Where is the Signal: Improving the Classification of SatNOGS Signals with Machine Learning

Tata Li

ʻIolani School

Hawaiʻi's Aspiring Aerospace Engineers Academy

May 30, 2025

Abstract

This project presents a machine learning approach to classifying spectrogram images, also known as waterfalls, in the Satellite Networked Open Ground Station (SatNOGS) Network. Using a Convolutional Neural Network (CNN), the classifier is trained on a diverse dataset sourced via the SatNOGS API, encompassing waterfalls from multiple satellites. The model accurately distinguishes the presence of signals, achieving 95.52% on a training set of 1985 images. Earlier versions employed a Support Vector Machine (SVM), progressing from a satellite-specific prototype to a generalized multi-satellite classifier. The full project includes modular scripts for training, prediction, and evaluation, along with tools for data collection and logging. A "lite" module for prediction can be accessed from GitHub (https://github.com/tlofbj/waterfall-classifier-project-v3.5cnn-lite). This project highlights how machine learning can be applied to automate and scale signal detection in the vast, often obscure, streams of spaceborne data.

1. Introduction

Background

The SatNOGS (Satellite Networked Open Ground Station) project is a "complete platform of a[n] open satellite ground-station network…to create a full stack of open technologies based on open standards" [1]. Among the key technologies of the stack is SatNOGS Network, which is a global management interface that enables any observer "to utilize all available Ground Stations and communicate with Satellites" for which the results are made public and free for distribution [2]. This makes SatNOGS Network an accessible platform for many researchers to receive and study signals from satellites, particularly in Low Earth Orbit.

However, one of the limitations of using the SatNOGS Network is its misclassification of observation signals, which according to SatNOGS's Wiki is rated by an algorithm to be categorized into either "unknown", "good", "bad", or "failed" [3]. The general idea of such categorization is to determine whether or not the presence of a satellite is reflected by the signal captured, so if a signal presents some sort of pattern or indications that it was sent by a satellite, it's a "good" observation.

The current classification is done by analyzing the signal in terms of the I/Q data, waterfall, audio, demodulated/decoded frame, and/or the decoded image [4]. Among these, the waterfall is a great visual representation of the data from which a human researcher can easily tell the presence of a signal based on visual patterns of shape and color. A sample is shown in Figure 1. Similar to observations, a waterfall can be categorized as "with signal", "without signal", or "unknown". Nonetheless, the accuracy of the current algorithm is quite poor, especially in categorizing weak signals where correct classification is more difficult yet the most important.

Purpose of Study

This project aims to improve the accuracy of signal classification with machine learning (ML), which has recently shown great capability in accurately distinguishing faint patterns in images that can be applied to signal identification in signals' waterfalls. The success of this project will not only enable researchers to use SatNOGS Network with elevated convenience and assurance over the status of observations but also demonstrate the prospect of using machine learning to automate and scale processes typically done by traditional algorithms in space technologies.

Research Objectives

1. To develop a signal binary classifier using Machine Learning (ML) on waterfalls
2. To build a modular toolkit for smooth training, evaluating, and predicting
3. To achieve generalization across different types of signals from various satellites with accuracy

2. Methodology

Approach

In this study, ML models were built, trained, and evaluated. Although we initially used the Support Vector Machine (SVM) provided by scikit-learn, we transitioned to using a Convolutional Neural Network (CNN) provided by PyTorch to achieve higher accuracy. The collection and processing of real waterfall data were necessary in this process. All processes were coded using Python 3.10.13. Code for the prediction module can be accessed from GitHub (https://github.com/tlofbj/waterfall-classifier-project-v3.5cnn-lite). Details for preparing the CNN are presented below.

Dependencies

- torch (PyTorch): Core machine learning library that contains the CNN
- torchvision: Image resizing, tensorizing, and normalizing
- PIL (Pillow): Image manipulation (cropping and conversion to grayscale)
- sklearn (scikit-learn): Evaluation metrics (classification report and confusion matrix)
- seaborn: Data visualization (confusion matrix heatmap)
- matplotlib: Basic plotting library
- numpy: Numerical and vector operations
- requests: HTTP requests for web scraping
- termcolor: Colored terminal output

Data Collection

We coded a web scraper with Python using the requests library that can send an HTTP request to SatNOGS's GET-request API at https://network.satnogs.org/api/observations/. The query string was manipulated such that waterfalls from observations of an intended status ("good" or "bad") were downloaded to two folders, "signal" and "no_signal", based on the

status. The Python script was able to automate this process using a for loop and download hundreds of waterfalls at a time.

Once images were downloaded, misclassified and ambiguous images were manually removed from download folders to ensure a correct and accurate representation of the two classes; the rest were separated into training and testing sets. At the end, 1985 images were used for this project. The final distribution of waterfall usage for the model is presented in Table 1:

Table 1: Usage of downloaded waterfalls

	Training	**Testing**	**Total**
Signal	1103	209	1312
No Signal	547	126	673
Total	1650	335	1985

Data Processing

The processing of waterfall images was entirely handled by Python scripts. We used Pillow (PIL) to open the image as grayscale, crop out the white margin and color coding legend, and save the image as a PIL-Image format, which better prepared the data for training. The PIL images were then resized to 256x256, converted to a NumPy tensor, and normalized using torchvision (mean=0.5, std=0.5). An approximate visualization of this process is shown in Figure 1:

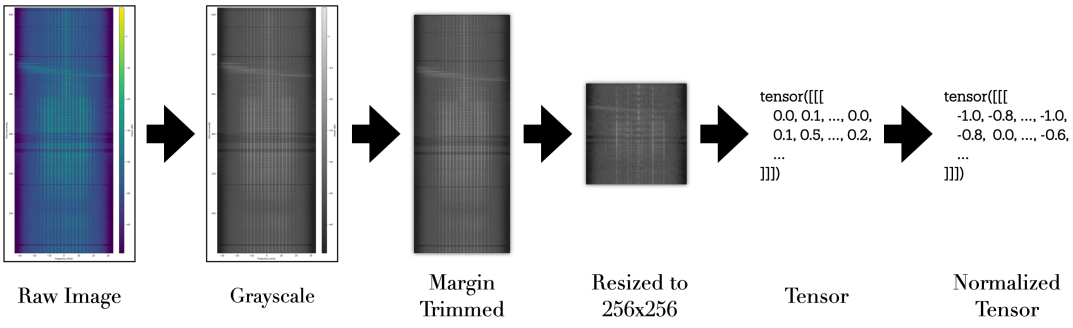

Figure 1: Visualization of image processing

Finally, the images were contained in a custom dataset class based on torch.utils.data. Dataset and loaded into an instance of torch.utils.data.Dataloader for training.

Building the Model

Multiple CNN models were trained using the acquired waterfalls. For each CNN model, however, 4 convolutional layers and 2 fully connected layers were used with varied parameters. Adam was used as the optimizer. The sequential layers and hyperparameters were adjusted as needed to achieve higher accuracy. Details for the hyperparameters are presented in Table 2.

Table 2: Hyperparameters for training

Number of Epochs	30–40*
Learning Rate	0.001–0.002
Batch Size	32

* With a few exceptions

Data Interpretation

Each model was evaluated after training was completed. While earlier models used a smaller testing set, more images were added in the latest models to reach 335 testing images (see Figure 1). Using accuracy metrics from sklearn and a confusion matrix visualized by seaborn, we were able to determine the percent accuracy of the model as well as the number of false negatives and false positives.

It is worth noting that the results produced by our evaluation will likely underestimate their accuracy in real life. The testing set we had compiled included a much higher proportion of weak signals and ambiguous cases than in reality.

3. Results

Ordered from the oldest to the newest reading from top to bottom, the results of eight different models trained over time are recorded in Table 3:

Table 3: Comparing different models and their results

Model Number	Training Set Size	Epochs	Batch Size	Learning Rate	Training Time (s)	Accuracy
1	1281	2	32	0.001	Not recorded	49.5%*
2	1281	30	32	0.001	Not recorded	89.5%*
3	1433	40	32	0.0015	1615.98	89.0%*
4	1436	30	32	0.002	1190.42	88.5%*
5	1316	30	32	0.0015	1141.68	96.41%*
6	1650	1	32	0.0015	45.20	62.39%
7	1650	40	32	0.0015	1862.06	94.33%
8	1650	40	32	0.0015	2461.34	95.52%

* Models 1–5 were evluated against oudated testing sets that were less indicative of actual performance; therefore, they are not considered for best performance (see Limitations).

The best model, which was selected between Models 6–8, is Model 8, achieving an accuracy of 95.52%. The confusion matrix of Model 8 is presented in Figure 2:

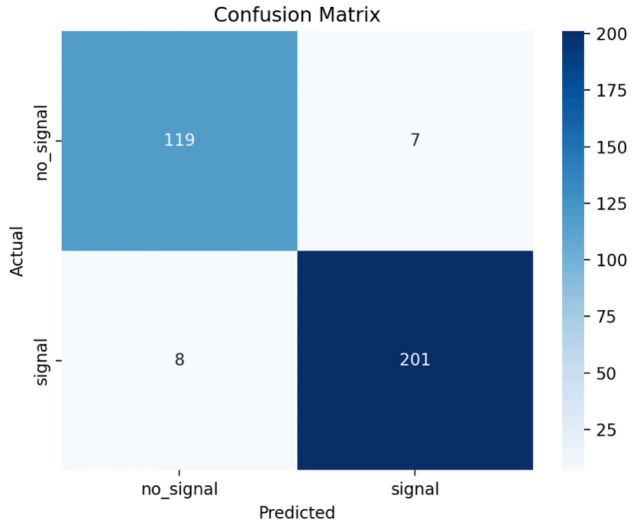

Figure 2: Confusion matrix of Model 8

This model is distinguished by the largest training set, a substantial number of epochs, and the longest training time. The loss over time plot is presented in Figure 3:

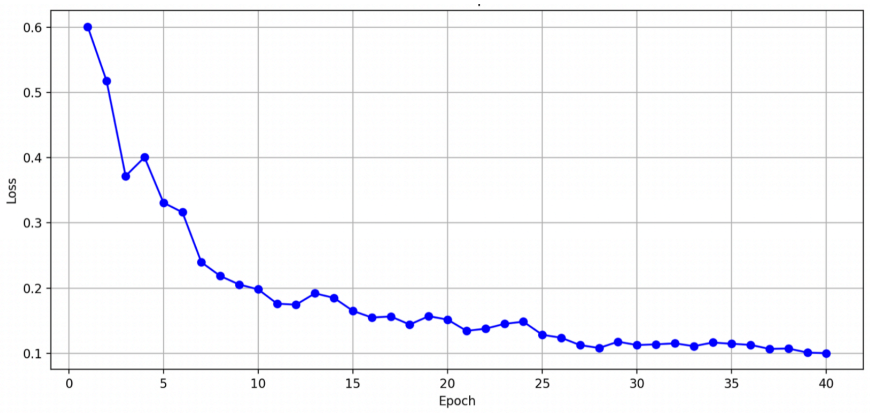

Figure 3: Loss over time for Model 8

While why Model 8 is more accurate than Model 7, which has the same hyperparameters, is unclear, increasing the kernel size and stride in the first pooling layer from 2 to 4 seemed to have improved the accuracy. The sequential for Models 8 is presented in Figure 4:

```
self.sequential = nn.Sequential(
    # Convolutional Layer 1: (1*128*128 -> 32*64*64)
    nn.Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size=2, stride=2),

    # Convolutional Layer 2: (32*64*64 -> 64*32*32)
    nn.Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size=2, stride=2),

    # Convolutional Layer 3: (64*32*32 -> 64*16*16)
    nn.Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size=2, stride=2),

    # Convolutional Layer 4: (64*16*16 -> 64*8*8)
    nn.Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size=2, stride=2),

    # Flatten feature maps to prepare for fully connected layers
    nn.Flatten(start_dim=1, end_dim=-1),

    # Fully Connected Layers for Prediction:
    nn.Linear(64*8*8, 256),
    nn.ReLU(),
    nn.Dropout(0.5), # (prevents overfitting)
    nn.Linear(256, 1),
    nn.Sigmoid()
)
```

Figure 4: Sequential of Model 8

Lastly, we also collected ample data from our failed initial attempt to use Support Vector Machines (SVMs) in classifying waterfall. Results are shown in Table 4:

Table 4: Hyperparameters for training SVMs

Model Number	Training Set Size	Kernel	C	Accuracy
1	539	linear	1.0	20.00%
2	539	sigmoid	1.0	27.50%
3	539	rbf	1.0	29.66%
4	675	poly	1.0	29.29%
5	565	poly	1.0	36.44%
6	675	poly	1.0	40.71%
7	539	poly	1.0	41.25%
8	643	poly	100.0	42.50%
9	643	poly	10.0	42.50%
10	643	poly	10.0	47.50%
11	675	poly	1.0	50.71%
12	643	sigmoid	1.0	50.00%
13	643	sigmoid	1000.0	53.75%
14	643	rbf	10.0	53.75%
15	643	poly	1.0	53.75%
16	643	rbf	0.1	56.25%

Evidently, SVMs were unable to effectively distinguish subtle patterns in these waterfall images, achieving a maximum accuracy only slightly above random chance.

4. Discussion

Key Findings

The high accuracy of CNN models indicates success in ML models' capabilities in classifying SatNOGS waterfalls. We have found that increasing the size of the training dataset and increasing the number of epochs have had the strongest positive impact on the performance of the model. Adjusting parameters in the sequential was also helpful though many of its effects are not well-understood. We have also found that SVMs are not suited for image classification involving complex patterns.

Additionally, in the process of expanding datasets, we have also realized the importance of selecting quality data and removing misclassified data. Although not completely reflected in Table 3, the sudden boost of accuracy from Model 4 to Model 5 was accompanied by a decrease in training set size, which was due to the removal of certain misclassfied or ambiguous data from training set, along with using a more moderate learning rate, which might have also contributed to this boost.

Limitations

The results are limited in many ways due to the haste of the development process. A key limitation was the incompatibilities between different versions of the code. As shown in Table 3, Models 1–5 (marked by an asterisk) ran on earlier versions that carried different sequentials while Models 6–8 ran on the new one. This made it difficult to re-evaluate the accuracy of older iterations, which is reflected by the fact that Model 6 appears to have a higher accuracy than Model 8 despite possibly being less capable due to the fewer varieties of signal patterns it was trained on.

In addition, the current modular toolkit is still unable to deal with different sequentials and CNN architectures, which strongly hinders the modular toolkit's ability to utilize, evaluate, and manipulate all CNN models previously developed.

Lastly, the model itself can definitely improve. In this research, only a few variations of the sequentials and hyperparameters were attempted for efficiency, but many other important parameters stayed constant. Batch size, for example, stayed at 32, and all CNN sequentials used 4 convolutional layers with 2 fully connected layers. These parameters could be adjusted to potentially achieve greater accuracy.

Further Research

Despite limitations, the best model produced by this project is likely better-performing than the current traditional algorithm that is used for SatNOGS Network's platform. Our next step is to integrate the model into the SatNOGS platform. On top of that, future work could be done to further the improvement of the program.

5. Conclusion

Overall, our best model, which achieved an outstanding accuracy of 95.52%, marks a success in regards to objectives 1 and 3. We were able to classify a range of different waterfalls from various satellites with ML. A light version of a deployable Python program for predicting, which can be found on GitHub (https://github.com/tlofbj/waterfall-classifier-project-v3.5cnn-lite), partially achieves objective 2 although deployable code for training and evaluating is still underway. Nonetheless, the project has demonstrated how machine learning can change the way binary classification of SatNOGS signals is done, making this task more accurate and automated.

Acknowledgement

First of all, a huge thank you needs to go to Hawai‘i’s Aspiring Aerospace Engineers Academy (HAAEA) under the directorship of Dr. Dilmurat Azimov and Ms. Melissa Onishi from the University of Hawai‘i at Mānoa. Their diligent effort in maintaining a network of young researchers and experienced college mentors in the program has been instrumental in making explorations like this project possible, where a high schooler like I have the chance to collaborate with a graduate researcher on a niche but important part of the aerospace area.

Specific to this project, however, the person I am most thank of his of course my mentor Mitch McLean from the University of Hawai‘i at Mānoa. He inspired me on the selection of project topic, introduced me to different machine learning models and techniques (especially in regards to SVMs), supported me with SatNOGS resources, and guided me through the gradual development of these project versions, from a single working SVM capable of binary classification of a single type of signal from a single satellite to a fully generalized CNN capable of differentiating any waterfall based on the presence of signal. We met and discussed weekly to ensure fruitful progress and his support was essential to the success of this project.

Additional thanks is given to Ms. Anna Heshiki from ‘Iolani School for connecting with HAAEA as a representative of my school, as well as, Ms. Kimberly Tsiang from ‘Iolani School who discussed with me about using CNNs.

References

[1] Libre Space Foundation, “About,” *SatNOGS*. [Online]. Available: https://satnogs.org/about/. [Accessed: May 26, 2025].

[2] Libre Space Foundation, “About the SatNOGS Network,” *SatNOGS Network*. [Online]. Available: https://network.satnogs.org/about/. [Accessed: May 26, 2025].

[3] Libre Space Foundation, “Operation,” *SatNOGS Wiki*. [Online]. Available: https://wiki.satnogs.org/Operation. [Accessed: May 26, 2025].

[4] Libre Space Foundation, “Artifacts,” *SatNOGS Wiki*. [Online]. Available: https://wiki.satnogs.org/Artifacts. [Accessed: May 26, 2025].