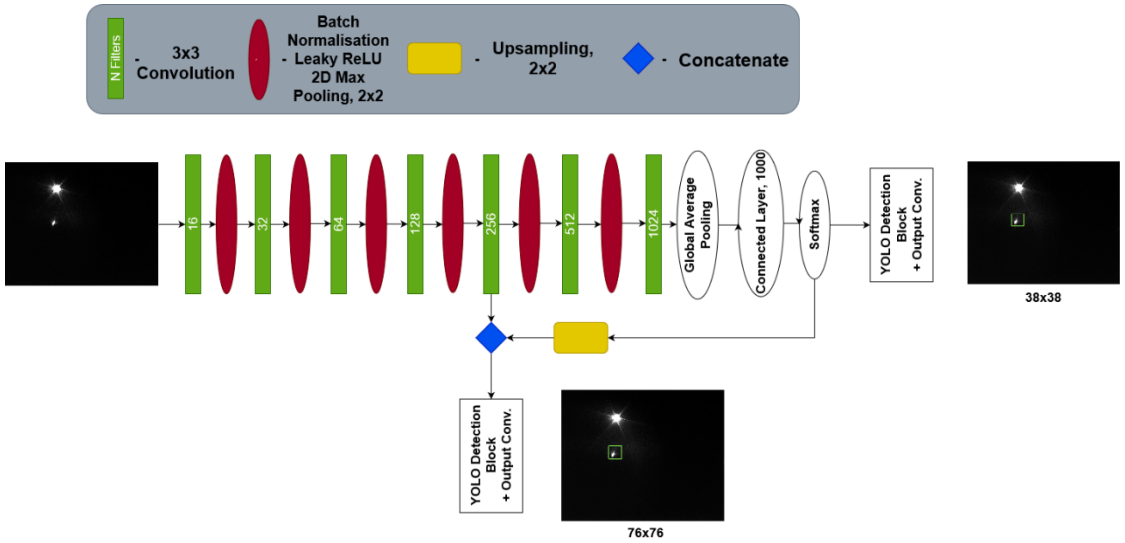

Figure 2: YOLOv3 Tiny architecture.

The skip connection in the model, where the output of an intermediate layer is concatenated with the final output, allows for detections of objects to be made at different scales in an image. The input to the model is a scaled image, normalised to the maximum pixel intensity based on the precision of the pixel values. After passing through the model, the outputs obtained are the classification for objects that have been detected in the image, the bounding boxes that contain these objects, and confidence scores around the detections.

The YOLO model was selected primarily due to the inference speed which is required for real time detection and the possibility of tracking. Using optimised hardware implementations, YOLOv3 can process over 200 frames per second (corresponding to an inference time of 5ms)[2, 3, 4]. Convolutional neural networks have been shown to have strengths in object detection tasks of this type[7].

After the bounding boxes have been detected using the ML model, the standard Centre of Gravity algorithm is used within the region of interest defined by the bounding box to detect the spot itself.

Network Healing Model

The model developed for the network healing use case was comprised of a temporal autoencoder (TAE) which learns a low-dimensional representation of the timeseries data with a multi-layered perceptron (MLP) to make predictions on latency. It is a novel architecture based on a similar model used for clustering of time series data[6]. The TAE is used to learn the most important features of the data at different temporal scales by using a convolutional layer followed by bi-directional long short-term memory (LSTM) layers to learn the short term and longer-term patterns respectively. The choice of algorithm was informed by the traffic data available in the simulator and the need to have a quantitative

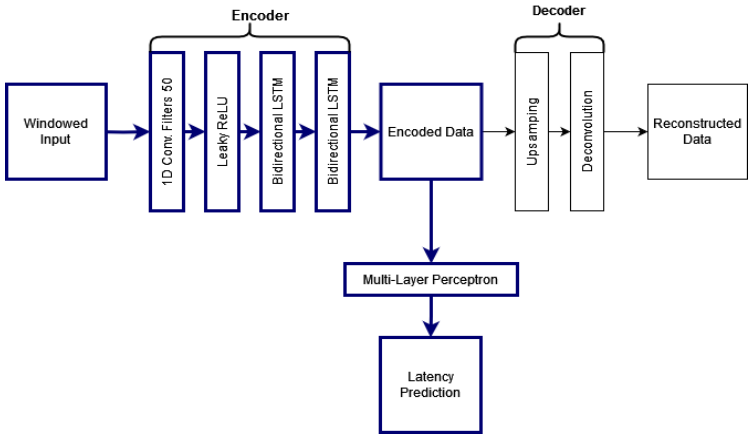

Figure 3: Architecture of the deep temporal regression (DTR) model. Blue blocks are used at inference.

model that can be used for well-informed routing choices.

For the ML agent to be able to make decisions as to where to re-route data, a regression model must be developed as this allows for the prediction of a continuous variable after training. This means the unsupervised clustering algorithms that were originally considered were not suitable for this problem as clusters would require classification in post of what clusters merit re-routing. To add label classes of link capacity means that the model cannot learn when re-routing decisions can be made. The concept was to use the model on a node to use its limited view of its own node edge statistics to predict the increase to the latency over the full path, and so the algorithm needs to be able to learn how temporal features of the network statistics affect the full path flow properties. These requirements led to the development of the deep temporal regression (DTR) model with the architecture shown in Figure 3.

TRAINING

The chosen use cases did not have readily available datasets for ML training. As such, datasets had to be created as part of this activity that were suitable for the scenarios.

Spatial Acquisition Training

The data required for training and testing were images recorded from the camera sensor that were representative of what would be seen during the acquisition phase. This required the presence of several key features: the incoming laser beacon at different intensities and positions, and different noise conditions.

The primary dataset for this use case was generated semi-synthetically from a source of two sets of 1000 images from an InGaAs camera sensor, provided by ESA. The first set of 1000 grayscale images consisted only of a very low background noise level and could represent the ISL scenario with dark space in the background. The second set was from the same camera, but opened to let a higher level of light onto the sensor and this could represent the spacecraft to OGS scenario with the Earth in the background casting light onto the sensor.

The source images did not include the incoming beacon spot, so this required the addition of this feature synthetically using Python image manipulation code. This allowed for generation of large numbers of images for the dataset through precise control of the beacon features (shape, position, and intensity). Incoming beacon spots were added using a kernel that creates a spot in the required Airy pattern and could be adjusted to different sizes and levels of visible ring lobes. The primary lobe was scaled to where it was only a few pixels across on the image, approximately 3 pixels across at its brightest.

The initial dataset used for training consisted of two subsets of 2000 images from both the low and high background noise image source sets. For both noise sets this resulted in a subset with the spot only a few pixels across with no visible ring, and a subset with the intensity of the spot increased until the ring was visible over the noise. The SNR in this training dataset averaged around 1.04dB for the high background noise images, and around 17.596dB for the low background noise images, with the maximum SNR at 22.839dB. SNR was calculated using Equation 1, where ‘S’ is the highest pixel value of the laser spot, and ‘N’ is the average pixel value of the background noise of the full image.

$$SNR = 10\log_{10}(\frac{S}{N}) \quad (1)$$

From this dataset 70% of images were used for model retraining, 10% for model validation and 20% for functional testing (ML model inference, without the use of a testbench). The most important part of the YOLO model which allows for high quality object detection with a simple network is the loss function. The loss function is a combination of mean squared error for the bounding box offsets; binary cross entropy for objectness score; and sparse categorical entropy for class prediction.

In addition to the synthetically generated dataset, another dataset was originally created using the CPL optical test bench which consists of the components of a CPL APT system arranged on an optical breadboard. Using a fast-steering mirror and optical attenuation filters, a dataset was created that showed the incoming beacon at different intensities and positions in different background noise conditions, as well as in the presence of an outgoing beacon which created blinding effects on the sensor. Various camera conditions were also included, showing levels of dead and hot pixels. Another version of the same ML model was trained and tested on this dataset.

However, the use of the optical test bench for performance testing was deemed to be too far from real conditions in terms of accuracy and relevance, so this dataset was subsequently used to only reinforce the adaptability of the ML model to different conditions for training instead of specific performance results.

The synthetically created dataset was the primary source for performance training and testing.

Network Healing Training

The datasets used for training the model were created using the CGI network simulator which consists of a Linux machine running Mininet. Mininet is an open-source network emulator which can emulate switches, hosts (data endpoints), and SDN controllers, along with aspects of network links and data flow. The simulator had an Open Network Operating System (ONOS) controller application which enabled traffic control and operated as an SDN controller. The emulated switches were in Open VSwitch (OVS) format and have flow tables which set the rules of which ports traffic should flow through at any given time, and these can be altered either by the SDN controller or by manual OVS commands.

The required data from the network to build the datasets are as follows:

- Switch node port statistics: bytes sent and received, buffer backlog, errors. Viewing total bytes sent/received over time allows patterns to be observed in the traffic at a node. For example, a reduction in the rate of bytes being received over time may indicate some degradation.

- Full path flow statistics: latency, jitter, throughput. Measuring full path latency allows patterns in behaviour observed at a node to be compared to the latency of the full path. For example, for the reduction in bytes received at a port discussed above, if there is an increase in latency at the same moment, this confirms degradation of the link, and not just a natural reduction in traffic flow.

Simulation runs were also performed on the simulator with variations on the following network parameters: starting throughput, level of throughput reduction, pattern of throughput reduction (steps or single drop), statistics sampling rate, simulation time and network topology. Two topologies were used for statistics gathering: a complex 8 node network for collecting statistics from multiple interfaces at once, and a simple topology of two nodes connected in a chain for mapping link statistics to latency. Statistics were collected from each side of network links to highlight any traffic patterns. The simulation runs collected for training included a range of starting throughputs and level of throughput reduction, in addition to different patterns of throughput reduction (single or multiple steps). The dataset also included differentiations for distinguishing between healthy throughput reduction (no latency increase) and degraded link quality (latency increases as throughput decreases). Starting throughputs varied from 100MBps to 2000MBps in 100MBps increments. In total there was 62 simulation runs ranging from 90 to 400 second duration.

To train the DTR model, there are two loss functions that are minimised simultaneously: the mean squared error between the input data and the reconstructed data from the TAE, and the mean squared error between the true latency value and the prediction from the MLP. Hyperparameters of the model were tuned using a grid search to find the optimal parameters. Using the dataset of 75,774 data points once it has been windowed which was split 70/20/10 into training, testing, and validation sets, an R^2 score on the test set was 0.595.

Training the model also involved the development of the software for the Agent applications. The ML Agent software was constructed from Python code and packaged into one program that could initialise and monitor the function of the Agents on the CGI simulator. The Agents can interface with the network to collect statistics from network node ports and have a stored map of the topology of the network. Once internal machine learning inference and path decisions have been made, the Agents can then enact updates to switch flow tables using OpenVSwitch commands.

TEST SCENARIOS & RESULTS

Specific scenarios were created for the purposes of performance testing of the machine learning models. The test results will be described following the outline of the test scenarios.

Spatial Acquisition Testing

Twelve simulated datasets of 500 images each were generated specifically for performance testing this use case: three levels of background noise (low, medium, and high), with each noise level having four intensity levels of the incoming beacon ranging from only the central lobe being visible and a few pixels across to bright central lobe and clear first ring lobe. To generate the medium noise images, a naïve method was selected for brevity where the high noise images had their intensities halved and the pixels shuffled to ensure that the images were random and simulated being drawn from a completely new distribution. The purpose of this was to test the performance of the model across a range of sensor conditions.

Examples are shown of each level of background noise in Figure 4. The spot intensities are at all at the highest spot level for comparison in this figure, and the images have been zoomed in to show the details of the spot. The full images are sized at 888x888 pixels. For the datasets these specific images belong to the average signal to noise ratios are as follows: (1) – 15.28dB, (2) – 4.14dB and (3) – 1.31dB.

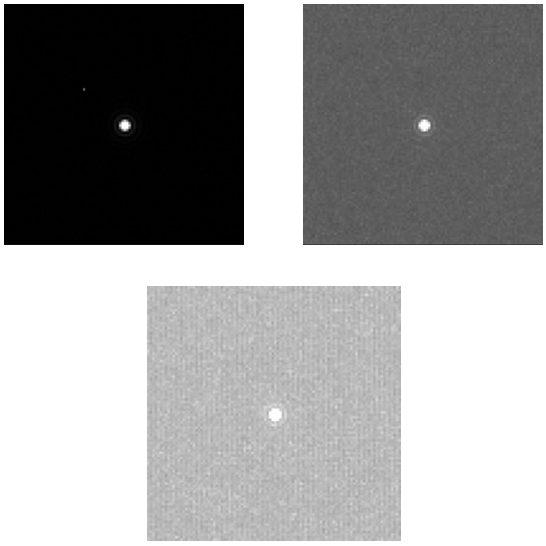

Figure 4: Examples of synthetically created images used for testing. Top left (1) – low background noise, top-right (2) – medium background noise and bottom-middle (3) – high background noise. All beacon spots have been set to the highest intensity at spot level 4.

The average SNR values for the rest of the datasets are as follows:

Table 1: Summary of average SNR values for synthetic datasets based on noise level and spot intensity level.

	Low Background Noise	Medium Background Noise	High Background Noise
Spot Level 1	14.02dB	3.14dB	1.19dB
Spot Level 2	14.13dB	4.03dB	1.23dB
Spot Level 3	14.82dB	4.07dB	1.31dB

The datasets were put through the ML model to perform inference which outputs bounding boxes around where the incoming beacon has been detected. Then the CoG algorithm was used on the ROI within the bounding box to find the centre of the spot. The pixel distance from the centre of the spot detected by ML assisted CoG algorithm to the true spot was measured. This was then compared to that of the pure CoG algorithm. Overall accuracy of the model was also measured.

Model accuracies were calculated using the intersection over union (IoU) metric and are shown in Table 2. Accuracy at IoU > N is the percentage of detections that have an IoU value greater than that 'N' value.

Table 2: Average IoU and Accuracy values for the YOLOv3 Tiny model.

Metric	Average	Maximum	Minimum
IoU of Detections	0.757	0.850	0.569
Accuracy at IoU > 0.7	0.636	0.931	0.183
Accuracy at IoU > 0.8	0.952	0.772	0.165
Accuracy at IoU > 0.9	0.336	0.574	0.078

Figure 5 shows the IoU at different SNR values. An IoU score of 0.7 and above is considered a positive detection as is standard in object detection scenarios. Accuracies over different SNR values are shown in

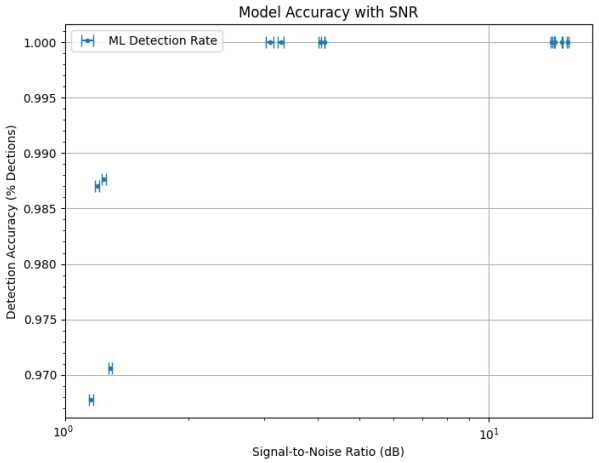

Figure 6. The accuracy values differ between Figure 5 and

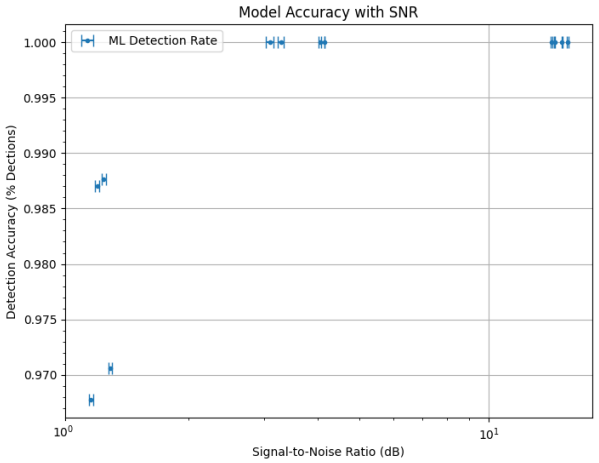

Figure 6 due to different definitions of a detection. For

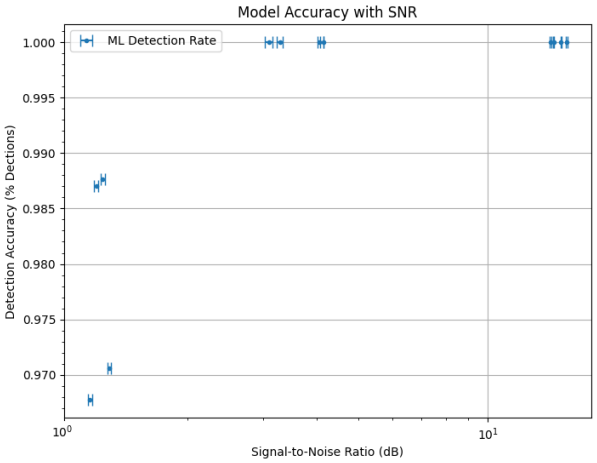

Figure 6, a detection is classified as successful if the laser spot exists anywhere within the bounding box therefore allowing the centre of gravity algorithm to be used to locate the spot centre.

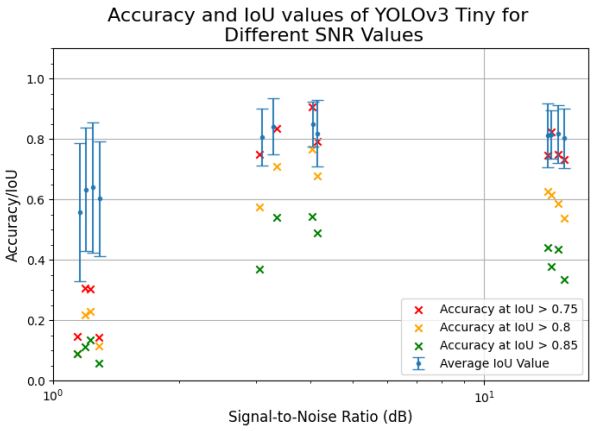

Figure 5: Average IoU and accuracy variation with SNR.

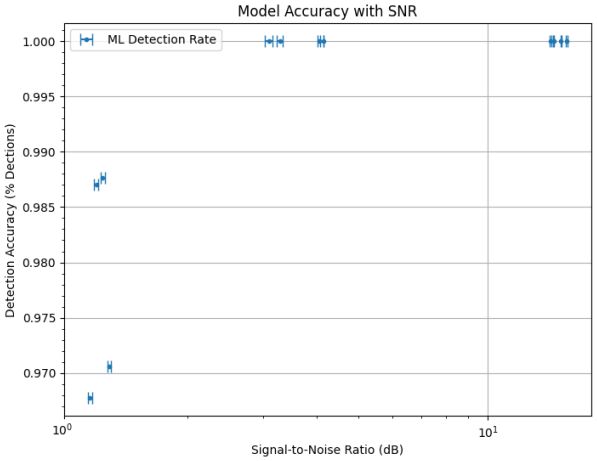

Figure 6: Model accuracy variation with SNR. Overall, for the full SNR range tested, the average ML model detection accuracy was 97.77%.

A plot comparing the ML YOLOv3 assisted CoG results to the pure CoG baseline algorithm for spot detection accuracy is shown in Figure 7. The pure CoG algorithm is configured with a handpicked threshold equal to 70% of the maximum image signal value (16bit images). The YOLOv3 ML model performs detection and outputs a detection bounding box of fixed size of 25px by 25px. This bounding box is treated as a region of interest over which the same CoG algorithm is applied to extract the spot centre pixel position in the ML assisted case.

As can be seen the ML assisted algorithm detections outperform the pure CoG algorithm in low SNR regions by over an order of magnitude better detection accuracy (based on the distance from the true spot). The CoG misdetections are resultant from the noise being above the threshold used by the CoG. The distance from the true spot signal centre of the ML model is below 9px and averages below 5px, this is likely due to the relatively big bounding box pixel size (25px by 25px) used for detection compared to the spot size (approximately 3px by 3px) over which the CoG is applied and affected by the noise.

For higher SNR conditions (> 3dB in the used dataset) the pure CoG algorithm and the ML assisted algorithm perform equally well which is expected as the noise levels are below the CoG threshold used.

The ML model success rate of detection per SNR from these results is shown by the detection accuracy in Figure 7. As the SNR lowers for the case of high background noise in the test images the ML model detection accuracy is above 90% allowing to boost the beacon acquisition detection rate in low SNR conditions compared to the pure CoG algorithm. Overall, for the full SNR range tested the average ML model detection accuracy was 97.77%.

As discussed at the end of the Spatial Acquisition Training section, the dataset created on the optical test bench was also used for testing the robustness of the model to different conditions prior to the focus on the synthetically created data. The test compared the detection distance from the true spot over a whole spiral of the fast-steering mirror performed for different outgoing laser powers for a pure CoG algorithm and ML assisted algorithm. The CoG algorithm used a threshold equal to 95% of the maximum image signal value (8bit images). The ML model bounding box pixel size (50px by 50px) was used for detection.

For a test run with 5% pixels in the image containing hot and dead pixel noise, it was shown that under all outgoing blinding laser powers the ML assisted algorithm can detect the incoming laser spot with an approximate spot distance of 8px to the true spot and average detection success rate of 80% for powers lower than 0.189mW.

As the powers of the outgoing blinding laser increased beyond 0.189mW the ML model detection rate decreases to an average of 75% and the distance from the spot varies up to 35px. Nevertheless, the ML model detection of the spot outperforms the pure CoG algorithm on all outgoing blinding laser powers used.

The inclusion of these additional test results showcases the versatility of the chosen machine learning model to different optical conditions.

Network Healing Testing

The aim of performance testing was to analyse the performance of the ML Network Healing Agents in the act of rerouting traffic in the presence of a single network fault. Then the performance will be compared to that of an SDN intent-based controller in the same scenarios. The Agents were deployed on each node in a created network topology in the CGI network simulator. For verification, they were configured to log collected traffic statistics, ML inferences, new path decisions, notifications sent and received, and any commands to update the flow rules on the switches. The throughput at the final node was also recorded to analyse the impact of throughput reductions on the traffic.

As with the training stage, different combinations of the network parameters were used to create scenarios for testing. The test procedure is as follows:

1. Configure network simulator and ML Agents.

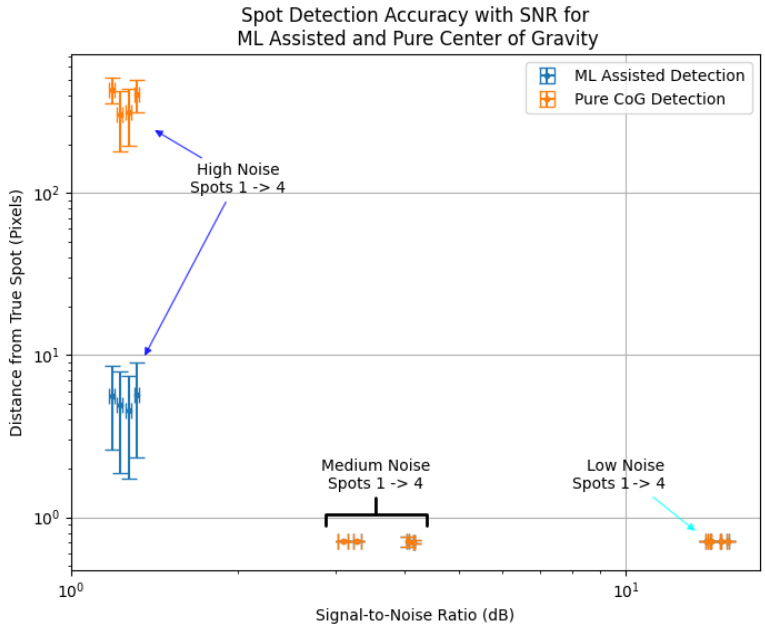

Figure 7: Pixel distance to the true spot centre (0.5px) for different SNR levels from the final test dataset when the CoG algorithm is used and when the ML model assisted algorithm is used. Note that some of the spot SNR levels are very close and as such are merged into a single data point.

2. Begin the test scenario.
3. Initiate degradation on specified link after 30 seconds of initial operation.
4. End simulation after 30 seconds of link degradation.
5. Repeat for all test scenarios.

The test scenarios were based on starting throughput, reduced throughput, and network topology. Table 3 shows the combinations of throughputs. These levels were chosen to give a wide spread of levels to analyse without requiring a large amount of time to be spent on generation of the test scenarios on the simulator. Each combination of throughputs will be performed once for each topology, giving 24 sets of results for both the ML and conventional SDN controller solution.

Table 3: Combinations of starting and reduced throughputs for performance testing.

Starting Throughput	Reduced Throughput	Starting Throughput	Reduced Throughput
2MBps	0.1MBps	1MBps	0.1MBps
	0.5MBps		0.5MBps
	1MBps		0.8MBps
	1.3MBps	0.5MBps	0.05MBps
	1.6MBps		0.1MBps
			0.25MBps
			0.4MBps

Different topologies were selected for the purposes of observing network behaviour in different conditions. The topologies varied in complexity and number of nodes. Figure 8 and Figure 9 show the chosen topologies.

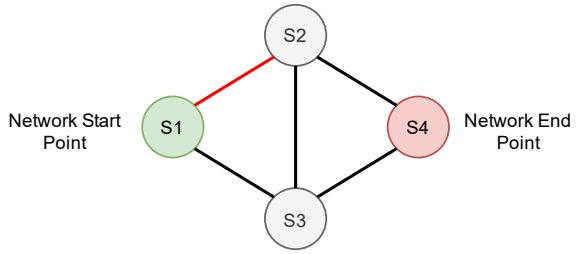

Figure 8: Topology 1 used for testing. Four switches with links shown. Red link represents link to have throughput reduced.

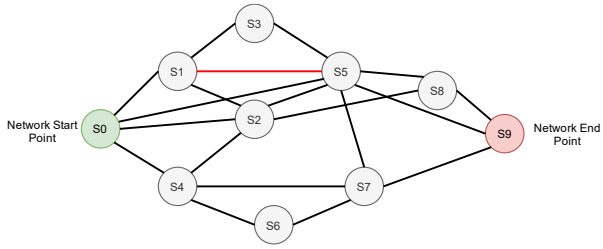

Figure 9: Topology 2 used for testing. Ten switches with links shown. Red link represents link to have throughput reduced.

The small topology shown in Figure 8 was chosen to observe the rerouting behaviours in a controlled and simple environment and get an initial impression of network behaviour when switching paths. The larger topology shown in Figure 9 has numerous links from each node. The larger network means that the ML Agent decisions can be analysed in a more complex environment with more alternative routing paths.

The chosen topologies are not representative of a full or real satellite constellation but represent a meaningful subset scenario, whereby an OGS with up to 20 satellites in view (as shown in Figure 1) may have cloud cover or other atmospheric interference blocking connection to some of those satellites. The ML enabled network switching could then be used to divert traffic from the satellite that needs to make connection with the OGS, to an intermediary satellite that does have clear view of the OGS to enable uplink/downlink as quickly as possible. Using smaller topologies in the testing goes towards addressing this scenario. To scale to a real constellation would also require more complex simulation and datasets. The subset scenario also enables the showcasing of the ML switching technology as a potential mitigation for the use of protected switching (another form of network redundancy) which can penalise network performance.

The main metrics for performance analysis are average throughput after fault injection and reaction time. Average throughput is the average of the bytes received per unit time at the final interface between fault injection and end of the simulation run. Reaction time pertains to the time between when throughput reduction was initiated, and the time that an ML Agent or the SDN controller makes some actionable routing decision. Reaction time is measured to gauge the responsiveness of the ML Agents as compared to the conventional SDN controller solution. The smaller topology will be compared to the larger topology for the ML solution and the conventional solution.

The results for average throughput after fault injection for topology 1 are shown in Figure 10 and Figure 11. Figure 10 shows the average throughput results for the starting throughputs of 1MBps and 2MBps. Figure 11 shows the results for starting throughput of 0.5MBps. The average throughput after fault injection is higher on average for the ML Agents than it is for the SDN Controller.

Across all test scenarios for topology 1, the throughput in the ML Agent scenario after the fault was injected was

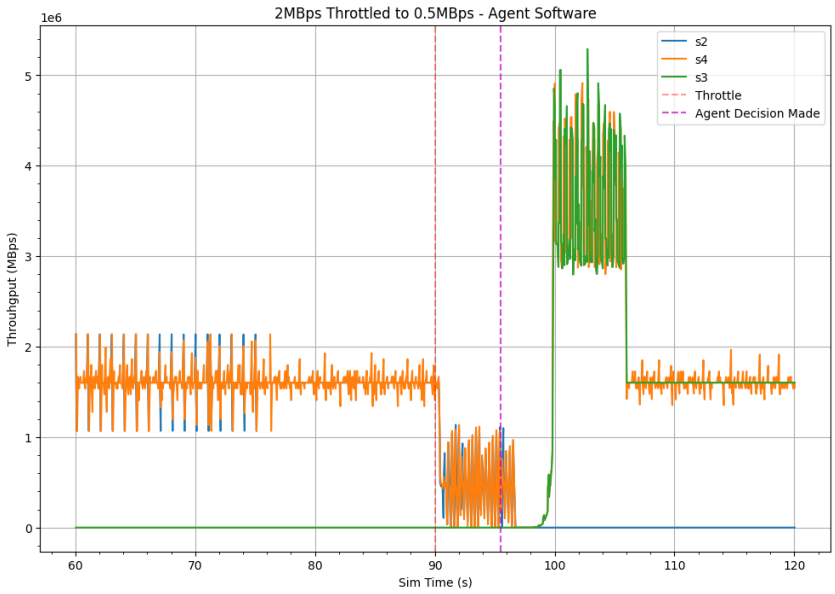

Figure 12: ML Agent response to throughput throttling (orange dashed vertical), with Agent decision marked (purple dashed vertical), and the different switch statistics responses highlighted. Topology one used.

an average of 456801.52 +/- 776.64 Bps higher than in the SDN controller scenario giving a percentage increase of 16.7%.

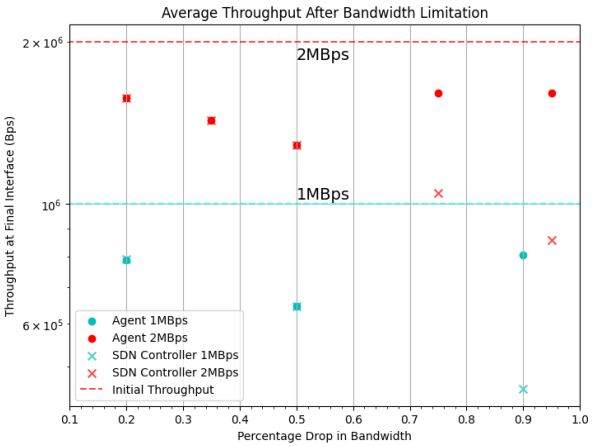

Figure 10: Average throughput after bandwidth limitation for ML agent response and SDN controller response. Grouped by starting throughput (1MBps and 2MBps) and percentage reduction in bandwidth. Topology one used. ML Agent performance improvement increases as percentage drop in bandwidth increases.

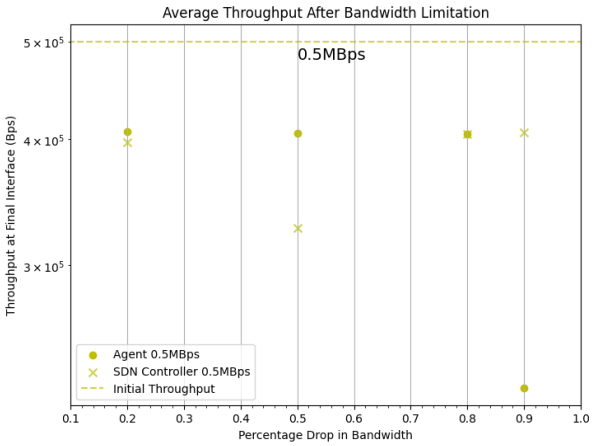

Figure 11: Average throughput after bandwidth limitation for ML agent response and SDN controller response. Grouped by starting throughput (0.5MBps) and percentage reduction in bandwidth. Topology one used.

The results for the starting throughput of 0.5MBps were separated out into Figure 11 as different behaviour was observed. The previous pattern of increasing throughput reduction resulting in decreasing average throughput in the SDN controller scenarios can be observed only in the first two datapoint sets. For throughput reduction of 80% and 90% it can instead be seen that one example is due to a simulation run where the SDN controller recovered from the fault, but the ML Agent failed, and the other is where both recovered but the average throughput was higher for the SDN controller after the fault was introduced.

For reaction time for topology 1, across all tests the ML Agents responded 39.2% faster than the SDN controller, average response time was 6.245s, and the fastest Agent response was 1.956s.

For 2MBps starting throughput and throughput then reduced to 0.1MBps and 0.5MBps the ML Agents responded in 5.685s and 8.464s respectively, whereas the SDN controller did not. For throughput reduced to 1MBps, 1.3MBps and 1.6MBps there was no response from either solution. For 1MBps starting throughput and throughput then reduced to 0.1MBps the Agent re-routed in 2.753s, with no response from the SDN controller. For throughput reduced to 0.5MBps and 0.8MBps there was no response from either solution. Finally, for 0.5MBps starting throughput and throughput then reduced to 0.25MBps and 0.4MBps only the Agents responded, with reaction times of 19.169s and 2.664s respectively. For throughput reduced to 0.1MBps, the Agents re-routed in 3.024s, with the SDN controller re-routing in 10.856s. For throughput reduced to 0.05MBps, the Agents re-routed in 1.956s, with the SDN controller re-routing in 8.013s.

Figure 12 shows a timeseries plot over the run for 2MBps starting throughput and throughput reduced to 0.5MBps. Each switch is highlighted, and the times of throughput reduction and Agent response are marked on the timescale. The initial path of S2 to S4 can be observed prior to the throughput throttling (marked by the orange dashed vertical line). In the plot, throughput is measured via the bytes received at a switch per unit time. The reduction in throughput in the S2-S4 path is observed in the signals and then the ML Agent makes a routing decision following its inference of the statistics being received (marked by the purple dashed vertical line). The time between throughput throttle and ML Agent routing decision is less than 6 seconds. Following the routing decision, the blue S2 signal drops to zero and the signal on S3 instead begins to ramp up, indicating a change in flow rules for a new S3-S4 path. The gap between S2 dropping and S3 starting is attributed to an artifact of how the flow rules are being implemented. The sharp increase in the throughput signals of both S3 and S4 is attributed to the buffers on the switches sending data being emptied quickly as the route is re-established following the switch. After approximately 6 seconds, the buffers return to normal, and the updated route can be observed at the original throughput level.

For topology 2, the results for average throughput after fault injection are shown in Figure 13 and Figure 14. For starting throughput level of 2MBps in Figure 13, the first 3 datapoints indicate that there is no response from the ML Agents as the results are the same as the SDN controller. Datapoints 4 and 5 show improvement in the average throughput over the SDN controller, which is following a steady downward trend. However, compared to Figure 10, the average throughput increase for ML Agent response over SDN controller response is lower than for the smaller topology. The same is true for the starting throughput of 0.5MBps, which only sees improvement in the fourth datapoint. This indicates the ML Agent currently is not performing as well in the larger topology.

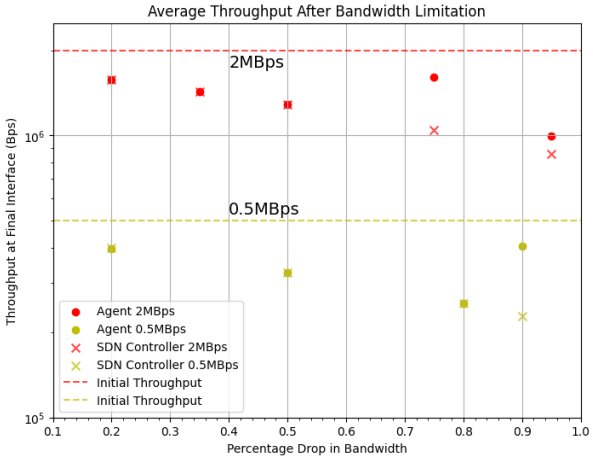

Figure 13: Average throughput after bandwidth limitation for ML agent and SDN controller. Datapoints are grouped by starting throughput (0.5MBps and 2MBps) and percentage reduction in bandwidth. Topology two used.

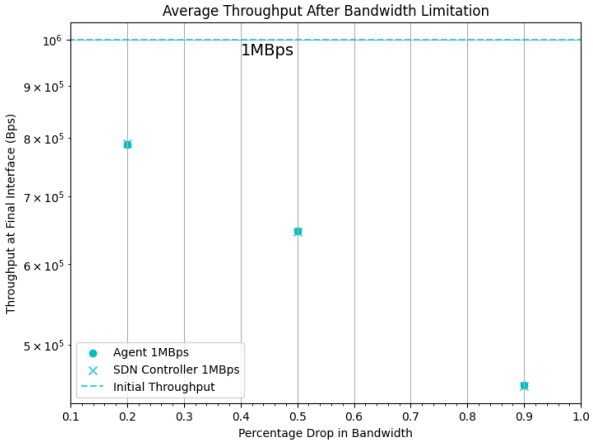

Figure 14: Average throughput after bandwidth limitation for ML Agent response and SDN controller response. Datapoints are grouped by starting throughput (1MBps) and percentage reduction in bandwidth. Topology two used.

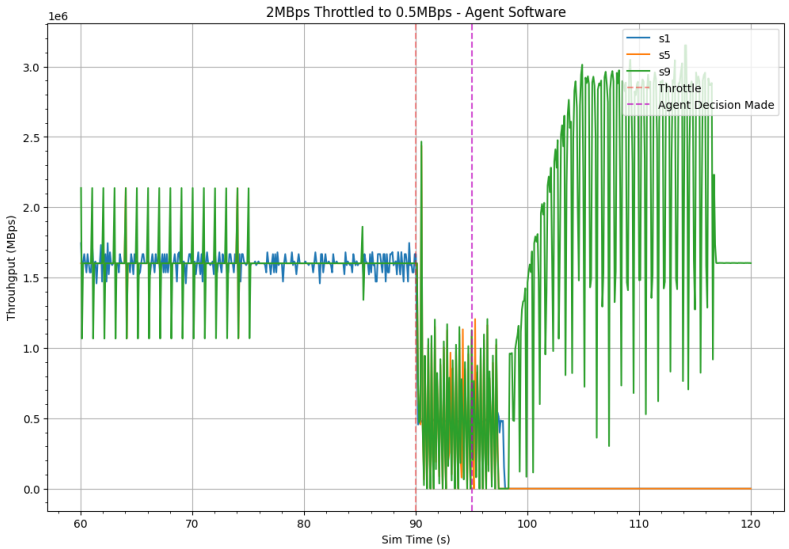

Figure 16: ML Agent response to throughput throttling (orange dashed vertical), with Agent decision marked (purple dashed vertical), and the different switch statistics responses highlighted. Topology two used.

Figure 14 shows that for a starting throughput of 1MBps, no single data point had improvement over the SDN controller response. However, from analysis of the log files recorded by the Agents, for the 50% and 70% drops in throughput, the Agents did detect the degradation. The issue instead lies with the re-routing path selection, as the larger number of possible routes in the larger network is causing the Agent to not calculate the shortest path correctly in these cases. This shows that the ML model used to detect the degradation in network link quality is performing better than can immediately be seen in the throughput plots, and that improvements could come from updating the path selection algorithm for larger networks.

An example of this is seen in Figure 15 which shows an excerpt from the Agent log on switch S5 during the test with 1MBps starting throughput with 50% throughput drop. It can be seen from the successive statistics windows that from one timestep to the next the Agent recognises an increase in latency resulting in a larger latency weight for the graph. However, the same path is calculated after the latency increase.

```
2023-11-23 11:29:31.821.
[[0.00335287]]
new ['s0', 's1', 's5', 's9']


2023-11-23 11:29:33.521.
[[0.82917875]]
new ['s0', 's1', 's5', 's9']
```

Figure 15: Excerpts from Agent log files showing detection of increase in latency with no path update. Information shows successive timesteps, an increase in the latency parameter in double square brackets, and the same path before and after the latency increase.

Overall, across all the datasets the ML Agents had an average throughput of 72.9 +/- 0.406 kBps higher than the SDN controller tests which translates to an increase of 12.21%. This is a smaller average increase than for the results for the smaller topology, but it does still show an improvement over the SDN controller's performance.

Reaction time results on topology 2 show that across all tests the ML Agents responded 14.3% faster than the SDN controller, average response time was 12.858s, and the fastest Agent response was 5.078s.

2MBps starting throughput tests the following reaction time responses were recorded: for reduced throughputs of 0.1MBps and 0.5MBps the ML Agents responded in 19.653s and 5.078s respectively, whereas the SDN controllers did not respond. The 0.1MBps datapoint was significantly slower than its counterpart in the topology 1. For reduced throughputs of 1MBps, 1.3MBps and 1.6MBps there was no response from the Agents or the SDN controllers.

For the 1MBps starting throughput tests there were no tests where the ML Agent reacted with routing decisions. Finally, for the 0.5MBps starting throughput tests the following reaction time responses were recorded: For the 0.05MBps reduction level, the reaction time was 13.844s. The SDN controller did not respond. For all

other examples there was no response from either ML Agent or SDN controller.

Compared to the reaction time results for 0.5MBps starting throughput for Topology 1, the performance is worse for the larger topology. Though this example also shows that the SDN controller did not respond to any of the throughput reductions whereas in the smaller topology examples it responded to the 0.05MBps and 0.1MBps reduction levels. This indicates that while the ML Agents are struggling more with the larger topology, they are still an improvement over the intent based SDN controller.

Figure 16 is an example of successful switching behaviour in the larger topology for the 2MBps to 0.5MBps throughput scenario. The original path in this scenario was S0 (host 1) – S1 – S5 – S9 (host 2), and the plot shows the throughput via received bytes at S1, S5, and S9. Five seconds after the throughput is reduced, the ML Agent recognises this degradation and makes a routing decision, marked by the purple dashed vertical line. The path decision made by the ML Agent is to S0 – S2 – S8 – S9. This can be seen in the plot as the signal lines for S1 and S5 drop to zero, and the signal for S9 ramps up as the buffers empty as the path is re-established and then returns to the original level.

CONCLUSIONS

To summarise, the aim of this activity was to demonstrate ML models that begin to tackle specific challenges in OCS, highlight the advantages ML can bring and analyse the feasibility of their use in these systems.

Over the two use cases which have been discussed throughout this document, there were several successes that can be taken forward and developed into technology ready to be used in an OCS which are summarised below.

It has been shown that machine learning can be used to improve signal acquisition when using pixel array detectors by reducing the region of interest over which to apply the CoG algorithm. These results need further exploring but it has been demonstrated that, at SNR values between 1-1.5dB, the ML assisted acquisition method shows an improvement of over an order of magnitude compared to a standard CoG method.

For the network healing use case with ML network Agents, improvements were seen in the overall throughput at the end point of the network when compared to a COTS SDN controller. This is a beneficial outcome as it means that the quality of service for the end user will be improved when using the ML agents to reroute traffic around a compromised link. This end user service would also be improved through the improvement of reaction time when using the ML agents. The agents were able to react to the fault in the network in less than 10 seconds and in some cases less than 5 allowing for this improved end user service.

The main challenges in this activity were associated with the development of custom datasets, and the availability of relevant test benches, specifically for optical data. Future work for the networking use case could start with production of higher fidelity synthetic data through simulation of a larger network with more realistic fault injection. Through this, the impact of the ML solution could be better measured when considering OGS handover. For spatial acquisition, synthetic datasets could be further developed, or real optical data could be incorporated. This would then lead to tests combining the use cases, through the creation of a simulator test bench that combines networking and optical communication.

Despite these challenges, the results of this feasibility activity have shown that there is a place for the use of machine learning in networked optical communications systems. A system that applies machine learning in both areas effectively would enable efficient and accurate acquisition of the optical ground station signal and optimal routing of traffic to the ground, effectively maximising the time connected to the ground during an available downlink window.

REFERENCES

1. ESA, “Developing Future Optical High-Capacity Satellite Networks: Hydron (High Throughput Optical Network)”, Available at: Developing Future Optical High-Capacity Satellite Networks: HydRON (High Throughput Optical Network) | ESA CSC
2. Joseph Redmon and Ali Farhadi, “YOLOv3: An Incremental Improvement”, Available at: https://doi.org/10.48550/arXiv.1804.02767, 2018
3. GluonCV Toolkit, “Gluon Model Zoo Documentation”, Available at: https://cv.gluon.ai/model_zoo/detection.html
4. Kumar et al, “YOLOv3-Tiny: Object Detection and Recognition using one stage improved model”, DOI: 10.1109/ICACCS48705.2020.9074315, 2020
5. Gai et al, “An improved Tiny YOLOv3 for real-time object detection”, In: Systems Science & Control Engineering: an open access journal, vol. 9, no. 1, 314–321, 2021

6. Madiraju et. al., “Deep Temporal Clustering: Fully Unsupervised Learning of Time Domain Features”. Available at: https://arxiv.org/abs/1802.01059, 2018

7. Mabaso et al, “Spot Detection in Microscopy Images Using Convolutional Neural Network with Sliding Window Approach”, In Proceedings of the 11th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2018) – Volume 2: BIOIMAGING, pages 67-74, DOI: 10.5220/0006724200670074

8. Ochoa-Aday et al, “Self-healing and SDN: bridging the gap”, in Digital Communications and Networks, Volume 6, Issue 3, Pages 354-368, 2020