A&A 621, A103 (2019)
https://doi.org/10.1051/0004-6361/201833833

Astronomy & Astrophysics

Deriving star cluster parameters with convolutional neural networks

I. Age, mass, and size

J. Bialopetravičius[1,2], D. Narbutis[1,2], and V. Vansevičius[1,2]

[1] Vilnius University Observatory, Saulėtekio Av. 3, 10257 Vilnius, Lithuania
e-mail: jonas.bialopetravicius@ff.vu.lt
[2] Center for Physical Sciences and Technology, Saulėtekio Av. 3, 10257 Vilnius, Lithuania

Received 12 July 2018 / Accepted 14 November 2018

ABSTRACT

Context. Convolutional neural networks (CNNs) have been proven to perform fast classification and detection on natural images and have the potential to infer astrophysical parameters on the exponentially increasing amount of sky-survey imaging data. The inference pipeline can be trained either from real human-annotated data or simulated mock observations. Until now, star cluster analysis was based on integral or individual resolved stellar photometry. This limits the amount of information that can be extracted from cluster images.

Aims. We aim to develop a CNN-based algorithm capable of simultaneously deriving ages, masses, and sizes of star clusters directly from multi-band images. We also aim to demonstrate CNN capabilities on low-mass semi-resolved star clusters in a low-signal-to-noise-ratio regime.

Methods. A CNN was constructed based on the deep residual network (ResNet) architecture and trained on simulated images of star clusters with various ages, masses, and sizes. To provide realistic backgrounds, M 31 star fields taken from The Panchromatic *Hubble* Andromeda Treasury (PHAT) survey were added to the mock cluster images.

Results. The proposed CNN was verified on mock images of artificial clusters and has demonstrated high precision and no significant bias for clusters of ages $\lesssim$3 Gyr and masses between 250 and 4000 $M_{\odot}$. The pipeline is end-to-end, starting from input images all the way to the inferred parameters; no hand-coded steps have to be performed: estimates of parameters are provided by the neural network in one inferential step from raw images.

Key words. methods: data analysis – methods: statistical – techniques: image processing – galaxies: individual: M 31 – galaxies: star clusters: general

1. Introduction

In recent years, methods using convolutional neural networks (CNNs) have drastically improved various object-recognition tasks from natural images, such as object classification and detection. Examples of image recognition competitions where CNNs have recently driven progress include ILVSCR[1] (Russakovsky et al. 2015), PASCAL VOC[2] (Ren et al. 2017), and Microsoft COCO[3] (Lin et al. 2014). As of now, a lot of these tasks can be solved with better accuracy than that of a human (Russakovsky et al. 2015). These methods are trained wholly on data and forgo any manual feature engineering steps that were usually required for computer vision applications.

The uptake of deep-learning-based methods has also been accelerating in the field of astronomy. Most of the activity is in galaxy classification (Dieleman et al. 2015; Huertas-Company et al. 2018; Domínguez Sánchez et al. 2018), gravitational lensing (Petrillo et al. 2017; Lanusse et al. 2018; Pourrahmani et al. 2018), and transient detection (Cabrera-Vives et al. 2017; Lanusse et al. 2018; Sedaghat & Mahabal 2018). There has also been work on astronomical image reconstruction (Flamary 2016), exoplanet identification (Shallue & Vanderburg 2018), point spread function (PSF) modeling (Herbel et al. 2018), and other topics. However, star cluster parameter estimation has not yet been attempted with these methods, even though this field is an ideal candidate since accurate inference from imaging data is sorely needed.

One of the benefits of CNN architectures is that parameter inference tasks can be solved in much the same way as object type classification. This opens up the possibility to perform astrophysical parameter inference from star cluster images both accurately and efficiently, utilizing all of the available information in each pixel of an image. However, the method first has to be trained on either human-annotated or simulated mock observations.

The Panchromatic *Hubble* Andromeda Treasury (PHAT; Dalcanton et al. 2012) provides high-quality multi-band imaging data[4] along with human-annotated star cluster catalogs (Johnson et al. 2012, 2015). These catalogs have previously been analyzed by Fouesneau et al. (2014) and de Meulenaer et al. (2017), who provided cluster age and mass estimates using integrated photometry. Based on resolved star photometry,

[1] ImageNet Large Scale Visual Recognition Competition.
[2] The PASCAL (Pattern Analysis, Statistical Modelling and Computational Learning) Visual Object Classes.
[3] Common Objects in Context.
[4] https://archive.stsci.edu/prepds/phat/

Johnson et al. (2017) derived cluster ages and masses. Previously, Caldwell et al. (2009) had derived cluster parameters of massive clusters in M 31 based on spectroscopy. Sizes were estimated from images using aperture photometry by Johnson et al. (2015). Therefore, the PHAT dataset provides a good basis for testing CNN-based methods.

In this paper, we propose a CNN architecture to estimate the ages, masses, and sizes of star clusters. The network is trained on realistic mock observations, with backgrounds taken from the PHAT survey. The method is tested on a different set of artificial clusters. The paper is organized as follows: Sect. 2 provides details about the PHAT survey data, Sect. 3 gives a summary of used deep-learning-based methods, Sect. 4 describes the proposed CNN and its training methodology, and Sect. 5 presents the results of testing the method on artificial clusters.

2. Data

The PHAT survey data is extensively described by Dalcanton et al. (2012). Here we use stacked, defect-free mosaic ("brick") images, which are photometrically (pixel values are in counts per second) and astrometrically (with available world coordinate system information) calibrated. Multi-band images for four bricks (19, 20, 21, 22) were obtained from the PHAT archive[5]. The bricks were chosen to be located in the outer part of the M 31 disk, and have low stellar background contamination. Although the PHAT survey was conducted in six passbands, only three (F336W, F475W, F814W) were used in this study because of their high signal-to-noise ratio (S/N). The frames in these passbands were observed with The Wide Field Camera 3 (WFC3) and The Advanced Camera for Surveys (ACS) instruments, and have different PSFs as well as exposure times.

Each passband image of a brick was transformed into a tangential projection with a common scale (0.05 arcsec pixel^{-1}) and size, oriented in such a way that the pixel grids of the images are aligned with the north and east directions. First, the world coordinate systems of images were aligned and then pixel values were transformed using the reproject[6] package from Astropy[7], conserving flux. Bicubic interpolation was used, providing appropriately resampled images for CNN training purposes.

The brick images contain a lot of saturated stars and extended objects, which could introduce bias to the CNN training procedure. To deal with this, we masked out real clusters and galaxies listed in catalogs by Johnson et al. (2012, 2015) as well as stars brighter than 18 mag in the G passband of the *Gaia* catalog (Gaia Collaboration 2016). These regions were omitted when generating backgrounds for artificial clusters.

To generate mock clusters of various ages, masses, and sizes, PARSEC isochrones[8], release 1.2S (Bressan et al. 2012), were used to sample stellar masses according to the Kroupa (2001) initial mass function. A fixed metallicity of $Z = 0.009$ and no interstellar extinction was assumed. From the isochrones the absolute magnitudes were taken, stars were placed at the same distance as M 31 (McConnachie et al. 2005, 785 kpc) and their fluxes were transformed to *Hubble* Space Telescope (HST) counts per second using HST calibrations for ACS (Avila 2017) and WFC3 (Dressel 2012) cameras.

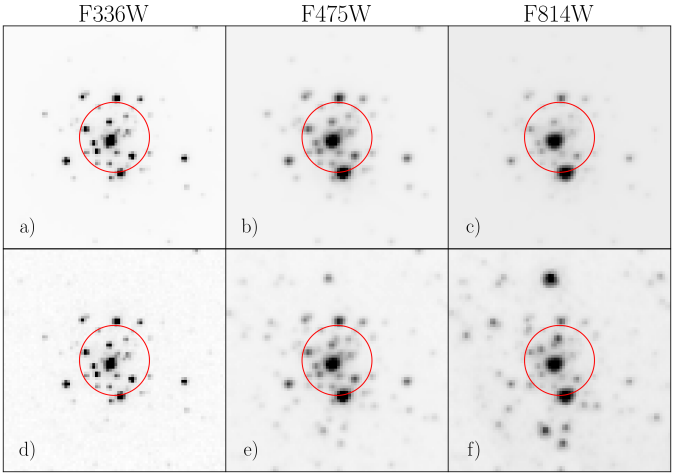

Fig. 1. An example of a mock cluster drawn with GalSim, without (*top panels*) and with a random background from PHAT (*bottom panels*). Passbands are indicated at the top of each panel. A cluster with typical parameters of $\log_{10}(t/\mathrm{yr}) = 8.0$, $\log_{10}(M/M_\odot) = 3.0$, and $r_h = 0.6$ arcsec is shown. The red circle of r_h radius encloses half of the stars of each cluster. The intensity of the images is normalized with the arcsinh function.

The spatial distribution of stars was sampled from the Elson–Fall–Freeman (EFF) model (Elson et al. 1987), placing the cluster in the center of the image.

TinyTim[9] PSFs (Krist et al. 2011) were used to draw the individual stars of the clusters. The PSFs used were 6×6 arcsec in size and drawn with the GalSim package (Rowe et al. 2015) in the coordinate system of aligned PHAT bricks. Artificial clusters were then placed on backgrounds taken from the transformed PHAT bricks. An example of a mock cluster both with and without a background is shown in Fig. 1.

3. Deep learning

3.1. Artificial neural networks

For a comprehensive review of artificial neural networks (ANNs) and the algorithms for their training, we refer the reader to Haykin (2009). Here, a short summary is presented as a basis for the methods used in this study.

The convolutional neural network used in this paper is a type of ANN, which is a machine-learning model comprising multiple artificial neurons, each taking a vector of inputs and producing a single output. The vector of inputs is multiplied by a vector of weights, summed, and then passed through a nonlinearity known as the activation function. One of the most common activation functions is the rectified linear unit (ReLU), defined as $f(x) = \max(0, x)$ (Nair & Hinton 2010).

An ANN can be trained using the gradient descent algorithm. This requires training samples, which consist of inputs to the network (or observations), and the expected outputs (targets). A loss function $\mathcal{L}$ is used to determine how well the outputs of the network match the expected outputs in the training set. The gradient of this loss function is computed for each sample with respect to the weights of the neural network, and then the weights are adjusted according to the gradient.

[5] https://archive.stsci.edu/prepds/phat/
[6] https://reproject.readthedocs.io/
[7] http://www.astropy.org/
[8] http://stev.oapd.inaf.it/cgi-bin/cmd
[9] http://tinytim.stsci.edu/cgi-bin/tinytimweb.cgi

In essence, the weights of the network w_{t+1} at iteration $t+1$ are updated from the weights w_t at iteration t by the formula $w_{t+1} = w_t - \eta\nabla\mathcal{L}$, where η is called the learning rate and is used to control the speed of the learning process. A high learning rate results in quick learning, but the model converges poorly due to large parameter steps near optima; meanwhile, small learning rates have the opposite effect. It is important to control this variable during training so that the training procedure converges.

In the most common form of an ANN, neurons are arranged in multiple layers, where each layer has some number of neurons taking inputs from the previous layer and producing activations for the next layer to process. Such an arrangement is called a feed-forward network. The algorithm used for optimizing such networks is called backpropagation, which provides a convenient way to calculate gradients at each layer using the chain rule. With this algorithm, the inputs are passed through the network, obtaining outputs. Gradients are then calculated and propagated backwards, adjusting the weights layer by layer so that the training error (loss) is minimized.

Since each neuron performs a nonlinear mapping of its inputs, stacking many layers of neurons results in a deeply nonlinear model, which is beneficial in many real-world tasks. However, backpropagation-based algorithms often cause gradients to vanish, and learning to stall in the lower layers, or gradients to explode, and learning to diverge. Usually, these problems are solved by either minimizing the number of optimizable parameters with more restrictive neural network architectures, or restricting the possible values of the weights of the network obtained during training.

3.2. Convolutional neural networks

A CNN is one way to regularize the weights of an ANN via architectural constraints. In a regular ANN, each neuron takes input from every neuron in the lower layers. Meanwhile, in a CNN each layer consists of learnable convolutional filters with a small number of parameters that exploit the 2D structure of images. Each convolutional filter may be as small as a 3×3 matrix, resulting in nine optimizable parameters. The filter is applied many times to its input by moving it by a step, called a stride, and a lot of outputs for the next layer are produced. Training such models results in increasingly more abstract feature detectors in the form of convolutional filters. The lower layers look for simple features, such as corners or edges of objects. Deeper layers combine these features to form more abstract concepts about objects present in an image, while discarding irrelevant information and noise. It seems that this hierarchical pipeline of feature extraction is essential to solving computer vision problems, as the neocortex of animals is known to work in a similar way (Kruger et al. 2013).

In classical computer vision, hierarchy was usually implemented by hand. Simple features were first detected and extracted, then combined into more complex aggregates that can describe whole objects, and later processed with simple machine learning algorithms. These hand-engineered approaches, however, had relatively low performance as real world data have a lot of noise, irregularities, irrelevant correlations and a high level of variation. Meanwhile, CNNs learn the necessary regularities from the data itself without any feature engineering. This is also why CNNs are such a favorable algorithm to apply to star clusters.

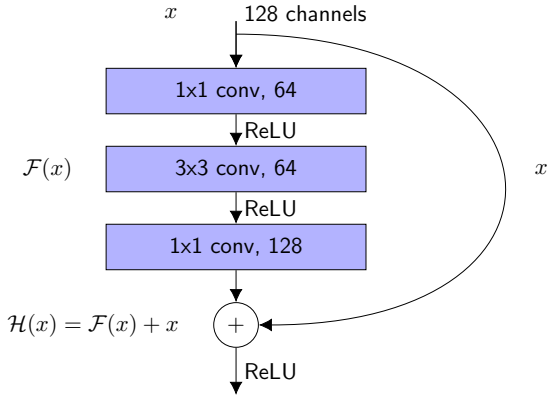

Fig. 2. Example building block of a ResNet, consisting of three sequential convolutional layers. The input is a 128-channel activation map, which is passed through 64 1×1 convolutional filters. The filters extract a 64-channel feature map. These features, after applying a ReLU activation, are then passed through 64 3×3 convolutional filters. The purpose of the first layer is to compress the channels for the 3×3 convolutional layer, which results in less optimizable parameters. Then, the ReLU activation is applied again and the final 1×1 convolutional layer expands the number of channels back to 128. Finally, these outputs are summed with the inputs via a skip-connection and passed through a ReLU activation.

3.3. Residual networks

During the past few years, many variants of CNN architecture have been proposed. It has become quite common in the literature to reuse these standardized architectures in order to have a common ground of reference for measuring improvements.

One such architecture is called the ResNet (He et al. 2016). Convolutional layers in a regular CNN learn feature detectors, taking inputs from lower layers and mapping them to a representation, called a feature map. The idea of a ResNet is that instead of learning to produce a simple feature map, a residual mapping on top of the features of the previous layer is learned. Denoting the inputs of a layer as x, we are looking for a mapping $\mathcal{H}(x)$, such that $\mathcal{F}(x) = \mathcal{H}(x) - x$ (the residual function). This gives the desired mapping as $\mathcal{H}(x) = \mathcal{F}(x) + x$. This has a convenient implementation for CNNs as a skip-connection, which consists of simply taking the activations of a layer and adding them to a deeper layer in the graph of the network (see Fig. 2).

The ResNet architecture allowed He et al. (2016) to achieve state-of-the-art results on a few standard image recognition datasets with a lower number of network weights than the alternatives at the time. The training of ResNet-type networks is stable regardless of network depth since good results are achieved with networks as shallow as 20 layers (ResNet-20) and as deep as 1202 layers (ResNet-1202). Such networks also seem to work well on object classification as well as detection and regression tasks. Therefore, they are ideal candidates for application in clusters.

4. Methods

4.1. CNN architecture

In this paper, the ResNet-50 version was used as a basis for the constructed CNN. The network was adapted from the variant used for the ImageNet dataset (Russakovsky et al. 2015). The usual inputs of this network are natural photographic RGB images (3 channels) with 224×224 pixels, which get compressed very quickly in the lower layers to narrow feature

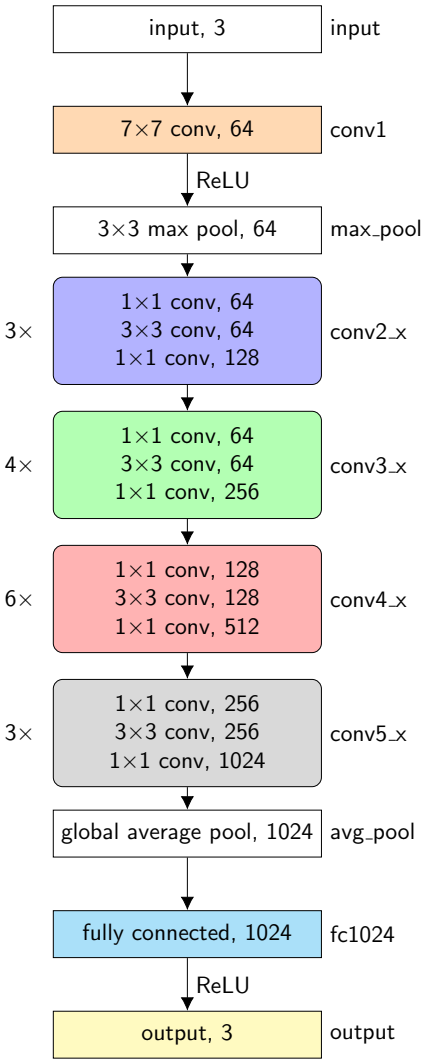

Fig. 3. A block diagram of the used CNN. The input image of a cluster passes through the network top to bottom, with the output result being age, mass, and size. All blocks with sharp corners depict singular layers, while blocks with rounded corners are groupings of layers (see example in Fig. 2), with the number on the left indicating how many times the group is repeated sequentially and the name on the right corresponding to the layer names in Table 1. The blocks in color are parts of the network with optimizable parameters. The last number in each row is the number of output channels from that layer.

mappings because of the large stride by which the convolutional kernels are moved. Meanwhile, our inputs are 80×80 pixels in size (3 channels: F336W, F475W, F814W), but we operate in a low-S/N regime. Therefore, we reduced the stride of the earliest convolution operations, which should allow the network to extract more low-level information. The network was implemented with the TensorFlow package[10] and is depicted in Fig. 3 and Table 1.

Considering that star clusters have rotational symmetry, rotating an image should not affect what the network infers about it. One way to alleviate this problem is with a method proposed by Dieleman et al. (2015): The original image of a cluster is rotated by 90° three times and passed through the same convolutional layers. The resulting layer activations are then averaged before the output of the network (second block from the bottom in Fig. 3). This enforces the idea that a rotated image of a cluster should result in the same high-level representations in the activations of the network, as the parameters do not depend on the rotation angle. The outputs of the network were represented as a separate neuron for each parameter (age, mass, and size).

[10] https://www.tensorflow.org/

Table 1. Designed 50-layer CNN, based on the ResNet architecture.

Layer	Output		Operation
	Size	Channels	
input	80×80	3	
conv1	80×80	64	7×7
max_pool	40×40	64	3×3 max pool/2
conv2_x	40×40	128	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 128 \end{bmatrix} \times 3$
conv3_x	20×20	256	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv4_x	10×10	512	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 3$
conv5_x	5×5	1024	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 3$
avg_pool	1×1	1024	global average pool
fc1024	1×1	1024	fully connected
output	1×1	3	linear

Notes. The layers of the network are listed top to bottom, starting from the images of clusters and with the final layer producing the age, mass, and size of the cluster. The convolutional layers are actually groups of blocks depicted in Fig. 2, with the "_x" in the name acting as a placeholder for the block number. The size of the outputs of each layer, both in spatial dimensions and in channel count, are listed on the second and third columns. The last column lists the operations that each layer performs. The layers or blocks with a stride of 2 are: max_pool, conv3_1, conv4_1, and conv5_1; as can be seen when input and output sizes differ by 2.

4.2. CNN training

The training procedure consists of propagating the images through multiple layers of convolutions until the outputs (in our case – the three cluster parameters) are obtained. These outputs are directly determined by the structure and weights of the network. The structure is fixed beforehand; however the weights start out random and need to be optimized in order to construct an accurate inferential model, mapping images of clusters to their parameters.

At each iteration of the training procedure, a batch of 64 clusters (which includes all 3 passband images and their 4 rotated variants) is taken and propagated through the network. The inferred output parameters are then compared to the expected outputs and all the weights are updated according to a loss function:

$$\mathcal{L}(\text{targets}, \text{outputs}) = \sum_{i=1}^{3} \text{smooth}_{\mathcal{L}_1}(\text{targets}_i - \text{outputs}_i), \quad (1)$$

where targets refers to the true values of the parameters of a given cluster, while outputs refers to the parameters given by the network; i indicates parameter number (age, mass, and size).

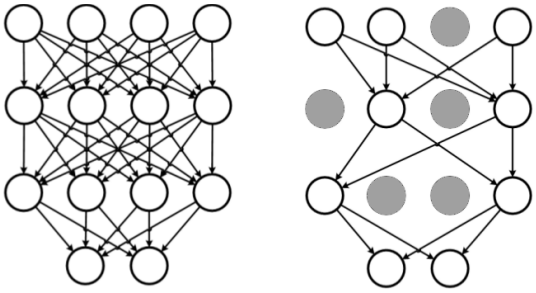

Fig. 4. Illustration of the dropout method (Hinton et al. 2012). The *left panel* shows a fully connected feed-forward neural network, while the *right panel* shows the same network with some of the neurons disconnected by dropout.

To compare these values and derive the training gradient for each of the parameters, the following function was used:

$$\mathrm{smooth}_{\mathcal{L}_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1, \\ |x| - 0.5 & \text{otherwise,} \end{cases} \quad (2)$$

which is a robust Manhattan distance-based loss function (Girshick 2015); it is more resilient to large prediction outliers than the more commonly used mean-squared-error loss. The Adam optimizer (Kingma & Ba 2014) was used to compute gradients for weight updates of the network.

The constructed CNN has approximately 7 million parameters. Even with a large number of training samples, optimizing this many parameters is problematic. Memorizing the images or unimportant peculiarities in the training data can become much easier than learning actual generalized tendencies that produce the output parameters we seek, resulting in overfitting. To combat this, we used the dropout method (Hinton et al. 2012), which discards neurons randomly from the network during training, at a probability of p_{dropout}. An example of this can be seen in Fig. 4. This reduces overfitting by preventing neurons from adapting to each other, in effect making the network more robust to missing outputs from neurons in other layers. This was only done during training with $p_{\mathrm{dropout}} = 0.5$, in the last fully connected layer seen in Fig. 3 (second block from the bottom). For inference, the fully trained network with all of the neurons is used.

4.3. Artificial clusters

The age of each cluster was sampled from the logarithmic range of $\log(t/\mathrm{yr}) = [6.6, 9.5]$ (with a step of 0.05 dex); mass was sampled from the logarithmic range of $\log(M/M_\odot) = [2.0, 4.0]$ to cover the majority of genuine young, low-mass M 31 clusters studied by de Meulenaer et al. (2017). The star count surface density radial profile $\mu(r)$ of the cluster was sampled from the EFF (Elson et al. 1987) model:

$$\mu(r) = \mu_0(1 + r^2/a^2)^{-\gamma/2}. \quad (3)$$

The parameters $a = [0.05, 6.4]$ arcsec and $\gamma = [2.05, 8.0]$ were sampled in a logarithmic space within the curves defined by constant r_h values (between 0.1 and 1.6 arcsec), as shown in Fig. 5. We define r_h as the radius of a circle on the sky enclosing half of the stars of a cluster. These values at the assumed M 31 distance (785 kpc) roughly correspond to real cluster sizes in M 31 (Vansevičius et al. 2009; Narbutis et al. 2014). The setup was chosen to target low-mass semi-resolved star clusters in order to demonstrate the capabilities of the CNN in low-S/N conditions.

Samples of artificial clusters were generated with these parameters and combined with backgrounds of M 31 stars. The process is as follows. A random image of a background is selected and its median value is determined. This median is then added to the image of an artificial cluster and then each pixel is sampled from a Poisson distribution, with its mean set to the value of the pixel. The median is then subtracted back from this image and the real background image is added. The counts per second of the images were then transformed with the logarithmic function $\log(x + 1)$. The resulting images were 80×80 pixels in size, which correspond to 4×4 arcsec, or 15×15 pc at the distance of M 31 (785 kpc). Examples of the generated clusters with different ages, masses, and sizes, covering most of the parameter space, are shown in Fig. 6.

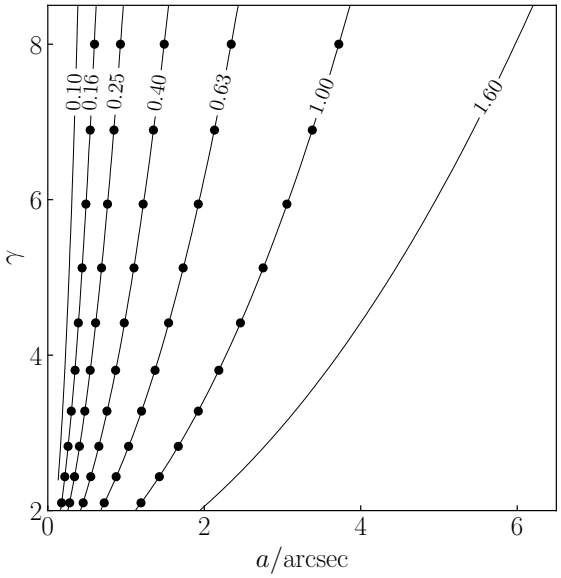

Fig. 5. Used EFF parameter space and corresponding lines of constant r_h. Clusters for CNN training were sampled from lines, while dots correspond to the values of r_h used to construct a grid of the artificial test samples.

We generated 200 000 images of artificial clusters as a training sample for our CNN. A batch size of 64 images per training step was used. To ensure that the training procedure of our CNN would converge, we experimented with various learning rates, starting from $\eta = 0.1$ down to $\eta = 0.0001$. The learning rate of $\eta = 0.01$ gave the best performance on the validation set, so this was the value used for the final training of the network. A few different learning rate schedules were also tested. Decreasing the learning rate twofold after every pass over the data gave the best results in our case. In order to control for overfitting, we trained the network for ten passes over the data while continuously monitoring its performance. The results on the validation set remained stable after about three passes, therefore we chose to stop after five passes over the training data.

In addition, to test whether our training sample of 200 000 clusters was sufficient for the network to generalize, we also trained our network with as little as 50 000 and as many as 400 000 cluster samples. During these experiments we did not observe a significant difference between the inference results when trained with smaller or larger sample sizes.

5. Results and discussion

The data sets for testing the CNN were prepared by drawing from the same age and mass ranges as described in Sect. 4.3. However, for ease of analysis, the EFF model parameter γ and

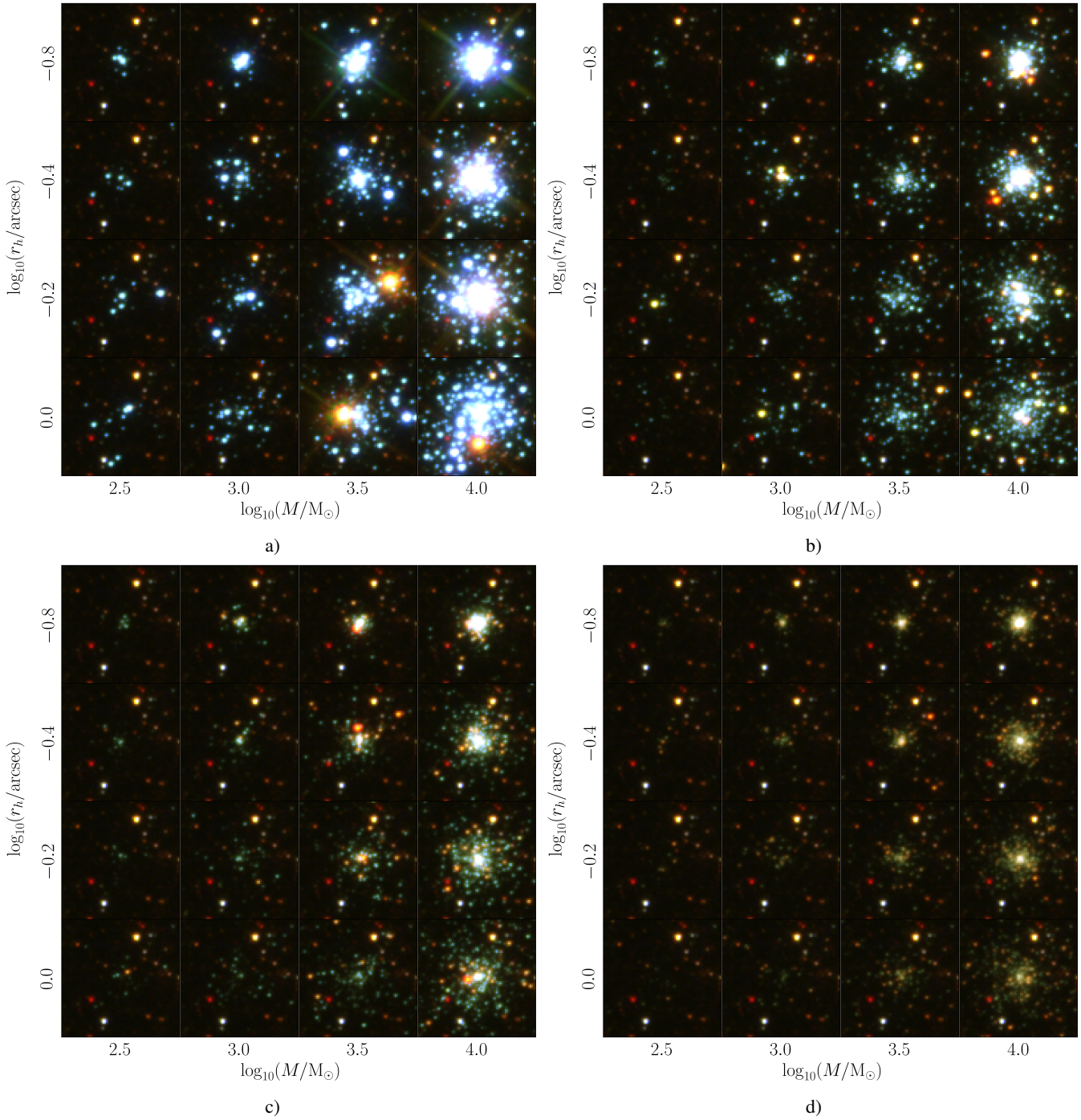

Fig. 6. Examples of generated clusters on a real background image. The ages of all of the displayed clusters are: $\log(t/\mathrm{yr}) = 7.0$ (*panel a*), 8.0 (*panel b*), 8.5 (*panel c*), 9.0 (*panel d*). The masses and r_h values are varied as shown on the axes. The intensity scale of the images was normalized with the arcsinh function. The background is the same for all of the displayed clusters for clarity.

a pairs, were chosen in such a way that the value of r_h would be equal to one of the following values: 0.16, 0.25, 0.4, 0.63, 1.0 arcsec (see Fig. 5). The procedure of image creation was also identical to that of the training data. The backgrounds for clusters were picked making sure that they would not overlap with the backgrounds used for the training set.

5.1. Parameter accuracy

To test the performance of the CNN, we built a bank of 10 000 artificial clusters by varying all three cluster parameters. The ages were sampled from the range of $\log(t/\mathrm{yr}) = [6.75, 9.25]$; mass was sampled from the range of $\log(M/M_\odot) = [2.25, 3.75]$ and r_h, as shown in Fig. 5. Figure 7 shows the differences between true and CNN-derived parameters plotted against the true parameter (age, mass, and r_h) values of the artificial clusters of the testing set.

Figure 7a shows no significant age difference between the true and derived values. The typical standard deviation of the age difference is estimated to be ∼0.1 dex. The youngest clusters show a slight systematic bias towards older ages, as can be seen by the nonsymmetric whiskers of the box plots. This is also

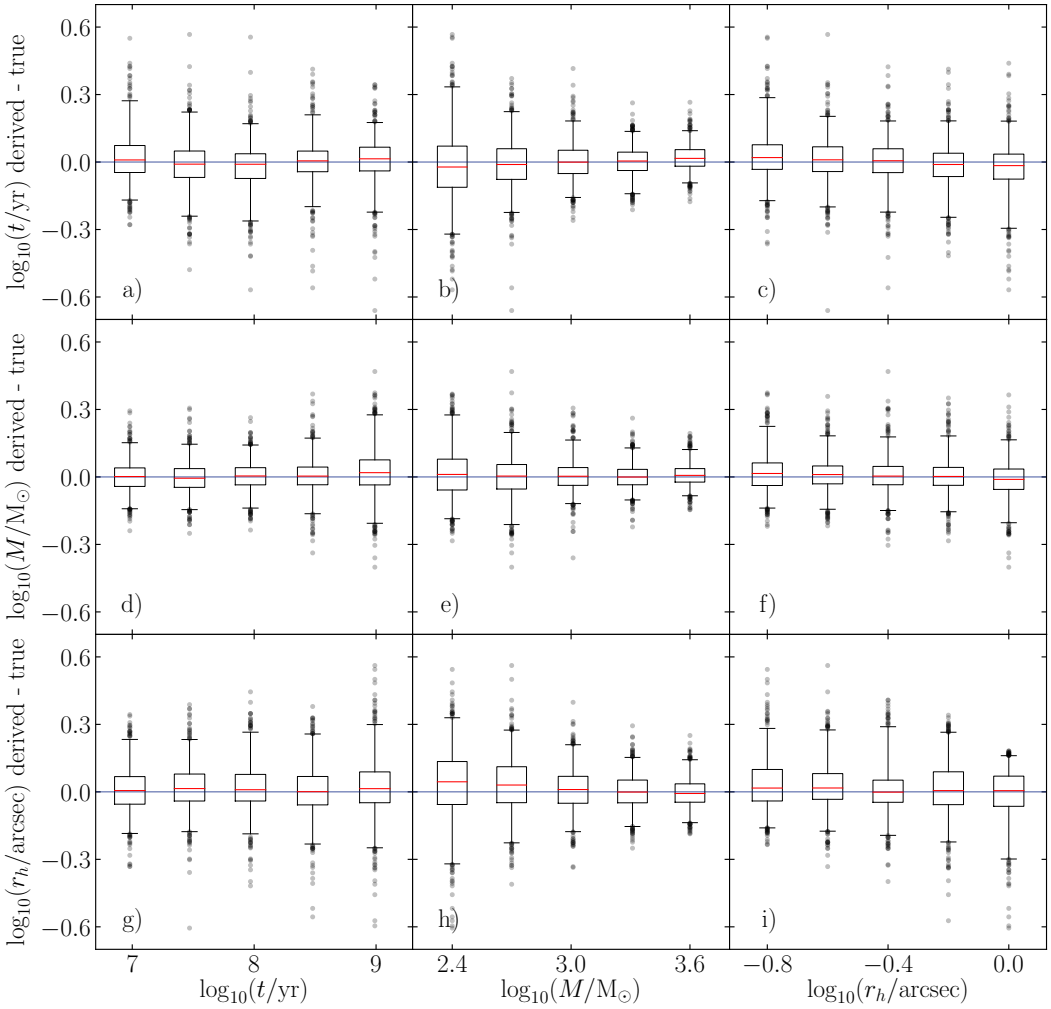

Fig. 7. Differences between CNN-derived and true parameters, plotted against true parameter (age, mass, and size) values of artificial clusters. The true ages and true masses are put into bins of 0.5 and 0.3 dex in width, respectively. The r_h bins correspond to EFF model parameters indicated by dots on the constant r_h values in Fig. 5. The widths of the boxes for age and mass correspond to half the widths of the bins. The spread of the parameter differences are displayed as box plots. The line in the middle of each box is the median error. Boxes extend from the first to the third quartiles. The whiskers denote the range between the 2th and 98th percentiles. Anything above and below the whiskers is plotted as separate points.

true for the oldest clusters in the test sample; only the bias is reversed. This could be explained as an age boundary effect due to the stellar isochrone age range limits used for the CNN training. Figure 7b shows the true and derived age value differences plotted against the true cluster masses. The median differences have no systematic shifts. However, the least massive clusters show a larger spread in mass error, with standard deviation of the differences as high as ∼0.15 dex. The errors get systematically lower as we move towards the higher masses, stabilizing at a standard deviation of ∼0.05 dex for the most massive clusters. This could be explained by the low S/N of $\log_{10}(M/M_\odot) = 2.4$ clusters, as can be seen from Fig. 6. No systematic errors are observed in Fig. 7c, with the standard deviations of the errors equal to ∼0.1 dex across the whole range of the true r_h values.

Figure 7d shows no systematic errors when deriving cluster age as cluster mass varies. However, the spread of errors does get larger towards older clusters. In Fig. 7e there are no systematic shifts; however errors are again larger for the lowest-mass clusters. This could also be explained by the very low S/N of $\log_{10}(M/M_\odot) = 2.4$ clusters. In Fig. 7f the derived mass values do not seem to vary with changing r_h values.

Figure 7g shows consistently derived r_h values with changing ages. Figure 7h shows systematically larger derived r_h values for low-mass clusters, as well as clearly larger errors. Figure 7i shows a clear systematic trend deriving larger r_h values for small clusters, as well as lower values for large clusters. The shift in large clusters, with r_h as high as 1.0 arcsec, could be explained by the fact that a significant portion of the stars of a cluster do not fit into the 4×4 arcsec images, the size of which is restricted by the chosen network architecture to minimize the background area for the majority of clusters.

5.2. Star cluster stochastic effects

In order to analyze the effects of star cluster stochasticity (initial mass function and star position sampling) on the inference results of the CNN, we built a grid of clusters (200 per node) with fixed age, mass, and size parameters. The only random effects were background, star position, and mass sampling.

Figure 8 shows three panels with different ages: $\log_{10}(t/\mathrm{yr}) =$ 7.0, 8.0 and 9.0. Each black dot corresponds to a node of 200 clusters of fixed true mass and r_h, as denoted on the axes. Ellipses

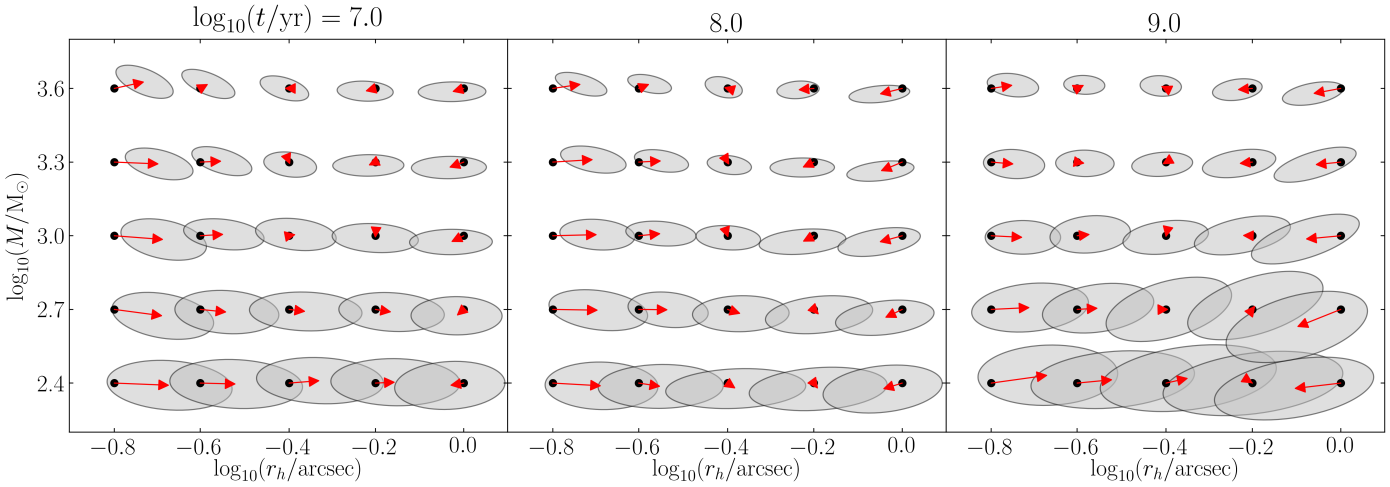

Fig. 8. Test results of CNN performance. Each black dot corresponds to the true parameters of 200 artificial clusters. The gray ellipses enclose one σ of the inferred values (accuracy), with the red arrows pointing to the means of the distributions (biases). The panels show mass vs. r_h for three different ages.

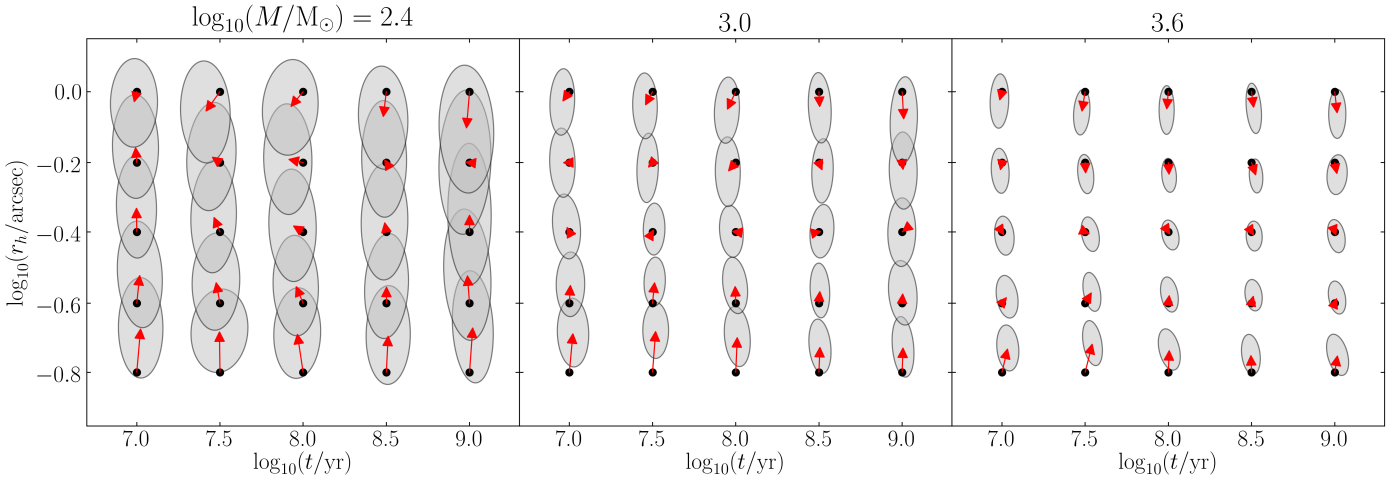

Fig. 9. As in Fig. 8, but panels show r_h vs. age for three different masses.

show the spread of the derived parameter values of these nodes. The arrows show the shift of the mean of the values and the boundary of the ellipses enclose one σ of the 2D distribution (~39% of the clusters). The sizes of the ellipses at low masses are approximately three times larger than at high masses, especially at the oldest ages. This is because of the low S/N of the old low-mass clusters, as can be seen in Fig. 6d. Sizes tend to be slightly overestimated for low-mass clusters. No significant correlated systematic shifts between the biases (red arrows) of derived mass and size can be seen.

Figure 9 shows three panels with different masses: $\log_{10}(M/M_\odot) = 2.4$, 3.0, 3.6. Black dots mark nodes of age and size as denoted on the axes. The spread of the derived age and size values is largest for the lower-mass clusters, being at least three times larger for $\log_{10}(M/M_\odot) = 2.4$ than for $\log_{10}(M/M_\odot) = 3.6$. However, at fixed mass, there is no correlated systematic shift between the derived mass and size biases of the cluster. The slight overestimation of sizes can be seen again for the lowest-mass clusters.

Figure 10 shows three panels with different sizes: $\log_{10}(r_h/\mathrm{arcsec}) = -0.8, -0.4, 0.0$. Black dots mark nodes of age and mass as denoted on the axes. As seen in the previous figures, the spread of derived values tends to increase towards the lower-mass clusters. However, there is no obvious influence of cluster size on the spread. A slight elongation of ellipses along the diagonal can be observed between the derived mass and age for the older grid clusters. This is because of the age-mass degeneracy, as the CNN has learned that older clusters have lower flux, and lower flux can be explained either by an older age or a lower mass. While for $\log_{10}(t/\mathrm{yr}) \lesssim 8.0$ (as seen in Figs. 6a,b) stochastic effects are dominant, for $\log_{10}(t/\mathrm{yr}) > 8.0$ (Figs. 6c,d), flux becomes the main factor in the mass-age correlation.

5.3. Overall CNN performance

The proposed CNN was verified on mock images of artificial clusters. It has demonstrated high precision and no significant bias for semi-resolved clusters with ages between $\log_{10}(t/\mathrm{yr}) = 7.0$ and 9.0, and masses between $\log_{10}(M/M_\odot) = 2.4$ and 3.6. Artificial cluster tests have demonstrated the effectiveness of CNNs in deriving star cluster parameters.

Even in the low-mass regime it is possible to recover both age and mass as seen in Fig. 10. However, the plots show only

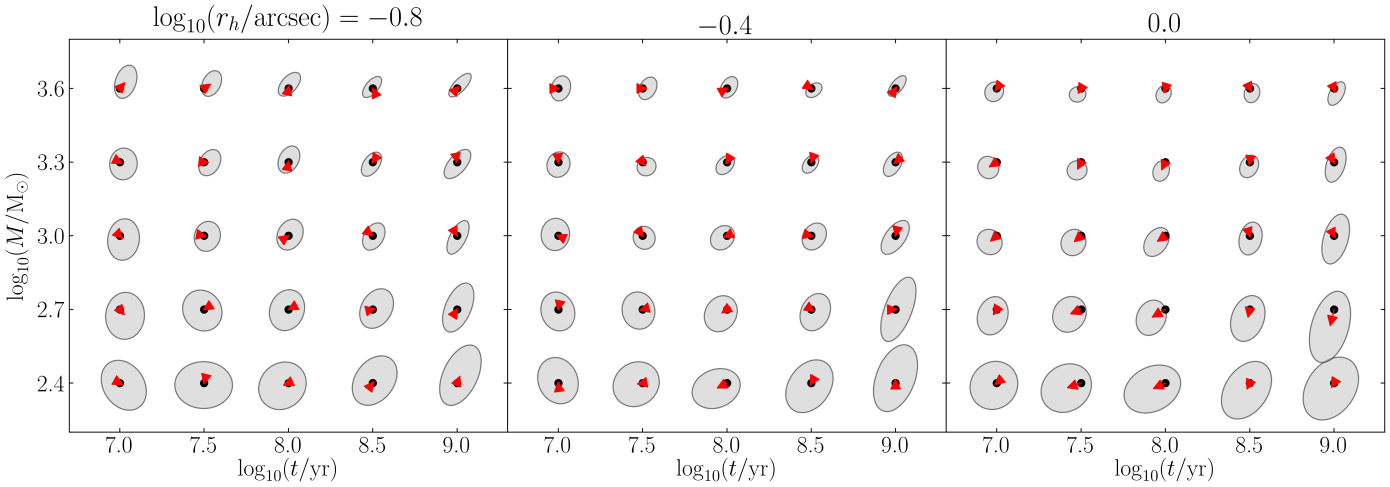

Fig. 10. As in Fig. 8, but panels show mass vs. age for three different r_h values.

one σ (~39%) of the 2D distributions, which means that for the lowest mass clusters, five age groups can be identified. As mass increases, the number of identifiable groups increases. The same trend is also seen in Fig. 7b and holds regardless of cluster size (see Figs. 7c,f). We note that even though this is true on average, there are still outliers with a parameter derivation error as large as 0.6 dex for the low-mass clusters (Fig. 7b).

The low-mass clusters shown in Fig. 6d are not visible to the naked eye. Nevertheless, the network still learns on the limited amount of signal and manages to produce valid parameter estimates. Even though the old-age lowest-mass clusters have been included, in real surveys they would be omitted because they would fall below the detection limit.

Meanwhile, for the cluster size, only three categories can be identified in the lowest mass regime as can be seen in Fig. 8. This is however unsurprising as the S/N of these clusters is low (see Fig. 6). This is especially true for the oldest clusters. However, higher masses allow us to identify up to five size groups; Fig. 7h also confirms this finding.

Currently, evolutionary and structural parameters of semi-resolved clusters are estimated separately and in two completely different ways. Usually, for age and mass estimates a grid of models is constructed by varying the parameters of interest and comparing the resulting mock observations to real observations (Bridžius et al. 2008; de Meulenaer et al. 2013). Meanwhile, a common approach for structural parameters is Markov chain Monte Carlo sampling of the parameter space (Narbutis et al. 2015). Both of these methods essentially sift through a large number of possible observations to get a likely parameter estimate or the distribution over the parameters.

Convolutional neural networks, while still requiring a sample of mock clusters for their training, are able to learn the properties of the system they are modeling within their weights and generalize more effectively to new examples. This also allows the derivation of both evolutionary and structural parameters in a unified way, using all of the available information in image pixels.

The information registered in image pixels is affected by many additional factors, which were not included in the CNN training; the PSF shape is expected to be the most significant among them for cluster size. In this paper, a fixed PSF simulated for the center of the detector was used. This was done for both the training cluster images and the test sets. In order to find the effects of variable PSFs on the performance of the network, tests were done with the PSF taken from a corner of the detector; however the effects of this were negligible. In order to test the robustness of the network to different views of the cluster, we also tried running the CNN on shifted and rotated cluster images. This however proved to have a negligible impact on the uncertainty of derived values.

Extinction and metallicity are also significant factors in age and mass derivation (de Meulenaer et al. 2014, 2015). Since for the first time we attempt to derive both evolutionary and structural parameters, we assumed zero extinction and fixed metallicity just to demonstrate CNN performance. At fixed extinction and metallicity the CNN demonstrates good accuracy for age and mass derivation. This implies that the CNN can extract more information than integrated photometry due to its ability to learn the appearance of the cluster – not only its integrated flux.

In comparison to methods where any sort of fitting or sampling procedure needs to be performed, a CNN can obtain parameter inference results from an image very efficiently. All experiments in this study were performed using a GeForce GTX 1080 graphics card. The initial training procedure on the 200 000 clusters takes ~6 h; however this only needs to be run once. Inference runs much faster, taking 40 seconds per 10 000 images of all three passbands.

5.4. Tests on real clusters

To validate our method on real star clusters we took the sample of clusters used by de Meulenaer et al. (2017). From those clusters we selected only objects located on PHAT bricks 19–22. The CNN in our study was trained ignoring extinction entirely, therefore it is not possible to correctly derive the parameters of significantly reddened clusters. However, extinction estimates for the chosen clusters have been published by de Meulenaer et al. (2017). In order to run our experiments, we corrected the cluster images in each passband for the extinction effect by increasing their corresponding flux.

We then used our CNN to infer the cluster age and mass. Figure 11 shows a comparison between our results and those of de Meulenaer et al. (2017). A good agreement between the derived values can be seen. However, this result can only be used for a preliminary validation of the method. By increasing the flux of our real cluster sample to remove the effects of extinction, we change the flux of not only the cluster itself, but also of its background. On the other hand, the artificial clusters used in this

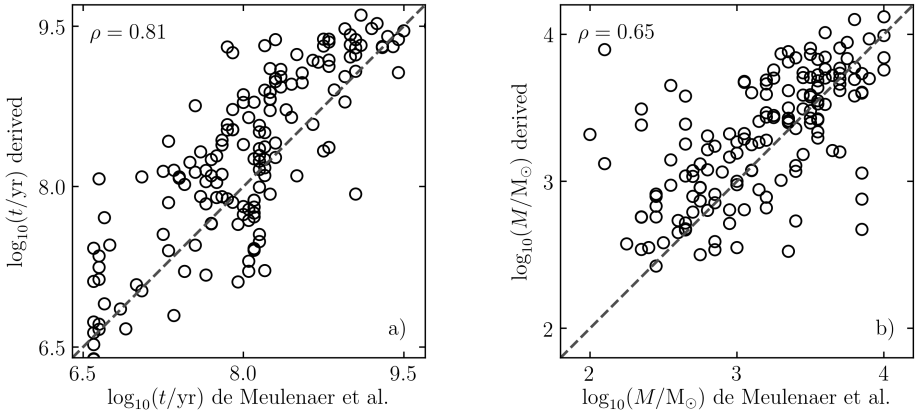

Fig. 11. Comparison of ages (*panel a*) and masses (*panel b*) derived by de Meulenaer et al. (2017) and our CNN for 157 real PHAT clusters located in bricks 19–22. The correlation coefficient ρ between values is displayed in the top-left corner of each panel.

study, while themselves drawn without extinction, are superimposed on real backgrounds which are affected by extinction. This makes any results on real clusters unreliable, introducing possible degeneracies. However, the preliminary results are promising and show a clear applicability to real clusters.

In order to deal with real clusters correctly, our CNN needs to be trained on images with various extinction levels, as well as to be able to predict extinction in the same way it currently predicts age, mass, and size. However, for a reliable derivation of extinction more photometrics filters are required. This work falls out of scope of this study, and is planned for the next paper in the series.

6. Conclusions

We propose a convolutional neural network based on the ResNet architecture for simultaneous derivation of both evolutionary and structural parameters of star clusters from imaging data. Artificial cluster images were combined with real M 31 backgrounds observed with the HST and used for training the neural network.

The proposed CNN was verified on mock images of artificial clusters. It has demonstrated high accuracy and no significant bias for semi-resolved clusters with ages between $\log_{10}(t/\mathrm{yr}) = 7.0$ and 9.0, masses between $\log_{10}(M/M_\odot) = 2.4$ and 3.6, and sizes between $\log_{10}(r_h/\mathrm{arcsec}) = -0.8$ and 0.0.

We have shown with artificial tests that CNNs can perform both structural and evolutionary star cluster parameter derivation directly from raw imaging data. This allows both unresolved and semi-resolved cases to be dealt with homogeneously, and multiple photometric passbands to be used in an integrated manner.

Acknowledgements. This research was funded by a grant (No. LAT-09/2016) from the Research Council of Lithuania. This research made use of Astropy, a community-developed core Python package for astronomy (Astropy Collaboration 2018). Some of the data presented in this paper were obtained from the Mikulski Archive for Space Telescopes (MAST). STScI is operated by the Association of Universities for Research in Astronomy, Inc., under NASA contract NAS5-26555. We are thankful to the anonymous referee who helped to improve the paper.

References

Astropy Collaboration (Price-Whelan, A.M., et al.) 2018, AJ, 156, 123
Avila, R. J. 2017, Advanced Camera for Surveys Instrument Handbook for Cycle 25 v. 16.0
Bressan, A., Marigo, P., Girardi, L., et al. 2012, MNRAS, 427, 127
Bridžius, A., Narbutis, D., Stonkutė, R., Deveikis, V., & Vansevičius, V. 2008, Balt. Astron., 17, 337
Cabrera-Vives, G., Reyes, I., Förster, F., Estévez, P. A., & Maureira, J.-C. 2017, ApJ, 836, 97
Caldwell, N., Harding, P., Morrison, H., et al. 2009, AJ, 137, 94
Dalcanton, J. J., Williams, B. F., Lang, D., et al. 2012, ApJS, 200, 18
de Meulenaer, P., Narbutis, D., Mineikis, T., & Vansevičius, V. 2013, A&A, 550, A20
de Meulenaer, P., Narbutis, D., Mineikis, T., & Vansevičius, V. 2014, A&A, 569, A4
de Meulenaer, P., Narbutis, D., Mineikis, T., & Vansevičius, V. 2015, A&A, 574, A66
de Meulenaer, P., Stonkutė, R., & Vansevičius, V. 2017, A&A, 602, A112
Dieleman, S., Willett, K. W., & Dambre, J. 2015, MNRAS, 450, 1441
Domínguez Sánchez, H., Huertas-Company, M., Bernardi, M., Tuccillo, D., & Fischer, J. L. 2018, MNRAS, 476, 3661
Dressel, L. 2012, Wide Field Camera 3 Instrument Handbook for Cycle 21 v. 5.0
Elson, R. A. W., Fall, S. M., & Freeman, K. C. 1987, ApJ, 323, 54
Flamary, R. 2016, ArXiv e-prints [arXiv:1612.04526]
Fouesneau, M., Johnson, L. C., Weisz, D. R., et al. 2014, ApJ, 786, 117
Gaia Collaboration (Brown, A. G. A., et al.) 2016, A&A, 595, A2
Girshick, R. 2015, 2015 IEEE International Conference on Computer Vision (ICCV), 1440
Haykin, S. S. 2009, Neural Networks and Learning Machines, 3rd edn. (Upper Saddle River, NJ: Pearson Education)
He, K., Zhang, X., Ren, S., & Sun, J. 2016, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770
Herbel, J., Kacprzak, T., Amara, A., Refregier, A., & Lucchi, A. 2018, JCAP, 7, 054
Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. 2012, ArXiv e-prints [arXiv:1207.0580]
Huertas-Company, M., Primack, J. R., Dekel, A., et al. 2018, ApJ, 858, 114
Johnson, L. C., Seth, A. C., Dalcanton, J. J., et al. 2012, ApJ, 752, 95
Johnson, L. C., Seth, A. C., Dalcanton, J. J., et al. 2015, ApJ, 802, 127
Johnson, L. C., Seth, A. C., Dalcanton, J. J., et al. 2017, ApJ, 839, 78
Kingma, D.P., & Ba, J. 2014, ArXiv e-prints [arXiv:1412.6980]
Krist, J. E., Hook, R. N., & Stoehr, F. 2011, in Optical Modeling and Performance Predictions V, 8127, 81270J
Kroupa, P. 2001, MNRAS, 322, 231
Kruger, N., Janssen, P., Kalkan, S., et al. 2013, IEEE Trans. Pattern Anal. Mach. Intell., 35, 1847
Lanusse, F., Ma, Q., Li, N., et al. 2018, MNRAS, 473, 3895
Lin, T. Y., Maire, M., Belongie, S., et al. 2014, in Computer Vision – ECCV 2014, eds. D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars, (Cham: Springer International Publishing), 740
McConnachie, A. W., Irwin, M. J., Ferguson, A. M. N., et al. 2005, MNRAS, 356, 979
Nair, V., & Hinton, G. E. 2010, Proc. 27th International Conference on Machine Learning, ICML'10, 807
Narbutis, D., Bridžius, A., & Semionov, D. 2015, Balt. Astron., 24, 305
Narbutis, D., Semionov, D., Stonkutė, R., et al. 2014, A&A, 569, A30
Petrillo, C. E., Tortora, C., Chatterjee, S., et al. 2017, MNRAS, 472, 1129
Pourrahmani, M., Nayyeri, H., & Cooray, A. 2018, ApJ, 856, 68
Ren, S., He, K., Girshick, R., & Sun, J. 2017, IEEE Trans. Pattern Anal. Mach. Intell., 39, 1137
Rowe, B. T. P., Jarvis, M., Mandelbaum, R., et al. 2015, Astron. Comput., 10, 121
Russakovsky, O., Deng, J., Su, H., et al. 2015, Int. J. Comput. Vision, 115, 211
Sedaghat, N., & Mahabal, A. 2018, MNRAS, 476, 5365
Shallue, C. J., & Vanderburg, A. 2018, AJ, 155, 94
Vansevičius, V., Kodaira, K., Narbutis, D., et al. 2009, ApJ, 703, 1872