Detecting Satellites with Object Detection: Challenges of Implementing Deep Learning Techniques for Space-based Images

Shane Ryall
Defence R&D Canada Ottawa, Canada

ABSTRACT

This paper details an experiment into the performance and implementation of using deep learning object detection models for satellite detection, on imagery taken from space-based Space Situational Awareness (SSA) sensors. While previous research in the field often relies on simulated or ground-based imagery, this research fills a much-needed gap in research on DL/SSA by addressing the specific difficulties encountered in space-based imagery, which is in short supply, and offers a realistic assessment of deep learning's potential in operational SSA contexts.

Over the course of the experiment, we compared the accuracy of a trained YOLO (You Only Look Once) neural network model against a concurrently developed conventional image processor, using a dataset of >30,000 labelled images from the NEOSSat microsatellite. This dataset, acquired as a part of the ongoing Canadian Satellite Tasking List (CSTL) catalogue maintenance experiment, provides a real-world test environment for our models in which the model would have to learn to deal with noisy and more realistic quality imagery.

The paper walks through the methodology taken for implementing a neural network for Resident Space Object (RSO) detection, exploring potential options for integration into an image processing loop for astrometric exploitation, and compare the performance of this method against our non-ML processing and RSO detection algorithms.

Ultimately, the experiment found that while object detection models are rapidly improving and highly effective for automated ground-based systems, from a pure accuracy point of view, deep learning techniques still underperform against segmentation-based processors. When trained on raw images, a YOLOv8 nano model was only able to achieve a maximum mAp50 of 78% in detecting at least 1 correct RSO per image. The main issue encountered was the rate of additional false positive detections on cosmic rays or broken streaks, where subsequent processing algorithms were required to discern and filter the detections. The ideal use for deep learning would be streak detection or RSO detection in low-risk clean images, where the target is not over exposed to undue noise or easily obscured by other objects in the image. Future experiments will explore methods of integrating the best of both methods in a hybrid processor, as opposed to exploring pure solutions.

1. BACKGROUND

An exponential increase in satellite launches and mega constellations over the past few years has led to a rapid growth in orbit congestion, especially in lower orbital regimes. This growing concern has made space surveillance and space situational awareness increasingly vital for ensuring the safety and functionality of these assets. This means there is an ever-increasing requirement for rapid processing pipelines to intake imagery and detect space objects at in a wide range of government and commercial data, with sufficient accuracy for both astrometric and photometric analysis. One of the challenges that current research into SSA image processing faces is that there are few space-based sensors that can provide open-source data to research organizations, with most of the current satellites locked behind militaries operational classifications, which makes development of computer vision algorithms for these systems more difficult.

Canada and DRDC are in a unique position to meet this challenge as currently DRDC operates one of these few SSA research satellites; NEOSSat, with an another SSA satellite Redwing, due for launch within the next 2-3 years. The DRDC-owned Near-Earth Orbit Surveillance Satellite (NEOSSat) is a research microsatellite telescope designed for SSA experimentation and asteroid astronomy. Jointly operated by DRDC and the Canadian Space Agency, NEOSSat shares satellite time on a rotating schedule between the two organizations, limiting observation frequency and duration. NEOSSat is mainly scheduled on publicly available TLEs produced by the SSN.

NEOSSat is in orbit at 785km and is equipped with a 15-cm optical telescope, an E2V 47-20 science CCD, and is capable of imaging objects to within 45 degrees solar elongation of the Sun. RSOs are primarily imaged using Track

Rate Mode (TRM), a satellite imaging method in which NEOSSat's camera slews at the angular rate of the satellite. This technique detects objects as points rather than streaks moving across the frame. Sequences of these images are then stacked to reject random energetic particle radiation signatures and enhance the RSO signal-to-noise ratio. This approach also provides a limiting visual magnitude of 16 when stacking NEOSSat imagery during image processing [2]. Track rate mode images of GEO objects are the main image source and type used in training and evaluating the models in this experiment. An example of NEOSSat imagery can be seen in figure 1.

A secondary objective behind this experiment was to compile archived imagery from past NEOSSat missions into a comprehensive training set for labelling and use in future deep learning experimentation. The bulk of the image products were produced as a result of the CSTL experiment, an ongoing investigation into the feasibility of maintaining an active satellite catalogue with NEOSSat, to see if Canada possessed the ability to maintain sovereign custody over Canadian owned satellites in orbit. During this experiment, which took place over the past 3 years, NEOSSat continuously monitored and imaged a list of 27+ GEO satellites, which included several ANIKs and Telstar satellites. A full list of the satellites monitored can be found in ref [2]. As of July 2024, approximately 30000+ images have been successfully processed and converted to labeled data by our legacy image processor, and ~10000 images are added to the dataset yearly.

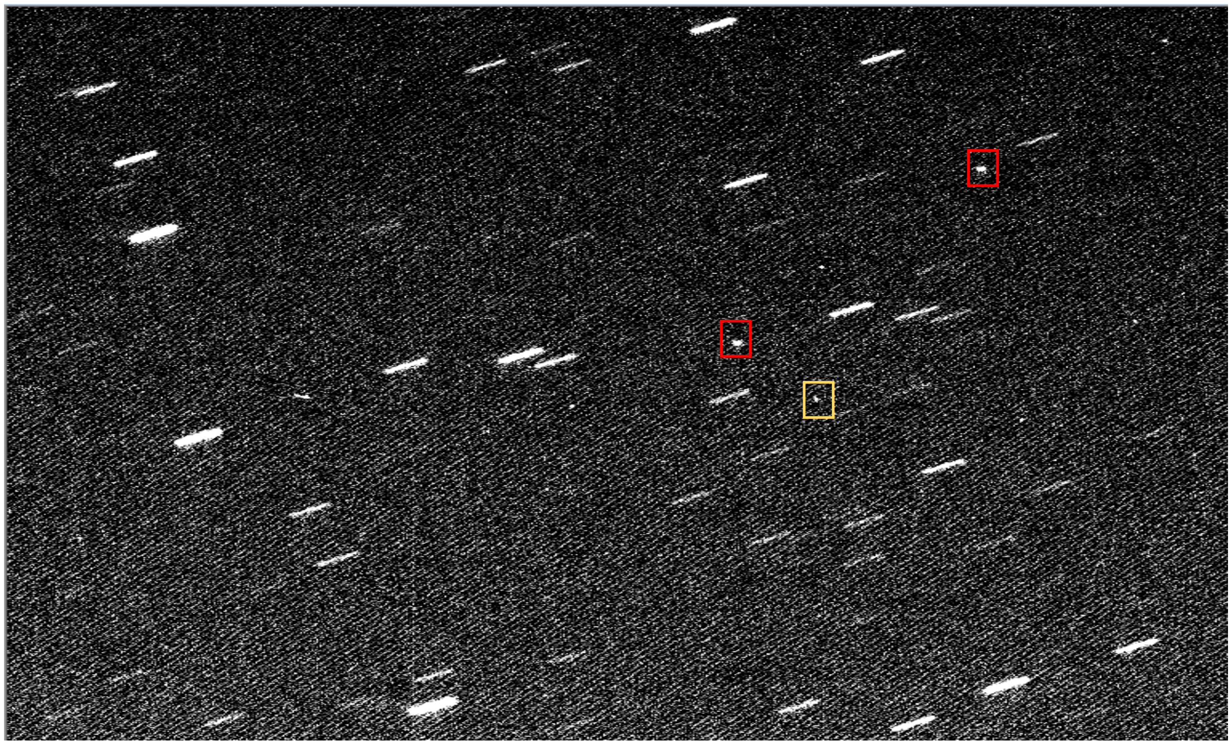
Figure 1. NEOSSat image in Track Rate Mode (TRM) The red squares are satellites, and the yellow square is a cosmic ray.

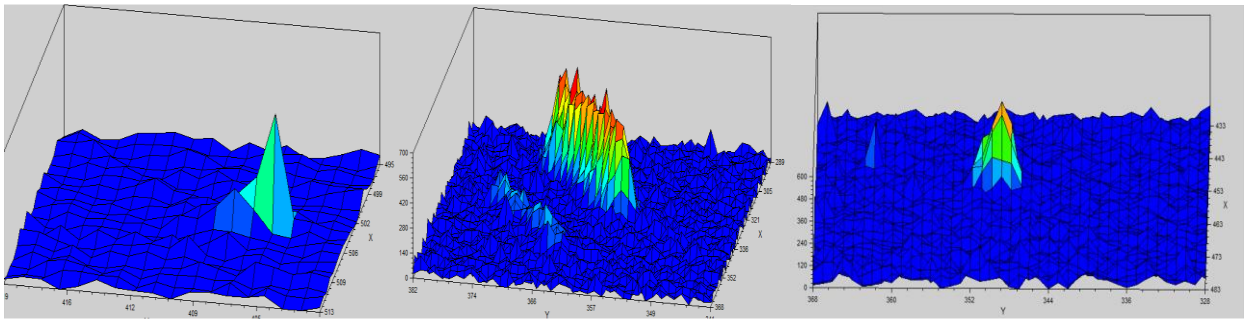
Figure 2. Space object patterns brightness vs pixel length. (Left) Cosmic Ray, (Center) Two streaks, (Right) Satellite

Figure 2 illustrates the pattern similarities between objects in a TRM image, specifically between cosmic rays and RSOs. Cosmic rays are high-energy particles that interact with the CCD, which cause sharp, sudden spikes in intensity across only a few pixels. In contrast, RSOs exhibit a Gaussian point-spread function (PSF) profile, where the intensity peaks at the center and gradually tapers off, blending into the background noise.

Current deep learning research for SSA processing is generally focused on RSO detection. As of the time of writing, the highest performing neural network architectures for object detection are YOLOv10 and Realtime Detection Transformer (RT-DETR). Object detection differs from deep learning classification models by, instead of classifying whether an image contains a specific class, the models aim to locate specific targets within the frame. YOLO works by dividing an image into a grid and predicting bounding boxes along with confidence levels for each grid cell. The key feature is that the model processes the entire image in one pass, allowing for faster real-time detection. A more comprehensive explanation of the YOLO architecture can be found in [3].

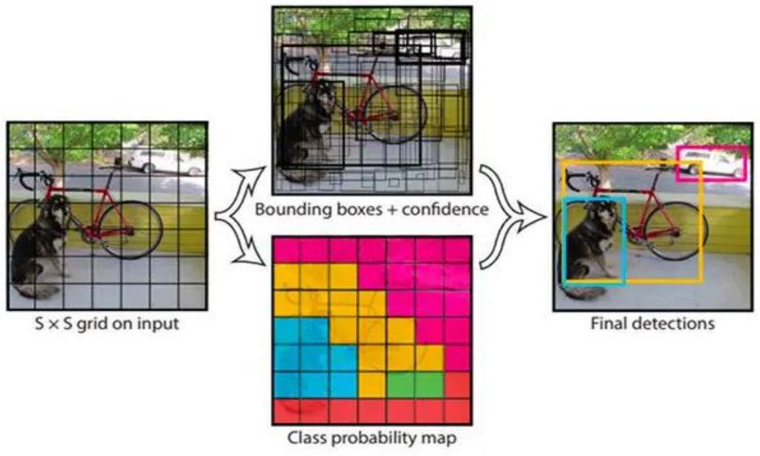

Figure 3. Example of class detection and localization using YOLO neural network

Several metrics are used to evaluate the performance of deep learning models in RSO detection, including accuracy, precision, recall, and mean Average Precision (mAP50 and mAP95). Commonly in deep object detection tasks, precision is prioritized to minimize false positives, in SSA it's more beneficial to tolerate a higher rate of false positives as long as the true RSOs are successfully identified. False positives can be filtered out in subsequent processing stages through pattern recognition however false negatives, where the network fails to detect any potential RSOs, are less useful and discarded due to minimal value. Mean Average Precision (mAP) evaluates the model's ability to accurately detect and localize RSOs across confidence thresholds.

2. EXPERIMENT STRATEGY

2.1 Data

The images used in this experiment are 768x768 pixel, single-channel, unfiltered grayscale images with a 16-bit dynamic range (brightness values from 0 to 65,535). The images are encoded to FITS (Flexible Image Transport System) standard, with a binning level of 1x1, which means that each pixel is sampled individually which maximizes spatial resolution. Exposure times vary between 5 and 10 seconds, and the CCD image with a gain setting of 1.1 electrons per Analog-to-Digital Unit (e-/ADU). The CCD readout noise is detected to be 8.0 electrons (e-), which adds some inherent noise to each observation due to the limitations of the sensor electronics.

Of the 30,000 images captured, all contain between 1 to a maximum of 3 RSOs. A minimum of one RSO per image was ensured since only images that were successfully processed by the conventional pipeline were included in the training dataset. Due to the method of NEOSSat's fine slew mode for tracking the satellite, the primary target RSO is generally located within ±15% of the center of the image.

The background noise in the images fluctuates between 30 and 150 pixel value, with an inherent sine wave pattern. Background stars in these images appear as elongated streaks due to the relative motion between the satellite and the target object. These streaks vary in orientation depending on the target's trajectory, with an average length of 25-35 pixels. Each image typically contains between 60 and 100 visible unbroken or semi-broken streaks.

False detections are primarily caused by cosmic rays, streaks that are clipped at the image borders, or faint broken streaks with circular fragments (eccentricity > 0.8). These artifacts present challenges for both conventional and deep learning-based detection models, and regardless of method employed post detection filtering and validation is necessary to eliminate these detections.

2.2 Conventional Image Processing

In order to assess where deep learning might best be integrated into a processor, the framework of the processing loop must be understood and tailored to the specific task in which the sensor is employed.

Our conventional processor, Birdseye, was developed as an iteration of a previous MATLAB based processor SQUID3, for use on the upcoming Redwing microsatellite. Primarily NEOSSat and Redwing are concerned with astrometric data. The ratio of processed images is less important than having a high degree of precision in the timing and location data of the target in the frame.

Conventionally SSA image processors are generally disjoint sets of functions, sequentially passing image data through each step as it is cleaned, amplified, and split into subcomponents for easier object identification. Part of the reason previous deep learning efforts have had a hard time achieving comparable performance is that the combination of dozens of smaller functions allows for a much finer hands-on control of the data as opposed to the black box approach of a neural network.

NEOSSat’s processing loop follows a specific cycle; image stacking, preprocessing, background removal, streak estimation and matched filtering, plate solving, satellite detection, and post-processing corrections. At the core of the image processors logic, it is just using segmentation techniques to identify the objects within the frame according to predicted patterns, and every other algorithm and function within the code exists to validate the shape of the detected object, clean noise and artifacts from the image, or amplify the signal to make it easier to detect above the noise.

The processor takes a 16 bit .fits image as input, and splits the image into the data and header metadata. The header data consists of the information associated with the image, the target, and the satellite modes and timings during imaging. The array (stacked or single) of image data is then fed into a background removal algorithm which is done either by iteratively sliding a window across the images and calculates the median background values as a product of the region of the image, as there can be several sources of noise fluctuations that naturally occur within the image.

The next step is to estimate and separate the streaks from the image. The streaks, or background stars make up a large portion of the bright pixels on the image, especially a space-based image which tends to contain more background stars than an image taken from the ground. The streaks should all have the same orientation and length as the streak shape is a function of the slew rate of the telescope and relative angle of the target to the observer, and are crucial for plate solving the image to identify precisely where in the sky the reference pixel is pointing. Conventionally the streaks are detected by segmenting and filtering objects that meet the same orientation across the image, with methods such as Radon transform and matched filtering applied to amplify the signal of the streaks for easier detection, especially for dim streaks near the noise floor.

Once you have the streak profiles, you apply centroiding methods to located the x and y coordinates of the weighted centroid of each streak, which is then converted to a proper format for insertion into a plate solving software which searches a star catalogue for the region of the sky which contains the stars in the positions you calculated. Regardless of RSO detection method, if you wish to use the RSO detection for orbit estimation you must implement some method of streak detection. Streak detection represents a possible avenue for deep learning exploitation as not all streaks need to be detected for a plate solve, they are more common than RSOs on the image, and have a unique and consistent shape that is generally not mistaken or obscured by other objects.

The final stage is RSO detection, once the streaks are detected and the image is plate solved, the processor subtracts the streaks from the background subtracted image data to leave an RSO-only image which can be fed into a segmentation algorithm or neural network to detect the RSOs in the image.

For orbit determination purposes, the processed observations must consist of angles-only data, specifically the Target ID, timestamp, right ascension, and declination of the RSO in J2000 coordinates, corrected for all aberrations. Accuracy of the centroid of the RSO is crucial, at the GEO range, even a single pixel off can translate to several arcminutes of error. Since object detection neural networks only output a bounding box for detections, we must perform subsequent segmentation and centroiding algorithms inside the box, to find the centroid xy-pixel location.

One of the key questions in deep learning implementation is at what processing point is ideal for network inference. Is it better or not to remove the background before training, before inference, or leave them as raw imagery? Ideally to have the highest likelihood of detection with a neural network you would train on single raw images which have a high degree of noise, and then infer on stacked images which have the cosmic rays and streak removed, so all that exists is the signal from the potential RSOs.

In this pipeline, deep learning algorithms have the potential to fit in several places; RSO detection, image cleaning, GANs for image simulation, streak detection, and post processing data exploitation.

2.3 Training and Tuning the Model

For this experiment we focused solely on training and tuning a neural network for RSO detection on raw images. The rationale being that the neural network should be capable of learning the inherent noise patterns in the images and filtering them out during the prediction and localization of RSO detections. We used the Ultralytics YOLOv8 framework, alongside PyTorch, as the primary object detection architecture for training and data preprocessing. Due to the Ultralytics package being optimized for training on 3-channel RGB 8-bit JPG and PNG images, modifications were required to handle our specific FITs image standard. The package had to be forked and modified in order to accommodate for our specific image data format. Rather than feed in the images, the image data was pre-read and loaded as 3x768x768 NumPy arrays.

An extra 2 zero channels were added to simulate a 3-channel image without inserting data that would throw off the learning process. Some papers have discussed compressing the data range from 16-bit to 8-bit as a method of speeding up and drastically simplifying the training process as the images can then be converted easily to PNG or JPG, however we have found that that this radically degrades the potential accuracy especially in space-based data, as the reduction from 0-65535 to 0-255 results in a significant loss of data resolution and patterns within the data, which is especially important when attempting to differentiate RSOs from cosmic rays and other very similar artifacts.

Due to limitations with the hardware, model and batch size had to be restricted to accommodate for VRAM limitations on the available GPU. This limitation potentially resulted in a loss of accuracy which could be attained with larger models, and is an avenue to explore in future experiments.

Data transformations were used to reduce overfitting and increase generalization as well as increase the data diversity and size of the dataset. Given the small size of the RSO, the majority of the data transformation preprocessing focused on physical transformations such as flipping, splicing, rotating, and avoided transformations such as stretching, color modifications, blurring, that obscured or changed the consistent circular pattern of the target.

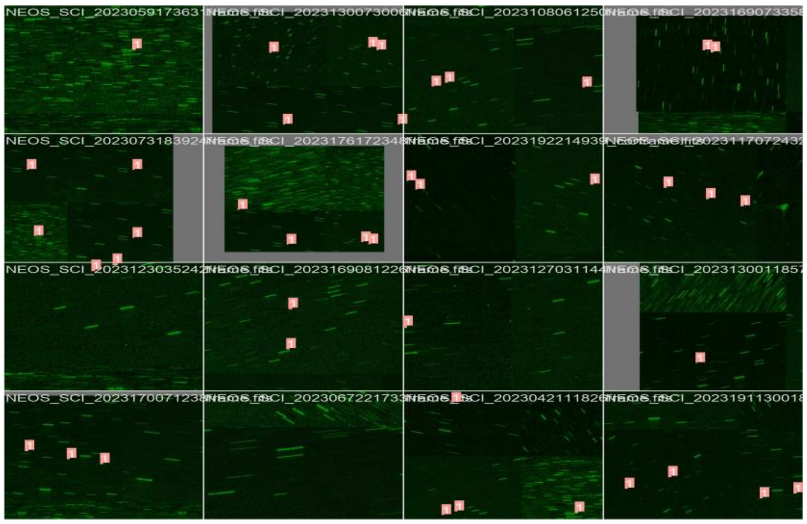
Figure 4. Examples of the transformed images and the RSO ground truths.

The hyperparameters tuned in this experiment were model size, batch size, optimizer, number of epochs, and learning rate. Model size was the initial hyperparameter tested. A basic hyperparameter configuration was trained on each of YOLOv8's 5 model sizes; n, s, m, l, xl. Although the patterns within the image are very sensitive to small deviations in pixel values and noise, SSA data might not necessarily require a complex model due to the consistency of object shape across images. This was reinforced by the results in Table 1 below demonstrating that the nano model, which has 1/20th of the parameters of the extra-large model, performed the best. In fact, there was a substantial degradation in performance solely by increasing the model size.

Table 1. Model performance vs Model Size

Model Name	**# Parameters**	**50-mAP (max)**	**Recall (max)**
Nano	3.2 M	0.717	0.69
Small	11.2 M	0.691	0.651
Medium	25.9 M	0.685	0.653
Large	43.7 M	0.628	0.645
Extra Large	68.2 M	0.643	0.629

Given the limitations on hardware, we took a manual approach to tuning, as automatic grid and random search tuning packages within Pytorch require a very high degree of compute and training time. A basic method of going through hyperparameters sequentially, identifying the best configuration for a particular hyperparameter out of a specific range, performed well at improving model performance. Once good hyperparameters for each are determined, go back to the start and re-tune the initial hyperparameters based on the highest performing complete configuration.

The next hyperparameter we tested was the number of epochs, which represents the number of times the entire dataset is passed through the network during training. Across 50 models, we found that performance increased with each epoch, reaching a stable plateau around epoch 30. The model's early convergence is likely a result of the simplicity of the patterns within the data and a reflection of the limited data diversity of patterns across images. Although our dataset is large, the RSOs and streaks share almost identical characteristics across images. Further increasing the number of epochs, increase the model's complexity and led to overfitting, where the validation accuracy began to degrade with additional epochs, showing that the model was memorizing the training set and not generalizing well.

Next batch size was tuned. Using the nano sized model and 50 epochs with a 10 epoch patience, we tested a number of batch sizes to gauge effects on model fit.

Table 2. Batch Size vs Validation Box Loss

Batch Size	Validation Loss (Box loss)	Batch Size cont.	Validation Loss cont.
8	1.34	128	1.35
16	1.21	256	1.48

32	1.21	512	1.67
64	1.29		

Batch size is the number of training examples the model weight's optimize against in one forward and backward pass. A small batch size allows the model to update its weights more frequently, which can lead to faster convergence but at the cost of noisier updates and more variability in training. This has benefit in optimization can sometimes help the model escape local minima but may result in less stable training. Conversely, a larger batch size can give more stable gradient estimates and smoother training, but requires more RAM, and slows down training and convergence since weight updates occur more infrequently. The ideal batch size was determined to be 16/32, which for a dataset and model size of our size is on the smaller end of potential batch sizes. Given the precision needed in detections and the consistency of RSO shape across images (high eccentricity), a lower batch size is preferred to reach a higher performance and lower loss values.

The choice of optimizer also plays a role in the optimizing the gradient and weights during training. Common optimization methods include Adam (Adaptive Moment Estimation), Stochastic Gradient Descent (SGD), and RMSProp. We tested several optimizers and while SGD was able to train the model vastly quicker than Adam (2-3x faster training speeds), Adam which combines SGD with momentum and RMSProp outperformed SGD in terms of convergence and lower loss values. Adam dynamically adjusts and schedules the learning rate for each parameter based on first and second moments of the gradients, allowing it to converge more efficiently, particularly on noisy datasets. This performed superiorly likely because for our data the information is highly variable due to noise from the cosmic rays, electronic noise, heat noise, and other artifacts.

Overall, the best hyperparameters for Yolov8 in our scenario were a nano model size, batch size of 16, 30 epochs, 10 patience, and data normalization to 0-1.

3. RESULTS

3.1 Metric Results

After manually tuning over ~50 models trained, our best model achieved a mAP50 of 0.81. It is likely that further gains could be made in performance by a more automated rigorous tuning process given enough time and compute, but given the constraints of this experiment that was not possible. The metrics of the final model are shown in figure 5 below.

The model shows steady decreases in bounding box regression loss (box loss) and distribution focal loss (DFL) over the 30 epochs of training. Box loss represents how well the model is predicting the size and location of the bounding boxes around the RSOs, and is the difference between the predicted bounding boxes and the ground truth coordinates. While the loss value itself is less important, the downward trend over the training iterations demonstrate the models increasing ability to narrow down the portion of the box that contains the RSO. The loss trends are mirrored in the validation loss which indicates that the model is learning to generalize, and not fitting itself to the training data.

Secondly, the model shows a steady increase across epochs in both precision and recall. This suggests that the network is learning and improving its detection accuracy throughout the training process. The increasing precision, which reaches 0.8391 by the final epoch, shows a reduction in false positives. Similarly, the increasing recall, which reaches 0.75102 at epoch 30, demonstrates that the model is getting better at discerning the actual RSOs present in the images. Additionally, the upward trend is balanced in both metrics implies that the model is able to generalize without overfitting to the training data.

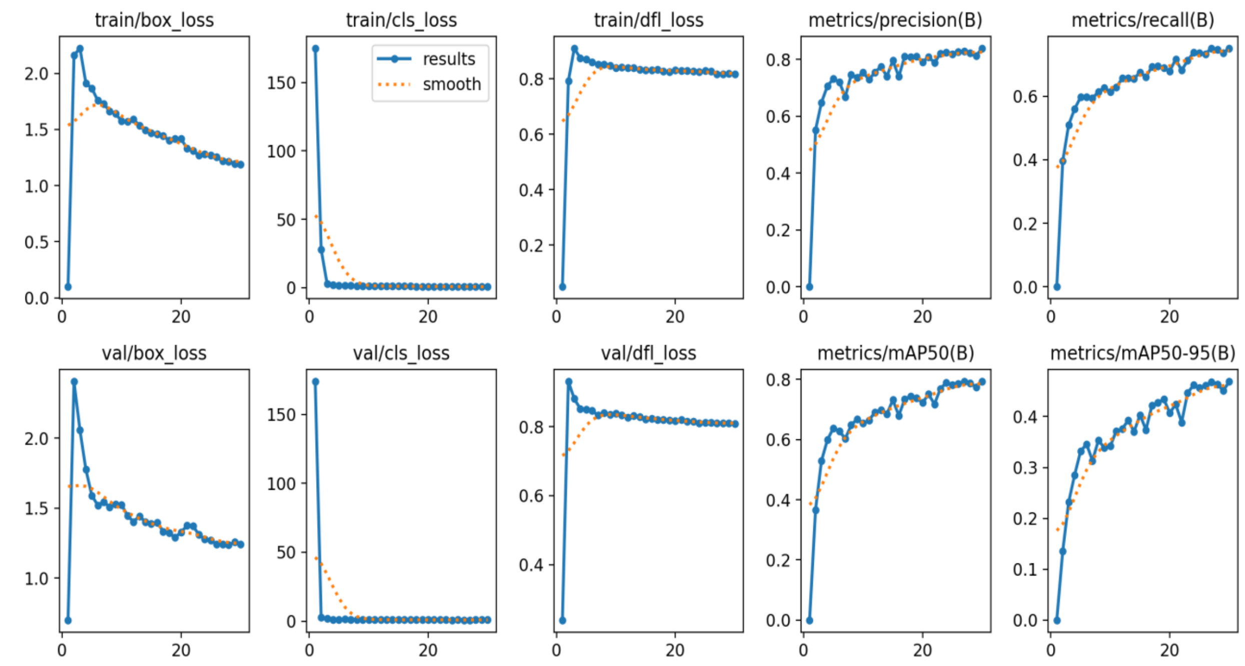

Figure 5. Training metrics of tuned model after 30 epochs.

The mAP50 and mAP95 show continually improvement as the model is trained, with 80% of satellites being able to be detected with at least a 0.5 confidence level. While we would ideally seek to have a high mAP95 which would imply that the model is certain of its prediction, given the small pixel area of an RSO, it is unlikely as even small common deviations in brightness of certain pixels within the PSF could throw off the model's prediction confidence.

A figure displaying sample detections during validation can be seen in figure 6 below. All 16 of the images have 1 true RSO in the center of the frame. During the validation inference, the model predicted the satellite location consistently with a high confidence interval, 0.7-0.8. However, it also frequently predicted several false positives, which may be cosmic rays or just clusters of increased noise. This was expected, and as discussed in a previous section, false positives can often be filtered.

The model fails in a few key areas, the most frequent of which being when the RSO is obscured in the frame. Space-based images tend to have a higher number of streaks, which makes overlap more likely. For the majority of the "good" images, the trained model is generally able to detect an RSO with relatively high accuracy, but using that as a sole approach would mean around 20-25% of images would have to be discarded due to failed or misdetections. In this particular failure case there is not much that can be done with a pure deep learning approach even with further training, as it requires more complex PSF deblending algorithms to differentiate. That being said, we do not necessarily need a 100% throughput of images being successfully processed, and so future systems could either have a fallback processor in the case of 0 detections, or an acceptable ratio of processed-to-taken imagery.

One additional consideration was that since the training data was automatically labelled using our legacy processor, the training data only consisted of images which could be successfully processed by the conventional processor, meaning that the performance of the neural network would inherently always be worse.

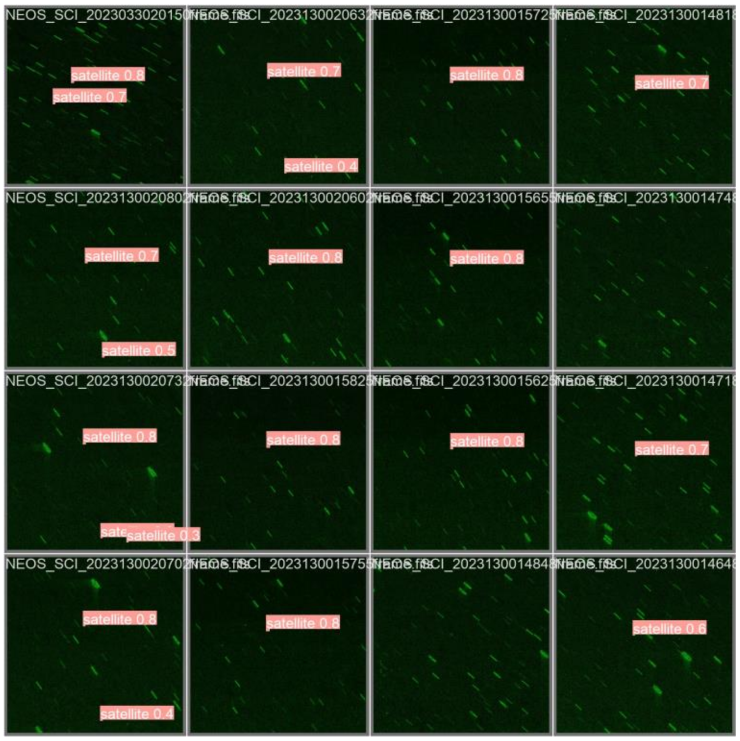
Figure 6. Results of inference tests on 16 validation images

3.2 Implementation

While the previous part of this experiment was validating that deep learning could provide an acceptable or comparable alternative to a conventional processor, an important aspect is finding where a neural network would fit into the pipeline. For organizations with small development teams, or who have limited time or expertise in developing SSA computer vision algorithms, a deep learning approach is more ideal, as what is lost in potential accuracy and rate of successfully detected satellites is made up for in development time and processor simplicity. The choice of approach of a complete solution depends on the goals of the sensor itself, whether there is legacy systems or you are starting from scratch, what degree of accuracy you require, and the throughput and required latency of the system. Deep learning undoubtedly performs faster and superior to a very basic segmentation approach, and is more future proof as when new images come in, new models can be rapidly trained and deployed for specific experiments or targets that may have unique characteristics.

However, if you have a high quality conventional processor, very noisy imagery, or a requirement for a high and precise satellite detection rate, deep learning does not compare to a non-ML solution which breaks down the image into subcomponents. Especially with noisy images, the questions arise of at what stage would you integrate a deep learning processor. If you wanted higher accuracy detections, it makes sense to input the cleaned RSO-only image into the network, which would find an RSO with a much higher degree of confidence. If you are doing astrometric experiments, you require the x and y weighted centroid locations of the streaks and satellites in the frame down to a specific pixel, whereas a neural network would return a bounding box. To bridge the gap, a processor would have to do segmentation even with deep learning inside the bounding box to get the object coordinates. If you are already cleaning the image, and must do segmentation in both methods, there is little benefit to implementing a neural network to detect objects which could be easily recognized in the heavily processed sub-image.

The average size of the RSO in the training set images was a diameter of 5-10 pixels, which means the margin for error on a detection is small. Ultimately we found that the network was unable to detect RSOs when they overlapped with streaks, or when the image had any significant noise artifacts or stray light corruption, which was extremely common with our sensor quality and image type. The network performed well and is ideal on ground-based images where the satellite PSF is blurred, circular, and atmospheric effects prevent a high degree of noise spikes.

Where neural network approach would shine and a potential avenue for future deep learning experimentation would be in streak detection. Conventionally most deep learning research focuses on RSO detection because that is the end goal of a processor, however streak detection is much more relevant towards pattern recognition. Streaks need to have their orientation estimated, signal amplified, and non streak objects filtered out from noisy segmentation results. Plate solving solutions such as pinpoint and astrometry only require 5-8 background stars to do a plate solve, which is easily achievable with a high degree of confidence for a neural network applied to image with 50-100 streaks. By having a neural network do streak detection it would save processing time and development time attempting to improve the detection of streaks, the algorithms of which are the most complex and time consuming to develop and tune within a conventional processor. The only roadblock to a streak detection neural network is that streaks are rarely labeled by processors, and obtaining a labelled dataset of streak objects would initially be extremely time consuming.

4. CONCLUSION

Ultimately, deep learning and object detection are valuable techniques in the field of space surveillance and SSA. Further improvements in ability and accuracy are only likely to grow as the field of artificial intelligence advances and GPU processing power improves. However, our research revealed moderate challenges in applying these methods to RSO detection and raised questions into the time-value trade-offs of what benefits could be derived over our current processing methodology. The neural network ultimately could not compete with the accuracy of conventional methods for RSO detection in NEOSSat imagery. Limitations primary stem from two factors; high levels of noise and cosmic ray interference in space-based imagery and lower image quality of NEOSSat images compared to ground-based observations. For pipelines which do not require the bleeding edge of accuracy or those which have a high image turnover, a deep learning approach does prove viable or even preferable depending on the sensor configuration.

Future research should explore three primary avenues:

1. Developing more sophisticated models and tuning methods with improved GPU compute.
2. Integrating deep learning as part of a hybrid solution, rather than a pure DL approach.
3. Exploring streak detection rather than RSO detection, as streaks are better suited to object detection paradigms.
4. Using deep learning to render 3d satellite models from close proximity satellite to satellite images.

A significant outcome of this experiment is the development of a NEOSSat machine learning dataset. This resource holds future research value for training neural networks on space-based assets. All images from this dataset can be accessed within the Unified Data Library [7] upon request.

5. ACKNOWLEDGEMENTS

The author wishes to acknowledge the Canadian Space Agency in supporting NEOSSat mission operations and the Royal Canadian Air Force - Director General Air and Space Force Development for their supporting the NEOSSat Space Domain Awareness R&D mission.

6. REFERENCES

1. Ryall, S., Thorsteinson, S., Malo, T., “Canadian Satellite Tracking List: Maintaining Canadian Object Custody with NEOSSat – Preliminary Findings”, *ASTRO 2021*, Canadian Aeronautics and Space Institute, (2021).
2. Ryall, S., Thorsteinson, S., “Canadian Satellite Tracking List: Maintaining Canadian Object Custody with NEOSSat”, *AMOS 2023*, Advanced Maui Optical and Space Surveillance Technologies, (2023).
3. Scott, R.L., Thorsteinson, S.E., “Key Finding from the NEOSSat Space-Based SSA Mission”, *AMOS Conference 2018*, Maui Economic Development Board, Maui, HI, 2018.

4. Johnson, C., Scott, R.L., Thorsteinson, S., “Space-Based Photometric Observations of the SpaceX Starlink Constellation Satellites – Preliminary Findings”, *ASTRO 2020*, Canadian Aeronautics and Space Institute, (2020).
5. Thorsteinson, S., “Space Surveillance from A Microsatellite – Metric Observation Processing from NEOSSat”, Masters Thesis, Royal Military College of Canada, accessible from http://hdl.handle.net/11264/1364, June 2017.
6. Vallado, D.A., Hujsak, R., Johnson, T., Seago, J., Woodburn, J., “Orbit Determination Using ODTK Version 6”, European Space Astronomy Centre, Madrid, Spain, 2010.
7. Space-Track.org, https://www.space-track.org/auth/login, accessed July 2023.
8. Unified Data Library, https://unifieddatalibrary.com, accessed July 2023.