Journal of Artificial Intelligence and Technology, 2024, 4, 1-8
https://doi.org/10.37965/jait.2023.0413

RESEARCH ARTICLE

Detection of Streaks in Astronomical Images Using Machine Learning

Charles Jeffries and Ruben Acuña
School of Computing and Augmented Intelligence, Arizona State University, Tempe, AZ 85281, USA

(Received 30 June 2023; Revised 21 August 2023; Accepted 21 August 2023; Published online 29 August 2023)

Abstract: Satellites in low earth orbit (LEO) pose a challenge to astronomy observations requiring long exposure times or wide observation areas. As the number of satellites in LEO dramatically increases, it motivates an increased need for methods to filter out artifacts caused by satellites crossing into observation fields. This paper develops and evaluates a deep learning model based on U-Net to filter these artifacts from collected data. The proposed model is compared with two existing filtering methods on a dataset generated using the state-of-the-art tool Pyradon. Although the initial application of deep learning does include some unpredictable behavior not found in traditional algorithms, the proposed model outperforms the existing methods in overall accuracy while requiring significantly less computational time. This suggests that the application of deep learning to satellite artifact removal which has previously been underdeveloped in the literature may be an appropriate avenue.

Keywords: astronomy; CNN; image processing; streak detection; U-Net

I. Introduction

The increasing number of human-made satellites in low earth orbit (LEO) [1] creates difficulties for astronomical observations surveying large sections of the sky, as well as observations with long exposure times [2]. For such observations, man-made satellites can create artifacts such as streaks when satellites move across the observation's field of view, as shown in Fig. 1. These streaks are caused by sunlight reflecting off the satellite and back towards the ground, resulting in a brightness, or apparent magnitude (m), that is much higher than the objects under observation. This saturates measurements in locations that the streak covers, rendering that data unrecoverable and introducing an artifact that can affect downstream analysis.

As established at the SATCON1 workshop [4] by NSF's NOIRLab and the American Astronautical Society, the effects of these artifacts on downstream scientific analysis can be significant. For example, the relatively bright streaks caused by satellites can confuse automated processing pipelines that are designed to identify stellar objects, resulting in erroneous downstream data. They can also result in decreased efficiency, as multiple observations may now be necessary where only one was previously, and decreased overall effectiveness as important data may be covered up by the streak of a transient satellite [4].

A number of detection algorithms have been developed to filter this compromised data. Traditional algorithms have had some success, which can be improved when multiple exposures of the same region of sky are combined [5] or when known locations of stars are considered. Another approach is to use precise tracking data to calculate where and when an object in LEO should cross the field of view of the observation and then proactively ignore data from those pixels at those times [6]. These approaches require

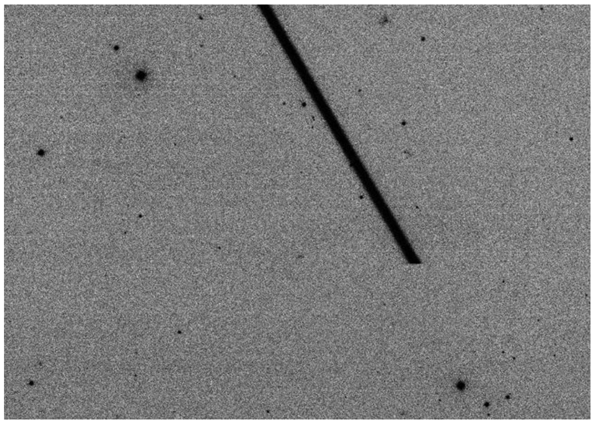

Fig. 1. An observation containing a satellite streak from the Trailblazer [3] repository. This image is characteristic of real-world observations, with embedded stars (black) and background noise (gray).

additional information and data to be collected, which may not always be available or accurate.

Machine learning provides an alternative to traditional filtering methods that can function on single observations with no other assistance while also supporting efficient scalability. This is possible because of advancements in the efficiency and capability of deep learning-based models through the extraction of features using pre-trained general models and then utilizing specially trained models to segment the desired features (e.g., [7]). One such model that follows this approach is U-Net [8] which is commonly used as the basis for many deep learning models, especially in medical imaging.

This paper proposes a deep learning model based on the existing U-Net platform that detects and masks out pixels in an observation determined to have been affected by the streak of a

Corresponding author: Ruben Acuña (e-mail: racuna1@asu.edu).

satellite or other LEO object. We show that this model, as an example of machine learning, is both significantly more accurate than traditional non-ML methods and is faster to execute. To compare our method more accurately with existing techniques, we also propose a novel metric, the Star Occlusion Factor (SOF), to evaluate data loss caused by masking pixels containing scientifically useful data. The research objectives of our work are:

RO1: Evaluate the effectiveness of a common medical-grade machine learning model and its suitability for use in filtering satellite streaks.

RO1.1: Perform a quantitative evaluation of the computing efficiency of the proposed machine learning model in comparison with existing methods.

RO1.2: Perform both quantitative and qualitative evaluation of the errors made by the proposed model in comparison with errors made by existing methods.

The rest of the paper is structured as follows: Section II presents the related work. Section III describes the data management processes used, the traditional methods, as well as the design of the ML model. Section IV presents the results achieved. Section V discusses the results and compares qualitative aspects of each method, and lastly, Section VI summarizes our contributions and conclusions.

II. RELATED WORK

Data management is an essential part of the astronomical process. As data quantities increase, especially with the advent of whole sky surveys which can generate terabytes of data in days [9], it is infeasible for incoming data to be processed manually. Therefore, the need for automated data processing and cleaning pipelines is greater than ever. Efforts to manage this flow of data have covered many different areas, from enabling efficient distribution and replication of the data [10] to automated labeling and categorization of the objects being imaged [11]. There is also significant interest in denoising and general cleaning of the data before it is analyzed (e.g., [12]).

Previous work in astronomical image denoising has focused on traditional algorithmic approaches for the detection and masking of streaks. Pyradon [13] is a tool suite for the detection and masking of satellite streaks utilizing a Radon transform mechanism. It exploits the fact that satellite streaks are typically straight and detects linear artifacts in supplied images. Pyradon also contains a set of tools for generating representative simulated datasets of astronomic samples containing satellite streaks. Some features that improve the representativeness of the generated samples include the use of a Point Spread Function that distorts light sources in the same way that the optics of a real telescope would, and the addition of Gaussian noise to simulate the background noise. ASTRiDE [14] is a tool for the detection of mostly linear artifacts in astronomic images. It utilizes boundary tracing to find the outline of all objects in the image and then filters down to only boundaries that are mostly linear in nature. Its primary use case is for the detection of unknown moving objects in LEO [15]. Some alternative approaches have also been proposed, such as performing analysis on spectrogram data rather than directly on optical data [16], with some success in higher noise environments.

Some initial ML approaches for streak detection have been developed; however, they have significant limitations. DeepStreaks [17] is a toolset for the detection of linear streak artifacts caused by objects in LEO. It uses several CNN models to detect and classify streaks in supplied images. However, since it does not generate data for the geometry of the streak, it cannot generate pixel masks of the detected streak.

Recent advances have been made in ML models capable of classifying individual pixels of an image as belonging to an object or not (image segmentation). One such model architecture is U-Net [8], which utilizes a multi-step approach of multiple convolutional neural networks (CNNs) chained together, combined with a series of upscaling networks to achieve high levels of effectiveness and efficiency for general-purpose image segmentation tasks. For example, it has been applied to road extraction [18], identification of buildings from satellite imagery [19], and identification of microorganisms [20]. U-Net was originally developed for medical image segmentation [21], but has already seen some use for the mitigation of artifacts in radio astrometric data [22].

CNNs reduce the pixel array of an image into a smaller array that describes the features contained in that image. The benefit of this approach is that less data are required to describe the image, and thus, it is more efficient to process. A CNN does this through a kernel describing how neighboring pixels affect a region as it is mapped onto the smaller convolved array. It then has a pooling layer that functions as a denoiser and either takes the maximum, or average, value found in a region of the convolved array and projects it into a smaller pooling array [23].

CNNs have also been applied in detection applications for particle streaks [24], both for boundary box generation as well as masking of streaks. This demonstrates the ability of CNNs to identify and mask a single streak even when noise and other artifacts are present in the proximity of the desired streak.

Another reason for the potential suitability of CNNs for this application is their exceptional performance in denoising applications [25]. Indeed, CNNs have been applied to the problem of denoising astronomical data so that further analysis can be performed more easily [26]. Several architectures have been used for this purpose, including U-Net [27]. CNNs in these kinds of applications consistently demonstrate improved efficiency when compared to traditional methods.

III. METHODS

A. DATASET

To evaluate our method, we constructed a data-driven workflow in Python to reproducibly generate a dataset utilizing Pyradon's simulation tool and perform our later evaluation. Training datasets that utilize generated data is a rising practice (e.g., [16,24,26]) and allows development of models even when real-world labeled training data is sparse or unavailable. The principle of domain randomization [28] allows our synthetic data to generalize to real-world data. Fig. 2 is an overview of our process for generating the dataset.

The dataset consists of 256×256 pixel images managed with NumPy [29], containing between 0 and 25 stars of varying brightness in random positions, distributed uniformly. Brightness varies evenly between fully dark and fully bright. Gaussian noise is added to the image in order to simulate sensor noise characteristically found in real-world observations (see [30]). A global predefined seed is used in order to ensure run-to-run consistency of generated noise while preserving randomization of stars between samples. Lastly, a simulated streak is added to each image with varying length, origin, and intensity to form the final image (e.g., Fig. 3). This image is representative of a standard single

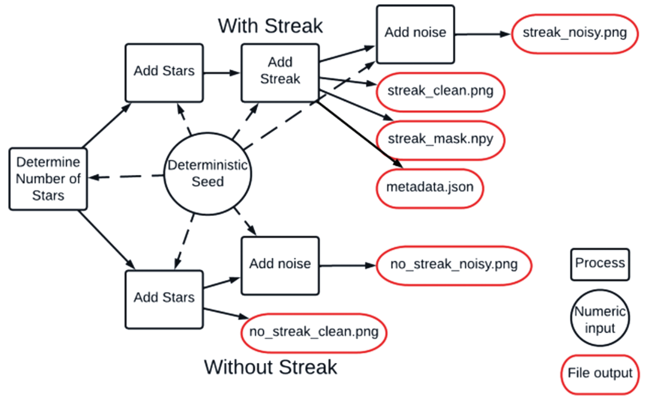

Fig. 2. Workflow for generating our dataset. During execution, several output image variants are saved for later analysis and processing.

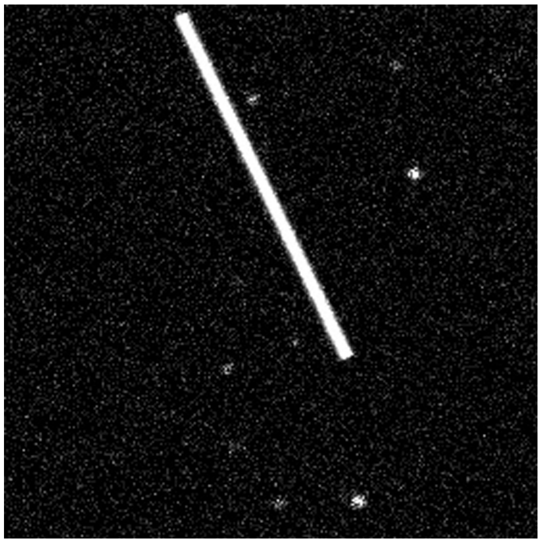

Fig. 3. A generated image. As in real-world images, it contains a streak, stars, and background noise.

capture which contains stars of varying intensity as well as a transient high apparent magnitude object such as a satellite. A binary pixel mask of the streak itself is saved alongside the image (e.g., Fig. 4) and later serves as ground truth for evaluation. All pixels which contain a streak in Fig. 3 are marked as true (white), while pixels that do not contain part of a streak are marked false (black). Variants of the images without a streak are also saved for use during performance analysis as well as metadata about the location and brightness of the streak. This approach is used to generate ten thousand samples in order to adequately characterize the degrees of freedom for the streak: origin, rotation, length, and brightness. Following convention [30], 80% of the dataset is used for training and the remaining 20% is held back for evaluation.

B. TRADITIONAL ALGORITHMS

We evaluated the performance of two approaches that use traditional algorithms, Pyradon and ASTRiDE, to provide context for our ML approach. For our evaluation, we apply each of the three approaches to the testing portion of the dataset (which contains

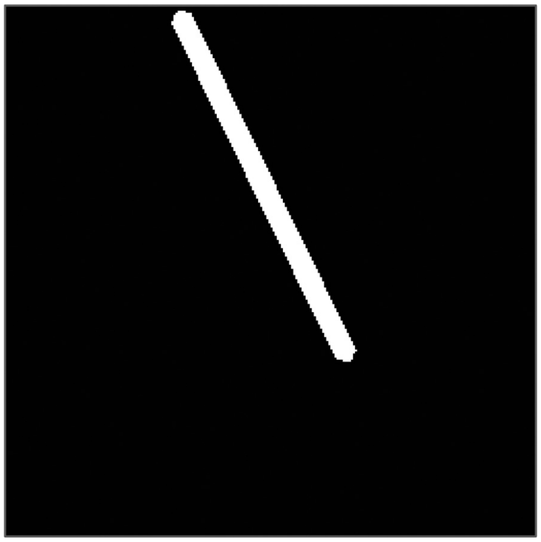

Fig. 4. Baseline streak mask. White pixels contain part of a streak.

sample images with streaks) and then process their respective results to compute masks such as shown in Fig 4. Once the output tools are unified into pixel masks, they can be fairly compared using the same metrics.

We first used one of Pyradon's utilities to perform streak detection. It takes in a grayscale image, utilizes a radon transform to locate streaks in the image, and then generates metadata for each found streak including position, rotation, and length. The equation for the Radon transform used by Pyradon is given in Equation (1) [13], where R is the resulting transform, x_0 is the starting coordinate of the line, Δx is the horizontal run of the line, and N_y is the total height of the image. Using Δx and N_y, we are able to conceptualize the slope of the line, and keeping the second axis as Δx allows the resulting transform to remain discrete and pixel-based rather than continuous as an angle. The specific algorithm utilized by Pyradon is a Fast Radon Transform, which uses dynamic programming to reduce redundant calculations in order to improve efficiency by several orders of magnitude. This is possible because of the discrete pixel nature of digital images. It can use this information to generate a mask of all pixels affected by the streak, which we save for analysis and comparison to the ground truth.

$$R(x_0,\Delta x) \equiv \sum_{y=0}^{N_y-1} I\left(x_0 + \Delta x \frac{y}{N_y}, y\right) \tag{1}$$

ASTRiDE's streak detection was performed using ASTRiDE's built-in streak detector. The detector functions by implementing boundary detection to trace the outline of all distinct objects in the image. This is done using scikit-image's [31] find_contours, which in turn uses a variant of the Marching Cubes [32] algorithm called Marching Squares to find the contours. A filter is then applied to all of the detected objects based on their outline's shape. Each shape is assigned a score according to how similar it is to a circle. For example, a circular object would score close to 1, a square object would score 0.78, and a long thread-like object would score close to 0. Outlines with a score higher than a threshold are then removed from consideration. The default threshold value of 0.2 was used. The output of the streak detector is a list of contours, or outlines, of all streaks detected. These contours must be converted into a pixel image in order to apply the same pixel-based evaluation method necessitated by the other approaches. We used scikit-image to construct a polygon of the enclosed area. We then converted the polygon to a binary pixel mask of all affected pixels and saved it for analysis and comparison to the ground truth.

C. MACHINE LEARNING APPROACH

We constructed a neural network based on the U-Net architecture and the needs of our domain. U-Net stacks CNN layers in order to efficiently extract feature information from an image (the encoder stage) and classifies pixels belonging to the desired feature (the decoder stage). U-Net is highly effective at generic image segmentation tasks (e.g., [15]). For the encoder, we used MobileNetV2 which is a robust and efficient CNN encoder for general-purpose image segmentation [33]. For the decoder, we used 4 layers of pi × 2pi× [34] to perform upscaling and image segmentation due to its suitability for general-purpose uses. The encoding stage model is pre-trained, and we trained the decoding stage specifically for our application.

To train our system, we used 8000 samples from the dataset generated using Pyradon. Training was performed over 20 epochs, using the Adam optimizer and the Sparse Categorical Cross-entropy loss function from Keras [35]. To ensure rapid training for iterative development of the ML model, the images as well as associated masks were downscaled to 128×128 pixels from their original size. Downscaling in this manner is a common strategy to improve training speed and effectiveness (e.g., [24,36]). The trained ML model was used to generate streak masks for the 2000 sample validation set. As traditional approaches, the training data were not used by the other approaches. The computed masks were then upscaled from their 128×128 native resolution to the standard 256×256 resolution using Nearest Neighbor interpolation. Like all approaches, this mask was saved for evaluation.

The execution time of each approach was measured using Python's time module and excluded the time to load the dataset into memory. Testing was performed on an Intel Core i7-12700H running at 4.14 GHz with an Nvidia RTX 3050ti GPU. Pyradon and ASTRiDE are CPU-bound approaches, while our machine learning model can leverage the GPU. The consumer nature of this hardware (i.e., not requiring highly specialized equipment) demonstrates the performance advantage and applicability of our work.

D. METRICS

Several standard metrics were used to evaluate the performance of each method tested. Initially, mean squared error (MSE) and mean absolute error (MAE) were used as baseline statistical metrics. As a more domain-appropriate measure, Intersection over Union (IoU) [37] was also applied. IoU is an error metric commonly used to evaluate the performance of image segmentation solutions. It is calculated by dividing the number of pixels in the intersection of the predicted mask and the ground truth mask and dividing by the union of all pixels in the predicted and ground truth masks. In the resulting score, a score of 1 represents perfect overlap, and a score of 0 represents no overlap between the predicted mask and the ground truth mask. The definition for IoU [37] is given in Equation (2), where A is the set of pixels in the ground truth mask and B is similarly extracted from the generated mask being evaluated.

$$IoU = \frac{|A \cap B|}{|A \cup B|} = \frac{TP}{FP + TP + FN} \tag{2}$$

However, none of these metrics are ideal for this problem domain because they treat all non-streak pixels as having the same scientific value regardless of what data they contain. Some pixels will contain data, such as stars, which is more useful than other pixels that contain only noise. There is also a potential for some methods of generating streak masks to erroneously include pixels containing stars because of their relatively high brightness value. This can have a negative impact on downstream analysis, such as star cataloging (e.g., [38]). A metric that can express this tendency can improve the accuracy of solution analysis for this problem.

Therefore, we propose a new metric for this domain, the SOF. SOF is designed specifically to evaluate the exclusion of target data in astronomic processing. SOF is calculated by dividing the number of pixels containing stars that have been masked by the total number of pixels containing stars present. The definition for SOF is given in Equation (3) where S is the set of pixels containing one or more stars and B is the set of pixels in the streak mask being evaluated.

$$SOF = \frac{|S \cap B|}{|S|} \tag{3}$$

This formulation is made possible by the use of generated data which includes ground truth data indicating the position and size of stars. In effect, SOF is a measurement of how much useful data are being excluded by the generated masks. Two variants of SOF were implemented: SOFa in which all stars in the frame are considered for the metric and SOFb which excludes stars containing pixels that have been covered by a streak from consideration. The former variant is an expression of the total data loss, while the second variant expresses only the data which were lost unnecessarily.

IV. EXPERIMENTAL RESULTS

Each approach was evaluated on 2000 samples derived from the Pyradon dataset. These samples were distinct from our ML model's training set. Generated masks for each method were collected (representing the ground truth from Pyradon) and compared to the true mask for each sample. MSE and MAE for the number of incorrect pixels in each mask were calculated for each method, as well as the average IoU. These results are shown in Table I. Average SOFa and SOFb were also calculated and are shown in Table II.

Table I. Error values for tested methods

Method	MSE	MAE	IoU
ML	10262.474	0.003145	0.803
ASTRiDE	516299.0805	0.008157	0.514
Pyradon	2494471.283	0.020974	0.431

Table II. Star Occlusion Factor values for tested methods and ground truth

Method	Average SOFa: including streak stars (%)	Average SOFb: excluding streak stars (%)
ML	9.25	1.50
ASTRiDE	6.34[†]	1.07
Pyradon	12.98	7.17
Ground Truth	9.27	0.00

[†] ASTRiDE has a tendency to omit the generation of a mask entirely in certain situations, making its score vacuously low.

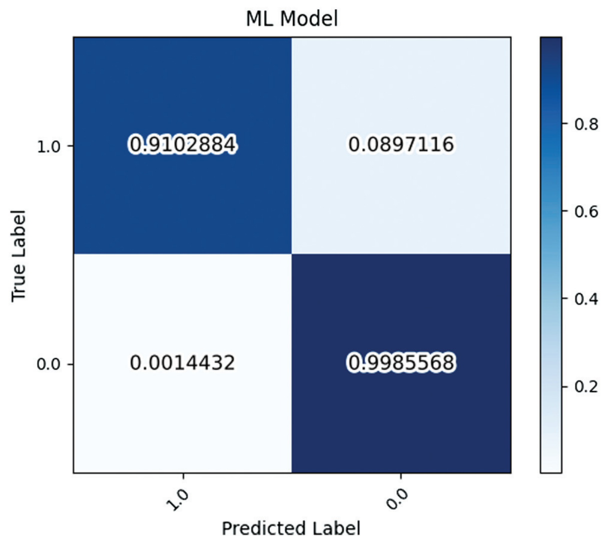

Fig. 5. Confusion Matrix for ML model.

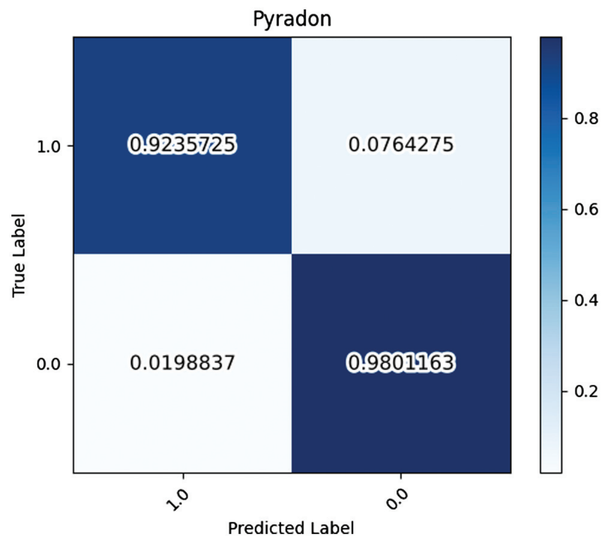

Fig. 7. Confusion Matrix for Pyradon.

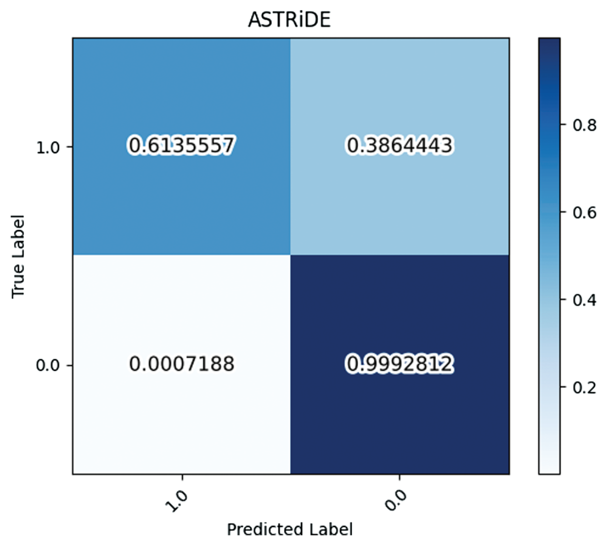

Fig. 6. Confusion Matrix for ASTRiDE.

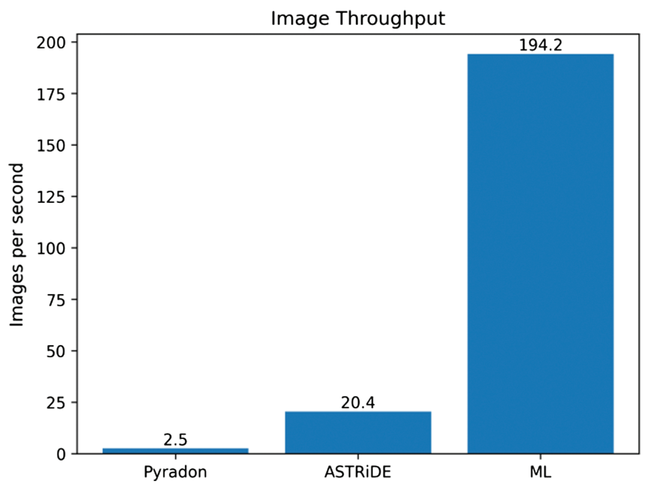

Fig. 8. Image processing throughput for tested methods.

Rates for true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) were calculated for each method in order to create a confusion matrix. Figures 5–7 contain the calculated confusion matrix for the ML model, ASTRiDE, and Pyradon, respectively.

The number of samples (2000) was divided by the measured execution time for each method, yielding the average image processing throughput for each method as shown in Fig. 8. For consistency, all benchmarks were performed on the same system (see earlier).

V. DISCUSSION

For MAE, ASTRiDE and the ML model appear to have similar performance; however, Pyradon is significantly less accurate overall. When MSE is considered, ASTRiDE performs much worse relative to the ML model. This indicates a number of cases where ASTRiDE had significant error which MSE amplifies. Pyradon continues to perform the worst, being over two orders of magnitude worse than the ML model.

From the confusion matrices, it is apparent that ASTRiDE struggles with FN labels, with over 33% of true streak pixels marked as negative by it. By contrast, the ML model and Pyradon have very low rates of false-negative detections. Analyzing false positives, ASTRiDE performs the best, with only a 0.07% false-positive detection rate. By comparison, the ML model achieves a false-positive rate of 0.14% and Pyradon 1.9%. Figure 9 gives a visual representation of the example streaks from Fig. 10 color-coded to denote FN, FP, TN, and TP.

Examination of sample predictions from each method gives insight into certain behaviors that may be undesirable. A prototypical example can be seen in Fig. 10, where ASTRiDE tends to include nearby noise artifacts in the detection mask, needlessly masking out pixels that may otherwise contain good data. There is also evidence that ASTRiDE is unable to create a mask in certain

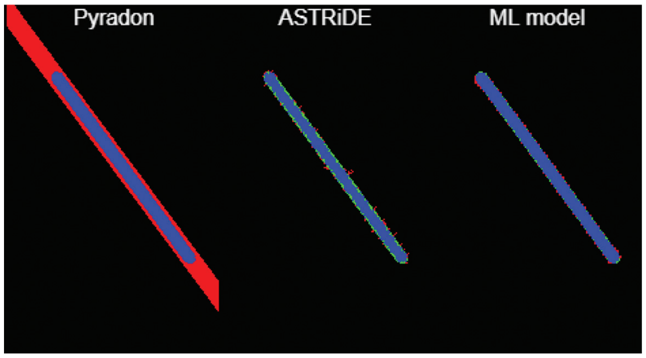

Fig. 9. Analysis of the pixel accuracy of each method tested. Black denotes TN, blue denotes TP, red denotes FP, and green denotes FN.

situations. For example, as seen in Fig. 11, ASTRiDE will not generate a mask when a streak crosses the border of the image. This is because the boundary tracing algorithm is unable to complete a polygon when the edge of the object is not visible in the image. The result is a large error across all metrics for these instances.

Pyradon also has issues with false positives which are easily visualized. Figure 10 shows Pyradon attempting to mask the same streak, but failing to correctly terminate at the ends of the streak.

In contrast, our ML model produces a uniform and well-formed mask of the same streak; however, it does so with some notable "aliasing" artifacts caused by the lower-resolution down-scaled images that the model requires as an input.

Analysis of IoU for each method indicates the same performance rankings as previous metrics; however, ASTRiDE's occasional failure to generate a mask becomes more apparent with its IoU score of 0.514 relative to Pyradon's score of 0.431. These instances of 0 overlap significantly harm ASTRiDE's score in this metric. Our ML model performs very well when measured by IoU, scoring 0.803, and indicating very strong overlap.

Analysis of the SOF for each method indicates how well each method avoids masking valid data, as seen in Table II. When measuring the SOFb which excludes stars covered by a streak, ASTRiDE performs the best at 1.07%, with our ML model close behind at 1.50%. Pyradon demonstrates a significant tendency to mask out valid pixels containing stars with an average SOFb of 7.17%.

When measuring SOFa which includes all stars in the frame, we gain the context of the total amount of valued data excluded by each method. The average SOFa from applying the ground truth mask to the image is 9.27% for our dataset. The ML model is close to this at 9.25%, the small reduction relative to the ground truth mask being

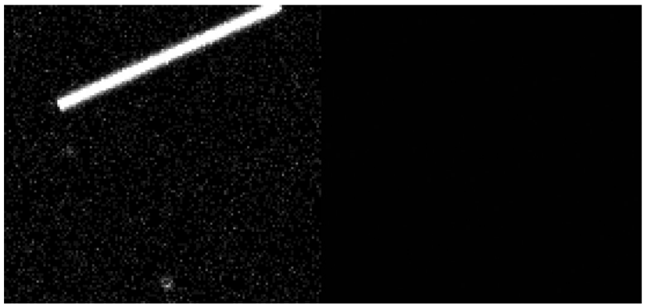

Fig. 11. An image containing a satellite streak that partially intersects with the edge of the frame and the corresponding mask generated by ASTRiDE.

indicative of a slightly undersized mask leaving exposed some intersected star pixels. Pyradon scores significantly worse at 12.98%. However, ASTRiDE has an average SOFa of 6.34%, below the value set by the ground truth. This is possible because of ASTRiDE's tendency to omit generating a mask at all in some situations. The result of this, in combination with ASTRiDE's low SOFa outside of pixels contained in streaks, is that ASTRiDE will not mask pixels containing both a streak and a star that otherwise should be masked, resulting in the anomalously low SOFa in this situation.

When comparing the execution speed of each method, our ML model is significantly faster than either of the traditional methods, being capable of processing nearly 10× as many images as ASTRiDE in the same amount of time. This could be useful in situations where large amounts of data need to be processed quickly in near real time, such as astronomical surveys.

VI. CONCLUSION

In this paper, we have developed and evaluated an ML approach for detecting and masking satellite streaks in astronomical data. The traditional methods ASTRiDE and Pyradon had problematic results with high levels of false negatives and false positives, respectively, while our method outperformed these traditional methods on the whole. We also have applied a novel metric, SOF, for the evaluation of masking methods' tendency to mask valid data. Under SOF, our model also demonstrated a significantly lower tendency to erroneously mask out valid star data than Pyradon and a tendency comparable to ASTRiDE. This demonstrates our model's capability to retain important valid data while masking invalid streak data.

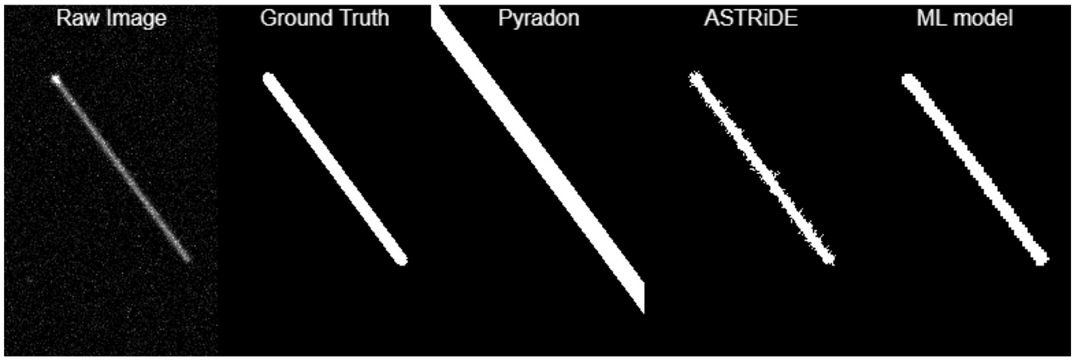

Fig. 10. Example detection masks for Pyradon, ASTRiDE, and our ML model compared to the ground truth and the raw image.

Simultaneously, our method required significantly less computing resources and was able to execute significantly faster than either of the traditional methods tested. Therefore, our novel ML model outperforms comparable traditional methods at this task, achieving a very low false-negative rate while maintaining an acceptably small number of false positives and retaining valuable data. At the same time, it provides these benefits on consumer-grade hardware.

In the future, we plan to evaluate the performance of other variants of U-Net as well as competing architectures. We could also investigate a hybrid system that utilizes aspects of both an ML model as well as some traditional detection techniques in order to achieve improved performance. Finally, we could explore the implementation of a ML model to remove noise and other distracting features from data in order to enhance the performance of traditional masking methods such as Pyradon and ASTRiDE.

CONFLICT OF INTEREST

The authors declare that they have no conflict of interest.

REFERENCES

[1] "Online Index of Objects Launched into Outer Space," United Nations Office for Outer Space Affairs. United Nations Office for Outer Space Affairs. [Online]. Available: https://www.unoosa.org/oosa/osoindex/search-ng.jspx.

[2] A. Venkatesan, J. Lowenthal, P. Prem, and M. Vidaurri, "The impact of satellite constellations on space as an ancestral global commons," *Nat. Astron.*, vol. 4, no. 11, Art. no. 11, Nov. 2020. DOI: 10.1038/s41550-020-01238-3.

[3] M. Rawls et al., "Trailblazer: an open data repository for satellite-streaked images," in *Am. Astronom. Soc. Meeting Abstracts*, American Astronomical Society, 2022, pp. 144.

[4] C. Walker et al., "Impact of satellite constellations on optical astronomy and recommendations toward mitigations," *Bull. AAS*, vol. 52, no. 2, Aug. 2020. DOI: 10.3847/25c2cfeb.346793b8.

[5] A. M. Meisner, D. Caselden, E. F. Schlafly, and F. Kiwy, "unTimely: a full-sky, time-domain unWISE catalog," *Astron. J.*, vol. 165, no. 2, p. 36, Jan. 2023. DOI: 10.3847/1538-3881/aca2ab.

[6] J. A. Hu, M. L. Rawls, P. Yoachim, and Ž. Ivezić, "Satellite constellation avoidance with the rubin observatory legacy survey of space and time," *Astrophys. J. Lett.*, vol. 941, no. 1, p. L15, Dec. 2022. DOI: 10.3847/2041-8213/aca592.

[7] C. Jeffries, *An Evaluation of the Methods of Removing Satellite Artifacts from Astronomic Data*, Mesa, AZ, USA: Arizona State University, 2023.

[8] O. Ronneberger, P. Fischer, and T. Brox, "U-net: convolutional networks for biomedical image segmentation," in *Med. Image Comput. Comput.-Assist. Interv.–MICCAI 2015: 18th Int. Conf., Munich, Germany, October 5–9, 2015, Proc., Part III 18*, Springer, 2015, pp. 234–241.

[9] C. Yang, X. Meng, Z. Du, Z. Duan, and Y. Du, "Data management in time-domain astronomy: requirements and challenges," in *Big Scientific Data Management*, J. Li, X. Meng, Y. Zhang, W. Cui, and Z. Du, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 32–43. DOI: 10.1007/978-3-030-28061-1_5.

[10] Z. Du, J. Hu, Y. Chen, Z. Cheng, and X. Wang, "Optimized QoS-aware replica placement heuristics and applications in astronomy data grid," *J. Syst. Softw.*, vol. 84, no. 7, pp. 1224–1232, Jul. 2011. DOI: 10.1016/j.jss.2011.02.038.

[11] C. S. Kochanek et al., "The all-sky automated survey for supernovae (ASAS-SN) light curve server v1.0," *Publ. Astron. Soc. Pac.*, vol. 129, no. 980, p. 104502, Aug. 2017. DOI: 10.1088/1538-3873/aa80d9.

[12] S. Beckouche, J. L. Starck, and J. Fadili, "Astronomical image denoising using dictionary learning," *Astron. Astrophys.*, vol. 556, p. A132, Aug. 2013. DOI: 10.1051/0004-6361/201220752.

[13] G. Nir, B. Zackay, and E. O. Ofek, "Optimal and efficient streak detection in astronomical images," *Astron. J.*, vol. 156, no. 5, p. 229, Oct. 2018.

[14] D.-W. Kim, "ASTRiDE: automated streak detection for astronomical images," *Astrophys. Source Code Libr.*, pp. ascl–1605, 2016.

[15] C. Zhang, R. Y. Sun, and S. X. Yu, "Combine the Space Situational Awareness and Time Domain Astronomy with Massive Optical Survey," in *1st Inter. Orbital Debris Conf.*, vol. 2109, 2019, p. 6161.

[16] D. Cutajar et al., "Track detection of high-velocity resident space objects in low earth orbit," *Adv. Space Res.*, vol. 71, no. 3, pp. 1670–1681, Feb. 2023. DOI: 10.1016/j.asr.2022.09.053.

[17] D. A. Duev et al., "DeepStreaks: identifying fast-moving objects in the Zwicky transient facility data with deep learning," *Mon. Not. R. Astron. Soc.*, vol. 486, no. 3, pp. 4158–4165, 2019.

[18] Z. Zhang, Q. Liu, and Y. Wang, "Road extraction by deep residual u-net," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 5, pp. 749–753, 2018.

[19] G. Chhor, C. B. Bartolome Aramburu and I. Bougdal-Lambert, "Satellite image segmentation for building detection using U-net", 2017. [online]. Available: https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/.

[20] J. Zhang et al., "LCU-Net: a novel low-cost U-Net for environmental microorganism image segmentation," *Pattern Recognit.*, vol. 115, p. 107885, Jul. 2021. DOI: 10.1016/j.patcog.2021.107885.

[21] G. Du, X. Cao, J. Liang, X. Chen, and Y. Zhan, "Medical image segmentation based on u-net: a review," *J. Imaging Sci. Technol.*, vol. 64, pp. 1–12, 2020.

[22] J. Akeret, C. Chang, A. Lucchi, and A. Refregier, "Radio frequency interference mitigation using deep convolutional neural networks," *Astron. Comput.*, vol. 18, pp. 35–39, 2017.

[23] S. Saha, "A comprehensive guide to convolutional neural networks," *Saturn Cloud Blog*. Saturn Cloud, Dec. 2018. [Online]. Available: https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/.

[24] C. Tsalicoglou and T. Rösgen, "Deep learning based instance segmentation of particle streaks and tufts," *Meas. Sci. Technol.*, vol. 33, no. 11, p. 114005, Aug. 2022. DOI: 10.1088/1361-6501/ac8892.

[25] M. Zheng, K. Zhi, J. Zeng, C. Tian, and L. You, "A hybrid CNN for image denoising," *J. Artif. Intell. Technol.*, vol. 2, no. 3, Art. no. 3, Apr. 2022. DOI: 10.37965/jait.2022.0101.

[26] Y. Zhang, B. Nord, A. Pagul, and M. Lepori, "Noise2Astro: astronomical image denoising with self-supervised neural networks," *Res. Notes AAS*, vol. 6, no. 9, p. 187, Sep. 2022. DOI: 10.3847/2515-5172/ac9140.

[27] A. Vojtekova et al., "Learning to denoise astronomical images with U-nets," *Mon. Not. R. Astron. Soc.*, vol. 503, no. 3, pp. 3204–3215, 2021.

[28] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 23–30. DOI: 10.1109/IROS.2017.8202133.

[29] C. R. Harris et al., "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: 10.1038/s41586-020-2649-2.

[30] S. S. Skiena, *The Data Science Design Manual*, Cham, Switzerland: Springer, 2017.

[31] S. Van der Walt et al., "scikit-image: image processing in python," *PeerJ*, vol. 2, p. e453, 2014.

[32] W. Lorensen and H. Cline, "Marching cubes: a high resolution 3D surface construction algorithm," *ACM SIGGRAPH Comput. Graph.*, vol. 21, p. 163, Aug. 1987. DOI: 10.1145/37401.37422.

[33] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF Conf. on Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, 2018, Salt Lake City, UT, USA pp. 4510–4520. DOI: 10.1109/CVPR.2018.00474.

[34] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogn. (CVPR)*, IEEE, Holoulu, HI, USA, pp. 1125–1134, Jul. 2017.

[35] F. Chollet and Others, "Keras." 2015. [Online]. Available: https://keras.io.

[36] H. Talebi and P. Milanfar, "Learning to resize images for computer vision tasks," in *2021 IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Montreal, QC, Canada: IEEE, Oct. 2021, pp. 487–496. DOI: 10.1109/ICCV48922.2021.00055.

[37] M. A. Rahman and Y. Wang, "Optimizing intersection-over-union in deep neural networks for image segmentation," in *Advances in Visual Computing*, G. Bebis, R. Boyle, B. Parvin, D. Koracin, F. Porikli, S. Skaff, A. Entezari, J. Min, D. Iwai, A. Sadagic, C. Scheidegger, and T. Isenberg, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 234–244. DOI: 10.1007/978-3-319-50835-1_22.

[38] E. Bica, D. B. Pavani, C. J. Bonatto, and E. F. Lima, "A multi-band catalog of 10978 star clusters, associations, and candidates in the milky way," *Astron. J.*, vol. 157, no. 1, p. 12, Dec. 2018. DOI: 10.3847/1538-3881/aaef8d.