Computer Assisted Methods in Engineering and Science
30(4): 461–479, 2023, doi: 10.24423/cames.527

Convolutional Neural Networks in the Detection of Astronomical Objects from the Messier Catalog

Witold BELUCH*, Paweł ŚLIWA

Faculty of Mechanical Engineering, Silesian University of Technology, Gliwice, Poland
** Corresponding Author e-mail: witold.beluch@polsl.pl*

This paper explores the application of convolutional neural networks in the field of amateur astronomy. The authors have employed the available astronomical datasets to develop a detector for identifying astronomical objects from the Messier catalog. A concept framework for creating such a detector for astronomical objects using artificial intelligence tools in the form of a detector based on convolutional neural networks is presented. Augmentation and pre-processing procedures have been used to extend the feature distribution in the training set. Examples confirming the effectiveness of the proposed detector of astronomical objects are presented.

Keywords: convolutional neural networks, astronomical objects detection, Messier catalog.

1. Introduction

Object detection belongs to the interdisciplinary scientific area known as computer vision. It involves the localization and classification of objects in digital images. The detector requires the submission of a digital image record, which is subsequently processed by a chosen computational intelligence method. Initially, computer vision relied on image descriptor techniques to create detection models. The algorithms using descriptors, which did not incorporate neural networks into their architecture, relied on representing characteristic features to describe items being searched for [33].

Currently, the processing and classification of images delivered to the detectors are most often performed by convolutional neural networks (CNNs). In 2010, during the ImageNet competition, detectors based on architectures containing CNNs and employing deep learning methods officially surpassed the performance of traditional image descriptor-based solutions [25].

CNNs, as we know them today, were first introduced by LeCun *et al.* [15] in 1989. They allowed the operation on inputs of one to three dimensions. The proposed LeNet-5 network was a multilayer CNN using a backpropagation algorithm and was used to classify handwritten digits [16, 17]. A key advantage of CNNs is the use of a machine learning algorithm performing automatic feature extraction, eliminating the need to prepare hand-engineered filters. The successful combined use of non-handcrafted and handcrafted features was presented in [23]. Krizhevsky *et al.* [13] proposed AlexNet, a deep CNN architecture, achieving a significant improvement over LeNet-5. Developments in computer vision and CNNs have resulted in the introduction of architectures such as GoogleNet [34], VGGNet [32], ZFNet [36] and containing residual connections ResNet architecture [10]. Currently developed models based on CNNs are used in various computer vision problems such as semantic segmentation [21], object detection [27], action recognition in video [31], 3D analysis [12], and natural language processing [6].

The present paper is focused on CNN applications in astronomy. The spectrum of CNN applications in modern astronomy includes solar activity prediction, telescope control systems, pulsar candidate selection, or stellar classification [35]. An example of the application of CNNs can be the problem of morphological classification of galaxies [7]. A binary classification of a galaxy's membership in the group of barred spiral galaxies based on CNNs was proposed in [1].

Professional real-time object tracking systems with long exposure times, such as the GoTo system [14], have become accessible to amateurs using astronomy telescopes. A popular tool for amateur astronomy is a telescope equipped with a CCD camera and a tracking system [37], which allows both real-time observations and astrophotography.

Amateur astronomy has greatly benefited from advancements in modern technology and equipment. Nowadays, thanks to the use of advanced tracking systems and long-exposure CCD cameras instead of a telescope and bare eyes, amateur astronomers possess more observational power than their counterparts in the 20th century. A major development in improving the observability of astronomical objects has been brought to amateurs by the use of advanced mounts, such as the equatorial mount [2]. Such mounts allow for precise adjustments in the telescope's angles, combined with electric motors operating in two axes, resulting in a precise tracking system. With such a tracking system, amateur astronomers can track the movement of celestial objects (effectively compensating for the Earth's rotation), allowing them to collect substantially more light through their telescope's mirror with the same surface area and optical quality using CCD cameras.

Deep-sky objects (DSO) listed in the Messier catalog are popular among amateur astronomers due to their high brightness and angular size. The first

version of this catalogue was published by Charles Messier in 1774 under the name the "Catalog of Nebulae and Star Clusters", which eventually contained 110 objects [22]. This specific number of objects is the subject of the research presented in this paper, carried out with the aim of establishing proof of concept for a Messier catalog detector. The authors' main motivation for developing the proposed approach was the absence of similar solutions for detecting objects from the Messier catalog in the existing literature.

The aim of this work was to implement a Messier catalog object detector using CNN. The Messier object detector can be employed as an addition to the GoTo control system predominantly employed in amateur telescopes. The GoTo system is an automatic system that controls the axes of electrical drives. The system can work in two modes. The first mode is object tracking, in which the axes move to keep the observed point in the telescope's eyepiece, tracking the object according to the speed of Earth's rotation. A telescope equipped with the GoTo system can also work in object search mode. The user enters the code of the astronomical object and the telescope, based on the GPS location and the current date, locates the object. Telescopes equipped with the GoTo system are often used in astrophotography due to their ability to accurately track objects, which usually requires a longer exposure time for the camera matrix. Current matrix technology enables connection to a computer and allows for observation on a monitor. The proposed detector, integrated into software, enables the detection of selected astronomical objects within the observed section of the sky using an astrophotography camera.

A novel approach is the application of machine learning and CNN-based architectures to address the problem of detecting DSO objects listed in the Messier catalog. This is conducted by presenting a proof of concept for the proposed detector.

The paper is organized as follows. Section 2 introduces the idea of CNNs. Section 3 presents the topic of object detection using the Faster R-CNN architecture, the applied metrics and the programming library used (TensorFlow). Section 4 is dedicated to data preparation, including data augmentation techniques and dataset splitting. Section 5 presents the results of the experiments. The paper concludes with a summary and a bibliography.

2. Convolutional neural networks

CNNs are a specialized class of deep neural networks (DNNs) in which at least one layer of the network performs a convolution operation, replacing the general matrix multiplication operation found in classical DNNs [16]. Due to the functionality of layers in CNNs, they can be divided into [25]:

- convolutional layers – containing filters responsible for feature extraction;
- subsampling (pooling) layers – ensure tensor size reduction;
- fully connected (FC) layers – responsible for further processing of feature maps and data classification.

Tensor in CNN denotes an image with a certain number of channels. Convolutional layers, together with pooling layers, are used in CNNs as the feature extractor, forming the backbone network whose purpose is to extract features from a provided input image and to reduce the dimensions of the resulting tensor. The specific number and types of layers influence the feature extraction and tensor size reduction. An exemplary CNN in the form of VGGNet-16 [32] is presented in Fig. 1.

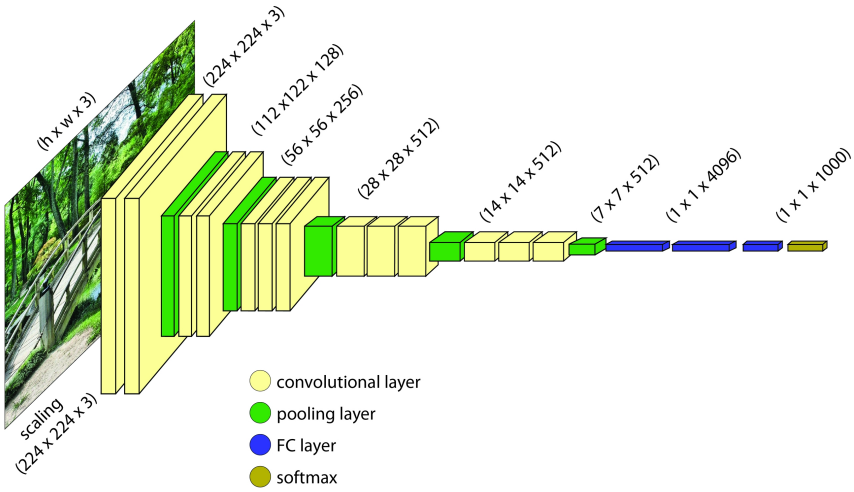

Fig. 1. VGGNet-16, an example feature extractor with FC layers.

The tensor supplied to the CNN inputs is subjected to a convolution operation using filters that are matrices of specific sizes and are initialized at random. Their weights are modified during learning [25]. The operation of a filter on an image encoded in three channels (R, G, B) is shown in Fig. 2. The filter with dimensions of $n \times n \times c$, where c denotes the number of channels, is shown surrounded by a dashed line. The filter moves across the channel matrix by a pre-set step (stride).

Each element of a particular channel of the input tensor of dimension j, k is subjected to the element convolution operation (*) according to the formula:

$$R_{(j-n+1)\,(k-n+1)} = M_{j\,k} * F_{n\,n}, \tag{1}$$

where M – the input matrix for given channel, F – the single filter matrix for given channel.

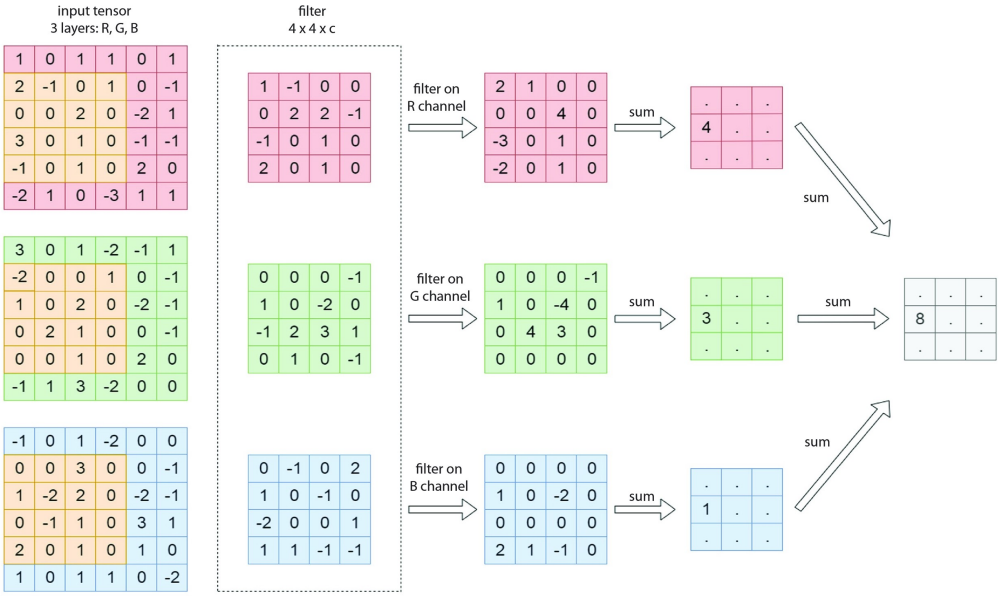

FIG. 2. Operation of a single filter.

The depth of the resulting tensor (and consequently the number of feature maps) is increased by applying multiple filters at the convolution stage. Neurons in the convolutional layer are not fully connected; instead, the output of each neuron in the convolutional layer is directed to selected neurons of the subsequent layer [28].

Subsampling (pooling) layers allow to reduce the dimensions of the resulting tensor while retaining relevant information [7]. The subsampling layers use non-overlapping frame shifting across input matrices in all channels to determine representative elements [4]. An example of the max pooling operation is presented in Fig. 3.

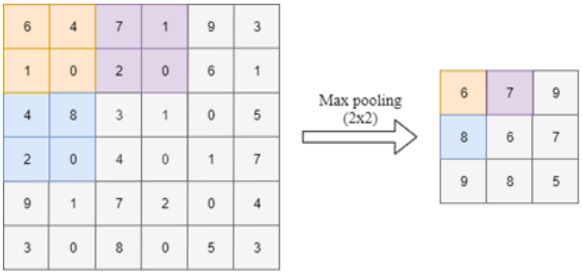

FIG. 3. Exemplary max pooling operation.

Then, the resulting tensor obtained from the backbone network is passed to the first FC layer, which requires input data in the form of a vector. To convert the vector into the input tensor, a flattening operation is performed,

projecting the tensor into a vector [3]. The task of the FC layers is to appropriate classify the received data features. The last FC layer in CNNs usually has a softmax activation feature enabling multiclass classification. A characteristic feature of the softmax function is that the values in the output vector specifying the probability of belonging to a given category always sum to one [8]. The use of the softmax function allows the categorical cross-entropy loss to be used as a cost function [18].

3. Object detection using CNNs

Object detection using CNNs involves several stages of data processing. A detector, understood as a trained CNN prepared to work in a specific environment, has to detect the presence of a given object in a provided element, typically an image, precisely locate the detected object, and categorize it into a given category [19]. The detector performs the task of object position regression present in the provided image and its classification. A detector with a sufficiently short prediction time, for which the data is buffered and provided in real time, can effectively work in a real-time system [11]. Applied detector solutions incorporating CNNs can be divided into one-stage and two-stage architectures. One-stage architectures, such as YOLO [29] or SSD [20], are usually characterized by their shorter inference times, while two-stage architectures, although slower, offer greater precision [24].

The CNN architecture used in this work is the faster R-CNN architecture which as shown in Fig. 4, is an extension of the fast R-CNN concept [9] in which the selective search algorithm is replaced by a region proposal network (RPN). The RPN, being a trainable neural network placed after the CNN's final convolutional layer, allows finding regions that might contain the objects of interest (regions of interest, RoI). The task of the RPN, being a binary classifier of object presence, is to determine whether the sought-after object is present in a given

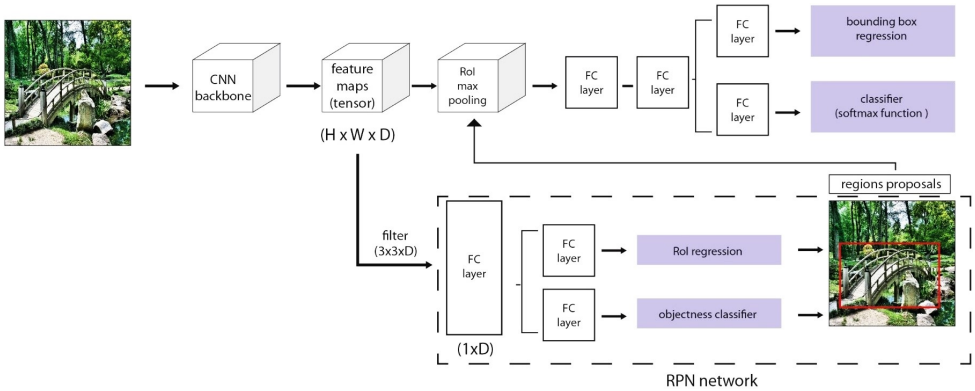

Fig. 4. Network image processing scheme with faster R-CNN architecture.

region, defined by generated anchor boxes, or whether the area represents a background.

The cost function $L_{\text{RPN}}(p,t)$ for the location is referred to as RPN loss and consists of classification and regularization cost components [30]:

$$L_{\text{RPN}}(p,t) = \frac{1}{N_{\text{cls}}} \sum_i L_{\text{cls}}(p_i, p_i^*) + \lambda \frac{1}{N_{\text{reg}}} \sum_i p_i^* \, L_{\text{reg}}(t_i, t_i^*), \tag{2}$$

where N_{cls} – the mini-batch size for a single image, which is equal to 256 (the sum of 128 positive and 128 negative samples extracted from the given image), N_{reg} – the number of anchor locations, $\sim$2400, $L_{\text{cls}}(p_i, p_i^*)$ – the classification loss, p_i – the output score from the classification branch for anchor i, p_i^* – the ground truth label (1 or 0), $L_{\text{reg}}(t_i, t_i^*)$ – the regression loss, t_i – the output prediction of the regression layer, t_i^* – the regression target, and λ – a parameter that regulates the contribution of the cost value of a given component to the total RPN network cost.

The classification loss L_{cls} is the log loss over two classes (object versus not object). The regression loss L_{reg} is activated only if the ground truth p_i^* is 1 (an anchor contains an object).

In the second stage, the goal is to classify the objects and refine their location. The backbone network and feature map are shared with the RPN. However, further processing through the FC layer requires a fixed size for each part of the feature map associated with a given RoI. This effect is achieved by using the RoI pooling layer. This layer applies max-pooling for each RoI while scaling the selected part of the feature map to a square and fixed size for all regions. The cost function $L(p,u,t^u,v)$ is defined as [9]:

$$L(p,u,t^u,v) = L_{\text{cls}}(p,u) + \lambda[u \geq 1] L_{\text{loc}}(t^u, v), \tag{3}$$

where p – the predicted class, u – the ground-truth class, t^u – the predicted tuple of localization bounding box, and v – the ground-truth target for localization task, L_{loc} – the loss for localization part of classifier.

The cost of classification is determined by the categorical cross-entropy loss function, primarily used in cases of multi-class classification [25].

The following metrics were used to assess detection performance [26]:

- recall metric:

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}; \tag{4}$$

- precision metric:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{5}$$

where TP – true positive, FN – false negative, and FP – false positive;

- AP (average precision) metric calculates the area under the precision-recall curve $p(r)$:

$$\mathrm{AP} = \int_0^1 p(r)dr, \tag{6}$$

where p – the value of precision for given classifier, r – the value of recall for given classifier;

- mAP (mean average precision) metric for a number of classes equal to c:

$$\mathrm{mAP} = \frac{1}{c}\sum_{i=1}^{c} \mathrm{AP}_i; \tag{7}$$

- AR (average recall) and mAR (mean average recall) metrics:

$$\mathrm{AR} = 2\int_{0.5}^{2} r(o)do, \tag{8}$$

$$\mathrm{mAR} = \frac{1}{c}\sum_{i=1}^{c} \mathrm{AR}_i, \tag{9}$$

where $o -$ IoU (intersection over union) value is defined as:

$$\mathrm{IoU}(P, GT) = \frac{P \bigcap GT}{P \bigcup GT} = \frac{P \bigcap GT}{P + GT - P \bigcap GT}, \tag{10}$$

where P – the prediction area, and GT – the ground-truth area.

IoU is a value determined by the prediction of a single bounding box, and it alone does not determine the quality of the prediction for the entire data set. The introduction of an IoU threshold into the calculation allows to reject predictions for which the IoU value is below a given level. The most used IoU thresholds are 0.50 or 0.75. For example, $\mathrm{mAP}^{\mathrm{IoU}=0.50}$ denotes the average precision determined for an IoU threshold of 0.50.

The TensorFlow object detection API [38] was used to implement and train the detector. This environment is a high-level application programming interface (API) that provides several models with sample configuration files and guidance on the operation of the interface. Figure 5 shows a simplified diagram of the layered structure of TensorFlow implementation. The lowest layer, performing low-level operations, is implemented in C++ programming language. The higher

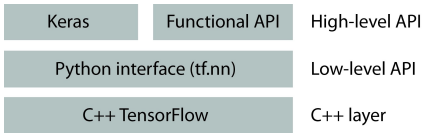

Fig. 5. Implementation layers in the TensorFlow environment.

layer is an implementation of a Python language interface using the lower layer. The top layer is the Keras library, which automates many operations and provides users with a high-level library of abstractions to implement (single shot networks) SSNs. Users can also employ the high-level functional API, which includes the TensorFlow object detection API. TensorFlow uses the CUDA architecture for optimal performance on GPU cores.

4. Data preparation

The aim of the study is to train a high-quality detector for astronomical objects from the Messier catalog. It is assumed that CNN should perform detecting objects in a fragment of the sky with dimensions $60' \times 60'$. The necessary photographs were obtained from the Digitized Sky Survey dataset downloaded in the FITS astronomical format from the Space Telescope Science Institute website [39].

The object searched for is provided together with the extent of the patch of sky on which it occurs. The Hubble Space Telescope (HST) Phase 2 images were downloaded together with a background covering an area of $30' \times 30'$, $45' \times 45'$ or $60' \times 60'$ squares. The images were converted from FITS to JPG format, and any corrupted images were removed.

4.1. Data augmentation

The requirement for the training set was to obtain the widest possible feature distribution, despite using a small number of real images. It should be emphasized that these augmentations should reflect the characteristics of actual objects. It is worth mentioning that every object in the Messier catalog has the same orientation from the perspective of an observer on Earth, so rotational augmentations, for example, are not relevant in this case. Similarly, augmentations that could change the proportions of a Messier object were not applied, as the proportions of a Messier object are a characteristic feature of that object and changing them could lead to recognition being unsuccessful. The main focus was on preparing a training dataset that reflects different observing conditions, such as sky contrast and brightness, and to simulate, for example, different levels of light pollution, the area of observation expressed in arc minutes, the location of the object and the proportion of the object area to the whole image area.

Since most of the Messier objects appear individually in the image (the exceptions might be, for example, M31 and M32, which the authors consider irrelevant also due to the fact of the significant difference in size between the two objects), it was decided to treat the Messier object detection as a detection of a single Messier object within the image. The detection was carried out in terms of object localization and classification. For this reason, it was necessary to augment the

existing images with dimensions of 30' × 30' and 45' × 45' by adjusting the contrast and brightness of the image, as well as the size and location of the ground truth box. Automatic and random modifications to contrast and brightness were applied.

The position and size of the ground truth box were modified using the RandomCrop function from the Albumentations library [5]. The function takes an image with its description and modifies the image by randomly cropping a segment with specified maximum dimensions and adjusting the ground truth box description accordingly to the image modification. The probability for the RandomCrop feature was set as 70%, leaving 30% of the features as unmodified. Finally, the set contains astronomical randomly rotated objects, with randomly modified brightness and with random size and position of the object within the element.

On the other hand, the test set was not augmented; each object was located in the center of each 60' × 60' image, the same as in the original FITS format.

4.2. Data splitting

A training set of 7000 elements, each with an of original resolution of 1600 × 1600 pixels, cropped randomly to different resolutions and a test set of 500 elements, each with resolution 2000 × 2000 pixels were generated together with a description in CSV format. Examples of the training set elements are shown in Fig. 6. Examples of all 110 classes were included in the training set.

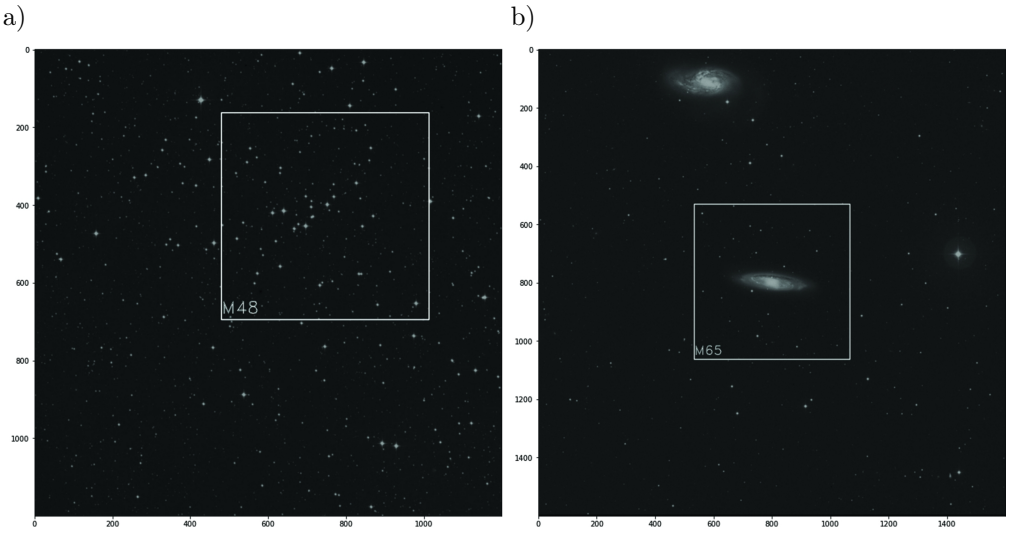

Fig. 6. Exemplary elements of the training set with ground truth boxes (the scale in pixels): a) example of randomly cropped to 1200 × 1200 M48 object, b) ultimately also modified to 1600 × 1600 M65 object.

The test set, composed of 500 elements with a resolution of 2000×2000 pixels, was generated together with a description in CSV format. The test set was created from $60' \times 60'$ images originally downloaded in FITS format from the website and randomly selected to ensure a representation of 110 classes. Examples of all 110 classes were included in the test set. Notably, the test set was not augmented, and the location, brightness and contrast of the objects were preserved.

5. Project assumptions and model architecture

For detector accuracy, the prioritization is placed on higher AR metric values over the mAP. This is because primary concern for the user is whether an existing object from the Messier catalog is found, making the value of the AR metric more important parameter than the accuracy of object classification. Therefore, the $\text{mAP}^{\text{IoU}=0.50}$ metric for validating the detector accuracy was chosen as the most important metric, as the metric $\text{mAP}^{\text{IoU}=0.75}$ is too restrictive for amateur users. The requirements for the quality of the detector model were values of $\text{mAP}^{\text{IoU}=0.50}$ metrics exceeding a value of 90%, while maintaining a value of AR metrics at not less than 75%. Due to the detector's operation in a relatively slow-moving environment, the faster R-CNN architecture was chosen for its high prediction accuracy.

The training of the CNN was carried out using different feature extractors as backbones: ResNet50, ResNet101 and Inception v2. An application of residual connections in ResNet networks [10] eliminates the problem of a vanishing gradient and time-consuming training in neural networks with many layers.

Residual connections refer to the network layer blocks forming a given ResNet module. A scheme of a single ResNet block is shown in Fig. 7, where: $\mathcal{F}(\mathbf{x})$ – the residual mapping replacing the original, and $\mathbf{x}$ – the input vector. When the difference between $\mathcal{H}(\mathbf{x}) - \mathcal{F}(\mathbf{x})$ is close to 0, the CNN is taught to skip a given module through the residual connection, and additionally the weights of the skipped block are not updated.

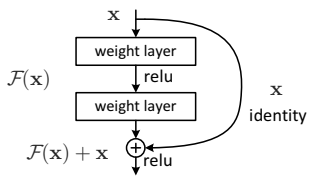

Fig. 7. Scheme of a single ResNet [10].

Similar advantages can be obtained by using Inception neural networks, in which the use of filters of various sizes in a single module of convolutional layers results in a change of the receptive field parameter [34].

6. Experiments and results

To maximize the level of detail, the dimensions of the input tensor were set as large as possible given the high resolution of the training and test images. The maximum acceptable image size, due to the limitation of the VRAM of the available graphics card, was a square of 1500×1500. Training hyperparameters were selected using a validation set separated from the training set, with 10% of the training set elements were included. The final verification was performed on the test set.

The main parameters of the Faster R-CNN were set as:

- the value of the learning rate $\eta =$ 1e-4 $\div$ 2e-6 (experimentally adjusted);
- *image_resizer*: *keep_aspect_ratio_resizer* with *min_dimension* and *max_dimension* set to 1500;
- *batch_size* = 1 (limitation of VRAM);
- *fine_tune_checkpoint* – pre-trained on the COCO dataset checkpoint;
- num_classes = 110;
- a parameter responsible for the size of the RPN output tensor *first_stage_features_stride* = 16;
- *first_stage_nms_score_threshold* = 0.0;
- *first_stage_nms_iou_threshold* = 0.7;
- parameters responsible for the first stage (RPN) loss: *first_stage_localization_loss_weight* = 2.0, *first_stage_objectness_loss_weight* = 1.0;
- parameters responsible for the second stage loss: *second_stage_localization_loss_weight* = 2.0, *second_stage_classification_loss_weight* = 1.0;
- *second_stage_post_processing*: *batch_non_max_suppression* with *score_threshold* set to 0.0, *iou_threshold* = 0.6 and *use_class_agnostic_nms* parameter activated.

Two different learning rate optimizers have been considered: momentum activated (with momentum coefficient equal to 0.9) and Adam (adaptive moment estimation) to compare their training effectiveness. The parameters such as AR, $\text{mAP}^{\text{IoU=0.50}}$ and $\text{mAP}^{\text{IoU=0.75}}$ were considered as the most relevant to the final user. The dependency of different metrics: AR, $\text{mAP}^{\text{IoU=0.50}}$ and $\text{mAP}^{\text{IoU=0.75}}$ on the number of training steps is presented in Figs. 8–10, respectively.

Summary results, showing information on the training metrics analyzed as a function of the number of learning epochs and the number of steps in which the maximum values were achieved, are presented in Table 1.

As a result of the training six different versions of the detector presented in Table 1, the requirement $\text{mAP}^{\text{IoU=0.50}} \geq 90\%$ was satisfied for all the backbone

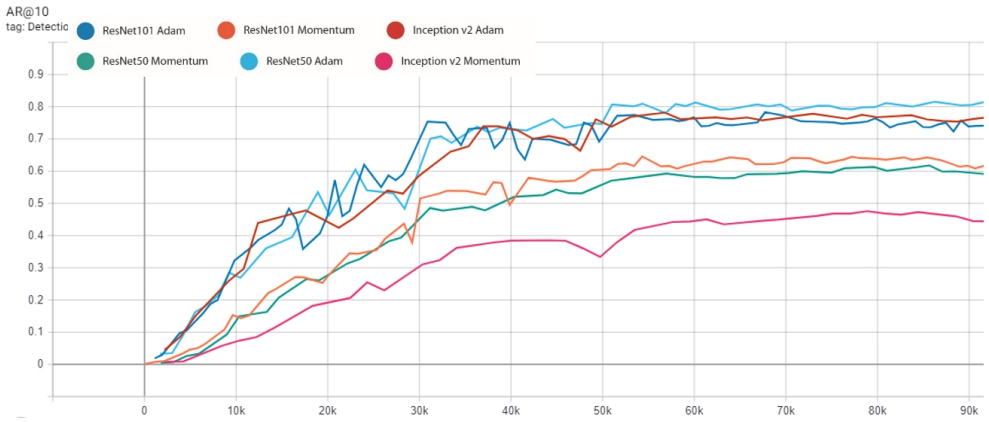

FIG. 8. Dependency of the AR metric on the number of training steps.

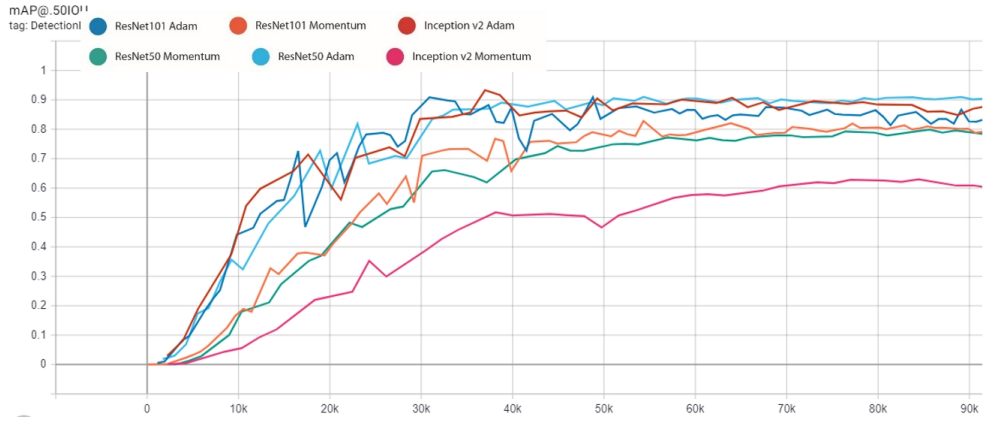

FIG. 9. Dependency of the $\mathrm{mAP}^{\mathrm{IoU}=0.50}$ metric on the number of training steps.

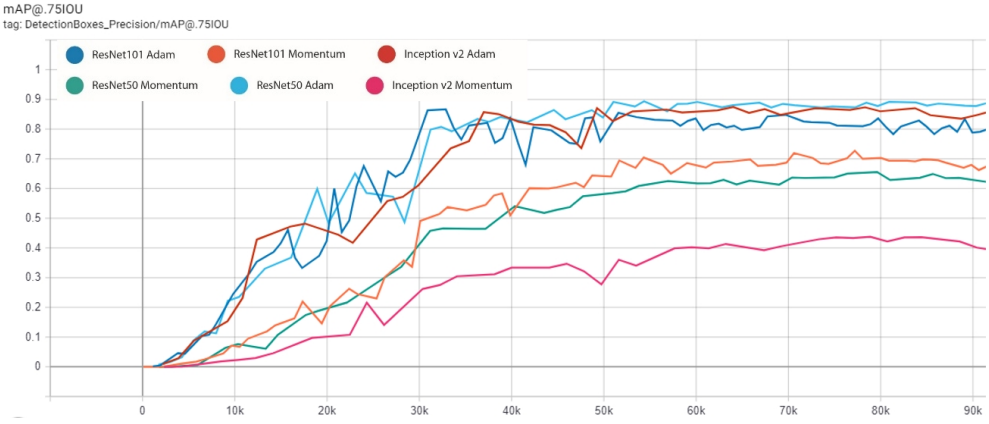

FIG. 10. Dependency of the $\mathrm{mAP}^{\mathrm{IoU}=0.75}$ metric on the number of training steps.

TABLE 1. Resulting accuracy metrics for the faster R-CNN detector.

Backbone	Learning rate optimizer	$mAP^{IoU=0.50}$ [%]	$mAP^{IoU=0.75}$ [%]	mAP [%]	AR [%]	Number of steps
ResNet50	Adam	91.01	89.40	79.23	80.98	54 350
	Momentum	79.89	64.91	55.86	61.79	85 640
ResNet101	Adam	90.87	86.32	72.13	75.49	30 900
	Momentum	82.84	70.45	58.93	64.56	54 320
Inception v2	Adam	90.47	87.07	73.83	76.11	49 260
	Momentum	62.85	43.53	39.60	46.86	77 020

types used, while only ResNet50 achieved the mAP metric value higher than 75%. The results obtained with the Adams optimizer outperformed those obtained with the momentum optimizer in all cases. Consequently, the ResNet50 with the Adam optimizer was selected as the best architecture and training configuration for the problem under analysis. Visualizations of exemplary detections from all 110 actual objects of Messier catalog using the best detector configuration are shown in Figs. 11–13.

Figures 11–13 demonstrate the model's good performance on the test set. Both localization and classification tasks have good quality, which is confirmed by the visualizations of the detections. The localization task is precise, capturing only the area that overlaps with the ground-truth. As the classification task is also solved correctly, the model is resilient to errors due to confusion between classes.

a) b)

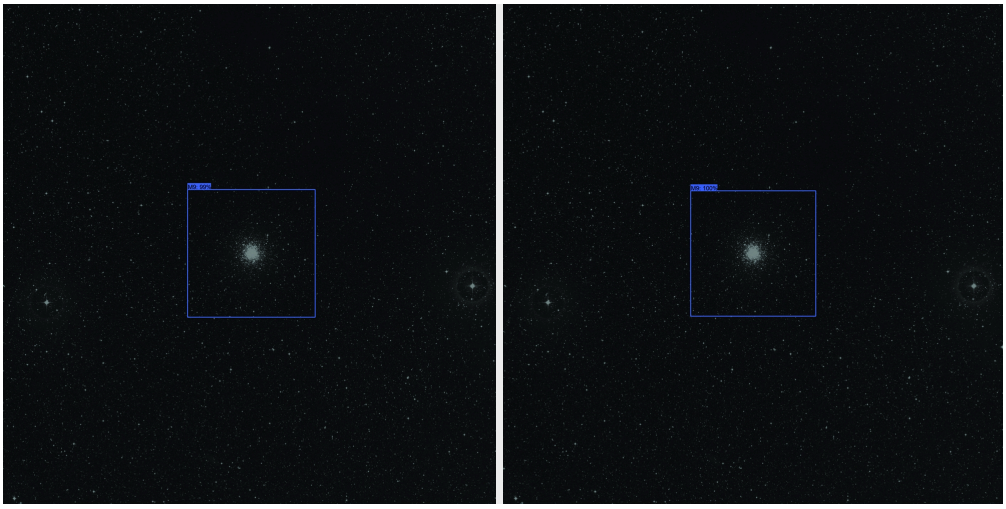

FIG. 11. M9 object detection a) result (confidence 0.99), b) ground truth (100%).

a) b)

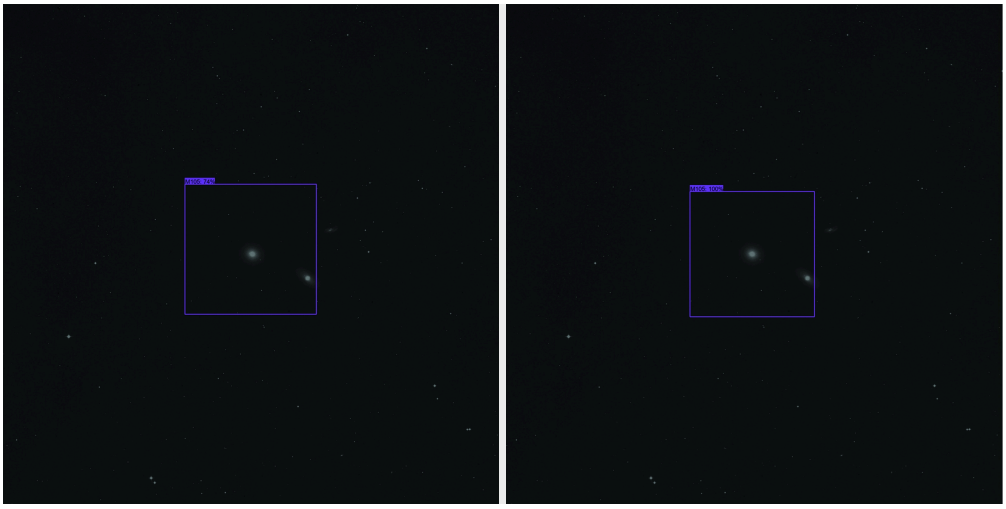

FIG. 12. M105 object detection a) result (confidence 0.74), b) ground truth (100%).

a) b)

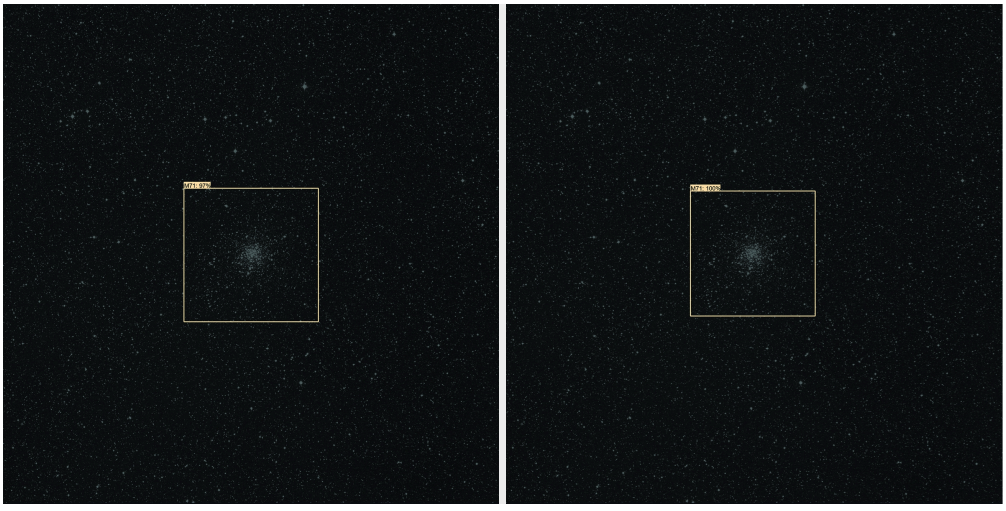

FIG. 13. M71 object detection a) result (confidence 0.97), b) ground truth (100%).

7. Conclusion

This paper investigated the application of deep learning solutions to amateur astronomical observing systems. CNNs in the form of the faster R-CNN architecture were used to create a detector for astronomical objects from the Messier catalog. Selected configurations of the faster R-CNN architecture were

tested with different backbone networks and learning optimizers. The tests indicated that in the case under consideration, the ResNet50 network with 50 layers yielded the highest efficiency and the Adams optimizer gave much better results than the momentum optimizer.

The use of a larger CNN for a framework such as ResNet101, in this case, leads to quicker convergence, but with slightly worse performance compared to ResNet50. This could be attributed to the known fact of higher overfitting risk when using larger CNNs, which in this case is slightly higher for ResNet101 compared to ResNet50 or Inception v2 as a backbone with the Adam optimizer.

Tests considering the momentum algorithm, while less relevant to the article's focus were performed for a single hyperparameter for the momentum variable. The momentum algorithm performs poorly with all backbones – a possible explanation for this phenomenon is that momentum is not an adaptive algorithm and may require a long time and resources to explore the state space for the best solution with methods such as grid search, including checking multiple hyperparameters.

Detector training of objects in the Messier catalog is not a typical problem of object detection. The colour variation of objects is marginal, and additionally, there is a large group of objects that are difficult to distinguish from each other, e.g., globular clusters. These challenges may impact object classification.

Most publications combining astronomy and deep learning focus on applying artificial intelligence to large datasets from professional observatories. However, the use of augmentation has made it possible to significantly increase the size of the training set and to further expand the feature distribution by solving the problem of different object location, size, sky brightness and contrast.

The images employed for training, validating and testing sets for the target, commercially useful version should maintain high sharpness and resolution. In addition, for commercial applications, it would be essential to train the obtained detector on images from real observations. Nevertheless, the solution presented in this paper provides a useful proof of a concept for obtaining a detector for astronomical objects serving the needs of astronomy and amateur astrophotography by means of deep learning methods.

References

1. S. Abraham, A.K. Aniyan, A.K. Kembhavi, N.S. Philip, K. Vaghmare, Detection of bars in galaxies using a deep convolutional neural network, *Monthly Notices of the Royal Astronomical Society*, **477**(1): 894–903, 2014, doi: 10.1093/mnras/sty627.

2. K. Aiuchi, K. Yoshida, M. Onozaki, Y. Katayama, M. Nakamura, K. Nakamura, Sensor-controlled heliostat with an equatorial mount, *Solar Energy*, **80**(9): 1089–1097, 2006, doi: 10.1016/j.solener.2005.10.007.

3. S.H.S. Basha, S.R. Dubey, V. Pulabaigari, S. Mukherjee, Impact of fully connected layers on performance of convolutional neural networks for image classification, *Neurocomputing*, **378**: 112–119, 2020, doi: 10.1016/j.neucom.2019.10.008.

4. Y. Boureau, J. Ponce, Y. LeCun, A theoretical analysis of feature pooling in visual recognition, [in:] *Proceedings of the 27th International Conference on Machine Learning (ICML'10)*, pp. 111–118, 2010.

5. A. Buslaev, V.I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, A.A. Kalinin, Albumentations: fast and flexible image augmentations, *Information*, **11**(2): 125, 2020, doi: 10.3390/info11020125.

6. R. Collobert, J. Weston, A unified architecture for natural language processing: deep neural networks with multitask learning, [in:] *Proceedings of the 25th International Conference on Machine Learning ICML '08*, pp. 160–167, 2008, doi: 10.1145/1390156.1390177.

7. S. Dieleman, K.W. Willett, J. Dambre, Rotation-invariant convolutional neural networks for galaxy morphology prediction, *Monthly Notices of the Royal Astronomical Society* **450**(2): 1441–1459, 2015, doi: 10.1093/mnras/stv632.

8. I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, MIT Press, 2016.

9. R. Girshick, Fast R-CNN, [in:] *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 2015, doi: 10.1109/ICCV.2015.169.

10. K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, 2015, arXiv: 1512.03385.

11. S. Hossain, D. Lee, Deep learning-based real-time multiple-object detection and tracking from aerial imagery via a flying robot with GPU-based embedded devices, *Sensors*, **19**(15): 3371, 2019, doi: 10.3390/s19153371.

12. S. Ji, W. Xu, M. Yang, K. Yu, 3D convolutional neural networks for human action recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **35**(1): 221–231, 2012, doi: 10.1109/TPAMI.2012.59.

13. A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, *Communications of the ACM*, **60**(6): 84–90, 2017, doi: 10.1145/3065386.

14. T.S. Kumar, R.N. Banavar, Design and development of telescope control system and software for the 50/80 cm Schmidt telescope, *Optical Engineering*, **52**: 081607, 2013, doi: 10.1117/1.OE.52.8.081607.

15. Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, L. Jackel, Handwritten digit recognition with a back-propagation network, *Advances in Neural Information Processing Systems*, **2**: 396–404, 1989.

16. Y. LeCun, Y. Bengio, Convolutional network for images, speech and time series, [in:] *The handbook of brain theory and neural networks*, M.A. Arbib [Ed.], The MIT Press, 1995.

17. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, [in:] *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998, doi: 10.1109/5.726791.

18. X. Li, W. Wang, Learning discriminative features via weights-biased softmax loss, *Pattern Recognition*, **107**: 107405, 2020, doi: 10.1016/j.patcog.2020.107405.

19. K. Li, W. Ma, U. Sajid, Y. Wu, G. Wang, Object detection with convolutional neural networks, [in:] *Deep Learning in Computer Vision: Principles and Applications*, CRC Press, 2020.

20. W. Liu *et al.*, SSD: single shot multiBox detector, [in:] B. Leibe, J. Matas, N. Sebe, M. Welling [Eds.], *Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science*, Vol. 9905, pp. 21–37, Springer, Cham, 2016, doi: 10.1007/978-3-319-46448-0_2.

21. J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, [in:] *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440, 2015, doi: 10.1109/CVPR.2015.7298965.

22. C. Messier, *Catalog of Nebulae and Star Clusters*, Memoirs of the French Academy of Sciences (in French: *Catalogue des Nébuleuses & des amas d'Étoiles*, Connoissance des Temps Pour l'Année), pp. 435–461, 1781.

23. L. Nanni, S. Ghidoni, S. Brahnam, Handcrafted vs non-handcrafted features for computer vision classification, *Pattern Recognition*, **71**: 158–172, 2017, doi: 10.1016/j.patcog.2017.05.025.

24. H. Nguyen, Improving Faster R-CNN framework for fast vehicle detection, *Mathematical Problems in Engineering*, **2019**: article ID 3808064, 11 pages, 2019, doi: 10.1155/2019/3808064.

25. B. Planche, E. Andres, *Hands-On Computer Vision with TensorFlow 2: Leverage deep learning to create powerful image processing apps with TensorFlow 2.0 and Keras*, Packt Publishing, 2019.

26. D.M.W. Powers, Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation, *Journal of Machine Learning Technologies*, **2**(1): 37–63, 2011.

27. S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, [in:] *NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems*, **1**: 91–99, 2015.

28. M.R.N. Rao, V.V. Prasad, P.S. Teja, Md. Zindavali, O. Reddy, A survey on prevention of overfitting in convolution neural networks using machine learning techniques, *International Journal of Engineering and Technology*, **7**(2): 177–180, 2018, doi: 10.14419/ijet.v7i2.32.15399.

29. J. Redmon, S. Divvala, R.B. Girshick, A. Farhadi, You only look once: unified, real-time object detection, [in:] *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91.

30. S. Ren, K. He, R.B. Girshick, J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **39**(6): 1137–1149, 2017, doi: 10.1109/TPAMI.2016.2577031.

31. K. Simonyan, A. Zisserman, Two-stream convolutional networks for action recognition in videos, [in:] *NIPS'14: Proceedings of the 27th International Conference on Neural Information Processing Systems*, pp. 568–576, 2014.

32. K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv/1409.1556, 2015.

33. E. Spyrou, H. Le Borgne, T. Mailis, E. Cooke, Y. Avrithis, N. O'Connor, Fusing MPEG-7 visual descriptors for image classification, [in:] *ICANN 2005 – International Conference*

on Artificial Neural Networks, 11–15 September, Warsaw, Poland, 2005, doi: 10.1007/11550907_134.

34. C. Szegedy *et al.*, Going deeper with convolutions, [in:] *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015, doi: 10.1109/CVPR.2015.7298594.

35. K. Wang, P. Guo, F. Yu, L. Duan, Y. Wang, H. Du, Computational intelligence in astronomy: A survey, *International Journal of Computational Intelligence Systems*, **11**: 575–590, 2018, doi: 10.2991/ijcis.11.1.43.

36. M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, [in:] *ECCV 2014. Part I*, D. Fleet, T. Pajdle, B. Schiele, T. Tuytelaars [Eds.], LNCS 8689, pp. 818–833, 2014.

37. *Amateur Astrophotography Magazine*, Astro Publishing Ltd., https://web.archive.org/web/20230324100114/, accessed August 18, 2023.

38. Github: The TensorFlow Object Detection API, https://github.com/tensorflow/models/tree/master/research/object_detection, accessed February 17, 2021.

39. The STScI Digitized Sky Survey, Space Telescope Science Institute in Baltimore, Maryland, https://archive.stsci.edu/cgi-bin/dss_form?target=M8&resolver=SIMBAD, accessed February 15, 2021.

Received February 24, 2022; revised version October 20, 2022;
accepted June 19, 2023.