Anomaly Detection Techniques in the Gaia Space Mission Data

Journal of Signal Processing Systems manuscript No.
(will be inserted by the editor)

Anomaly Detection Techniques in the Gaia Space Mission Data

Marco Roberti · Alessandro Druetto · Deborah Busonero · Rossella Cancelliere · Davide Cavagnino · Mario Gai

Received: date / Accepted: date

Abstract In this paper we deal with classification of anomalous data detected by the data reduction system of the Gaia space mission, in operation since 2013. Given the size and complexity of intermediate data and plots for diagnostics, beyond practical possibility of full human evaluation, the need for automated signal processing tools is becoming more and more relevant. Our classification task consists in discriminating among "normal" data and data affected by anomalies, which at present are grouped into four different classes. We investigate the use of some clever pre-processing approaches that allow the application of a tailored technique based on the Hough transform, and of some machine learning tools, evidencing that the task can be exactly solved in the former case. In the latter case, random forests and support vector machine provide less than satisfactory performance, while convolutional neural networks achieve very good classification accuracy, up to 91.22%. Statistics show satisfactory results also in terms of precision and recall of each class.

Keywords Anomaly Detection · Deep learning · Big Data · Astrometry

Marco Roberti – m.roberti@unito.it
Alessandro Druetto – alessandro.druetto@unito.it
Rossella Cancelliere – cancelli@di.unito.it
Davide Cavagnino – davide@di.unito.it
Computer Science Department, University of Turin
Via Pessinetto 12, 10149 Torino (TO), Italy

Deborah Busonero – deborah.busonero@inaf.it
Mario Gai – mario.gai@inaf.it
National Institute for Astrophysics, Astrophysical Observatory of Turin
Via Osservatorio 30, 10025 Pino Torinese (TO), Italy

1 Introduction

In this paper we investigate the use of automatic diagnostic techniques in the framework of the Gaia mission [43] of the European Space Agency (ESA), which will provide an all-sky catalogue of position, proper motion and parallax of about 1.7 billion objects among Milky Way stars and bright galaxies.

Gaia, described in Sec. 2, is providing a much clearer view of the dynamics of our Galaxy, and therefore setting a robust foundation to a number of astrophysical issues. It is now at an advanced stage of its observations, and the intermediate products of the data reduction correspond to an overall dataset in the petabyte range.

The current study explores the huge set of yet unexploited Gaia plots by means of different image processing and machine learning tools, in order to assess automatic diagnostic capabilities even in presence of extreme natural phenomena. We focus on the identification of transients and peculiar operating conditions i.e. i) identification of runaway conditions on the Gaia plots (with parameters drifting beyond appropriate limiting values), ii) identification of one or more missing data in the plots, and iii) states of excess noise, leading to error bar increase beyond given thresholds.

Our work deepens and expands what done in [12], where a preliminary study on a simple synthetic dataset is presented. The positive outcomes led us to a wider analysis, which now includes the use of real data and more advanced classification schemes, and the comparison between two different approaches, respectively based on the Hough Transform and on different machine learning techniques. The former is tailored to the shape of the data used for the current exploration, whereas the latter, also very well performing on the problem at

hand, may provide the flexibility, robustness, and tolerance required for forthcoming, more complex diagnostic tasks.

The Hough transform [13,23] is a feature extraction technique used for straight line and circular shape identification in image analysis, image processing, and digital image processing. Because of these properties it also allows to find, by a voting procedure, imperfect instances of objects within a certain class of shapes. For example, it can be used to identify straight lines or more complex ones with the application of specific filters, such as bee trajectories [52], or in its circular formulation to identify round shapes, such as eye iris [32].

Among machine learning techniques, we adopt Convolutional Neural Networks (CNNs), whose popularity in recent years increased dramatically thanks to their ability to exploit large datasets [22] and to show at the same time outstanding performance; Random Forests (RFs) and Support Vector Machines (SVMs) are also explored in order to provide a wider range of perspectives.

CNNs were firstly introduced in 1989 to recognize handwritten ZIP codes [33] but only the advent of modern, much larger datasets makes their training effective: the breakthrough came in 2012, when Krizhevsky et al. [30] achieved the highest classification accuracy in the ILSVRC 2012 competition, using a CNN trained on the images of ImageNet dataset.

Since this revival, CNNs have been successfully applied in a broad range of tasks, ranging from diagnosis and classification ([37,10,14]), object and motion detection, assistive technologies [40], just to name a few.

CNNs have also recently found increasing usage in astrophysical applications, for better exploitation and inter-calibration of the large datasets produced by modern sky surveys. Some relevant examples include the development of CNNs for the derivation of fundamental stellar parameters (i.e. effective temperature, surface gravity and metallicity) [29], the studies of galaxy morphology [51], the high-resolution spectroscopic analysis using APO Galactic Evolution Experiment data [35], and the determination of positions and sizes of craters from Lunar digital elevation maps [48].

Also, in [54] ExoGAN (Exoplanet Generative Adversarial Network) is presented, a new deep-learning algorithm able to recognize molecular features, atmospheric trace-gas abundances, and planetary parameters using unsupervised learning.

In section 2 we recall the main features of the Gaia mission and of the data used in this work; section 3 discuss how our classification task benefits from Hough transform, while section 4 presents the results obtained in applying machine learning tools. Finally, in section 5 we draw our conclusions, also outlining options for future work.

2 The astronomical problem

Gaia is the ESA space mission aimed at Global Astrometry at few μas, producing an all-sky catalogue of position, proper motion and parallax, complete to the limiting magnitude $V = 20\ mag$. The final full accuracy catalogue is foreseen for 2024; at the time of writing, we are getting close to the early third Gaia catalogue release, planned on the fourth quarter of 2020. The Gaia Data Release 2 [18] is publicly available, and it has been used for a number of astrophysical applications[17,20].

Gaia operates in scanning mode through two telescopes separated by a base angle of 106.5°, feeding a common focal plane (FP) of one hundred CCDs, with continuous full-sky observation. The FP is divided in three regions: the Sky Mapper-Astrometric Field (SM-AF), the Blue and Red Photometers (BP, RP) and the Radial Velocity Spectrometer (RVS), respectively devoted to astrometric [36,19], photometric [15], and spectroscopic measurements. The data used in this paper concern the first region, with 7×9 CCDs. Actually, one CCD in the array is devoted to service functions (metrology), so that only 62 CCDs are considered in our analysis.

The Gaia data processing is managed by the Data Processing and Analysis Consortium (DPAC), in charge of the scientific part of the Gaia ground segment. The DPAC is organized in Coordination Units (CUs), each in charge of specific parts of the whole reduction chain. CU3, in particular, takes care of the so-called core processing, i.e. it will take the observations of a suitable subset of well-behaved stars (e.g. single stars, photometrically and astrometrically stable, not too faint, etc.), and it will reconstruct their five astrometric parameters (parallax and two components of both angular position and proper motion), in addition to instrumental parameters and the satellite attitude.

Several statistics and plots are generated to track the Gaia instrument response: every day, about one hundred thousand plots are produced, in order to support the investigation on a number of operational aspects by choice of the relevant ones. For our investigation we used the output of the CU3 Astrometric Instrument Model (AIM) pipeline, running at the Data Processing Center in Turin (Italy).

Moderate variation of instrument parameters may be taken into account by the data reduction system; in case of larger variations, human intervention on the

satellite may be required. This corresponds to insurgence of critical conditions of excessive payload variation (e.g. optical transmission degradation by contamination), or external disturbances (e.g. solar flares). Gaia, during most of its lifetime, fortunately operated quite well, so that most of the plots were not studied in detail. However, analysis of specific samples evidenced that a number of minor identifiable events were ignored, that surely deserve to be included in the data reduction to improve on final mission precision. To ensure that an alert is issued, triggering the adequate corrective action, it is necessary to implement an automatic detection of such condition, so far managed mostly by human supervision. The use of the toolsets described in this paper will also provide the practical means of implementing a fully detailed review of the whole mission observations, which could not be done by direct human inspection.

2.1 Data description

In the Gaia data processing, several intermediate data are stored only in the graphical form of plots, rather than numerical values in a database; this eased the human evaluation of peculiar situations during commissioning, because each plot evidences the overall trend over a day for the selected parameter. The first tests have been targeted on the family of plots showing the daily statistics of matches between readout windows and actual positions of each observed star. Such plots are generated on every day of operation for each of the 7×9 CCDs, and for each telescope. In normal operating conditions, the distribution is expected to be random, with zero or fixed mean, and spread of order of half a pixel. As the electro-optical instrument response is variable over the field of view (FoV), such plots are similar, but with different mean value.

A data segment corresponding to 30 min of observation provides an average value and an error bar, due not only to photon statistics fluctuations, but also to "cosmic scatter", i.e. different characteristics of the many thousand detected celestial sources. Each daily plot includes therefore 48 points with errors.

A sample plot corresponding to the spread of estimated along scan star positions for the CCD corresponding to Strip 1, Row 1, is shown in Fig. 1a. The plots have format 1500×927 pixels.

The abscissa is the mission running time, in satellite revolutions; the vertical axis is in micro-meters (μm), referred to the center of the readout window. One detector pixel is $10\mu m$; a slip by more than one pixel in either direction of the average photo-center requires readjustment of the on-board parameters used to compute the read-out window placement. A plot instance evidencing one runaway condition, with values located below the lower threshold, is shown in Fig. 2a, where a solar flare also causes larger error bars and increased noise; in Fig. 3a we observe some missing data.

3 Image diagnostics and Hough transform

As stated earlier, we aim at detecting anomalies in a set of daily plots, all having the same structure: they are composed by 48 vertical segments, one for each 30 minutes time span in a day.

In order to exactly detect the plot segment parameters, i.e. line length, line position and line absence, we decided to process each plot by means of the Hough Transform [13,23]. This feature extraction technique is widely used both in straight line and circular shape identification (e.g. [38,44]).

The Hough Transform algorithm uses a two-dimensional array, called an accumulator, to detect the existence of a line that will be described by $r = x \cos\theta + y \sin\theta$. For each pixel at (x, y) and its neighborhood, if there is enough evidence of a straight line, the algorithm will calculate the parameters (r, θ). Subsequently, it will increment the value of the accumulator bin where the parameters fall into. The most likely lines can then be extracted by finding the bins with the highest values: a threshold is typically applied to find the peaks corresponding to such values.

The result of the Hough transform is a two-dimensional array, similar to the accumulator: one dimension represents the angle θ and the other dimension represents the distance r. Each element of the matrix, indexed by (r, θ), has a value equal to the sum of the points (i.e. pixels) that were successfully recognised on the corresponding line.

A convenient strategy appears to be the separation between actual reconstruction of the plot segments, and anomaly diagnostics on the reconstructed data. Hereafter, we describe the two phases.

Hough Transform reconstruction is typically split in two parts: *line detection* to find line positions in the image and *feature extraction* to find exact matching segments. The main steps and used parameters for the application of the Hough Transform can be briefly summarised as follows:

1. binarization of the original plot, with a luminance threshold value of 0.6;
2. application of the Hough transform to the binarized plot;
3. largest frequency peaks extraction from the transformed plot to find the main lines of interest;

4. line segments extraction from the transformed plot and the peaks, with a minimum segment length of 70 (this is the *feature extraction* part of the analysis).

The fixed constant parameters of the previous procedure where determined by examining the plot images which have a fixed structure, in particular in terms of luminance and distance between vertical lines in the plots. The luminance threshold value (0.6) can be determined by observing the histograms of the images and finding the (dark) grey levels assigned to the vertical line segments and the (light) grey levels of the background. The extraction of the frequency peaks has the purpose to find lines related to a meaningful number of aligned points and which are not too close one to the other. To avoid to consider too short segments their extraction is parameterized with a minimum length (70) as reported in the last step of the procedure. After testing various combinations of such values, and also considering the structure of our images, we found the aforementioned 0.6 for luminance and 70 for segment length, that were the two that perform better.

3.1 Hough Transform for plot analysis

The code for the experiments we describe in this section is implemented in Python 3.7[1], using the Computer Vision library OpenCV 3.4.2 [7].

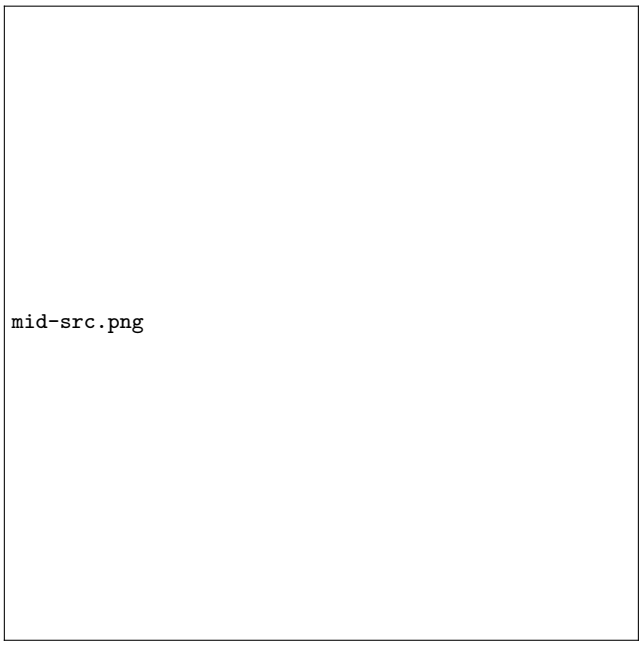
In order to identify present or missing lines, we exploited the Hough Transform results: we implemented a vertical scan (over the lines) with a 5×5 mask to identify presence and exact coordinates of *upper*, *mid* and *lower* points of each segment. They are shown in Figure 1b, respectively *extremes* in blue and *midpoints* in red. The dimension of the mask derives directly from the structure of the analysed plot, since these points are drawn as a group of pixels in the shape of a diamond (5 pixels wide and 5 pixels tall).

In Figure 1a one example of *normal* plot is shown, and in Figure 1b we see the result of the aforementioned line recognition method. We remark that all points are correctly identified, as demonstrated by the positioning of blue and red dots over significant points.

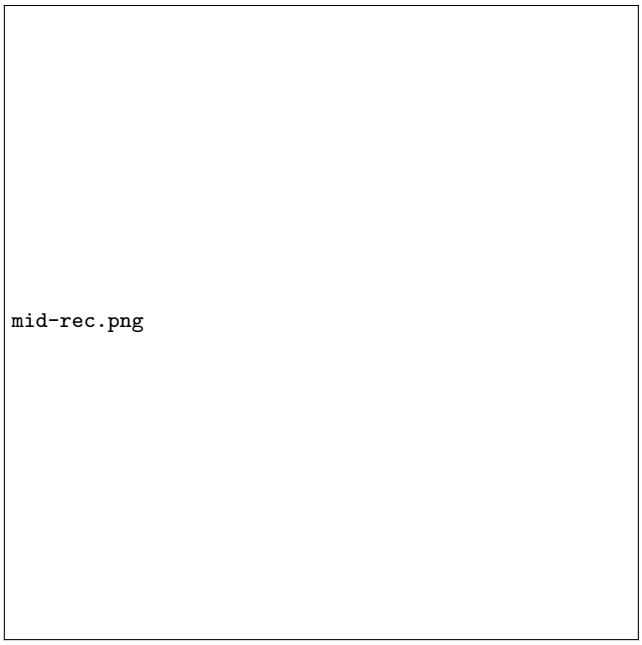
The anomaly diagnostic phase analyses lines and points found by Hough Transform.

In Figure 2 there is an example of a plot taken during an episode of solar storm. In the rightmost part of the source plot, Figure 2a, error bars are larger, showing a huge variance, induced by the aforementioned solar storm. Figure 2b evidences that all relevant points, when present, are successfully recognized.

[1] `https://www.python.org/`



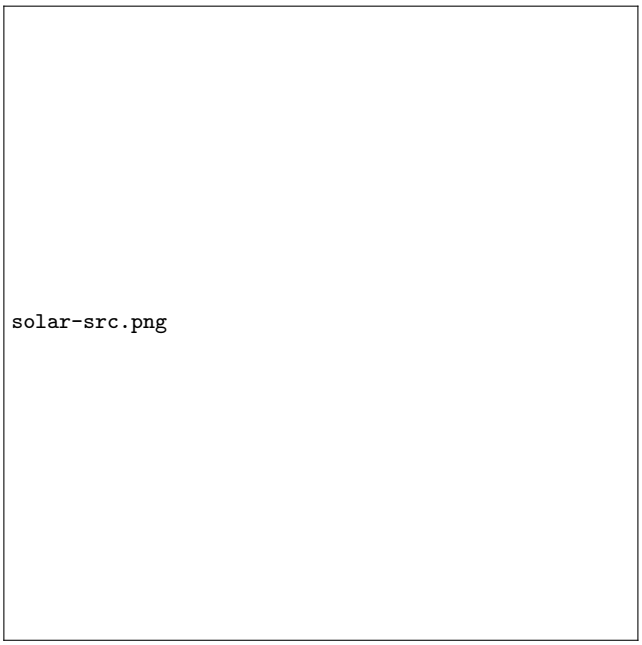
(a) Source



(b) Analysis

Fig. 1: Example of line coordinates recognition for a normal plot.

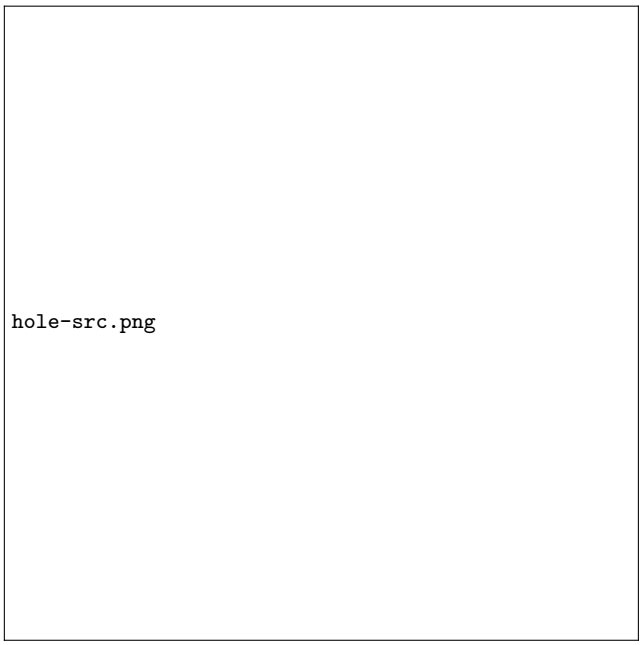
A particular kind of anomaly is the total absence of a line. In Figure 3 some kind of system failures generated a gap in the data. Again, in the plot in Figure 3a, no shift is present nor huge variance, but a line is missing toward the left end. Figure 3b evidences that our method successfully skips the missing line; we also remark that even multiple missing lines (contiguous or not) can be easily identified.



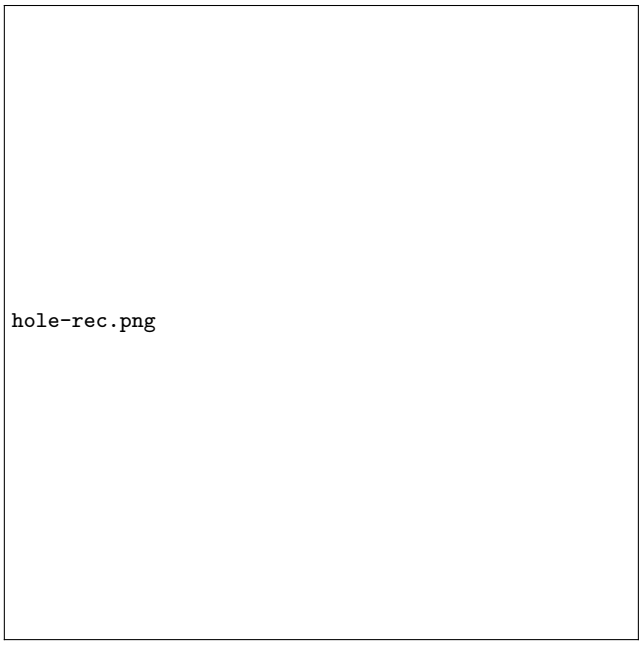
(a) Source

(b) Analysis

Fig. 2: Correct line recognition in a plot showing data obtained during a solar storm



(a) Source



(b) Analysis

Fig. 3: Line recognition over a plot with missing line.

Application of the Hough Transform (and of the aforementioned vertical scan) thus leads to excellent results, identifying every line attribute in all plots and hence providing an exact approach to the anomaly detection problem.

3.2 Result presentation

Through the knowledge of the pixel coordinates of all lines' extremes and midpoints (where present) we can successfully discriminate regular plots from those affected by one or more known anomalies. The procedure works as follows:

– if the file corresponding to any CCD is missing or does not contain any line, the entire plot is classified as absent;
– if one of the 48 lines positions of the plot has neither upper, mid nor lower point, then such line is missing and the plot is classified as problematic;
– if the maximum line length of the plot exceeds the maximum line length of the reference set, within tolerances, it means that the variance is too large for some lines and the plot is classified as problematic;
– if the average midpoint of the plot is displaced by more than ±5 units (within tolerances) from that of the reference plot, then a significant shift is present and the plot is classified as problematic;
– otherwise (no anomaly detected!) the plot is classified as regular.

We implemented the following three different possible output formats from our analysis, which can be further specialised depending on user convenience:

Layer 0 We recall that for each day we have data from two FOVs and 62 CCDs: to have a quick glance of the results, on the terminal is shown a line describing the situation of each FOV.

Layer 1 In a spreadsheet file the user can find two tabs, one for each FOV, graphically representing the status of each CCD. Green means no problems, yellow means that some problems are found in this particular plot, and red means that the entire plot is empty, or that the file containing the plot is missing.

Layer 2 Finally, in a text file there is the complete log of all problems found for every CCD and every FOV, with the complete report of all problems found.

In Figure 4 the Layer 0 of analysis is presented: from the first progress bar, that represents the number of plots read for the first FOV, we deduce that one of the plots is missing (98% instead of 100%). We also see that in both FOVs there are anomalous plots, so that the suggestion is to open the two files containing further analysis.

In Figure 5 the Layer 1 output format is shown: the sheet related to FOV 0, clearly marked in red, evidences that data from S1-R1 CCD are missing.

The most detailed (Layer 2) result analysis is stored in a text file, an example of which is shown in Figure 6; only the first lines are listed, for the sake of clarity. A comment row is produced for every anomalous line detected in the S(n)-R(m) CCD plot. The first row again evidences that S1-R1 plot is missing.

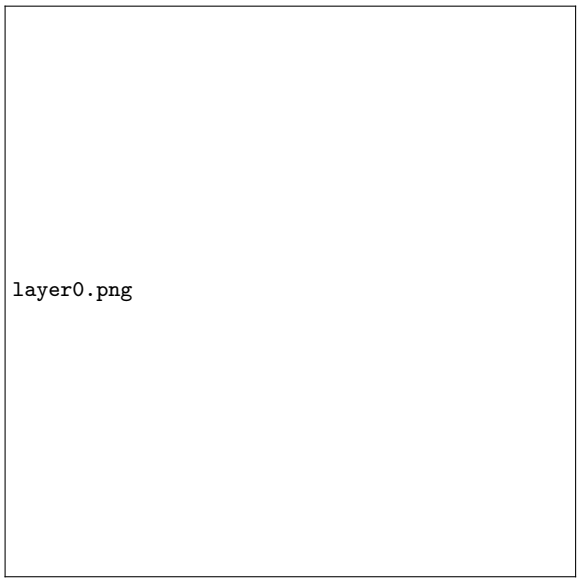


Fig. 4: Layer 0 - quick report.

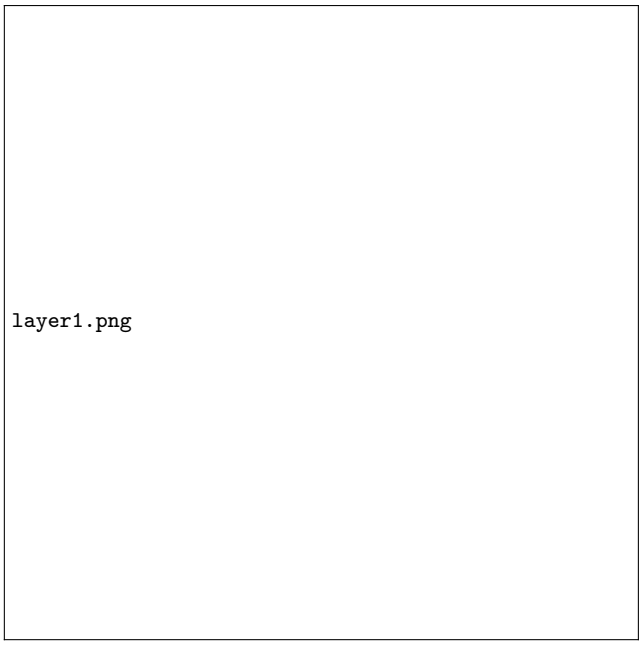


Fig. 5: Layer 1 - the spreadsheet file graphically showing the status of each CCD.

4 Machine Learning approach

As discussed in section 3, the image processing approach leads to exact classification. However, the high quality of this performance is obtained thanks to a tailored technique, which exploits the knowledge of the segment structure of the plots under analysis. In this section we will apply some machine learning based methods, in order to explore also a data-driven approach, not strictly related to a predefined plot structure. This at-

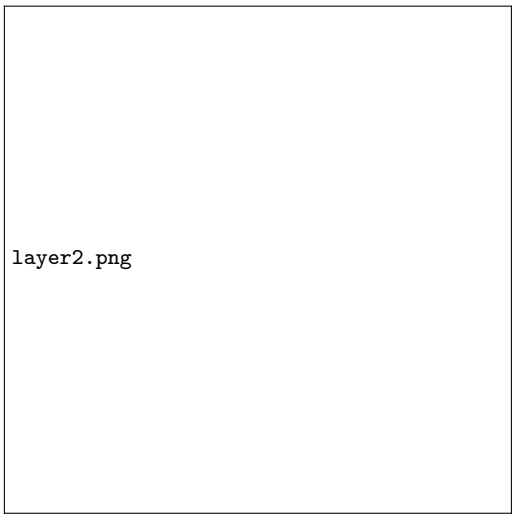


Fig. 6: Layer 2 - text file with the complete analysis.

tempt is particularly interesting because it may provide the flexibility required for forthcoming, more complex diagnostic tasks.

In subsection 4.1, we describe our data pre-processing procedure and justify the need for a data generator. The subsequent subsections describe the models we used: Principal Component analysis (subsection 4.2) jointly exploited with Support Vector Machines (subsection 4.3) or Random Forests (subsection 4.4), and Convolutional Neural Networks (subsection 4.5).

The code is implemented in Python 3.7 taking advantage of Matplotlib 3.1 [25], Scikit-learn 0.21 [41], Hyperopt [5] and Keras 2.2 [11] on Tensorflow 2.0 [1]. All experiments are run on an Intel Core i9-9900KF CPU and, when required, an NVIDIA Titan RTX GPU. All the results described in the following are the outcome of 10 tests: means and standard deviations are presented.

4.1 Building Data

As described, individual plots are not informative *per se* about the presence of anomalies, that can be instead observed when cross-checking with a reference trend. A reference plot is thus chosen as representative among those not featuring anomalies; an example is shown in Figure 1a.

In order to get more intelligible data and reduce the input size, we create the so-called "difference plots" by subtracting, pixel-by-pixel, the current image and its reference one. This results in final monochromatic images whose pixels values are in $\{-1, 0, 1\}$, that are the actual inputs for our models; some examples are shown in Figure 7. We can identify (i) pixels that have the same values in the original plot and in its reference (0, in gray), (ii) pixels filled in black in the original plot, but not in the reference plot (-1, in black), and (iii) pixels filled in black in the reference plot, but not in the original plot (1, in white). This procedure clearly puts in evidence discrepancies between current and reference images, so easing the classification task.

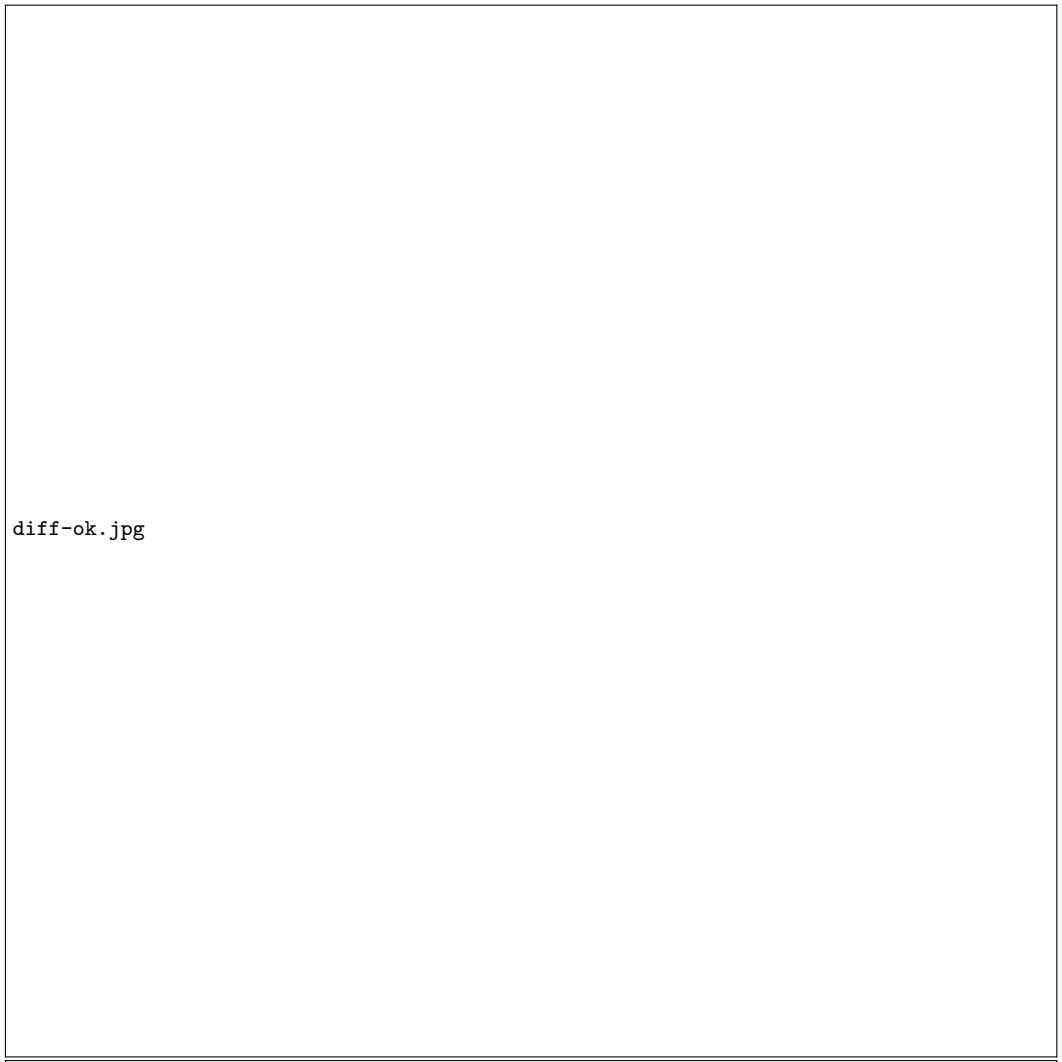
Our data are organized in a pair $(\boldsymbol{X}, \boldsymbol{y})$, where $\boldsymbol{X}$ is the set of monochromatic input images $\boldsymbol{x_i} \in \mathbb{R}^n$ and $\boldsymbol{y}$ is the set of corresponding output classes $y_i \in \mathbb{R}$.

The dataset consists of 1241 labeled plots, barring the 124 reference ones. Such cardinality is not enough for a training set when using modern machine learning techniques such as deep CNNs. We therefore preserve these real plots, considering them as our test set $(\boldsymbol{X_{test}}, \boldsymbol{y_{test}})$, and opt for the development of a generator of synthetic difference plots, whose output is used as training set $(\boldsymbol{X_{train}}, \boldsymbol{y_{train}})$.

The pseudo-code of the generator script is reported in Algorithm 1. In short, it (1) samples the reference plot's values, i.e. its abscissa values and the corresponding average ordinate values and error bars' lengths; (2) creates the current instance by perturbing such values according to a sampled class; (3) generates the two corresponding plots; (4) generates the difference plot. The perturbation is done by means of Algorithm 2, which applies class-dependent and class-independent perturbations to the reference plot's values. All parameters have been chosen in order to obtain a good quality when comparing with the test difference plots. The generator makes use of the perturbation procedure described in Algorithm 2, which simply applies the proper modification to the generated reference data, according to the given class. In order to implement the anomalies already described in sections 2.1, our classes are specified as follows:

1. `ok` means absence of anomalies;
2. `shift` means that the lines are significantly shifted on the vertical axis;
3. `error` means that one or more error bars are at least twice as long as normal;
4. `hole` means that one to three lines are missing;
5. `big_hole` means that more than three lines are missing;

It is worth noting that the generated data has the same distribution independently from the number of instances, as the distribution $\mathcal{D}(\pi)$ is an explicit parameter: in this way, sets of different cardinality can be generated in order to train the explored techniques. We also underline that the generator is able to synthesize, when required, unbalanced data or multi-labeled plots.


diff-ok.jpg


diff-ko.jpg

Algorithm 1 The generator's pseudo-code. $\mathcal{U}$ is the uniform distribution; $\mathcal{N}$ is the normal distribution.

Require: $|\boldsymbol{X}| > 0$: size of the dataset that has to be generated
Require: $\mathcal{D}(\boldsymbol{\pi})$: classes probability distribution
  for $i = 1$ **to** $|\boldsymbol{X}|$ **do**
    // Sample the reference plot's features
    $t_0 \leftarrow 0.23 + \text{sample}[\mathcal{U}\{4000, 9000\}]$ // The time of the first data segment
    $A \sim \mathcal{N}(0, 1.5)$ // The amplitude of the average values' sinusoid trend
    $\phi \sim \mathcal{U}(0, 12)$ // The sinusoid's horizontal shift
    $T \sim \mathcal{N}(0, 3)$ // The sinusoid's vertical shift
    $z \sim \mathcal{N}(0, 0.5)$ // Additive white Gaussian noise

    // Create reference plot
    $times \leftarrow \{t_0 + \frac{t}{12} \mid 0 \leq t < 48\}$ // Abscissa values
    $values \leftarrow \{T + A \cdot \cos(\phi + \frac{\pi}{6} \cdot t) + z \mid 0 \leq t < 48\}$ // Ordinate values
    $errors \leftarrow \{e_t \sim \mathcal{N}(4.5, 0.5) \mid 0 \leq t < 48\}$ // Error bars' lengths
    $ref \leftarrow \text{plot}(times, values, errors)$ // Reference plot

    // Create instance plot by "perturbating" the reference
    $\boldsymbol{y}_i \sim \mathcal{D}(\boldsymbol{\pi})$ // Class of the instance
    $times, values, errors \leftarrow \text{perturbate}(\boldsymbol{y}_i, times, values, errors)$ // Algorithm 2
    $img \leftarrow \text{plot}(times, values, errors)$ // Instance's plot

    // Create difference plot
    $diff \leftarrow \text{normalize}(img - ref)$
    $\boldsymbol{X}_i \leftarrow diff$
  end for
  return $\boldsymbol{X}, \boldsymbol{y}$

4.2 Principal Component Analysis

Principal Component Analysis (PCA) is one of the most used and well-known dimensionality reduction methods; it relies on the creation of new features, called Principal Components, each one corresponding to a linear combination of the original ones. The data's first principal component is the maximum variance direction; the second one is the direction of maximum variance linearly independent from the first component, and so on.

PCA is particularly helpful when Support Vector Machines (SVM) and Random Forests (RF) are used, as these methods are subject to the curse of dimensionality; this is an actual issue in our case because of the dimension n of each input (1500×927 pixels). As we do not know in advance how many principal components are needed in either case, this value is an hyperparameter to be optimized.

In contrast, CNNs do not need dimensionality reduction as they include, by design, a set of feature extractors.

4.3 Support Vector Machine

A Support Vector Machine [6] is a widely used [34, 50] linear classifier that "constructs the unique decision boundary that maximises the distance to the nearest training examples (the support vectors)" [16]. The decision boundary is by definition

$$\hat{s}(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} - t, \quad \boldsymbol{w} \in \mathbb{R}^n, \tag{1}$$

where $\boldsymbol{x}$ is the independent variable and $\boldsymbol{w}$ and t are the model's learned parameters. The distance to the nearest training examples is referred to as "margin".

The training phase consists in solving the following large optimisation problem (soft-margin formulation), that has quadratic complexity:

$$\begin{aligned} \boldsymbol{w^*}, t^*, \xi_i^* = \underset{\boldsymbol{w}, t, \xi_i}{\arg\min} \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad y_i(\boldsymbol{w} \cdot \boldsymbol{x_i} - t) \geq 1 - \xi_i, \\ \xi_i \geq 0, \quad 1 \leq i \leq n \end{aligned} \tag{2}$$

in which every example $\boldsymbol{x_i}$ is associated with one slack variable ξ_i that allows it to be within the margin, or even to be misclassified. The real-valued parameter C weighs a regularization term, trading off slack variables minimisation and margin maximisation [16].

In order to allow the SVM architecture to deal with complex non linearly separable data, kernel methods [2, 6] are used. The kernel functions we experimented with are the following:

$$\begin{aligned} \text{K}_{polynomial}(\boldsymbol{x_i}, \boldsymbol{x_j}) = (\gamma \langle \boldsymbol{x_i}, \boldsymbol{x_j} \rangle + r)^d \\ \text{K}_{Gaussian}(\boldsymbol{x_i}, \boldsymbol{x_j}) = e^{-\gamma \|\boldsymbol{x_i} - \boldsymbol{x_j}\|^2} \end{aligned} \tag{3}$$

Algorithm 2 The plot perturbation pseudo-code. $\mathcal{U}$ is the uniform distribution; $\mathcal{N}$ is the normal distribution; $\mathcal{T}$ is the triangular distribution.

Require: $cls \in \{\texttt{ok}, \texttt{shift}, \texttt{error}, \texttt{hole}, \texttt{big\_hole}\}$: the class the returned plot has to belong to
Require: $times, values, errors \in \mathbb{R}^{48}$: the abscissa and ordinate values and error sizes of the reference plot
// Time shift (class-independent)
$\Delta_t \sim \mathcal{U}\{-1000, 1000\}$ //
$times \leftarrow \{t + \Delta_t \mid t \in times\}$

// Values shift (class-independent)
$i \sim \mathcal{U}\{0, 47\}$
for $j = 0$ **to** 47 **do**
$values_i \leftarrow \{values_{(i+j) \bmod 48}\}$
end for

// Class-dependent perturbations
if $cls = \texttt{shift}$ **then**
$T \leftarrow \text{sample}\{+1, -1\} \cdot \text{sample}[\mathcal{N}(4, 10)]$ // Shift's direction and amplitude
$values \leftarrow \{v + T \mid v \in values\}$
else if $cls = \texttt{error}$ **then**
$i_{start} \sim \mathcal{U}\{0, 46\}$ // First index of longer error bars
$i_{end} \sim \mathcal{U}\{1 + i_{start}, 47\}$ // Last index of longer error bars
$factor \sim \mathcal{U}(2, 3.5)$ // Amplitude of the bars' stretch
for $i = i_{start}$ to i_{end} **do**
$errors_i \leftarrow factor \cdot errors_i$
end for
else if $cls = \texttt{hole}$ **then**
$i_{start} \sim \mathcal{U}\{1, 43\}$ // First missing bar's index
$i_{end} \leftarrow i_{start} + \text{sample}[\mathcal{U}\{1, 3\}]$ // Last missing bar's index
delete indices $i \in [i_{start}, i_{end}]$ from *times*, *values* and *errors*
else if $cls = \texttt{big\_hole}$ **then**
$i_{start} \sim \mathcal{U}\{1, 39\}$ // First missing bar's index
$i_{end} \leftarrow i_{start} + \text{sample}[\mathcal{U}\{4, 7\}]$ // Last missing bar's index
delete indices $i \in [i_{start}, i_{end}]$ from *times*, *values* and *errors*
end if

// Avoiding same noise between reference and instance (class-independent)
$values \leftarrow \{v + \text{sample}[\mathcal{N}(0, 0.25)] \mid v \in values\}$
$errors \leftarrow \{e \cdot \text{sample}[\mathcal{T}(0.7, 1.5, 1)] \mid e \in errors\}$

return *times*, *values*, *errors*

The first column of table 1 shows the hyperparameters to be optimized in order to obtain a SVM architecture which solves at best the anomaly detection problem.

As discussed in [4] random search performs better than grid search, as it samples each hyperparameter value from its given, independent distribution; we therefore performed 1000 random sampling from the distributions shown in the aforementioned table, using 500 synthetic difference plot for training. The best hyperparameter's values, shown in the third column, are provided by a 3-fold cross-validation procedure.

This approach provides a poor test classification accuracy: the mean value over 10 different runs is 69.78% (81.35% on the validation set); it is recorded in Table 3 for the sake of comparison. We believe that the absence of variance of this model is due to its inherent design: as a linear separator, it has very few parameters and, consequently, low variance and high bias [16]. PCA and the soft-margin formulation further sharpen this trait.

4.4 Random Forest

A Random Forest [8] is an ensemble estimator that trains N decision trees on various sub-samples of the original dataset. Every sub-sample has the same size as X_{train}, but the sampling is done with replacement, thus preserving about 63.21% of the data. The Random Forest's prediction is the average of every decision tree's one, which improves the output accuracy while keeping over-fitting at bay.

The number of decision trees used, along with their unpredictable branching likelihood, results to a potentially very high memory usage: for this reason, we limited the training dataset to 200 synthetic difference plots (≈ 40 plots per class).

Table 1: Hyperparameters, optimizing sampling distribution and best values for SVM and RF architectures.

Model	Hyperparameter	Sampling distribution	Best value
SVM	PCA's retained variance (%)	$\mathcal{U}(0.20, 0.75)$	0.2487 (4 PCs)
	Kernel type	{polynomial, Gaussian}	polynomial
	γ	$\log\mathcal{U}(10^{-3}, 10)$	$5.1711 \cdot 10^{-2}$
	r (for polynomial kernel)	$\mathcal{U}(0, 2)$	1.6530
	d (for polynomial kernel)	$\{2, 3\}$	2
	C	$\log\mathcal{U}(10^{-2}, 10^{+2})$	7.5561
RF	PCA's retained variance (%)	$\mathcal{U}(0.25, 0.75)$	0.4544 (14 PCs)
	N	$\mathcal{U}\{20, 200\}$	171

Table 1 shows the hyperparameters to be optimized in order to obtain the best possible Random Forest for the classification task. Random search [4] is used, sampling each hyperparameter value from the distributions shown in the table, and as for SVMs, we performed 1000 random hyperparameters sampling. The third column shows the best values for hyperparameters, according to 3-fold cross-validation accuracy.

The obtained mean accuracy over 10 different runs on the test set is $69.69 \pm 0.12\%$ ($75.14 \pm 0.46\%$ on the validation set). As for SVMs, this result clearly shows that Random Forests is not a very successful approach, considering also the high memory requirements of this model.

4.5 Convolutional Neural Network

A Convolutional Neural Network (CNN) [33] is "a specialized kind of neural network for processing data that has a known grid-like topology" [22]. Images are a typical case. In general, we can refer to CNNs as any "neural network that uses convolution in place of general matrix multiplication in at least one of its layers" [22].

Architectural details. In order to deal with the anomaly detection problem, we focus on a neural network composed by N_{Conv} convolutional blocks, followed by N_{FC} fully connected blocks.

Every convolutional block consists of a convolution followed by a max-pooling operation [53], a ReLU activation function [27,39,21] and a batch normalization [26] layer. Convolutional blocks take as input three-dimensional tensors: the first one directly deals with image tensors, whose dimensions are image's width and height and number of color channels. Each input image is scaled according to a factor, treated as a hyperparameter.

Every fully connected block is composed by a dropout layer [49], an affine transformation, and a ReLU nonlinearity, except for the output neurons whose activation is, conforming to the task, a softmax function [9]. Fully connected blocks take vectors as input: the last convolutional block's output is therefore flattened so that its width, height and feature number are merged. Similarly, the CNN's output is a C-dimensional vector whose values, each one representing the input's probability of belonging to the corresponding class, sum to 1. The model is shown in Figure 8.

CNNforGAIA.png

Fig. 8: A typical Convolutional Neural Network architecture

We addressed the risk of over-fitting, potentially caused by the large number of parameters, in a threefold way: (1) from the data side, we used a large training set, made by 20,000 samples (10% of which is kept as the validation set); (2) from the architectural side, we regularize the convolutional blocks via batch normalization, and fully connected ones via dropout (see above). Besides, convolution and pooling layers inherently promote generalization thanks to respectively parameter sharing and dimensionality reduction; (3) from the optimization side, we applied weight decay, the more widely used form of L2 regularization [31,24].

Because of the very high number of hyperparameters to be optimized when using CNNs, a plain random search is very likely to fail, as it only fits well on architectures having a low effective dimensionality [47].

We therefore used Bayesian optimization: such approach "uses a Bayesian regression model to estimate both the expected value of the validation set error for each hyperparameter and the uncertainty around this expectation" [22]. In particular, we took advantage of the Tree-structured Parzen Estimator (TPE) algorithm, which is able to jointly optimize the network architecture as well as the required hyperparameters [3]. The input distributions and the corresponding outputs are listed in Table 2.

Preliminary experiments suggested the use of the Adam optimization algorithm [28] over Stochastic Gradient Descent [45,46] with momentum [42], aiming at minimizing the cross-entropy between the network's output and the target.

Result discussion. Table 3 shows the accuracies obtained by the above discussed machine learning approaches; deep learning turns out to be the most suitable method for anomaly detection, reaching a mean test accuracy over 10 different runs of $87.02 \pm 3.64\%$ ($94.09 \pm 1.80\%$ on validation set). This is a remarkable performance since test set includes real data describing extreme and rare natural phenomena, such as solar storms, the correct classification of which constitutes a true challenge.

Figure 9 shows the test set confusion matrix detailing classification results of our best performing model, that reaches an accuracy of 91.22%. We can notice that the main diagonal contains the majority of samples, as usual when a task is correctly solved. The percentages on light squares refer to the total number of instances in the dataset.

Performance concerning the `ok` class is crucial, as an important feature required by domain experts is the ability to distinguish plots containing an anomaly whatsoever, which will be further analyzed by a human expert, from regular ones, which will be dropped. The precision ($\frac{TP}{TP+FP}$) value of 91.51% on the `ok` class obtained in this case is therefore highly significant, with only a few anomalous plots (78) misclassified as OK.

Confusion between `error` and `ok` classes is due, in our opinion, to the continuous variation range of the actual size of the error bars, which makes the reference hard threshold (a factor 2) set by human experts difficult to fit.

Besides, 48 plots with missing data are misclassified as `ok`: this can be understood since the great majority of `hole`-labeled plots in the test set only miss one point, which might not be "perceived" easily by the network.

Finally, we highlight that the `shift` class is always successfully recognized.

4.5.1 Multi-label classification

Another approach to anomaly detection consists in the multi-label classification of the difference plots. In this case, the model can label every input either as `ok` or with one or more anomalies. On the one hand, this gives the possibility to correctly recognize plots with a more complex structure, i.e. presenting more than one anomaly. In this case the model is not constrained by the assumption, potentially wrong, of mutual exclusivity of anomalies; thus it is more general by design. On the other hand, the increased complexity of the task may result into an accuracy loss, that has to be taken into account for real-life applications.

We used Hyperopt [5] to solve the optimization problem that imposes the probability of every label π_l to appear in a fair number of instances:

$$\pi^* = \arg\min_{\pi} \sum_{l} (|\{ \mathbf{X}_i \mid l \in y_i \}| - |\{ \mathbf{X}_i \mid l \notin y_i \}|)^2 \quad (4)$$

Of course, the label `ok` excludes all the other ones, and labels `hole` and `big_hole` are mutually exclusive: these properties are easily learned by our neural model.

From the architectural point of view, the only difference consists in the activation of the output layer; now the sigmoid function is used, in order to disentangle every label's output, so that more than one of them can be close to 1 at the same time.

Preliminary experiments showed that weighting the loss in order to give more importance to the `ok` class leads to better performance.

As in the single-label case, the number of required hyperparameters is high, so that we used Bayesian optimization to find a good architectural configuration. In particular, we adopted the Tree-structured Parzen Estimator (TPE) algorithm [3], deriving the values shown in Table 2.

In order to compare the multi-label model with the single-label one, we introduce the "collective accuracy" and the "individual accuracy" metrics. According to the former, a given image is correctly classified if the model assigns the correct values to all labels, while the latter operates label-by-label. Therefore an image correctly classified on four labels out of five is still considered a failure in terms of collective accuracy, in spite of achieving 80% individual accuracy.
Despite the harder task, the collective accuracy reached by the model is $67.42 \pm 6.50\%$ on the test set ($85.79 \pm 0.93\%$ on the validation set).

Table 2: CNN hyperparameters, optimizing sampling distribution and best values. Indented hyperparameters are lists of values, whose lengths depends on the outer ones (i.e. N_{Conv} or N_{FC}).

Hyperparameter	**Sampling distribution**	**Best value**	
		Single-label	Multi-label
Scaling factor	$\mathcal{U}(0.1, 1)$	98.05%	93.50%
N_{Conv}	$\{2, 3, 4\}$	4	4
Kernel sizes	$\mathcal{U}\{2, 5\}$	4,4,2,3	4,4,2,3
N. of filters	$\log\mathcal{U}\{2, 32\}$	8,11,12,3	11,11,14,11
Max-pooling sizes	$\mathcal{U}\{2, 5\}$	3,4,4,4	4,4,3,4
N_{FC}	$\{1, 2, 3, 4\}$	2	1
N. of neurons	$\log\mathcal{U}\{4, 1024\}$	263,197	161
Dropout rate	$\mathcal{U}(0, 0.5)$	31.97%	29.26%
Learning rate	$\log\mathcal{U}(5 \cdot 10^{-4}, 10^{-1})$	$5.12 \cdot 10^{-3}$	$6.68 \cdot 10^{-4}$
L2 regularization coefficient	$\log\mathcal{U}(10^{-5}, 10^{-1})$	$1.58 \cdot 10^{-4}$	$3.11 \cdot 10^{-4}$
Batch size	$\log\mathcal{U}\{1, 32\}$	22	2

Table 3: Test accuracy for machine learning methods

Model	Test accuracy (%)
Support Vector Machine	69.78 ± 0.00
Random Forest	69.69 ± 0.12
CNN (single-label)	$\mathbf{87.02} \pm 3.64$
CNN (multi-label)	67.42 ± 6.50

Our multi-label classifier exceeds 90% individual accuracy in both test and validation, with values of $90.24 \pm 1.92\%$ and $96.55 \pm 0.21\%$ respectively; we remind that this performance is not directly comparable with the values reported in Table 3.

The excellent model behavior on individual label accuracy opens up interesting application possibilities in successfully detecting multiple simultaneous anomalies.

We also check specifically the individual label accuracy achieved on the `ok` class in terms of precision $\frac{TP}{TP+FP}$, which is one of the main goals. Our multi-label model appears to be the most suitable one in this respect, as its performance is $95.11 \pm 0.47\%$ on the test set ($95.06 \pm 0.57\%$ on the validation set). We remark that the two values are very close, suggesting good generalization capabilities; besides, they significantly overperform those obtained with the single-label model, and are more stable in terms of standard deviation.

5 Conclusions

We deal with the issue of detection and classification of anomalous data on images from the intermediate processing in the data reduction system of the Gaia space mission. We investigate the application of both an exact diagnosis technique and different machine learning tools, evidencing that the task can be successfully solved.

The *ad hoc* usage of Hough transform allows to correctly detect all relevant points in the plots under study, thus providing the most complete information required to identify image anomalies.

Among the explored machine learning tools, random forests and support vector machine do not achieve very good results.

CNNs show the best performance, achieving the best image accuracy of 91.22% (single-label case), while in terms of precision for the `ok` class we reach over 95% for the best model (multi-label case).

The results are promising with respect to possible adoption in the Gaia data reduction system of the best performing tools. We remark that multi-label CNNs demonstrated the capability of filtering out 19 out of 20 normal plots, thus going a long way to alleviating the efforts required from human experts.

Conflict of interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.

Acknowledgements The activity has been partially funded by the Italian Space Agency (ASI) under contracts Gaia Mission, The Italian Participation to DPAC, 2014-025-R.1.2015 and 2018-24-HH.0.

This research has been partially carried on in the context of the Visiting Professor Program of the Italian Istituto Nazionale di Alta Matematica (INdAM).

References

1. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray,

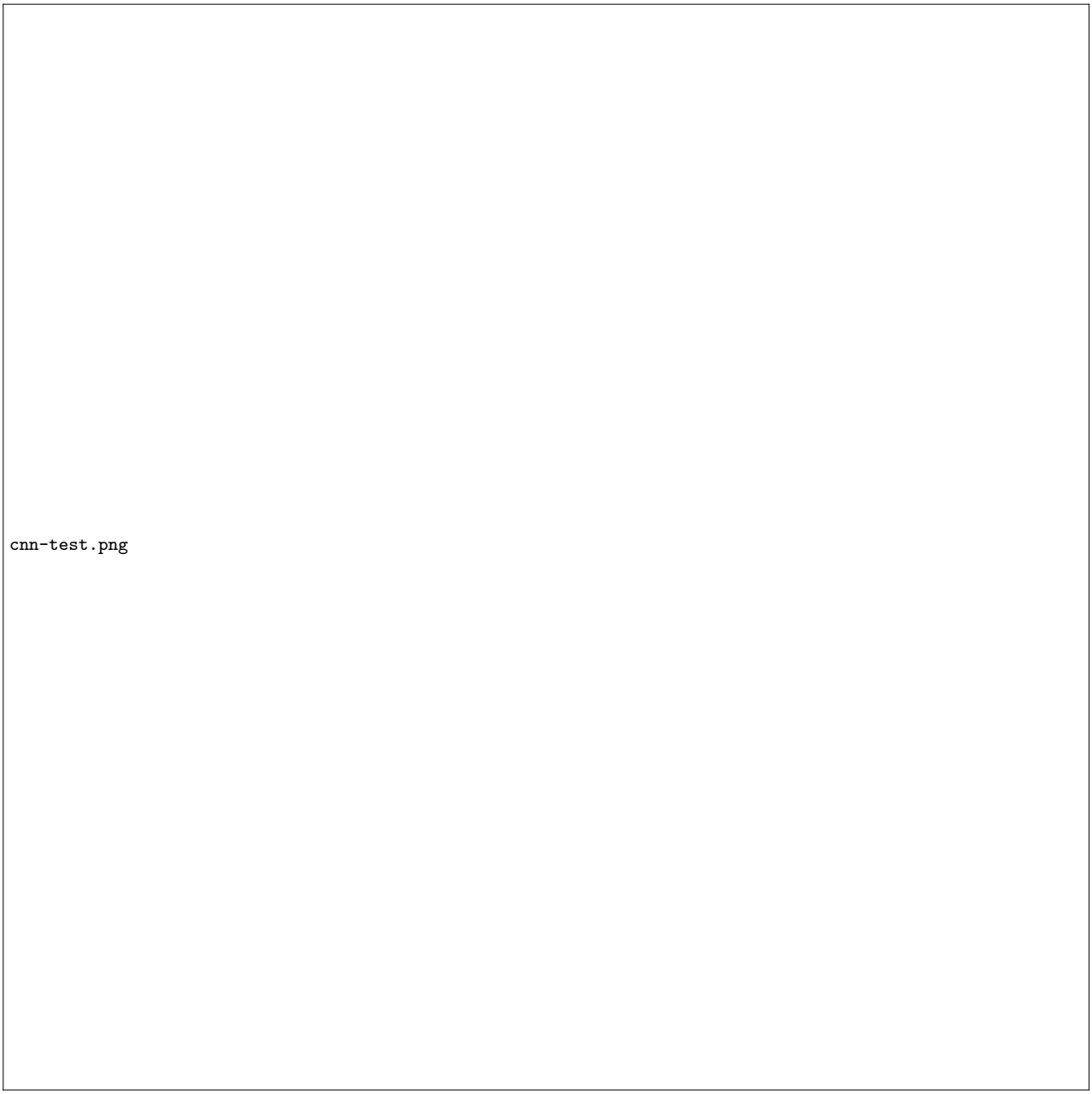


Fig. 9: Confusion matrix of our best performing single-label CNN.

C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

2. M. A. Aizerman. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control*, 25:821–837, 1964.
3. J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pages 2546–2554, 2011.
4. J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305, 2012.
5. J. Bergstra, D. Yamins, and D. D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of the 30th International Conference on Ma-*

chine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 115–123. JMLR.org, 2013.
6. B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the Fifth Annual ACM Conference on Computational Learning Theory, COLT 1992, Pittsburgh, PA, USA, July 27-29, 1992*, pages 144–152. ACM, 1992.
7. G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.
8. L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
9. J. S. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In F. Fogelman-Soulié and J. Hérault, editors, *Neurocomputing - Algorithms, Architectures and Applications, Proceedings of the NATO Advanced Research Workshop on Neurocomputing Algorithms, Architectures and Applications, Les Arcs, France, February 27 - March 3, 1989*, volume 68 of *NATO ASI Series*, pages 227–236. Springer, 1989.
10. J. Choi, H. Eun, and C. Kim. Boosting proximal dental caries detection via combination of variational methods and convolutional neural network. *J. Signal Process. Syst.*, 90(1):87–97, 2018.
11. F. Chollet et al. Keras. `https://keras.io`, 2015.
12. A. Druetto, M. Roberti, R. Cancelliere, D. Cavagnino, and M. Gai. A deep learning approach to anomaly detection in the gaia space mission data. In I. Rojas, G. Joya, and A. Català, editors, *Advances in Computational Intelligence - 15th International Work-Conference on Artificial Neural Networks, IWANN 2019, Gran Canaria, Spain, June 12-14, 2019, Proceedings, Part II*, volume 11507 of *Lecture Notes in Computer Science*, pages 390–401. Springer, 2019.
13. R. O. Duda and P. E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, Jan. 1972.
14. L. Eren, T. Ince, and S. Kiranyaz. A generic intelligent bearing fault diagnosis system using compact adaptive 1d CNN classifier. *J. Signal Process. Syst.*, 91(2):179–189, 2019.
15. D. Evans, M. Riello, F. De Angeli, J. Carrasco, P. Montegriffo, C. Fabricius, C. Jordi, L. Palaversa, C. Diener, G. Busso, et al. Gaia data release 2-photometric content and validation. *Astronomy & Astrophysics*, 616:A4, 2018.
16. P. A. Flach. *Machine Learning - The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, 2012.
17. Gaia Collaboration, C. Babusiaux, F. van Leeuwen, M. A. Barstow, C. Jordi, A. Vallenari, D. Bossini, A. Bressan, T. Cantat-Gaudin, M. van Leeuwen, and et al. Gaia Data Release 2. Observational Hertzsprung-Russell diagrams. *Astronomy and Astrophysics*, 616:A10, Aug. 2018.
18. Gaia Collaboration, A. G. A. Brown, A. Vallenari, T. Prusti, J. H. J. de Bruijne, C. Babusiaux, C. A. L. Bailer-Jones, M. Biermann, D. W. Evans, L. Eyer, and et al. Gaia Data Release 2. Summary of the contents and survey properties. *Astronomy and Astrophysics*, 616:A1, Aug. 2018.
19. Gaia Collaboration, F. Mignard, S. A. Klioner, L. Lindegren, J. Hernández, U. Bastian, A. Bombrun, D. Hobbs, U. Lammers, D. Michalik, and et al. Gaia Data Release 2. The celestial reference frame (Gaia-CRF2). *Astronomy and Astrophysics*, 616:A14, Aug. 2018.
20. Gaia Collaboration, F. Spoto, P. Tanga, F. Mignard, J. Berthier, B. Carry, A. Cellino, A. Dell'Oro, D. Hestroffer, K. Muinonen, and et al. Gaia Data Release 2. Observations of solar system objects. *Astronomy and Astrophysics*, 616:A13, Aug. 2018.
21. X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In G. J. Gordon, D. B. Dunson, and M. Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, volume 15 of *JMLR Proceedings*, pages 315–323. JMLR.org, 2011.
22. I. J. Goodfellow, Y. Bengio, and A. C. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016.
23. P. E. Hart. How the Hough transform was invented. *Signal Processing Magazine, IEEE*, 26:18 – 22, 12 2009.
24. G. E. Hinton. Learning translation invariant recognition in a massively parallel networks. In *International Conference on Parallel Architectures and Languages Europe*, pages 1–13. Springer, 1987.
25. J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.
26. S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In F. R. Bach and D. M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015.
27. K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *IEEE 12th International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, September 27 - October 4, 2009*, pages 2146–2153. IEEE Computer Society, 2009.
28. D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, page 15, 2015.
29. R. Kou, P. Petit, F. Paletou, L. Kulenthirarajah, and J.-M. Glorian. Deep learning determination of stellar atmospheric fundamental parameters. In *SF2A 2018*, pages 167–170, Paris, France, Jan. 2018.
30. A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114, 2012.
31. A. Krogh and J. A. Hertz. A simple weight decay can improve generalization. In J. E. Moody, S. J. Hanson, and R. Lippmann, editors, *Advances in Neural Information Processing Systems 4, [NIPS Conference, Denver, Colorado, USA, December 2-5, 1991]*, pages 950–957. Morgan Kaufmann, 1991.
32. V. Kumar, A. Asati, and A. Gupta. Hardware Accelerators for Iris Localization. *Journal of Signal Processing Systems*, 90(4):655–671, 2018.
33. Y. Lecun. *Generalization and network design strategies*, chapter Learning, pages 143–156. Elsevier, 1989.

34. K. Lee, S. Kung, and N. Verma. Low-energy formulations of support vector machine kernel functions for biomedical sensor applications. *J. Signal Process. Syst.*, 69(3):339–349, 2012.
35. H. W. Leung and J. Bovy. Deep learning of multi-element abundances from high-resolution spectroscopic data. *Monthly Notices of the Royal Astronomical Society*, 483(3):3255–3277, 2019.
36. L. Lindegren, J. Hernández, A. Bombrun, S. Klioner, U. Bastian, M. Ramos-Lerate, A. de Torres, H. Steidelmüller, C. Stephenson, D. Hobbs, and et al. Gaia Data Release 2. The astrometric solution. *Astronomy and Astrophysics*, 616:A2, Aug. 2018.
37. X. Liu, J. Du, J. Yang, P. Xiong, J. Liu, and F. Lin. Coronary artery fibrous plaque detection based on multi-scale convolutional neural networks. *J. Signal Process. Syst.*, 92(3):325–333, 2020.
38. J. B. McDonald and J. B. M. Donald. Application of the Hough transform to lane detection and following on high speed roads. In *in Motorway Driving Scenarios", in Proceeding of Irish Signals and Systems Conference*, page 9, 2001.
39. V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In J. Fürnkranz and T. Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 807–814. Omnipress, 2010.
40. P. Nakjai and T. Katanyukul. Hand sign recognition for thai finger spelling: an application of convolution neural network. *J. Signal Process. Syst.*, 91(2):131–146, 2019.
41. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
42. B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.
43. T. Prusti, J. De Bruijne, A. G. Brown, A. Vallenari, C. Babusiaux, C. Bailer-Jones, U. Bastian, M. Biermann, D. W. Evans, L. Eyer, et al. The gaia mission. *Astronomy & Astrophysics*, 595:A1, 2016.
44. Qi-Chuan Tian, Quan Pan, Yong-Mei Cheng, and Quan-Xue Gao. Fast algorithm and application of Hough transform in iris segmentation. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)*, volume 7, pages 3977–3980 vol.7, 2004.
45. H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.
46. D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.
47. N. Schilling, M. Wistuba, and L. Schmidt-Thieme. Scalable hyperparameter optimization with products of gaussian process experts. In P. Frasconi, N. Landwehr, G. Manco, and J. Vreeken, editors, *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I*, volume 9851 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 2016.
48. A. Silburt, M. Ali-Dib, C. Zhu, A. Jackson, D. Valencia, Y. Kissin, D. Tamayo, and K. Menou. Lunar crater identification via deep learning. *Icarus*, 317:27–38, 2019.
49. N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014.
50. V. Tsoutsouras, K. Koliogeorgi, S. Xydis, and D. Soudris. An exploration framework for efficient high-level synthesis of support vector machines: Case study on ECG arrhythmia detection for xilinx zynq soc. *J. Signal Process. Syst.*, 88(2):127–147, 2017.
51. D. Tuccillo, M. Huertas-Company, E. Decencière, S. Velasco-Forero, H. Domínguez Sánchez, and P. Dimauro. Deep learning for galaxy surface brightness profile fitting. *Monthly Notices of the Royal Astronomical Society*, 475(1):894–909, 2018.
52. C. Yang and J. Collins. Improvement of Honey Bee Tracking on 2D Video with Hough Transform and Kalman Filter. *Journal of Signal Processing Systems*, 90(12):1639–1650, 2018.
53. Y. Zhou and R. Chellappa. Computation of optical flow using a neural network. In *Proceedings of International Conference on Neural Networks (ICNN'88), San Diego, CA, USA, July 24-27, 1988*, pages 71–78. IEEE, 1988.
54. T. Zingales and I. P. Waldmann. Exogan: Retrieving exoplanetary atmospheres using deep convolutional generative adversarial networks. *The Astronomical Journal*, 156(6):268, 2018.