Enhancing space surveillance through the Unistellar collaborative observation program

Janin-Potiron, P.[a], Cañameras, R.[a], Neichel, B.[a], Gray, M.[a], Marchis, F.[b,c], Lambert, R.[c], Quere, F.[b], Blaclard, G.[b], Malvache, A.[b], Beltramo-Martin, O.[d], Cantegreil, J.[d], Boutte, P.[d], Zancanaro, Y.[d], Gervail, A.[e], and Correia, C. M.[f]

[a]Aix Marseille Univ, CNRS, CNES, LAM, Marseille, France
[b]Unistellar, 5 Allée Marcel Leclerc 13008 Marseille
[c]SETI Institute, 339 Bernardo Ave, Suite 200 Mountain View, CA 94043, United States
[d]SpaceAble, 13-15 rue Taitbout 75009 Paris. France
[e]Observatoire de la Combe de Lancey, 341 route du Boussant, France
[f]Space ODT Lda, Av da França 492, 4050-277 Porto

ABSTRACT

The DOSSA (Decentralization of Space Situational Awareness) project, led by SpaceAble in collaboration with Unistellar and the Laboratoire d'Astrophysique de Marseille, aims to enhance space surveillance through a collaborative effort involving amateur astronomers, researchers, and industrial networks, leveraging advanced technology and data collection to improve space situational awareness in an era of escalating satellite numbers. This project aims to create a comprehensive sky map of objects transiting around Earth. It benefits from a large dataset collected by the Unistellar telescope network, a global network comprised primarily of robotically controllable evScopes 1 and 2 telescopes, each featuring a 11.4 centimeter aperture diameter. We develop dedicated deep learning algorithms to account for the relatively compact diameters of the telescopes and to extend detection thresholds. Firstly, we use traditional convolutional neural networks (CNNs) based classifiers to detect images including a satellite streak, and secondly, we use UNet to identify pixels affected by such streaks. Both neural networks are trained on realistic simulations generated using our optical Fourier simulation software, by combining observed sky backgrounds and synthetic satellite streaks spanning a wide range of orbital parameters. With this strategy, we reach excellent performance on the segmentation of images in test set, with a recall of 79.6% pixels belonging to the satellites masks, at a false positive rate of 0.001%. For 90% of streak pixels recovered by the neural networks, we obtain a precision of 96.3% on the predicted satellite masks. This exceeds the performance of non-ML algorithms and paves the way to measuring accurate satellite positions over broader magnitudes ranges and down to lower S/N, in order to increase the precision on their orbital parameters.

Keywords: Satellite trails, sky mapping, collaborative science, Unistellar telescope network, amateur observation, artificial intelligence, deep learning

1. INTRODUCTION

In astronomy, image segmentation is a key component of image analysis pipelines, either as part of the data reduction stage, or to select and characterize sources of astronomical interest. The first case includes, for instance, the identification of artefacts such as cosmic rays in images of variable depth and resolution.[1,2] The second case covers, for instance, the identification of asteroid trails,[3] the detection and characterization of tidal tails, or the study of filamentary structures across the Galactic plane.[4] Fully-convolutional neural networks have become state-of-the-art for these segmentation tasks, allowing to significantly exceed the performance of traditional source-finding algorithms such as SExtractor,[5] while also drastically decreasing the inference time. This paves the way to developing similar pipelines to analyse satellite streaks with higher efficiency than traditional, non-machine learning approaches (e.g. Hough transform).

Further author information: (Send correspondence to Pierre Janin-Potiron)
Pierre Janin-Potiron: E-mail: pierre.janin-potiron@lam.fr

Ground-based and Airborne Telescopes X, edited by Heather K. Marshall, Jason Spyromilio, Tomonori Usuda, Proc. of SPIE Vol. 13094, 1309455
© 2024 SPIE · 0277-786X · doi: 10.1117/12.3018639

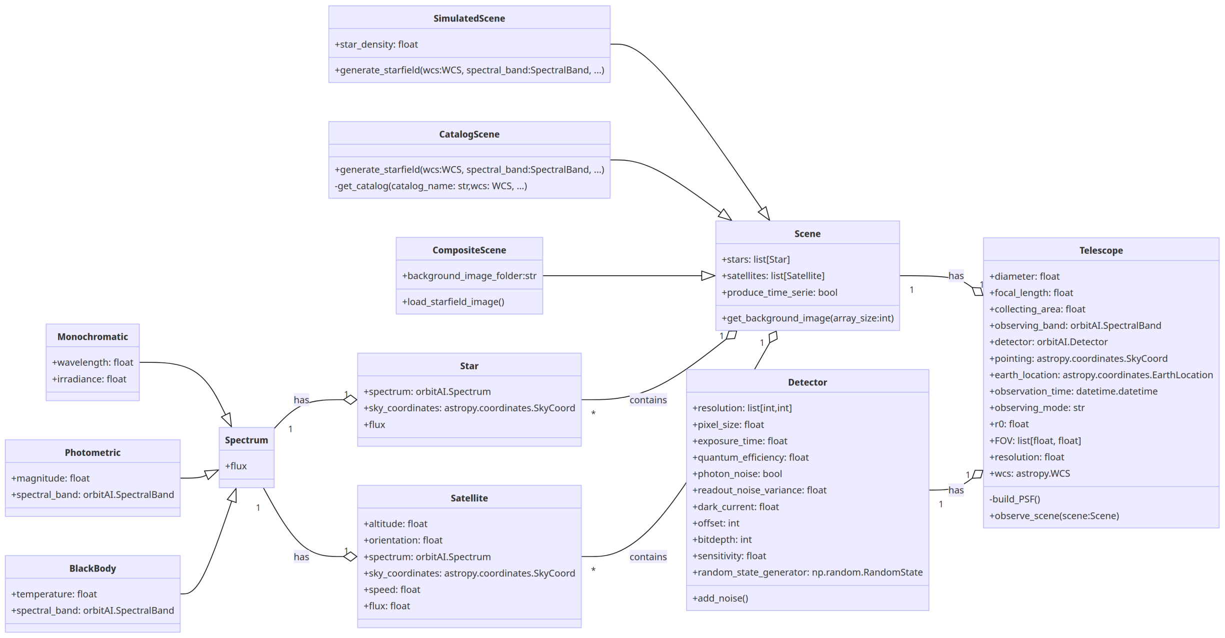
SimulatedScene
+star_density: float
+generate_starfield(wcs:WCS, spectral_band:SpectralBand, ...)
CatalogScene
+generate_starfield(wcs:WCS, spectral_band:SpectralBand, ...)
-get_catalog(catalog_name: str,wcs: WCS, ...)
CompositeScene
+background_image_folder:str
+load_starfield_image()
Scene
+stars: list[Star]
+satellites: list[Satellite]
+produce_time_serie: bool
+get_background_image(array_size:int)
Telescope
+diameter: float
+focal_length: float
+collecting_area: float
+observing_band: orbitAI.SpectralBand
+detector: orbitAI.Detector
+pointing: astropy.coordinates.SkyCoord
+earth_location: astropy.coordinates.EarthLocation
+observation_time: datetime.datetime
+observing_mode: str
+r0: float
+FOV: list[float, float]
+resolution: float
+wcs: astropy.WCS
-build_PSF()
+observe_scene(scene:Scene)
Monochromatic
+wavelength: float
+irradiance: float
Photometric
+magnitude: float
+spectral_band: orbitAI.SpectralBand
BlackBody
+temperature: float
+spectral_band: orbitAI.SpectralBand
Spectrum
+flux
Star
+spectrum: orbitAI.Spectrum
+sky_coordinates: astropy.coordinates.SkyCoord
+flux
Satellite
+altitude: float
+orientation: float
+spectrum: orbitAI.Spectrum
+sky_coordinates: astropy.coordinates.SkyCoord
+speed: float
+flux: float
Detector
+resolution: list[int,int]
+pixel_size: float
+exposure_time: float
+quantum_efficiency: float
+photon_noise: bool
+readout_noise_variance: float
+dark_current: float
+offset: int
+bitdepth: int
+sensitivity: float
+random_state_generator: np.random.RandomState
+add_noise()
has
has
has
has
contains
contains
1
1
1
1
1
1
1
1
1
1
*
*

Figure 1. `orbitAI` class diagram.

Recently, Ref. 6 have developed a versatile ML pipeline for identifying artefacts in astronomical images, including the segmentation of extended features such as satellite streaks, diffraction spikes, or nebulosities. While their method is adaptable to wide range of contaminants, the underlying training images are taken from deep, wide-field imaging surveys in the optical/near-infrared conducted with modern observing facilities (e.g. CFHT, Subaru, etc). This makes their data set heavily dominated by seeing-limited images from the best astronomical sites around the world, and with a relatively low range in depth. Due to the domain shift issues affecting neural networks, directly applying this pipeline to shallower, short-exposure images from amateur astronomers would provide moderate to poor performance, even for the segmentation of bright satellite streaks. In contrary, one need to use representative training data, approaching as much as possible the resolution, depth, and imaging quality of the target survey.

In this paper, we follow this approach and develop a deep learning framework optimized for the detection and segmentation of satellite streaks in shallow, short-exposure images from Unistellar telescopes distributed worldwide. In particular, we focus on the eVscope 1 and eVscope 2 that largely outnumber other Unistellar telescopes part of the overall network. We aim at reaching optimal detection capacities at the low-S/N regime, in order to meet our goals in characterizing the faintest possible satellites and space debris. We start with a description of the image simulation pipeline in Sec. 2, followed by the training, validation and tests of neural networks in Sec. 3 and 4.

2. IMAGE SIMULATION

The process of image simulation within the DOSSA project relies on the `orbitAI` Python package, specially built for our specific needs. In this section, we delve into the fundamental components of this package, highlighting its key features: the `Telescope` unit discussed in Sec. 2.1, the `Scene` unit detailed in Sec. 2.2, and the `Detector` unit elaborated upon in Sec. 2.3. The package's core concept is visually represented through a class diagram, depicted in Fig. 1.

2.1 Telescope

The `Telescope` class is the core of the `orbitAI` module, providing the essential functionalities to simulate an image of a satellite crossing an observation. We detail the controllable parameters in Sec. 2.1.1 and the three

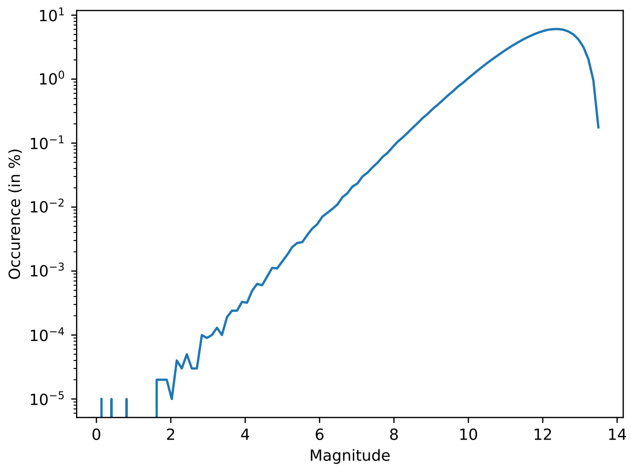

Figure 2. Star magnitude probability density function used in `SimulatedScene` mode.

observing modes in Sec. 2.1.2.

2.1.1 Parameters

- `diameter`: the physical diameter of the aperture (in m),
- `collecting_surface`: the effective collecting surface of the telescope(in m^2),
- `focal_length`: the effective focal length of the telescope (in m),
- `observing_band`: the observing band in which the observation is made (*e.g.* U, B, V, etc.),
- `pointing`: the alt-az coordinates of the center of the field pointed by the telescope (an astropy `SkyCoord` object),
- `observing_mode`: one of the three following - `diffraction_limited`, `seeing_limited` or `from_background` - the last one not being implemented yet,
- `r`$_0$: Fried parameter, used in the case of a `seeing_limited` observation.

2.1.2 Observing modes

Our simulation tool offers three observing modes, each corresponding to a distinct `Scene` object type (see section 2.2). These modes differ in how they generate the background image, which represents the star field behind the satellite streak.

- `SimulatedScene`: The background is entirely simulated. Star positions are randomly distributed, and their magnitudes follow the distribution in Fig. 2. This distribution was derived from star density data obtained from Gaia DR3 (see Fig.2 from Ref. 7).
- `CatalogScene`: As the name suggests, this mode uses real star positions and magnitudes from catalogs like Hipparcos or Gaia to construct the background image. This approach is ideal for algorithms like Astrometry.net [astrometry.net] and offers future potential, as discussed in Sec. 2.5.
- `CompositeScene`: This mode combines realism with simulation by overlaying a simulated, noise-free satellite streak onto a real background image captured with a telescope. This creates images that closely resemble real-world observations, including features like optical aberrations which are absent in purely simulated backgrounds. This makes them valuable training data for neural networks by incorporating realistic features, as discussed in Sec. 4.

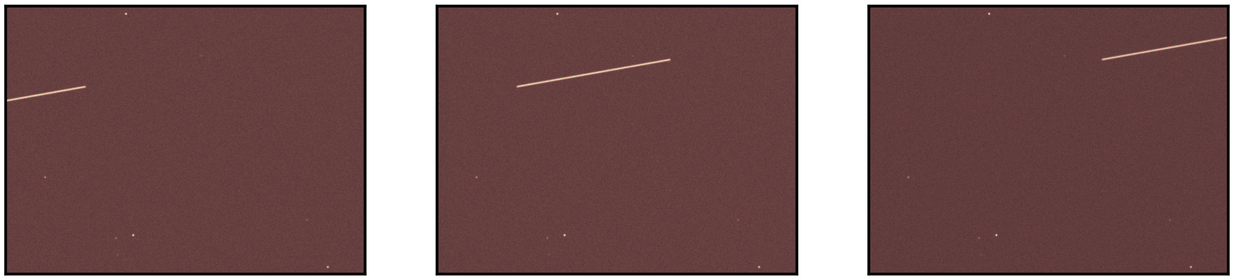

Figure 3. Example of time series produced with `orbitAI` with a satellite orientation of 10° and an altitude of 900 km. The module produces three consecutive single frames.

2.2 Scene and its components

The `Scene` class encapsulates all information about the field of view observed by the `Telescope`. We detail its key attributes in section 2.2.1. These attributes primarily consist of two class objects: `Star` (described in section 2.2.2) and `Satellite` (described in section 2.2.3). Additionally, section 2.2.4 delves into the subtleties of building a satellite streak within this scene.

2.2.1 Scene parameters

There are only three parameters accessible through the `Scene` class.

- `stars`: a list of `Star` object containing the stars present in the field of view (empty for `CompositeScene`).
- `satellites`: a list of `Satellite` objects containing the satellites present in the field of view. Currently, the module only handles the presence of one satellite at a time.
- `produce_time_series`: a boolean indicating whether the output should be a single frame or a time series (multiple consecutive frames, as shown in Fig. 3).

2.2.2 Star parameters

- `spectrum`: a `Spectrum` object that calculates photon flux for diverse inputs.
- `sky_coordinates`: alt-az coordinates of the star as an astropy `SkyCoord` object.

2.2.3 Satellite parameters

- `altitude`: the altitude of the satellite (in km).
- `orientation`: the orientation of the satellite relative to the horizontal plane (in degree).
- `spectrum`: A reference to the same ‘Spectrum‘ object used by the ‘Star‘ class (see Sec. 2.2.2).
- `sky_coordinates`: alt-az coordinates of the star as an astropy `SkyCoord` object.

2.2.4 Building streak

This section delves into the process we developed for generating reliable satellite streaks. The process comprises two key steps: (1) constructing a unitary streak and (2) convolving it with the telescope's PSF. The first step presents the most significant challenge and will be the focus of this section. We define unitary streak as the theoretical unconvolved trace left by a satellite on a detector plane. This trace correspond to a line segment with a fixed length of Δ that originates from point A and terminates at point B. Figure 4 (left) illustrate this definition. The goal of the unitary streak building is simple : provide an exact placement of the streak in the detector plane especially so that the WCS coordinates matches exactly between detector plane and sky coordinates. We achieved the desired outcome on the third attempt after exploring two alternative approaches. Here are these three steps in chronological order:

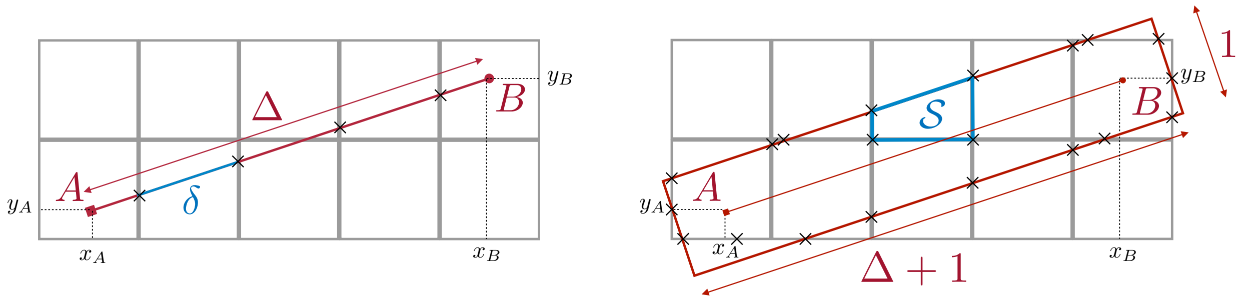

Figure 4. Left: line perfect algorithm scheme. Black crosses mark the intersection between the unitary streak and the pixel grid. Right: pixel perfect algorithm scheme. Black crosses mark the intersection between the unitary rectangle streak and the pixel grid.

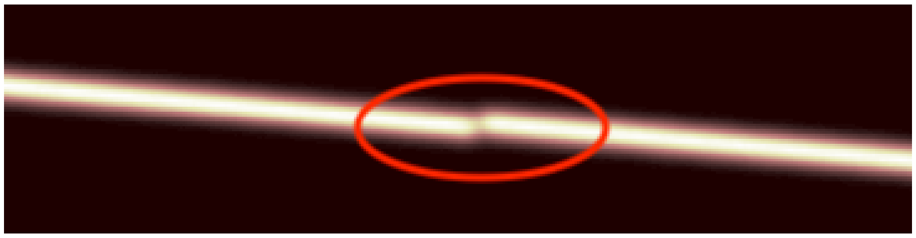

Figure 5. Observed misalignment in consecutive time series frames demonstrating the limitations of the purely computational subpixel streak placement (first approach in Sec. 2.2.4).

- **Computational algorithm**: Our initial approach involved creating a one-dimensional array filled with ones. The array's length was set to the ceil part of the streak length (*i.e.* $\lceil\Delta\rceil$). To account for the streak length's fractional part, we assigned a fractional value to the last element. Subsequently, we rotated this array using the SciPy's image rotation function with spline interpolation. Finally, we positioned the resulting image with a subpixel shift to achieve the desired location. However, as evident in the time series frames superimposed in Fig. 5, this approach resulted in poor streak placement accuracy.
- **Line perfect algorithm**: In our second approach, we adopted a different strategy. We calculated the coordinates of the unitary streak's intersection points with the pixel grid (represented as black crosses on Fig. 4, left). Finally, for each pixel intersected by the streak, we assigned a value proportional to the segment length within that pixel (length δ of the blue line on Fig. 4, left). This approach yielded improved accuracy compared to the previous one. However, limitations remained for subpixel displacements or rotations. As shown in Fig. 6 (top center), where the unitary streak is centered relative to the pixel grid, we observe that for rotation angles $|\theta| \leq \arctan(1/\Delta)$, the generated streak appears identical to a vertical one. Similarly, displacements smaller than a pixel produce the same effect.
- **Pixel perfect algorithm**: Finally, the third approach proved successful. Building upon the previous method, we introduced a second dimension to the unitary streak, transforming it into a rectangle with a length of $\Delta + 1$ and a width of 1. Rather than calculating the length of a segment within each pixel, we now assign a value to each pixel based on the area $\mathcal{S}$ of the polygon formed by the intersection points and one or two corners of the pixel itself (see Fig. 6, right). This approach successfully resolves the limitations associated with subpixel displacements and rotations, as evidenced by the results in Fig. 6 (bottom center)

2.3 Detector

The `Detector` class offers two essential functionalities: (1) adding noise, mimicking the inherent randomness of real detectors, and (2) applying a Bayer filter mosaic effect, replicating the color filter pattern present in image sensors.

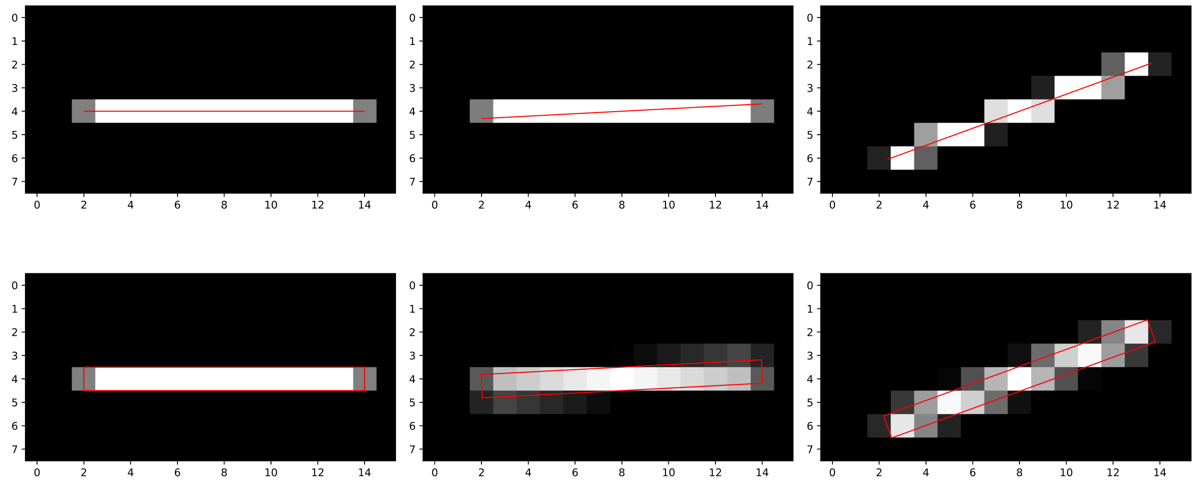

Figure 6. Top row: "line perfect" streak. Bottom row: "pixel perfect" streak. Left: vertical streaks. Middle: streaks at a 3° angle. Right: streaks at a 20° angle.

2.3.1 Parameters

- `resolution`: resolution of the image expressed as `width` × `height` in pixel,
- `pixel_size`: size of the pixel in μm,
- `exposure_time`: exposure time in second,
- `quantum_efficiency`: detector's quantum efficiency in e^-/ph,
- `photon_noise`: a boolean stating if photon noise should be applied to the image,
- `readout_noise_variance`: the variance of the readout noise in e^-,
- `dark_current`: dark current in e^-/s,
- `offset`: constant offset in ADU,
- `bitdepth`:
- `sensitivity`: sensitivity of the detector in ADU/e^-.
- `bayer_order`: literal representation of the Bayer filter pattern (*e.g.* RGGB).
- `white_balance`: a `WhiteBalance` object containing the relative gain of each color channel.

2.3.2 Noise model

The adopted noise model is a simplified version of the model presented in Ref. 8. Figure 7 illustrates its block diagram within the `orbitAI` framework. The physical units of each variable are indicated in red throughout the diagram. Model steps are the following:

1. **Photon arrival**:
   (a) The model starts by considering the incoming photon flux, measured in units of photons per square meter per second (ph/m^2/s).

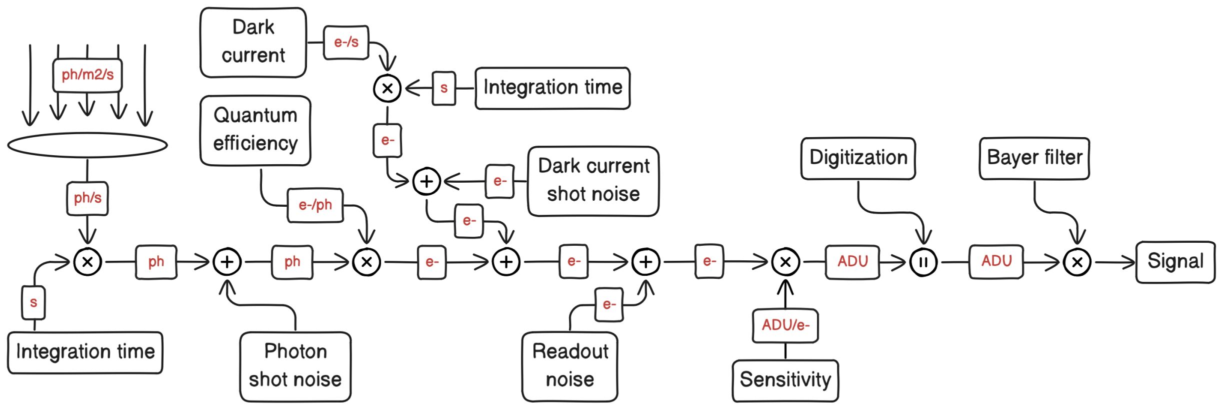

Figure 7. Block diagram of the noise modeling used in the DOSSA simulations. The rightmost block stands for digitization of the signal by the ADC.

  (b) This flux enters the detector's pupil and is integrated over time, resulting in the total number of photons captured by the image.
  (c) A normalized PSF having a unit integral value then multiplied by this number of photons. This step accounts for the spatial distribution of the photons within the image due to the detector's optical properties.
  (d) The result of this multiplication is a noise-free image, initially represented in units of photons.

2. **Noise addition**: The noise-free image undergoes four transformations representing the different noise sources.
   (a) Photon noise is added from a Poisson distribution with an expected value for each pixel corresponding to the pixel value itself.
   (b) The image is then converted from photons to electrons by multiplying the signal by the detector's quantum efficiency.
   (c) Dark current is added as a constant Poisson noise across the entire detector, with an expected value of dark_current $\times$ integration_time.
   (d) Readout noise is added using a normal distribution with zero mean ($\mu = 0$) and a variance of `readout_noise_variance` (σ^2).

3. **Digitization**:
   (a) Camera gain is applied by multiplying the signal by the detector's sensitivity.
   (b) Digitization step converts the image to a digital format by taking the integer part of the signal and truncating it to a user-defined bit depth.
   (c) Bayer filter is finally applied by multiplying the signal by a mosaic filter (e.g., RGGB) to capture color information.

2.4 Perceptual evaluation of `orbitAI` images

In this section, we present a visual comparison between images generated by `orbitAI` and real-world images captured with an eVscope 2. Figure 8 shows two images of the same scene. The left image is a real capture from the eVscope 2, while the right image is a simulation created using `orbitAI`'s `CatalogScene` functionality. We successfully reproduced the background star field with accuracy, and the satellite streak appears positioned precisely. While the initial comparison reveals promising similarities, a key difference lies in the overall image

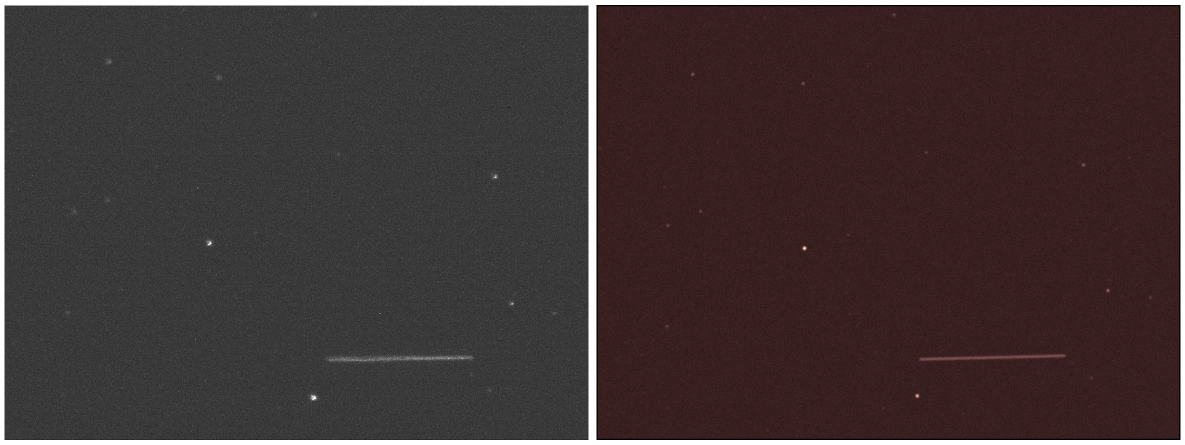

Figure 8. Left: satellite streak observed with a Unistellar eVscope 2. Right: simulated satellite streak produced by `orbitAI` using a `CatalogScene`.

quality. The real eVscope 2 image (Fig. 8, left) exhibits degradation compared to the simulated image (Fig. 8, right). This is mainly due to the effects of atmospheric turbulence and/or optical aberrations within the telescope itself. These factors distort the PSF and ultimately reduce the image's sharpness and fidelity. To achieve greater realism, potential improvements for `orbitAI` simulations are discussed in Section 2.5.

To gain further insights, we constructed histogram plots for the images showcased in Fig. 8. These histograms, presented in Fig. 9, reveal a good level of agreement between the simulated (orange) and real-world (blue) data distributions. The histograms also show a noticeable difference in the distribution of low and high pixel values between the real and simulated images. The simulated images appear to lack pixels in both the very low and very high intensity ranges. The absence of low-value pixels in the simulations could be due to limitations in modeling non-uniformities within the detector noise. Including a more comprehensive noise model in `orbitAI` might improve the simulation's accuracy in this aspect. The scarcity of high-value pixels might be related to missing hot pixel simulation in `orbitAI`. Investigating and incorporating hot pixel behavior into the simulation process could address this discrepancy.

To mitigate the discrepancies observed in histogram analysis, we opted to utilize the `CompositeScene` mode, simulating just the satellite streak onto a real background image. This approach minimizes the influence of potential inconsistencies between the real and simulated data. By focusing solely on the simulated satellite streaks, we aim to achieve the best possible match between the training data fed to the neural network and the real-world data it will encounter during inference.

2.5 Future improvements

While `orbitAI` currently generates simulated images suitable for neural network training, we are actively seeking improvements. This section outlines a non-exhaustive list of future development areas to further enhance our algorithms.

- Adding jitter: Telescopes might experience vibrations, causing the satellite's trajectory on the detector plane to deviate from the straight line seen in our simulations. To mimic this effect, we can transform the rectangle from Fig. 6 into a "wavy rectangle" that reflects the satellite's non-linear path.
- Adding flux variation: During image acquisition, satellite flux can vary due to two main factors. First, atmospheric scintillation, caused by high-altitude turbulence in the Earth's atmosphere, can lead to fluctuations in the signal intensity. Second, as the satellite moves through space, its orientation relative to the sun changes, affecting the amount of reflected light captured by the telescope and contributing to an observed change in brightness.

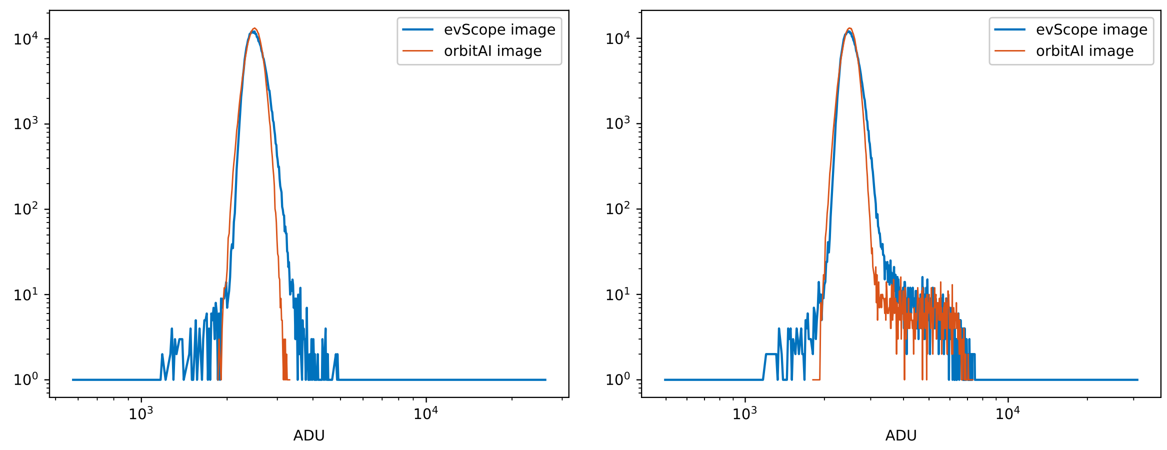

Figure 9. Histograms of real `eVscope` images (in blue) and images simulated in `CatalogScene` mode with `orbitAI` (in orange). Left: histograms for a scene without satellite. Right: histograms for a scene with a satellite. The scale is log-log.

- Adding phase aberrations: While travelling to the detector, the wavefront coming from the scene encounters phase aberrations, mostly due to atmospheric turbulence and optical aberrations within the telescope itself, leading to a quality loss in the final image.These phase aberrations can be modeled in the simulation by applying a phasescreen to the pupil plane.
- Extracting the PSF from the background image for improved realism: To achieve a more realistic composite image where the simulated streak seamlessly blends with the background, we can extract the PSF from the background itself. This extracted PSF can then be used to generate the satellite streak, ensuring a more accurate representation. However, this method requires the presence of at least a few bright stars in the image. The absence of such stars limits the applicability of this technique.
- Setting satellite position from its orbital elements: Currently, the `Telescope` object points to a user-defined sky region through the `pointing` attribute and the `Satellite`'s coordinates are set relatively to this value. However, for a more general simulation framework, we propose incorporating orbital elements into the `Satellite` object as this would allow us to validate and improve our orbital element estimation algorithms

3. SATELLITE DETECTION WITH DEEP LEARNING

Analyzing satellite streaks for space surveillance not ony requires to identify the presence of such satellites over broad ranges of telescope images, but also to accuretly identify the pixels belonging to the streaks in order to reconstruct the satellite orbital parameters. We follow a two step approach to detect exposures including satellite streaks and to identify pixels affected by such streaks. First, the binary classification of images with or without streak is conducted with convolutional neural networks (CNNs) described in Sec. 3.1. Second, the segmentation of images with a detected streak is conducted with a UNet described in Sec. 3.2. Both types of networks are trained on `orbitAI` simulations, either in `SimulatedScene` or in `CompositeScene` mode, using various data sets with different parameter distributions. In Sec. 4, we further test the networks trained on composite simulations which provide better performance.

3.1 Image classification with CNNs

The neural networks for classification are simple CNNs inspired from AlexNet.[9] Given the large dimensions of eVscope 1 and 2 images of 664×496 and 1032×768 pixels, respectively, the CNNs are trained on smaller cutouts and the model is then applied to the entire images using sliding windows.

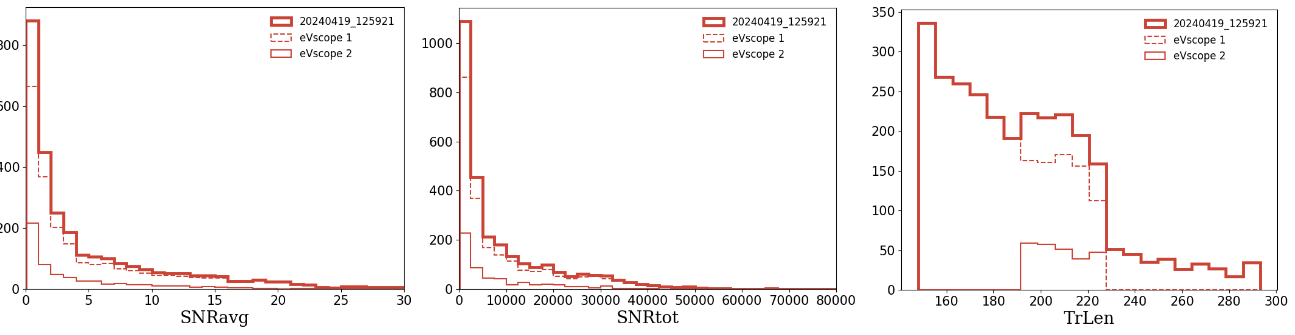

Figure 10. Properties of satellite streaks injected in the composite simulations. Left: Average S/N per pixel within the ground-truth satellite masks. Middle: Total signal obtained by summing all pixels within the ground-truth masks. Right: Streak length in pixels.

The baseline CNN has two convolutional layers with 5×5, and 4×4 kernels, and 32, and 64 filters, respectively. Max-pooling layers with 4×4 kernels and stride = 4 are inserted between the convolutional layers. This is followed by two fully-connected hidden layers with 512, and 16 neurons, and a single-neuron output layer. Non-linear ReLU activations are applied between each layer, and the last fully-connected layer has a sigmoid activation, resulting in the output score in the range [0;1].

Several variations around this baseline CNN have been tested. We tried adding a third and a fourth convolutional layer, but given the simplicity of features over the training images, this did not improve the classification performance. In addition, we tried using larger convolutional kernels, increasing the number of filters per convolutional layer, using 2×2 kernels and stride = 2 in the max-pooling layers, and increasing the number of fully-connected layers. These CNNs perform similarly to the baseline and are not discussed in Sec. 4.

All CNNs are trained on a set of composite simulations from `orbitAI` obtained by injecting simulated streaks on real eVscope 1 and 2 images of stars and sky background. We use 71 sequences of satellite observations taken by amateur astronomers until April 2024. These sequences include between 200 and 400 exposures of 150 ms, and span a broad range in observing conditions and data quality. The sequences are inspected visually and with a source-finding algorithm based on Sextractor[5] to identify and discard exposures including a detected satellite or a streak candidate. After cleaning, this results in a total of 26103 telescope images.

To simulate the streaks, we randomly draw satellite altitudes between 600 and 900 km, corresponding to the low Earth orbit (LEO), using a uniform distribution. The satellite magnitudes and orientations are drawn from flat distributions between 5 and 10 mag, and between 0 and 360 deg, respectively. The low S/N of these short, 150 ms exposures prevent to extract the seeing FWHM per individual images from the star light profiles, and we therefore use a representative range in Fried parameter r_0 between 2 cm and 4.5 cm. The ground-truth satellite masks from the simulations are obtained by convolving the linear satellite streak with a disk of diameter taken as the PSF FWHM. Only the injected streaks fully enclosed within the fields-of-view of eVscope 1 and 2 images are accepted.

While these simulations only contain of 71 independent fields-of-view, the random injection of streaks at various positions, orientations, and brightnesses guarantees the unicity of each 26103 individual image. The S/N of simulated streaks are evaluated in two different ways:

- using the average signal per pixel within the ground-truth satellite masks ($\mathrm{SNR_{avg}}$),
- and using the total signal obtained by summing all pixels within the ground-truth masks ($\mathrm{SNR_{tot}}$).

In both cases, the background noise is obtained from the width of the Gaussian function fitted to the background image before streak injection. The histograms of $\mathrm{SNR_{avg}}$, and $\mathrm{SNR_{tot}}$ over the training set are shown in Fig. 10, together with the lengths of simulated streaks in pixel units.

We use smaller image cutouts to train and validate the CNNs more efficienctly. Various sizes of image cutouts were tested, with values ranging between 128 and 512 pixels, and better performance are obtained for CNNs trained and tested on 128×128 pixel cutouts. In addition, we also tried varying the minimal number of pixels within the satellite masks, for positive examples in the training set. No significant impact was found when varying this lower limit between 50 and 500 pixels, and we eventually fixed to a minimum of 100 pixels.

The normalisation of images has a major impact on the performance of the CNN classifier on real eVscope images. Different types of image normalisations were tested, such as normalising pixel values to the range [0:1], or subtracting the mean and dividing by the variance computed either over the entire telescope images, over the small 128×128 pixel cutouts, or over sky background regions. The baseline normalisation procedure that showed better performance consists in applying the following transformation to each image cutouts before training, validation, and test:

$$\widetilde{\mathrm{D}} = \mathrm{arcsinh}\left(\frac{\mathrm{D} - \mu_{\mathrm{img}}}{\sigma_{\mathrm{img}}}\right), \tag{1}$$

where D and $\widetilde{D}$ are the input and normalised images, respectively, μ_{img} and σ_{img} are the average and standard deviation of pixel values within the entire 128×128 pixel cutouts. The arcsinh term is crucial in reducing the relative offset between small and large pixel values (as also found by Ref. 6).

In total, 33408 and 8352 normalised image cutouts are kept for training and validation, respectively, with a balanced ratio of positive and negative examples. Parameter optimization is achieved with mini-batch stochastic gradient descent with a batch size of 32. We minimize the binary cross-entropy loss expressed as:

$$\mathrm{BCE}(\mathrm{y}, \mathrm{p}_{\mathrm{CNN}}) = -\frac{1}{\mathrm{N}} \sum_{\mathrm{k}=0}^{\mathrm{N}} \mathrm{y}_{\mathrm{k}} \log(\mathrm{p}_{\mathrm{k}}) + (1 - \mathrm{y}_{\mathrm{k}}) \log(1 - \mathrm{p}_{\mathrm{k}}), \tag{2}$$

where y_{k} and p_{k} are the values of the ground-truth and predicted labels respectively, and N is the number of images per batch. Values of p_{k} result from the sigmoid activation and range from 0 to 1. After a preliminary search for the best neural network hyperparameters, we fixed the learning rate to a value of 0.001 and we use an Adam optimizer. The networks are trained over 100 epochs and saved at the epoch with minimal validation loss.

3.2 Image segmentation with UNets

The identification of pixels affected by satellite streaks is conducted with state-of-the-art neural network architectures for image segmentation tasks. As baseline, we use the UNet architecture presented by Ref. 2 as part of their Cosmic-coNN pipeline for cosmic ray identification in images from Las Cumbres Observatory (extending the work from Ref. 1 for HST imaging). This is a type of encoder-decorder convolutional neural network, providing satellite mask predictions with same dimensions as the input. Despite the different morphological signatures between cosmic rays and satellite streaks, the Cosmic-coNN network architecture is optimized for segmentation of low-resolution astronomical images, and is sufficiently flexible for broad range of image sizes and resolutions. It is also better suited than the original UNet from Ref. 10 that was developed for medical imaging.

The Cosmic-coNN UNet (see Fig. 3 of Ref. 2) starts with a single telescope image and a convolutional layer with $3{\times}3$ kernel and ReLU activation. The decoder has two downsampling stages with $2{\times}2$ max-pooling layers and one convolutional layer per stage. The bottleneck includes a single convolutional layer with $3{\times}3$ kernel and ReLU activation, and its features maps are four times smaller than the input telescope images. The decoder comprises two upsampling stages with a $2{\times}2$ transposed convolution with stride of 2, and one convolutional layer per stage. Convolutional layers have stride of 1 and either have 32 or 64 filters. Skip connections allow to copy and concatenate information from the various stages of the encoder to the decoder, in order to propagate both low- and high-level morphological features to the decoder. The last layer has a sigmoid activation.

We addition, we tried modifying the architecture of this baseline UNet by adding or removing one downsampling/upsampling stage in order to increase or decrease the network depth. On the one hand, removing one stage lowers the generality of the model to different feature sizes and substantially degrades the reconstruction of satellite masks. On the other hand, adding one downsampling/upsampling stage does not have a strong impact.

We also tested various numbers of features maps per convolutional layer. Finally, we trained the original UNet from Ref. 10 without obtaining better performance, and we therefore kept the Cosmic-coNN architecture as baseline.

The UNet is trained and validated on images of 256×256 pixels obtained by extracting random cutouts from the composite simulations described in Sec. 3.1. We use a total of 30000 cutouts with a balanced ratio of 50% images with injected streaks, and 50% without streak, in order to train the networks to exclusively extract the pixels affected by the satellites and to ignore stars and artefacts over the fields. To construct the set of images with satellite streaks, we randomly draw simulations from the overall set of 26103 images, and we randomly draw a cutout center, while also requiring a minimum of 100 pixels within the simulated masks. Ground-truth masks are set to binary pixel values equal to 0 and 1, for pixels outside and within the satellite masks, respectively. The baseline image normalisation of CNN classifiers is then applied to these ground-truth images.

These simulations in `CompositeScene` mode are covering a large diversity in observing conditions, streak properties, and spatial density of background stars in the field, while also representing the differences in detector properties between eVscope 1 and 2 telescopes. This strategy helps train a neural network able to generalize to future sets of observing sequences registered by amateur astronomers, regardless to variations in seeing FWHM, or to the presence of stars, and artefacts in the fields. In line with future space surveillance objectives, the model also needs to guarantee reliable satellite mask predictions down to low S/N. Other UNet trained on data sets fully simulated with `orbitAI` do not meet these requirements and are discarded.

Similarly to the CNN classifiers, the UNet is trained up to 100 epochs with mini-batch stochastic gradient descent. We minimize the binary cross-entropy loss per pixel, as commonly done in classification problems:

$$\mathrm{BCE}(\mathrm{y},\mathrm{p}) = -\frac{1}{\mathrm{N}} \sum_{\mathrm{k}=0}^{\mathrm{N}} \sum_{\mathrm{j}=0}^{\mathrm{Y}} \sum_{\mathrm{i}=0}^{\mathrm{X}} \mathrm{y}_{\mathrm{i,j,k}} \log(\mathrm{p}_{\mathrm{i,j,k}}) + (1-\mathrm{y}_{\mathrm{i,j,k}}) \log(1-\mathrm{p}_{\mathrm{i,j,k}}), \tag{3}$$

where $\mathrm{y}_{\mathrm{i,j,k}}$ and $\mathrm{p}_{\mathrm{i,j,k}}$ are the values of the ground-truth and predicted satellite masks for pixel (i,j) from image k, respectively, and X, Y, and N, are the image dimensions in pixel units and the number of images per batch, respectively. The UNet is trained over 100 epochs with early stopping, and we obtain a small generalization gap between the training and validation loss curves.

4. EVALUATION OF PERFORMANCE AND DISCUSSION

4.1 Construction of the test sets

4.1.1 Testing the CNN classifiers

The CNN classifiers are tested on a range of observing sequences of LEO satellites collected by amateur astronomers with eVscope 1 and 2 telescopes between January and March 2024. Each sequence comprises nearly 400 individual exposures of 150 ms, for a total duration of 1 min, allowing to detect the satellites crossing the fields-of-view over multiple exposures. We use a total of 67 test sequences corresponding to observations of 67 different satellites over different fields-of-view with variable spatial densities of stars. These sequences were inspected both visually and with a source-finding algorithm based on Sextractor,[5] in order to identify the exposures containing a detected satellite streak and to build a ground-truth catalog of positives (exposures with streak) and negatives (exposures without streak). Exposures with streaks crossing the image borders are considered as positives. On average, the streaks of primary satellites targeted by the observers are detected over three to 10 exposures along the 400-image sequences. A secondary, untargetted satellite is detected in 5 out of 67 sequences.

After labelling, the test set contains a total of 387 positive images with detected satellite streaks, and 23283 negative images with stars, sky background, but no detected streaks. This contains a wide diversity of scenes with a total number of stars ranging from two to 15, various types of observing conditions, and imaging quality. The sequences with lower data quality were taken with variable clouds, and some exposures are affected by field-of-view drifts, resulting in distorted light profiles for stars in a subset of exposures. This allows to test the robustness of the CNN classifications in presence of realistic observing artefacts. The total S/N of streaks from primary satellites in these 67 sequences range from 200 to 50,000, with a peak in the range 8000-10,000. Overall, about 20% sequences are observed with eVscope 1 telescopes, and 80% with eVscope 2.

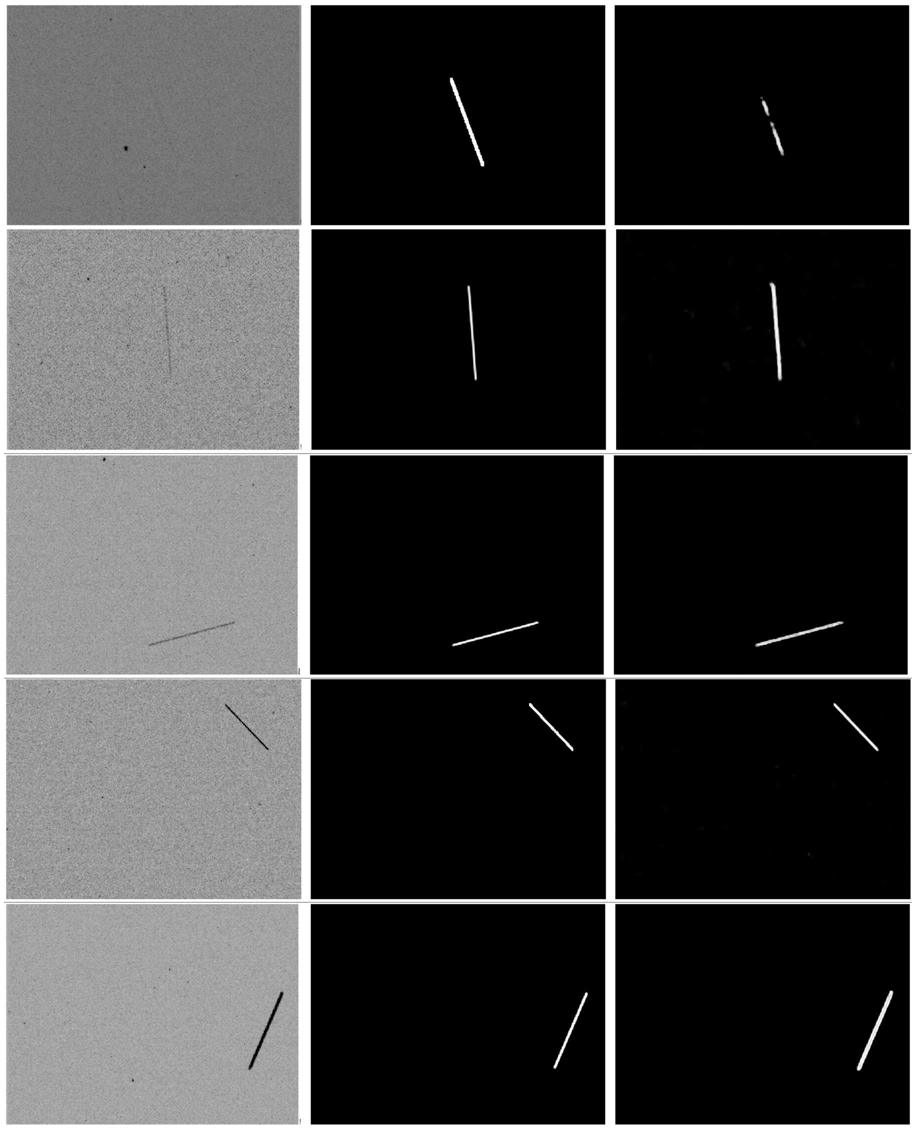

Figure 11. Example of satellite streaks with different values of $\mathrm{SNR}_{\mathrm{avg}}$ simulated with `orbitAI`. The `CompositeScene` images with injected streaks are shown on the left panels, and the ground-truth satellite masks on the middle, while the right panels show the binarized satellite masks predicted by the UNet. From top to bottom: streak with $\mathrm{SNR}_{\mathrm{avg}} \in [0;0.5]$ (blue curve in Fig. 14), with $\mathrm{SNR}_{\mathrm{avg}} \in [0.5;1]$ (orange curve), with $\mathrm{SNR}_{\mathrm{avg}} \in [1;3]$ (green curve), with $\mathrm{SNR}_{\mathrm{avg}} \in [3;10]$ (red curve), with $\mathrm{SNR}_{\mathrm{avg}} \in [10;30]$ (purple curve).

The CNNs are applied to the telescope images of 664×496 or 1032×768 pixels using sliding windows of $128{\times}128$ pixels with an overlap of 20 pixels. The CNN scores are computed separately for each sliding window, without accounting for correlations between neighbouring windows. We then assign a score prediction per image $\mathrm{p}_{\mathrm{img}}$, by combining the CNN score $\mathrm{p}_{\mathrm{CNN,l}}$ for each individual sliding window l. Several methods have been tested, and the best performance were obtained by simply assigning the highest score to the overall image as follows $\mathrm{p}_{\mathrm{img}} = \max\left(\mathrm{p}_{\mathrm{CNN,l}}\right)$. Since real satellite streaks are longer than the window size, we also tried combining the scores from neighboring windows, but without improving the performance discussed in Sec. 4.

4.1.2 Testing the segmentation UNets

The UNet is tested on a subset of 2174 composite simulations described in Sec. 3.1, and following the parameter distributions shown in Fig. 10. The UNet is trained on 256×256 pixel cutouts and its application to the 664×496 and 1032×768 pixel images from eVscope 1 and 2 telescopes can be conducted in several ways. Since the UNet is a fully convolutional architecture, the trained model can in principle be applied directly to segment the entire images. This however leads to suboptimal performance and we rather use an overlapping sliding window of 256×256 pixels, equal to the dimensions of the training images. For a given pixel, the UNet score is taken as the average of all scores evaluated from the different sliding windows covering this pixel.

The segmentation accuracy gradually increases when decreasing the step size between sliding windows. As a tradeoff between accuracy and computational cost, we use steps of 10 pixels, which is sufficient to generate smooth satellite masks. We also discard a 10-pixel wide band around the image borders to exclude pixels with a single UNet score. This also ensures that performance are not altered by imaging artefacts near the borders.

4.2 Performance metrics

The performance of the neural networks for classification and segmentation are evaluated with a range of metrics.

For the CNN classifiers, we use the receiver Operating Characteristic curves (ROC) corresponding to the true positive rates plotted as a function of false positive rates by varying the network scores threshold in the range $[0;1]$. The true positive rate (TPR or recall) is defined as:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{AP}}, \tag{4}$$

where TP is the number of images with satellite streak correctly identified by the CNN, and AP is the total number of images with streak in the test set. The false positive rate (FPR) is defined as:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{AN}}, \tag{5}$$

where FP denotes images with a streak candidate wrongly identified by the CNN, and AN corresponds to the number of images without satellite streak in the test set.

We also compute the area under ROC curve (AUC). Networks with better classification performance have higher TPR, lower FPR, and therefore a higher AUC approaching a value of 1. The classification step is aimed at efficiently selecting individual exposures with candidate satellite streaks among the entire observing sequences, for further analysis and segmentation with the UNet. We therefore measure the TPR at a FPR of 1%, to optimize the recall at low contamination. Exposures incorrectly classified by the CNN as positives can be discarded during the second, segmentation stage, using the overall shape and area of the predicted satellite masks.

For the segmentation UNet, we also evaluate the ability to infer accurate satellite masks using the ROC curves. This is a pixel-wise binary classification problem, with the TPR and FPR now measured by summing over pixels in each image, and then over all images in the test set. The true positive rate is therefore:

$$\mathrm{TPR} = \frac{\text{number of streak pixels correctly classified}}{\text{total number of ground truth streak pixels}}, \tag{6}$$

and the false positive rate is defined as :

$$\mathrm{FPR} = \frac{\text{number of nonstreak pixels incorrectly classified as positives}}{\text{total number of ground truth nonstreak pixels}}. \tag{7}$$

We aim at optimizing the TPR, namely the fraction of satellite streak pixels correctly classified, for low FPR, in order to correctly reconstruct the satellite masks with a limited number of additional pixels incorrectly flagged by the neural network. To limit the number of false positive to 10 pixels maximum per image, we measure and

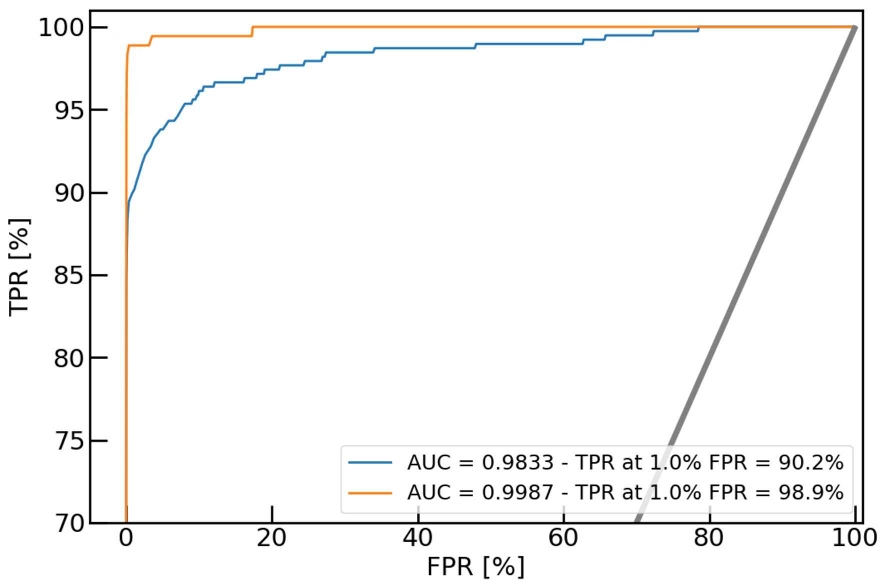

Figure 12. Performance of the baseline CNN evaluated on the overall test set (blue curve) using the Receiver Operating Characteristic curves (ROC). We optimize the TPR for a contamination level of 1%. The orange curve illustrates the higher performance obtained after excluding streaks cropped in the telescope images.

optimize the recall at a very low FPR of 0.001%. While the AUC and TPR at 0.001% FPR help quantify the level of contamination, these metrics can be combined with the precision, which is defined as:

$$\text{precision} = \frac{\text{number of streak pixels correctly classified}}{\text{number of predicted streak pixels}}. \tag{8}$$

Precision is only sensitive to positive pixels, and therefore insensitive to the low ratio of streak over non-streak pixels in real telescope images. Similarly to the ROC, we also plot the precision-recall curves to compare the relative performance of individual neural networks. Finally, we quantify the precision for a fixed threshold in TPR (or recall) of 90% to compare and optimize the precision of predicted UNet masks.

4.3 Performance of the classification networks

The performance of the baseline CNN classifier on the test set is shown in Fig. 12. We obtain an AUC of 0.9833 and a TPR at 1% FPR of 90.2%. These excellent performance demonstrate the ability of the CNN to generalize to real observing sequences.

Most exposures with a satellite streak missed by the CNN actually correspond to satellites crossing the image borders. These exposures are providing incomplete masks and are therefore useless for estimations of orbital parameters. We replotted the ROC curves after excluding these exposures from the test set and found a major improvement in performance, with an AUC of 0.9987 and a recall at 1% FPR of nearly 99%.

The CNN is able to identify and exclude images without satellite streak despite the diversity in observing conditions and imaging quality covered by these 67 sequences. In particular, the network correctly classifies the few exposures affected by unstable telescope pointing and showing extended star light emissions, despite the similarities with the morphological signatures of satellites. This network classifies an entire observing sequence of 400 exposures within 10 s on a single CPU. This allows to efficiently and robustly exclude empty exposures before the segmentation stage, which is also a major advantage in terms of computational time.

4.4 Performance of the segmentation networks

Our UNet model reaches excellent performance on the overall test set of 2174 images, with an AUC of 0.9923 and a TPR (or recall) of 79.6% at a FPR of 0.001% as shown in Fig. 13. When lowering the score threshold, the TPR

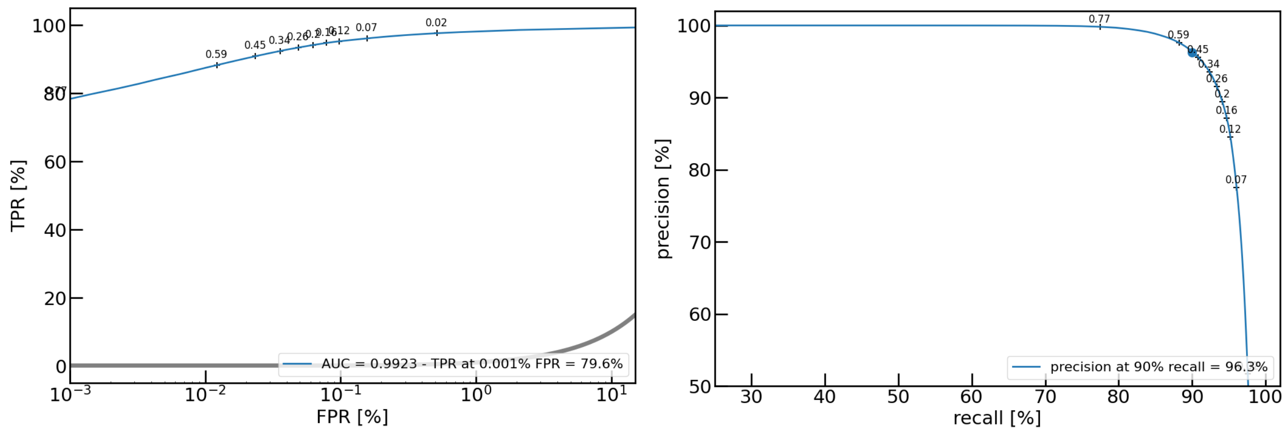

Figure 13. Performance of the baseline UNet (architecture from Cosmic-coNN) evaluated on the overall test set. The small crosses mark the evolution of network score thresholds along the curves. Left: Receiver Operating Characteristic curves (ROC). We optimize the TPR for a low FPR of 0.001% (extreme-left part of the plots), which corresponds to elevated ratios of streak pixels correctly identified, for a maximum of 10 false positive pixels. Right: Precision-recall curves. We optimize the precision at a recall of 90%, as marked with the colored circles.

increases to $\simeq$85% and $\simeq$90% at 0.01% and 0.1% FPR, respectively. Moreover, for thresholds corresponding to 90% of streak pixels recovered by the UNet (90% recall), we obtain a precision of 96.3% on the predicted satellite masks. These performance illustrate our accurate reconstructions of the linear streaks, without discontinuities, and without detection of secondary contaminant streaks over the fields (see lower panels in Fig. 11). The few pixels incorrectly classified as streak pixels (false positives) either match the position of the brightest stars in the fields, or fall over empty background regions. This broad distribution of FP pixels helps further clean the masks with separate criteria based, for instance, on shape, or compactness. Overall, these aspects make the predicted satellite masks robust enough to serve as inputs for orbital parameter measurements with separate pipelines.

Our goal is to obtain a versatile model able to segment satellite streaks of arbitrary length and S/N, over stellar fields with variable number densities and magnitude distributions, and over various types of observing conditions. We evaluate the performance of the UNet in separate bins to further quantify the impact of:

- telescope version (eVscope 1 or 2),
- S/N of the injected satellite streak, using both $\mathrm{SNR}_{\mathrm{avg}}$ and $\mathrm{SNR}_{\mathrm{tot}}$ definitions,
- total length of the streak in pixels,
- value of R0 applied to the simulated satellite mask,
- number of stars in the background image.

We find that performance mainly depend on the S/N of the injected streak. With use the $\mathrm{SNR}_{\mathrm{avg}}$ definition and divide the test set into five bins with $\mathrm{SNR}_{\mathrm{avg}} \in$ [0;0.5], [0.5;1], [1;3], [3;10], and [10;30], to keep similar number of images per bin. Since the ground-truth masks are convolved with the PSF, averaging over all pixels enclosed in these masks drives the $\mathrm{SNR}_{\mathrm{avg}}$ to these low values. Examples of simulated streaks from these five bins are shown in Fig. 11. In the highest S/N bin, we reach an AUC$\simeq$1, an excellent TPR of 91.3% at 0.001% FPR, and a nearly perfect precision of 99.8% at 90% recall. The performance gradually decrease for lower S/N, as shown with the ROC and precision-recall curves in Fig. 14. Compared to the values in the highest S/N bin we find that, at 0.001% FPR, the recall decreases by 2.4% for $\mathrm{SNR}_{\mathrm{avg}} \in$ [3;10], by 6.1% for $\mathrm{SNR}_{\mathrm{avg}} \in$ [1;3], by 18.3% for $\mathrm{SNR}_{\mathrm{avg}} \in$ [0.5;1], and by 50.3% for $\mathrm{SNR}_{\mathrm{avg}} \in$ [0;0.5]. Variations in the precision-recall curves are more moderate, except for the lowest S/N bin reaching a precision of only 29.8% at 90% recall. We find similar results when dividing the test set into $\mathrm{SNR}_{\mathrm{tot}}$ bins.

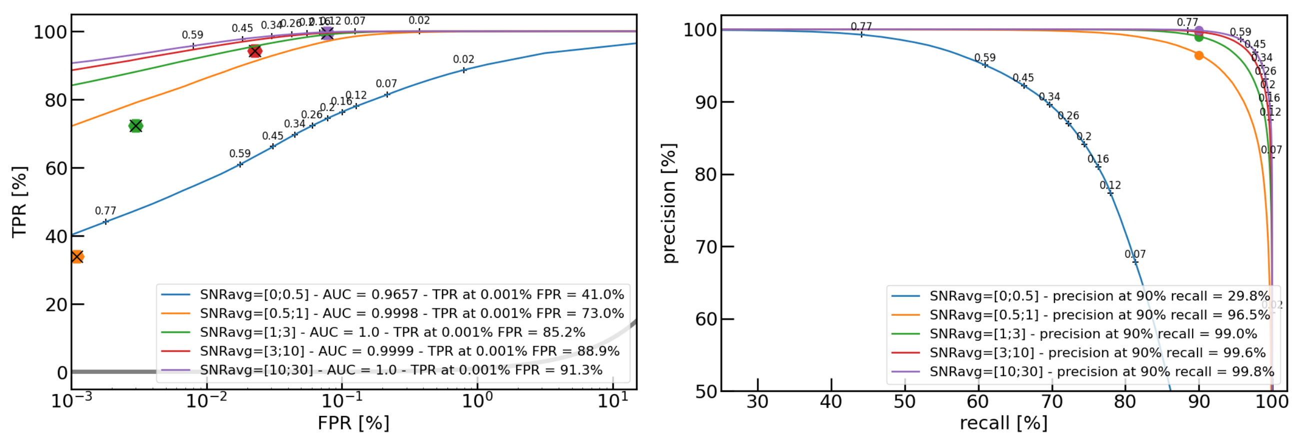

Figure 14. Same as previous figure, after dividing the test set into bins of $\mathrm{SNR}_{\mathrm{avg}}$. The large hexagons shows the performance of image segmentation with the SExtractor script, which serves as the baseline non-ML method.

The top row in Fig. 11 shows that simulated streaks with $\mathrm{SNR}_{\mathrm{avg}} < 0.5$ are typically near the noise level, and hardly distinguishable by eye. Even though these tricky examples dominate the overall distribution of images in the training set (see Fig. 10), the UNet has difficulties in extracting these types of streaks as a single, linear feature. The predicted satellite masks recover the correct position of the faint injected streaks, but they are typically separated into multiple, discontinuous portions. On a positive note, since the number of FP pixels remains very low for streaks with $\mathrm{SNR}_{\mathrm{avg}} < 0.5$, the UNet predictions help identify the satellite positions securely.

Besides the dependence on S/N, we find a significant variation in performance with the telescope version. At 90% recall, the UNet predictions have precisions 12.7% higher for images taken with eVscope 1 telescopes than images from eVscope 2 (96.8% compared to 84.1%). Since the S/N distributions remain comparable for test images taken with eVscope 1 and 2 telescope, these variations could be due to differences in the detector properties, making the Bayer matrix more prominent in eVscope 2 images.

We measure little or no change in performance as a function of streak length, number of stars, and value of R0. Table 1 shows that the three metrics are relatively stable for these various bins. For instance, the precision at 90% recall only changes by 6% as a function of the number of stars enclosed in the fields-of-view. The ROC curves divided into bins of streak length do not show a clear evolution, suggesting a comparable level of contamination for simulated streaks of variable length. The precision-recall curves, however shows stronger variations, with a precision at 90% recall 20% lower for streaks with length >230 pixels. This illustrates the complementarity between diagnostics inferred from the ROC and precision-recall curves.

5. CONCLUSION

We have developed a deep learning pipeline to detect satellite streaks in short-exposure images taken by amateur astronomers as part of the Unistellar network. Our two-step approach optimized for analysing large data sets starts with a CNN classifier separating images with and without streak, followed by the segmentation of images with streak candidates based on a UNet. Both networks are trained on images simulated with our Python package `orbitAI`, which injects synthetic streaks on observed sky backgrounds. Our CNN classifier shows an excellent recall of 90.2% at 1% FPR on a set of 387 positive and 23283 negative real images from eVscope 1 and 2 telescopes. Performance become nearly-perfect when focusing on streaks enclosed within the fields-of-view, which are most useful for orbital parameter measurements. The segmentation step reaches a recall of 79.6% pixels within satellite masks, at a false positive rate of 0.001%. For 90% recall, we obtain a precision of 96.3% on the predicted satellite masks. This largely improves the performance of non-ML algorithms and paves the way to measuring accurate satellite positions over broader magnitudes ranges and down to lower S/N.

In the future, we will further improve the performance in the low S/N regime, in order to meet our goals on the characterization of LEO satellites and smaller space debris. We will adapt the CNN to perform multiclass

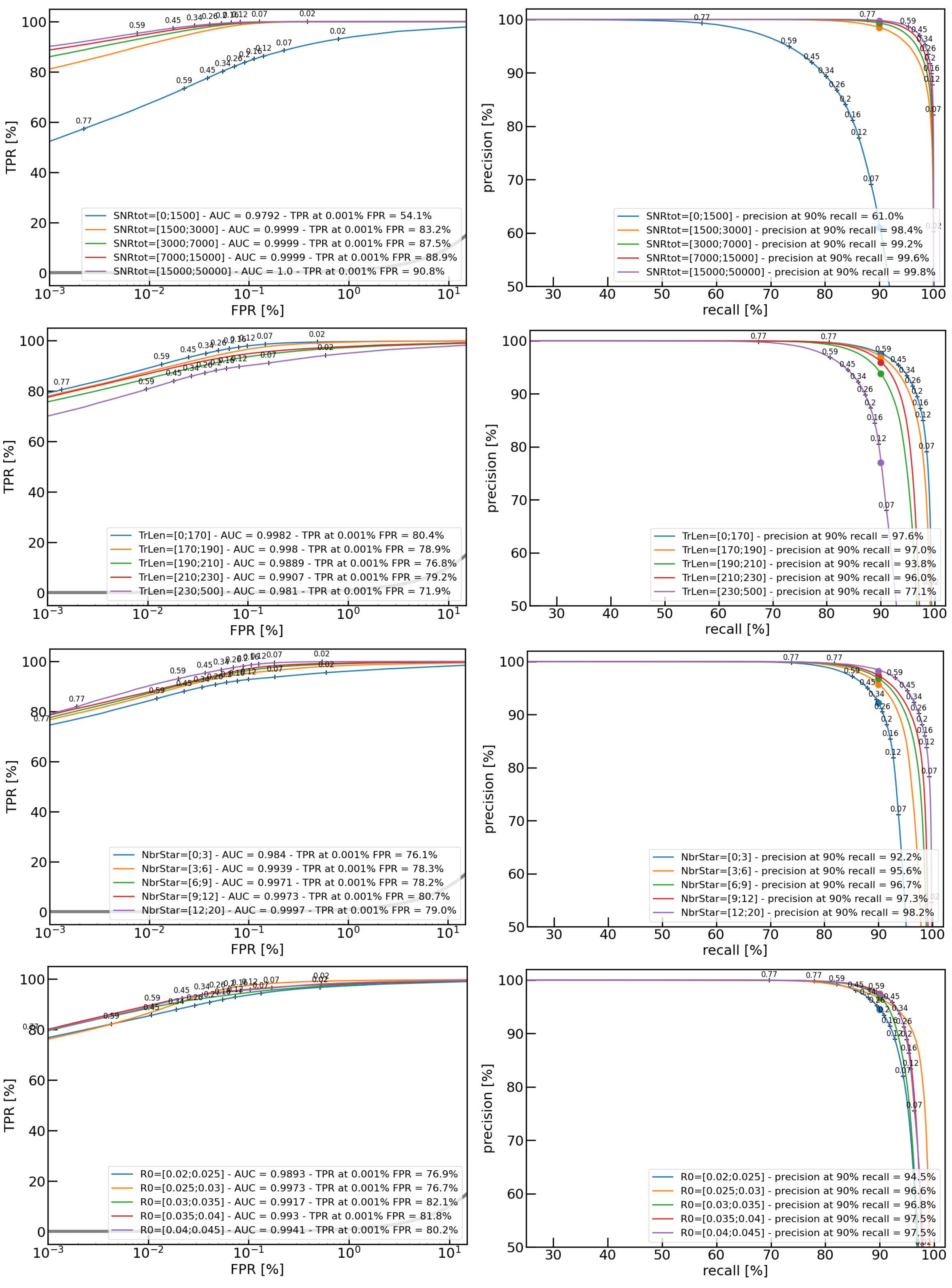

Figure 15. Same as previous figure, after dividing the test set into bins of $\mathrm{SNR_{tot}}$, streak (or trail) length, absolute number of stars, and value of R0. Left: Receiver Operating Characteristic curves. Right: Precision-recall curves.

Table 1. Evaluation of the UNet performance for segmentation of satellite streaks in the test set.

Test set	Area under ROC	TPR at 0.001% FPR	precision at 90% recall	number of test images
all	0.9923	79.6%	96.3%	2174
eVscope 1	0.9929	80.5%	96.8%	1854
eVscope 2	0.9825	74.6%	84.1%	320
$\mathrm{SNR_{avg}} \in$ [0;0.5]	0.9657	41.0%	29.8%	337
$\mathrm{SNR_{avg}} \in$ [0.5;1]	0.9998	73.0%	96.5%	347
$\mathrm{SNR_{avg}} \in$ [1;3]	1.0	85.2%	99.0%	551
$\mathrm{SNR_{avg}} \in$ [3;10]	1.0	88.9%	99.6%	559
$\mathrm{SNR_{avg}} \in$ [10;30]	1.0	91.3%	99.8%	380
$\mathrm{SNR_{tot}} \in$ [0;1500]	0.9792	54.1%	61.0%	594
$\mathrm{SNR_{tot}} \in$ [1500;3000]	1.0	83.2%	98.4%	358
$\mathrm{SNR_{tot}} \in$ [3000;7000]	1.0	87.5%	99.2%	398
$\mathrm{SNR_{tot}} \in$ [7000;15000]	1.0	88.9%	99.6%	359
$\mathrm{SNR_{tot}} \in$ [15000;50000]	1.0	90.8%	99.8%	462
Streak length $\in$ [0;170]	0.9982	80.4%	97.6%	737
Streak length $\in$ [170;190]	0.9980	78.9%	97.0%	495
Streak length $\in$ [190;210]	0.9889	76.8%	93.8%	459
Streak length $\in$ [210;230]	0.9907	79.2%	96.0%	363
Streak length $\in$ [230;500]	0.9810	71.9%	77.1%	166
Number of stars $\in$ [0;3]	0.9840	76.1%	92.2%	480
Number of stars $\in$ [3;6]	0.9939	78.3%	95.6%	920
Number of stars $\in$ [6;9]	0.9971	78.2%	96.7%	980
Number of stars $\in$ [9;12]	0.9973	80.7%	97.3%	160
Number of stars $\in$ [12;20]	0.9997	79.0%	98.2%	80
R0 $\in$ [0.02;0.025]	0.9893	76.9%	94.5%	482
R0 $\in$ [0.025;0.03]	0.9973	76.7%	96.6%	531
R0 $\in$ [0.03;0.035]	0.9917	82.1%	96.8%	477
R0 $\in$ [0.035;0.04]	0.9930	81.8%	97.5%	523
R0 $\in$ [0.04;0.045]	0.9941	80.2%	97.5%	207

classification and separate exposures with faint satellite streaks from the brighter, obvious streaks, in order to process the low S/N exposures separately. We will then train a specific UNet optimized for these faint streaks and we will carefully monitor its performance in the low S/N regime, and over broad ranges of data qualities and observing conditions.

ACKNOWLEDGMENTS

This work benefited from the support of the BPI with France2030 SSA-DOSSA project, the French National Research Agency (ANR) with APPLY (ANR-19-CE31-0011) and LabEx FOCUS (ANR-11-LABX-0013), the Programme Investissement Avenir F-CELT (ANR-21-ESRE-0008), the Action Spécifique Haute Résolution Angulaire (ASHRA) of CNRS/INSU co-funded by CNES, the ECOS-CONYCIT France-Chile cooperation (C20E02), the ORP-H2020 Framework Programme of the European Commission's (Grant number 101004719), STIC AmSud (21-STIC-09), the french government under the France 2030 investment plan, the Initiative d'Excellence d'Aix-Marseille Université A*MIDEX, program number AMX-22-RE-AB-151. This research has made use of computing facilities operated by CeSAM data center at LAM, Marseille, France.

REFERENCES

[1] Zhang, K. and Bloom, J. S., "deepCR: Cosmic Ray Rejection with Deep Learning," *Astrophysical Journal* **889**, 24 (Jan. 2020).

[2] Xu, C., McCully, C., Dong, B., Howell, D. A., and Sen, P., "Cosmic-CoNN: A Cosmic-Ray Detection Deep-learning Framework, Data Set, and Toolkit," *Astrophysical Journal* **942**, 73 (Jan. 2023).

[3] Kruk, S., García Martín, P., Popescu, M., Merín, B., Mahlke, M., Carry, B., Thomson, R., Karadağ, S., Durán, J., Racero, E., Giordano, F., Baines, D., de Marchi, G., and Laureijs, R., "Hubble Asteroid Hunter. I. Identifying asteroid trails in Hubble Space Telescope images," *Astronomy and Astrophysics* **661**, A85 (May 2022).

[4] Zavagno, A., Dupé, F. X., Bensaid, S., Schisano, E., Li Causi, G., Gray, M., Molinari, S., Elia, D., Lambert, J. C., Brescia, M., Arzoumanian, D., Russeil, D., Riccio, G., and Cavuoti, S., "Supervised machine learning on Galactic filaments. Revealing the filamentary structure of the Galactic interstellar medium," *Astronomy and Astrophysics* **669**, A120 (Jan. 2023).

[5] Bertin, E. and Arnouts, S., "SExtractor: Software for source extraction.," *Astronomy and Astrophysics, Supplement* **117**, 393–404 (June 1996).

[6] Paillassa, M., Bertin, E., and Bouy, H., "MAXIMASK and MAXITRACK: Two new tools for identifying contaminants in astronomical images using convolutional neural networks," *Astronomy and Astrophysics* **634**, A48 (Feb. 2020).

[7] Recio-Blanco, A., de Laverny, P., Palicio, P. A., Kordopatis, G., Álvarez, M. A., Schultheis, M., Contursi, G., Zhao, H., Torralba Elipe, G., Ordenovic, C., Manteiga, M., Dafonte, C., Oreshina-Slezak, I., Bijaoui, A., Frémat, Y., Seabroke, G., Pailler, F., Spitoni, E., Poggio, E., Creevey, O. L., Abreu Aramburu, A., Accart, S., Andrae, R., Bailer-Jones, C. A. L., Bellas-Velidis, I., Brouillet, N., Brugaletta, E., Burlacu, A., Carballo, R., Casamiquela, L., Chiavassa, A., Cooper, W. J., Dapergolas, A., Delchambre, L., Dharmawardena, T. E., Drimmel, R., Edvardsson, B., Fouesneau, M., Garabato, D., García-Lario, P., García-Torres, M., Gavel, A., Gomez, A., González-Santamaría, I., Hatzidimitriou, D., Heiter, U., Jean-Antoine Piccolo, A., Kontizas, M., Korn, A. J., Lanzafame, A. C., Lebreton, Y., Le Fustec, Y., Licata, E. L., Lindstrøm, H. E. P., Livanou, E., Lobel, A., Lorca, A., Magdaleno Romeo, A., Marocco, F., Marshall, D. J., Mary, N., Nicolas, C., Pallas-Quintela, L., Panem, C., Pichon, B., Riclet, F., Robin, C., Rybizki, J., Santoveña, R., Silvelo, A., Smart, R. L., Sarro, L. M., Sordo, R., Soubiran, C., Süveges, M., Ulla, A., Vallenari, A., Zorec, J., Utrilla, E., and Bakker, J., "Gaia Data Release 3. Analysis of RVS spectra using the General Stellar Parametriser from spectroscopy," *Astronomy and Astrophysics* **674**, A29 (June 2023).

[8] Konnik, M. and Welsh, J., "High-level numerical simulations of noise in ccd and cmos photosensors: review and tutorial," (12 2014).

[9] Krizhevsky, A., Sutskever, I., and Hinton, G. E., "Imagenet classification with deep convolutional neural networks," in [*Advances in Neural Information Processing Systems*], Pereira, F., Burges, C., Bottou, L., and Weinberger, K., eds., **25**, Curran Associates, Inc. (2012).

[10] Ronneberger, O., Fischer, P., and Brox, T., "U-net: Convolutional networks for biomedical image segmentation," (2015).