A&A 611, A2 (2018)
DOI: 10.1051/0004-6361/201731201

Astronomy & Astrophysics

Deep convolutional neural networks as strong gravitational lens detectors

C. Schaefer[1], M. Geiger[1], T. Kuntzer[1], and J.-P. Kneib[1,2]

[1] Institute of Physics, Laboratory of Astrophysics, École Polytechnique Fédérale de Lausanne (EPFL), Observatoire de Sauverny, 1250 Versoix, Switzerland
e-mail: christophernstrerne.schaefer@epfl.ch
[2] Aix-Marseille Université, CNRS, LAM (Laboratoire d'Astrophysique de Marseille) UMR 7326, 13388 Marseille, France

Received 19 May 2017 / Accepted 22 October 2017

ABSTRACT

Context. Future large-scale surveys with high-resolution imaging will provide us with approximately 10^5 new strong galaxy-scale lenses. These strong-lensing systems will be contained in large data amounts, however, which are beyond the capacity of human experts to visually classify in an unbiased way.
Aims. We present a new strong gravitational lens finder based on convolutional neural networks (CNNs). The method was applied to the strong-lensing challenge organized by the Bologna Lens Factory. It achieved first and third place, respectively, on the space-based data set and the ground-based data set. The goal was to find a fully automated lens finder for ground-based and space-based surveys that minimizes human inspection.
Methods. We compared the results of our CNN architecture and three new variations ("invariant" "views" and "residual") on the simulated data of the challenge. Each method was trained separately five times on 17 000 simulated images, cross-validated using 3000 images, and then applied to a test set with 100 000 images. We used two different metrics for evaluation, the area under the receiver operating characteristic curve (AUC) score, and the recall with no false positive ($\text{Recall}_{0\text{FP}}$).
Results. For ground-based data, our best method achieved an AUC score of 0.977 and a $\text{Recall}_{0\text{FP}}$ of 0.50. For space-based data, our best method achieved an AUC score of 0.940 and a $\text{Recall}_{0\text{FP}}$ of 0.32. Adding dihedral invariance to the CNN architecture diminished the overall score on space-based data, but achieved a higher no-contamination recall. We found that using committees of five CNNs produced the best recall at zero contamination and consistently scored better AUC than a single CNN.
Conclusions. We found that for every variation of our CNN lensfinder, we achieved AUC scores close to 1 within 6%. A deeper network did not outperform simpler CNN models either. This indicates that more complex networks are not needed to model the simulated lenses. To verify this, more realistic lens simulations with more lens-like structures (spiral galaxies or ring galaxies) are needed to compare the performance of deeper and shallower networks.

Key words. gravitational lensing: strong – methods: numerical – methods: data analysis – techniques: image processing – cosmology: observations – dark matter

1. Introduction

Future strong gravitational lense (SL) studies will help further constrain cosmology and galaxy evolution. As of today, galaxy-scale lenses have been used successfully to constrain the Hubble constant by measuring the time-delay of lensed images of quasars independently of other measurement techniques (Bonvin et al. 2016; Suyu et al. 2017). The magnification of lensed source-objects allows observations and studies of background objects at much higher redshifts than are usually visible to telescopes (Kneib et al. 2004; Richard et al. 2011; Atek et al. 2015). Measurement of galaxy-scale SLs can accurately constrain the total mass of the galaxy by probing the dark matter structure. This can be used to estimate the fraction of dark matter in galaxy halos when used in combination with weak-lensing analysis (Gavazzi et al. 2007) or by itself (Jiang & Kochanek 2007; More et al. 2011; Sonnenfeld et al. 2015). It can also be used to constrain the slope of the inner mass density profile (Treu & Koopmans 2002a,b; More et al. 2008; Koopmans et al. 2009; Cao et al. 2016) and the initial stellar mass function (Treu et al. 2010; Ferreras et al. 2010; Leier et al. 2016). One of the largest lens catalogs was produced by the Sloan Lens ACS Survey (SLACS) with about 100 observed lenses (Bolton et al. 2008). These SLs were discovered by selecting lens candidates from the spectroscopic database of the Sloan Digital Sky Survey (SDSS). Lens candidates were chosen by identifying the spectroscopic signature of two galaxies in the spectra, one galaxy at a greater distance than the other. These candidates were then verified by follow-up observation using the *Hubble* Space Telescope.

Historically, SLs were found serendipitously by human inspection of data. However, a systematic search by experts is too time-consuming to be a practical proposition for future large-scale surveys unless it were to involve citizen scientists. For example, the number of new lens systems from the *Euclid* mission (Laureijs et al. 2011) and from the Large Synoptic Survey Telescope (LSST Science Collaboration et al. 2009) survey is expected to reach at least 10^5 SLs among 10^9 objects (LSST: Oguri & Marshall 2010; HST: Pawase et al. 2014; *Euclid*: Collett 2015). Similarly, the amount of SLs found by the

SKA survey is expected to be on the same order of magnitude (McKean et al. 2015). Efficient automated gravitational lens-finding techniques are urgently needed.

The Spacewarps project[1] was an attempt to use and train non-experts at lens classification. Through an interactive website, amateur scientists were trained to sort through data from CFHTLS (Marshall et al. 2015). They found 29 promising new lens candidates in the survey (More et al. 2016), but this method will likely be too slow and too much subject to human error for future data sets. Semi-automated methods like arc detectors using clustering techniques have been used with some success (Lenzen et al. 2004; Cabanac et al. 2007) and have been further improved. Joseph et al. (2014) and Paraficz et al. (2016) added machine-learning to these techniques, using a Principal Component Analysis (PCA) based approach to remove the foreground galaxy from the image and facilitate the detection of arcs. Recently, Petrillo et al. (2017) and Jacobs et al. (2017) started using convolutional neural networks (CNNs) for lens detection. CNNs belong to a class of efficient image-classifier techniques that have revolutionized image processing (Lecun et al. 1998). In astrophysics, they have been applied successfully to galaxy morphology (Huertas-Company 2015), redshift estimation (Hoyle 2016), and spectra analysis (Hála 2014).

The *Euclid* Strong-Lensing working group, in collaboration with the Bologna lens factory[2], has started the Galaxy-Galaxy Strong-Lensing challenge[3] (GGSLC: Metcalf et al., in prep.) in light of future large-scale imaging surveys such as the *Euclid* mission. The goal was to determine the best technique for finding gravitational lenses for both ground-based and space-based imaging.

Our goal was to explore and optimize CNN architectures for lens classification. We successfully applied it to the GGSLC and were awared first and third place in the two categories of the GGSLC. In this paper, we present the CNN lens finder in detail that we created for the GGSLC challenge and discuss the advantages and disadvantages of CNN lens classifiers when applied on simulated and real data. The paper is organized as follows. Section 2 gives a brief overview of artificial neural networks (ANN) and CNNs and their usage in image processing. Section 3 outlines the details of our algorithm implementation and the two winning CNN architectures of the challenge, while in Sect. 4 we present some interesting alternative architectures. Section 5 summarizes the results of the different architectures we applied to GGSLC data, and we discuss them.

2. Theory

2.1. Artificial neural network

Artificial neural networks are machine-learning techniques inspired by the study of the human brain (Hebbian learning: Hebb 1950). ANNs are capable of learning classification or regression tasks in N dimensions by training using a set of labeled examples. This makes them easily applicable to complex problems for which explicitly programmed solutions or mathematical models are difficult to write. The main drawback of ANNs is the computation cost of the training procedure. More modern training techniques coupled with advances in GPU processing power made ANNs versatile and capable of being applied to almost any data set. They are created by stacking layers of neurons together.

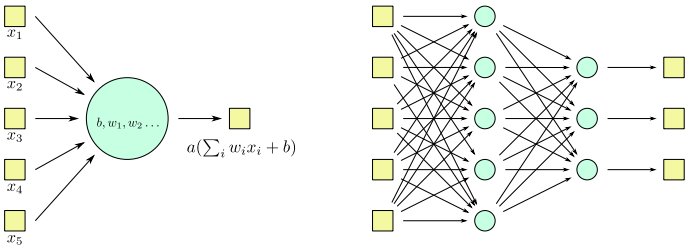


Fig. 1. *Left*: structure of a neuronal unit. Each neuron implements a linear combination (using weights w_i and a bias b) of its input $\boldsymbol{x}$ followed by a nonlinear activation function $a(\boldsymbol{x})$. *Right*: ANN structure. Neurons in the same layer all receive the same input. The stacking of layers allows the ANN to define a model parametrized by the weight variables of the network.

Each neuron implements a linear combination (using weights w_i and a bias b) of its input $\boldsymbol{x}$ followed by a nonlinear activation function $a(\boldsymbol{x})$,

$$y(\boldsymbol{x}) = a\left(\sum_{i=1}^{N} w_i x_i + b\right), \tag{1}$$

where N is the dimension of the inputs.

A layer consists of multiple neurons applied to the same input. Each output is passed as an input to the next layer. This cascade of nonlinear combinations of inputs ends at the output layer (see Fig. 1). In a classical ANN, all possible connections are established and exploited, in short, it is fully connected. A neuron in a given layer will transmit its outputs to all neurons in the next layer. Every layer between the input and the output layer is called a hidden layer. The initial input layer is sometimes also called a front layer. An ANN model is parametrized by the weights w and biases b of the neurons.

These weights and biases are trained iteratively. ANNs make predictions when presented with a training input. As the model parameters are randomly initialized, the first predictions are very different from the ground truth of the input.

The ANN then evaluates the error according to some predefined cost function and computes appropriate corrections to the parameters. These prediction errors are propagated backward through the layers, from the output to the front layer, and induce parameter updates. The technique is known as back-propagation (Rumelhart et al. 1986) and is commonly built on gradient descent for the computation of the parameter updates.

2.2. Convolutional neural network

Deep ANNs, models that have more than one or two hidden layers, perform better than shallow networks. The mathematical evidence for this statement is still scarce, but it is empirically observed. The continued growth in computation power made ANNs interesting for scientific application. However, computational cost of training increases with depth, and limitations in gradient-based procedures are challenging performance obstacles. Training with gradient methods generates a so-called vanishing-gradient problem, first identified by Hochreiter (1991). The magnitude of the gradient diminishes as it is back-propagated through the layers. The typical result is that layers close to the front layer effectively stop learning. While still affected by the vanishing-gradient problems, CNNs limit its effect by reducing the number of connections and sharing weights. This mitigation motivated the development of CNNs and their

[1] https://spacewarps.org/
[2] https://bolognalensfactory.wordpress.com/
[3] metcalf1.bo.astro.it/blf-portal/gg_challenge.html

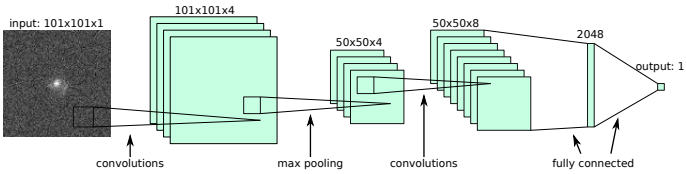


Fig. 2. Example of a CNN architecture: The input image undergoes a series of convolution layers into a series of feature maps. The first convolution transforms the 101×101 pixel image into four 101×101 pixel feature maps. To lower computation cost, max-pooling layers are used in between convolutions. They reduce the dimensionality of the image, dividing the size of the image by two. A fully connected layer then combines all feature maps for the classification.

subsequent application to image recognition (Lenet-5 model, Lecun et al. 1998).

The breakthrough for CNN came when Krizhevsky et al. (2012) created an architecture that won the 2012 ImageNet Large-Scale Visual Recognition Challenge[4]. His submission achieved a classification error of only 15.3% compared to the second-best submission with 26.2% obtained by a method not based on a neural network. CNNs have been used extensively in image processing ever since.

Our CNNs (Fig. 2) are created by stacking the following layers: convolution layers convolve the input image by a number of small kernels (or features maps, typically of dimension 3×3 to 7×7). The parameters to be optimized during training are the individual kernels. These weights are shared by all neurons in the layers (the kernels are the same for the whole layer). Pooling layers reduce the dimensionality of the input to decrease the number of parameters and avoid overfitting. The most common pooling technique is the max-pooling method. It partitions the input image into non-overlapping quadrants and yields the maximum value in the quadrant. Fully connected (fc) layers are the classic ANN neuron layer. Every input is connected to every neuron of the layers. They are used as the final CNN layers to merge the information contained in the feature maps into the desired output form. Dropout layers are only active during training. They randomly sever half the connections between the two layers they separate (Hinton et al. 2012). This is done to reduce coadaption of the neurons (learning the same features) and reduce overfitting. Batch normalization layers normalize and shift the output along a small input sample $B = \{x_{1...m}\}$ following the equation

$$y_i = \gamma \frac{x_i - \mu_B}{\sigma_B^2} + \beta, \tag{2}$$

where μ_B and σ_B are the mean and the variance over B. γ and β are two model parameters of the layer. Batch normalization is used to increase the training speed of the CNN (Ioffe & Szegedy 2015).

Convolution layers take advantage of the local spatial correlation in the data. Stacking multiple convolution layers implies a global treatment of the signal, making the network shift-invariant (i.e., features will be detected independently of their position). This make CNNs especially effective when treating images (Mallat 2016).

2.3. Data set of the Galaxy-Galaxy Strong-Lensing Challenge

The data for the GGSLC was provided by the Bologna lens factory challenge. The Bologna lens factory project is a complex lens-simulation project. It is based on the Millennium simulation (Lemson & Virgo Consortium 2006), with modeling of the gravitational lensing effect using the `Glamer` ray-tracing tool (Metcalf & Petkova 2014) and with `MOKA` to create the multi-plane dark halos and their substructures (Giocoli et al. 2012). The models and the parameters used to generate the simulations were blinded for the duration of the challenge.

Each image of the SL challenge was a 101×101 pixel stamp centered around an object. Participants had to submit a confidence value $p \in [0, 1]$ for each image. An object with a high confidence value was interpreted as a lens. Two categories of data were proposed with separate data sets, each with 20 000 training and 100 000 test images: (i) a space-based data set that consisted of images in a single visible band (simulating exposures of the *Euclid* instrument VIS); and (ii) a ground-based data set with images taken in four bands (U, G, R, and I) with a lower singal-to-noise ratio (S/N) and random masking of pixels, mimicking noisy data.

The ratio of lenses to non-lenses in the simulated data was much higher than in reality, around one-to-one, as an imbalance of examples (called skewed classes) can lead to biases. The results are have been made public, and a detailed discussion of the simulations and results will be provided in Metcalf et al. (in prep.). Our baseline architecture submission to GGSLC ranked first in the space-based data category and third in the ground-based category (Fig. 9). CNNs in general dominated the challenge. CNN-based methods filled the seven best submission in both categories.

3. CNN lensfinder: architectures

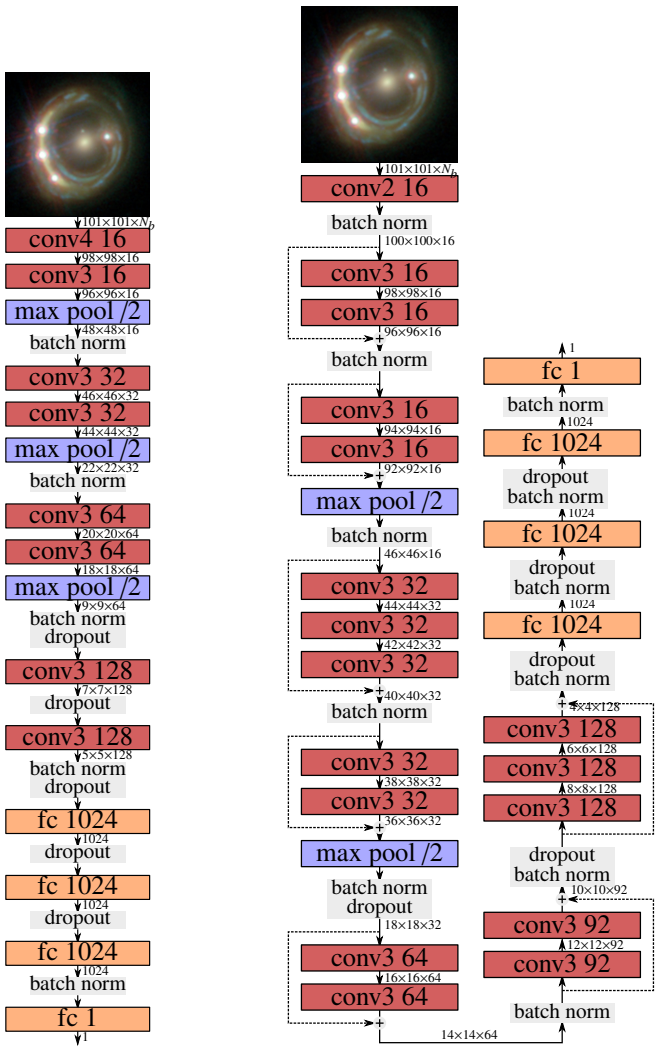
For this paper, four different types of CNN architectures were applied to the training data of the GGSLC: a simple CNN architecture that forms the baseline comparison for the paper, a so-called residual architecture based on the paper by He et al. (2015), and two further architectures with additional invariant properties. The final version of each architecture was selected after a heuristic study of the parameter space.

3.1. Baseline architecture

The baseline architecture as shown in Fig. 3a was inspired from typical CNN architectures that performed well in the ImageNet competition (Simonyan & Zisserman 2014). It is organized by stacking convolution blocks. This simple baseline architecture achieved first place in the space component of GGSLC. A convolution block is the superposition of 2 convolutional layers followed by a pooling layer to reduce the dimensionality of the image and a batch-normalization layer. The baseline architecture is comprised of 8 convolutional layers, organized into 3 convolution blocks and 2 stand-alone layers, and 3 fully connected layers at the top. There is thus a total of 11 layers. With the exception of the initial layer, every convolution layer uses 3×3 convolution kernels for efficiency reasons (Simonyan & Zisserman 2014). The first convolution layer uses a 4×4 kernel to yield an even number of pixels for easier manipulation. At each convolution block, the number of features was doubled, resulting in 256 features in the last block. The fully connected layers used either 1024 or 2048 features.

For each layer, we chose a modified version of the rectifier linear unit (ReLU) activation function because of its sparse representation capability (Glorot et al. 2011; Arpit et al. 2016). The

[4] http://image-net.org/



(a) Baseline **(b)** Residual

Fig. 3. Visualization of the baseline and residual architecture for the CNN lensfinder: the convolution blocks (red) indicate the size of the kernel and the number of features. The fully connected blocks (yellow) indicate the number of features. The arrows indicate the flow of the data, and between the blocks, we show the dimensionality of the input ($N_{\rm pixel} \times N_{\rm pixel} \times N_{\rm features}$). The last fully connected layer yields a confidence value of the object being a lens. The initial layer has $N_{\rm b}$ features, either one or four, depending on the category of the data (space and ground, respectively). Batch normalization and dropout layers are indicated as gray blocs.

activation is given by

$$f(x) = \frac{1}{\sqrt{\pi - 1}} \left(\sqrt{2\pi} \max(0, x) - 1 \right). \quad (3)$$

Inputs of the networks have dimension of $101 \times 101 \times N_{\rm b}$, where $N_{\rm b}$ is the number of bands ($N_{\rm b} = 1$ for space and $N_{\rm b} = 4$ for ground). The wavelength-dependent information (in the third dimension) is handled naturally by extending the kernel dimension from two to three (spatial to spatial plus wavelengths).

3.2. Residual architecture

A common way for improving CNN is to increase the depth, that is, the number of convolutional layers. With creating increasingly deeper CNNs comes the vanishing-gradient problem detailed before. At some point in the training process, the accuracy starts to saturate and degrade, generating an upper limit to the possible depth of CNNs. To compensate for this, He et al. (2015) introduced residual learning. In the GGSLC challenge, Francois Lanusse's deep lens classifier (Lanusse et al. 2018) used residual learning to create a 46-layer deep CNN that won the ground part of the challenge. We adapted our residual architecture to analyze the advantages and disadvantages compared to the baseline CNN.

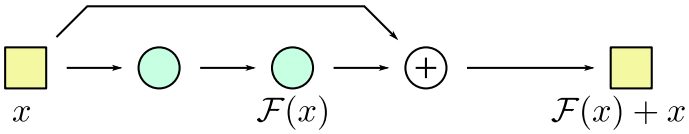


Fig. 4. Structure of a residual block: The feature maps $F(\boldsymbol{x})$ from two stacked convolutional layers are added to input $\boldsymbol{x}$. Each green circle represents a convolutional layer.

In a classical convolution layer, the feature map is created from scratch, that is, it learns an unreferenced mapping. The end-goal of the training process is to find parameters that minimize the cost function. We denote by $H(\boldsymbol{x})$ the optimum feature map and by $F(\boldsymbol{x})$ the map currently held in the parameters. In other words, the training updates $F(\boldsymbol{x})$ until

$$H(\boldsymbol{x}) = F(\boldsymbol{x}). \quad (4)$$

In contrast, residual networks train by optimizing a residual mapping $\boldsymbol{x}$, or the difference between the ideal and the real feature map. He et al. (2015) stated that it is easier to optimize the residual feature map than the unreferenced map,

$$H(\boldsymbol{x}) = F(\boldsymbol{x}) + \boldsymbol{x}, \quad (5)$$

where $\boldsymbol{x}$ is the identity mapping obtained by using shortcut connections skipping the convolution layers (Fig. 4). Our residual architecture as shown in Fig. 3b is 20-layer deep with 3 small residual blocks, 4 large residual blocks, and 3 fully connected layers with 1024 features. The small residual block is composed of 2 convolutions and 1 shortcut, keeping the same number of features. The large residual block is composed of 3 convolutions and 1 shortcut followed by a convolution layer, doubling the number of features.

3.3. Implementation details

Other than the differences in the approach to the problem, the networks shared a number of implementation details that we outline here.

- Cost function: we chose the binary cross-entropy cost function as the cost function C driving the training,

$$C = -\frac{1}{N} \sum \left\{ y \ln(y_{\rm p}) + (1 - y) \left[(1 - \ln(y_{\rm p}) \right] \right\}, \quad (6)$$

  where N is the number of training examples, y is the ground truth, and $y_{\rm p}$ is the classification prediction.
- Data augmentation: to increase the number of training samples, we used data-augmentation techniques. The goal is to generate more examples out of the original training set by exploiting physically invariant transformations, for example, rotating the image by 90 degrees. The benefit of increasing the training set size is to reduce overfitting.Taking advantage of the dihedral group symmetry (Fig. 6) of the lens problem,

the training sets were augmented using 90-degree rotations and flipping operations. We did not use rotation angles different than 90 degrees to avoid having to interpolate in pixel space.

- Training: the challenge training set was subdivided into a training set (of 17 000 images) that was used by the networks to learn and a validation set (3000 images strong) to check the performances on an independent set. The performance was monitored every 1000 steps by evaluating predictions made on the validation set. At each training step, we randomly selected batches of 30 images (15 lenses and 15 non-lenses) and ran the learning procedure for ∼250–300 epochs using the `ADAM` minimization algorithm (Kingma & Ba 2015). We trained five networks with the same architecture and selected the best-performing individual.
- Library: the models were implemented using the `Tensorflow` library (Abadi et al. 2015) on a GeForce GTX 1060 graphic card. The training time took approximately 1 h/100 epochs for the baseline model and 2 h/100 epochs for the residual model. The final prediction of the classification for the challenge on the 100 000 test images took approximately 20 min.

3.4. Image invariance

The idea behind these two next architectures was to deal with the inability of most lens finders to recognize and handle the invariant features of gravitational lenses. CNNs are, by design, already invariant to translation, but not to rotation, scaling, and flipping. The pretraining data augmentation phase renders them more robust to these symmetry operations, but not invariant. By modifying the CNN architecture so as to be invariant or more robust to different types of symmetries, we expect to reduce identification errors. The following sections describe how we increased the invariance of our models.

3.4.1. Views architecture

Several models trained to accomplish the same task form a committee. Predictions of a committee typically result in some sort of weighted combination of its members' predictions. They have been used to improve classification results for example on the MNIST[5] problem (Ciresan et al. 2011) or to detect anomalies in the predictions (Nguyen et al. 2014).

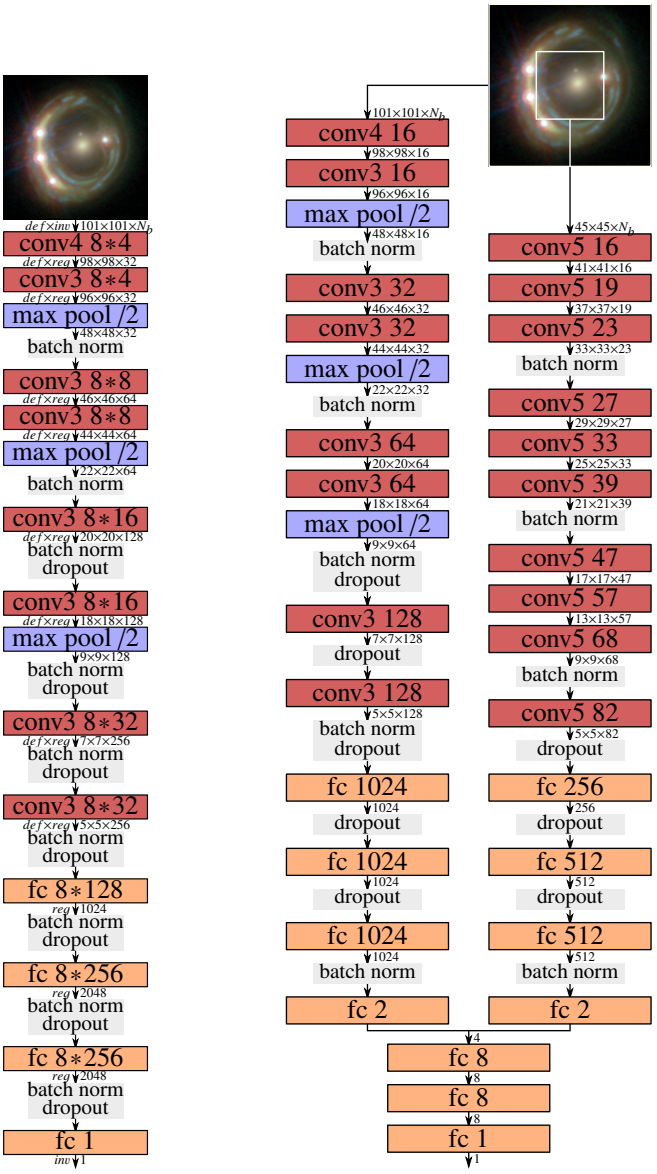
The views architecture (Fig. 5b) trains two neural networks separately to look for lenses of different sizes. The first network looks at the whole image, detecting large lenses spanning the whole image. The second uses only the central part of the image. By combining the prediction of the two networks, smaller lenses should be detected while not neglecting the detection of the larger lenses. In other words, the first network takes as input the whole image, like the baseline model, while the second only accepts a smaller stamp of 45×45 pixels. To simplify the smaller network, we used only 5 × 5 convolution layers and fewer features at each layer.

3.4.2. Invariant architecture

The invariant architecture adds additional invariant properties to the model. While relatively untested, this has been used with success for a galaxy morphology classifier on Galaxy Zoo data (Dieleman et al. 2015, 2016). The invariant architecture takes advantage of the dihedral symmetry of the lens-finding problem (Fig. 6) by using dihedral equivariant convolutional layers that we refer to as Dec layers.

At the level of the input layer, eight operations of the same convolution kernel, transformed by a different transformation of the dihedral group, are applied to the input image. The output is divided into eight different output channels (see Fig. 7),

$$y_i = \mathrm{Conv}(x, F_i) \quad i \in \{0, \ldots, 7\}, \tag{7}$$



(a) Invariant **(b)** Views

Fig. 5. Visualization of the invariant and views architecture for the CNN lensfinder: the convolution blocks (red) indicate the size of the kernel and the number of features. The fully connected blocks (yellow) indicate the number of features. The arrows indicate the flow of the data, and between the blocks, we show the dimensionality of the input ($N_{\mathrm{pixel}} \times N_{\mathrm{pixel}} \times N_{\mathrm{features}}$). The last fully connected layer yields a confidence value of the object being a lens. The initial layer has N_{b} features, either one or four, depending on the category of the data (space and ground, respectively). Batch normalization and dropout layers are indicated as gray blocs.

[5] MNIST database of handwritten digits, `http://yann.lecun.com/exdb/mnist/`

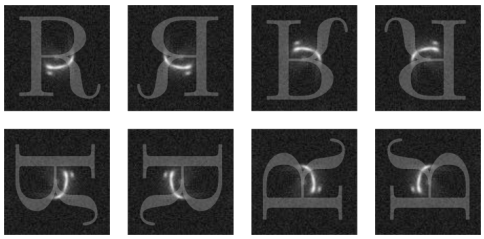


Fig. 6. Representation of the dihedral symmetry group: An optimal lens finder should be invariant to the operations of this group (i.e., flipping and rotation).

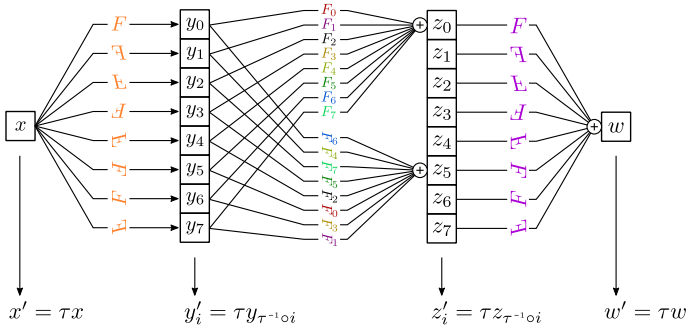


Fig. 7. Dihedral equivariant architecture: Kernels with identical colors but different orientation are identical kernels to which a different dihedral operations has been applied. Phase 1: seperation into eight channels, one for every input channel and member of the dihedral group. Phase 2: convolution of the eight channels with eight separate kernels. Each output channel from a Dec layer is the sum of all the input channels convolved by all feature kernels of the layer transformed by one of the dihedral operations. Phase 3: the eight channels are summed, giving a dihedral invariant result.

where i is one of the eight specific dihedral transformation and M_i is the filter to which a dihedral transformation has been applied.

Compared to the baseline version, for the Dec layer there are eight different convolution kernel instead of one: one kernel for each transformation of the dihedral group (F_i, $i \in \{0, \ldots, 7\}$). Each kernel is initialized and trained separately from each other. Each output channel in a Dec layer is the sum of all the input channels convolved by all the different feature kernels of the layer transformed by one of the dihedral operations (Fig. 7). The result of the eight channels, y_j, is a dihedral invariant quantity,

$$y_j = \sum_{i=0}^{7} \mathrm{Conv}(x_i, jF_{j^{-1} \circ i}) \quad j \in \{0, \ldots, 7\}. \tag{8}$$

The two layers illustrated in Fig. 7 have the property of being invariant with respect to the dihedral group. Our invariant architecture is shown in Fig. 5a and follows the same fundamental scheme as the baseline architecture. Since using eight channels increases computation time and makes the model more prone to overfitting, the number of features of the convolutional layers is divided by four. The invariance was tested by checking that rotated and flipped versions of the same image are attributed the same score by the classifier.

4. Results

In this section we describe the results of the different architectures applied to the GGSLC data.

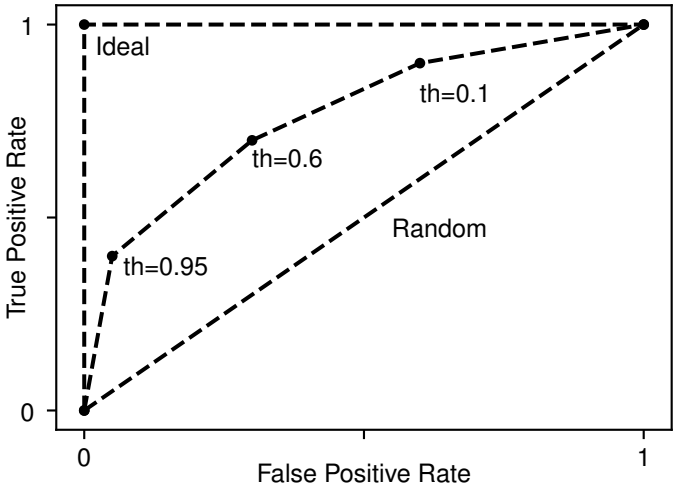


Fig. 8. Receiver operating characteristic (ROC) curve: The GGSLC ranked the classifiers as a function of the area under the ROC curve (AUC). For a perfect classifier, the score is 1, and for a random classifier, it is 0.5.

4.1. Performance metric

We first start by a brief overview of the performance metrics we used to quantify the performance of the lens classification.

- The true-positive rate (TPR) measures how well the classifier detects lenses from the whole population of objects,

$$\mathrm{TPR} = \frac{N_{\text{True positives}}}{N_{\text{True psitives}} + N_{\text{False negatives}}}. \tag{9}$$

  This metric is also known as recall. The best algorithms have a TPR close to 1.
- The false-positive rate (FPR) measures the contamination of the positive detections by false positives,

$$\mathrm{FPR} = \frac{N_{\text{False positives}}}{N_{\text{True negatives}} + N_{\text{False positives}}}. \tag{10}$$

  The best algorithms have an FPR close to 0.
- The receiver operating characteristic (ROC) is a visual representation of the TPR and FPR. Since they depend on the threshold $t \in (0, 1)$ defined to distinguish objects as lenses or non-lenses, the ROC curve (Fig. 8) is created by plotting TPR(t) as a function of FPR(t) for $t \in (0, 1)$. The challenge ranked the classifiers as a function of the area under the ROC curve (AUC), which is the integral of the ROC curve between an FPR of 0 and 1. A perfect classifier would score 1, while a randomly predicting classifier would score 0.5.

4.2. Training, submission, and results

After the challenge deadline, we tested our four architectures on the GGSLC data. As for the baseline architecture, we used our 17 000-image training set and the 3000-image validation set. Each architecture was trained five separate times. In Table 1 we show the result of this committee training. The performance is also evaluated in Table 3 on the challenge test data as the ground-truth values were released to participants after the submission deadline. The standard deviation of the five runs is also given. The two metrics used to evaluate the performance of the

Table 1. Training results: rach architecture was run five separate times.

Space	Training	Validation
baseline	0.9920 ± 0.0020	0.9764 ± 0.0017
views	0.9898 ± 0.0030	0.9753 ± 0.0021
residual	0.9958 ± 0.0028	0.9765 ± 0.0024
invariant	0.9997 ± 0.0003	0.9719 ± 0.0016
Ground	Training	Validation
baseline	0.9953 ± 0.0090	0.9905 ± 0.0082
views	0.9980 ± 0.0010	0.9924 ± 0.0006
residual	0.9990 ± 0.0023	0.9932 ± 0.0027
invariant	$0.9999 \pm 3 \times 10^{-6}$	0.9880 ± 0.0030

Notes. The training and validation AUC scores are the mean of these runs. The error is the standard deviation.

Table 2. Confusion matrix (baseline architecture, GGSLC challenge) for TPR0.

Space-based	Classified as non-lens	Classified as lens
Non-lens	59742	40
lens	20957	19264
Ground-based	Classified as non-lens	Classified as lens
Non-lens	50042	17
lens	21754	28194

Notes. The TPR0 threshold was chosen by the GGSLC organizers for no false-positive in the first 10 000 images of the test set.

Table 3. Test, $\text{Recall}_{0\text{FP}}$, and $\text{Recall}_{1\text{FP}}$ results.

Space	Test AUC	$\text{Recall}_{0\text{FP}}$	$\text{Recall}_{1\text{FP}}$
baseline	0.9322 ± 0.0016	0.01 ± 0.02	0.04 ± 0.04
committee b.	0.9326	0.01	0.01
views	0.9324 ± 0.0013	0.26 ± 0.06	0.28 ± 0.07
committee v.	0.9343	0.30	0.32
residual	0.9322 ± 0.0006	0.23 ± 0.04	0.29 ± 0.03
committee r.	0.9346	0.29	0.30
invariant	0.9332 ± 0.0006	0.27 ± 0.04	0.28 ± 0.05
committee i.	0.9399	0.32	0.33
Ground	Test AUC	$\text{Recall}_{0\text{FP}}$	$\text{Recall}_{1\text{FP}}$
baseline	0.9761 ± 0.0011	0.44 ± 0.13	0.49 ± 0.08
committee b.	0.9773	0.50	0.55
views	0.9746 ± 0.0011	0.35 ± 0.19	0.43 ± 0.17
committee v.	0.9759	0.35	0.39
residual	0.9775 ± 0.0006	0.44 ± 0.06	0.46 ± 0.07
committee r.	0.9795	0.50	0.55
invariant	0.9774 ± 0.002	0.39 ± 0.11	0.45 ± 0.05
committee i.	0.9813	0.49	0.49

Notes. Each architecture was run five times. The test scores are the mean of these runs.

methods are the (i) the AUC and (ii) the zero false-positives, $\text{Recall}_{0\text{FP}}$ (that is, the fraction of lenses recovered with zero false-positives).

The AUC results are much better for ground-based data than for space-based data (Fig. 9) although the images have a lower S/N than the space-based images. This increased performance could be due to the increased amount of information in the form of the four bands, instead of the single VIS-like band for space. The lower S/N in the ground-based data does not seem to hinder prediction.

The baseline, views, invariant and residual architectures achieved equally good AUC results on the validation set and the test set within the standard deviation of the runs. This is surprising because deeper networks, like the residual one, are expected to perform better than shallower models. The scores are too close to the optimum to confidently distinguish between the architectures. The most likely explanation is that the simulated data were too simple for the CNN lensfinder. The simulations did not include spiral galaxies or some other ring-like objects capable of confusing gravitational lens classifiers. A more complex method was therefore not needed to classify the data correctly. The difference that can be seen between the validation scores and the test scores in Tables 1 and 3 can be attributed to a slight overfitting. This is probably due to the small size of the validation set we used in comparison to the test set.

The invariant architecture has a lower validation AUC score than the others, but performs equally well on the test set. This may indicate that the invariant architecture generalizes the lens model better than other architectures. This could be due to the imposed invariant properties, as we have given the model some additional knowledge. This has no effect on the final test score but could become important when applying the CNN lensfinder to real data. Since the amount of known galaxy-scale lenses is small, a sufficiently large training set for a CNN lensfinder can only be obtained by simulated lenses (see Petrillo et al. 2017, for a CNN lensfinder applied to CHFTLS data). The caveat here is that CNNs trained on simulation might miss lenses because the simulated training set was unrealistic. The better the CNNs generalize the lens model, the lower the chance that they will missidentify objects. Ideas exist to force CNNs to focus on the lens model. One is to use multiple different simulations to create lenses (Jacobs et al. 2017). Adding dihedral invariance to CNNs could be another way of doing this.

The ground-based results are extremely encouraging, especially because of the purity of the score. In a classification problem with a 1-to-1000 ratio between lenses and non-lenses, algorithms with even a very small contamination can be dominated by false positives. The baseline model performs well on the metric $\text{Recall}_{0\text{FP}} = 0.44$ in the ground-based test set (Table 3). In a more realistic setting with a ratio of lenses to non-lens objects, we would have found 22 out of the 50 lenses in a test set containing 100 000 images without any false positives.

Table 3 shows that the standard deviation of $\text{Recall}_{0\text{FP}}$ is large. Using the CNN that performed best on the validation set does not guarantee the best $\text{Recall}_{0\text{FP}}$ or even best AUC score (Figs. 10 and 11). The metrics vary depending on the individual result of the training run. To mitigate this, we grouped the five training runs of our model in a committee of CNNs. The committee output is taken as the average of their prediction. By compensating for each other's shortcomings, committees stabilize the results and achieve a better-than-average result for the AUC metric as well as for the $\text{Recall}_{0\text{FP}}$ metric (Table 3, Figs. 12 and 13). The invariant model especially is improved by this and obtains the best scores for the space-based data.

5. Conclusions

We presented a strong gravitational lens finder based on convolutional neural networks (CNN). The method showed strong

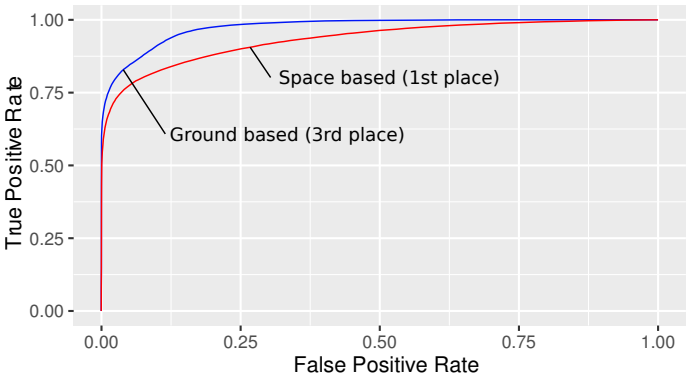


Fig. 9. ROC curve of our baseline architecture submission to the GGSLC challenge. The solid line is the curve from our submission. Blue is the ground-based data category, red is the space-based data category.

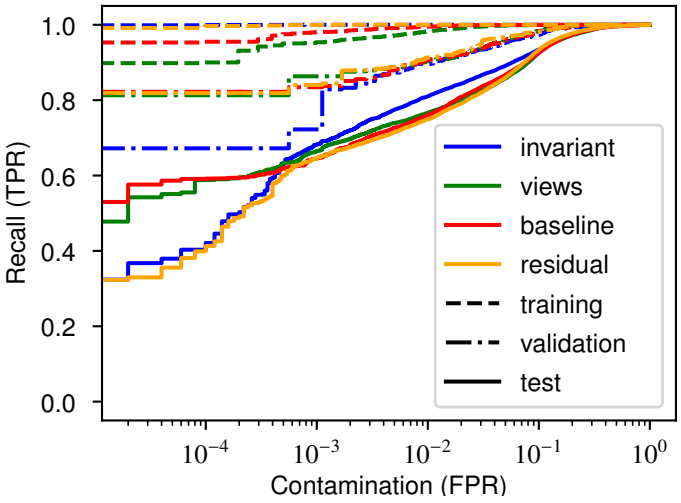


Fig. 10. Logaritmic ROC curves on ground-based data. Training (dotted line), validation (half-dotted line) and test (solid line) score of all four architectures. Data come from the best of five runs in terms of validation set score.

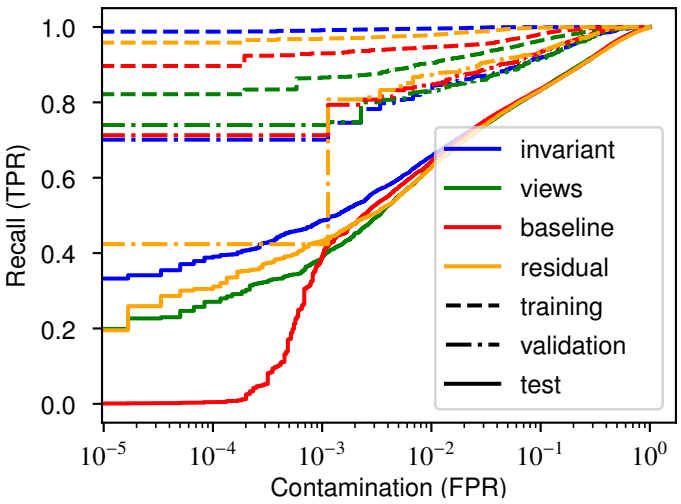


Fig. 11. Logaritmic ROC curves on space-based data. Training (dotted line), validation (half-dotted line) and test (solid line) score of all four architectures. Data come from the best of five runs in terms of validation set score.

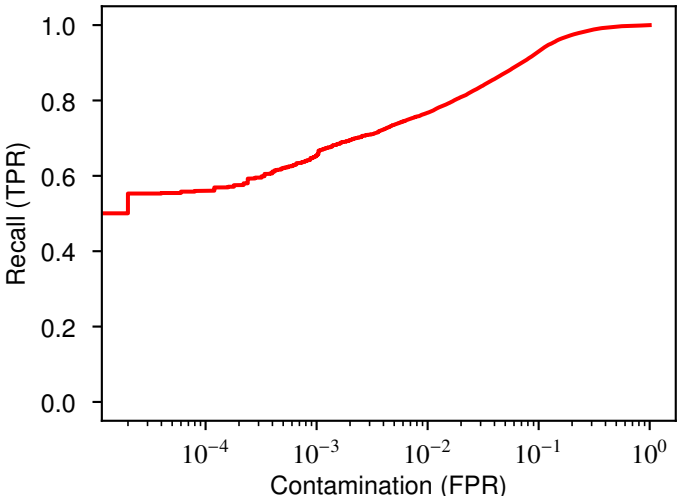


Fig. 12. Logaritmic ROC curve of the baseline committee on ground-based data. The curve is the result of the baseline committee (five baseline CNNs taken together). The shaded areas represent the minimum and maximum values from the five stand-alone baseline CNNs.

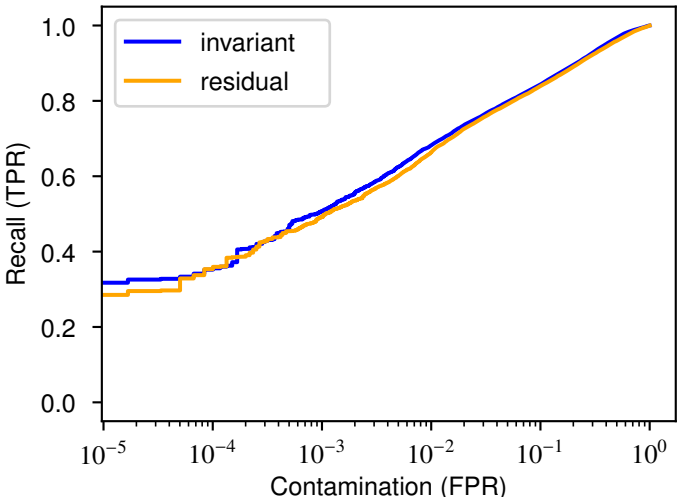


Fig. 13. Logaritmic ROC curve of the invariant and residual committee on space-based data. The curve is the result of the committee (five invariant or residual CNNs taken together). The shaded areas represent the minimum and maximum values from the five stand-alone invariant or residual CNNs.

performances on simulated data. It won the first place and third place in the Strong Gravitational Lens Challenge (GGSLC), respectively, in the space-based and ground-based data category. We have also presented three other variations of that lensfinder, among which, a residual CNN based on the recent architecture developed by He et al. (2015).

We found that CNNs perform better on ground-based data than on space-based data despite the lower S/N. This is probably due to the additional bands, which add information, but this still has to be confirmed. This can be done, for instance, by limiting the ground-based data to one band and comparing to the other results. All four CNNs achieved almost perfect ROC curve scores on the simulated data, with the highest area under the ROC curve (AUC) score up to 0.9775 for ground-based and 0.933 for space-based data. They also achieved a recall with a zero false-positive ($\text{Recall}_{0\text{FP}}$) of 50% for ground-based and of 32% for space-based data. We showed that the best $\text{Recall}_{0\text{FP}}$ results were achieved

by committees of CNNs instead of single CNNs. Committees of CNNs consistently scored the best AUC scores. We also observed that adding rotation invariance to CNNs grouped together in committees produces the best space-based $\text{Recall}_{0\text{FP}}$ score.

Because all results are almost equally good, more conclusions about the best CNN model cannot be drawn. Most likely the simulations did not include enough lens-like objects capable of inducing false positives in the lensfinder, that is, the simulations were likely not realistic enough. This might explain why, contrary to expectations, the residual CNN has not performed better than the others. We will further explore CNN algorithms in the future GGSLC.

Acknowledgements. The authors would like to acknowledge Frederic Courbin, Colin Jacobs, Ben Metcalf, and Markus Rexroth for their help and advice on the subject. C.S. and J.P.K. acknowledge support from the ERC advanced grant LIDA. T.K. acknowledges support from the Swiss National Science Foundation.

References

Abadi, M., Agarwal, A., Barham, P., et al. 2015, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, software available from tensorflow.org
Arpit, D., Zhou, Y., Kota, B. U., & Govindaraju, V. 2016, ArXiv e-prints [arXiv:1603.01431]
Atek, H., Richard, J., Kneib, J.-P., et al. 2015, ApJ, 800, 18
Bolton, A. S., Burles, S., Koopmans, L. V. E., et al. 2008, ApJ, 682, 964
Bonvin, V., Tewes, M., Courbin, F., et al. 2016, A&A, 585, A88
Cabanac, R. A., Alard, C., Dantel-Fort, M., et al. 2007, A&A, 461, 813
Cao, S., Biesiada, M., Yao, M., & Zhu, Z.-H. 2016, MNRAS, 461, 2192
Ciresan, D. C., Meier, U., Gambardella, L. M., & Schmidhuber, J. 2011, in 2011 International Conference on Document Analysis and Recognition, 1135
Collett, T. E. 2015, ApJ, 811, 20
Dieleman, S., De Fauw, J., & Kavukcuoglu, K. 2016, ArXiv e-prints [arXiv:1602.02660]
Dieleman, S., Willett, K. W., & Dambre, J. 2015, MNRAS, 450, 1441
Ferreras, I., Saha, P., Leier, D., Courbin, F., & Falco, E. E. 2010, MNRAS, 409, L30
Gavazzi, R., Treu, T., Rhodes, J. D., et al. 2007, ApJ, 667, 176
Giocoli, C., Meneghetti, M., Bartelmann, M., Moscardini, L., & Boldrin, M. 2012, MNRAS, 421, 3343
Glorot, X., Bordes, A., & Bengio, Y. 2011
Hála, P. 2014, Ph.D. Thesis [arXiv:1412.8341]
He, K., Zhang, X., Ren, S., & Sun, J. 2015, ArXiv e-prints [arXiv:1502.01852]
Hebb, D. O. 1950, Science Education, 34, 336
Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. 2012, ArXiv e-prints [arXiv:1207.0580]
Hochreiter, S. 1991, Diploma, Technical University Munich, Institute of Computer Science
Hoyle, B. 2016, Astron. Comput., 16, 34
Huertas-Company, M. 2015, IAU General Assembly, 22, 2252228
Ioffe, S., & Szegedy, C. 2015, ArXiv e-prints [arXiv:1502.03167]
Jacobs, C., Glazebrook, K., Collett, T., More, A., & McCarthy, C. 2017, MNRAS, 471, 167
Jiang, G., & Kochanek, C. S. 2007, ApJ, 671, 1568
Joseph, R., Courbin, F., Metcalf, R. B., et al. 2014, A&A, 566, A63
Kingma, D. P., & Ba, J. 2015, in International Conference on Learning Representations [arXiv:1412.6980]
Kneib, J.-P., Ellis, R. S., Santos, M. R., & Richard, J. 2004, ApJ, 607, 697
Koopmans, L. V. E., Bolton, A., Treu, T., et al. 2009, ApJ, 703, L51
Krizhevsky, A., Sutskever, I., & Hinton, G. E. 2012, in Advances in Neural Information Processing Systems Conf.
Lanusse, F., Ma, Q., Li, N., et al. 2018, MNRAS, 473, 3895
Laureijs, R., Amiaux, J., Arduini, S., et al. 2011, ArXiv e-prints [arXiv:1110.3193]
Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. 1998, Proc. IEEE, 86, 2278
Leier, D., Ferreras, I., Saha, P., et al. 2016, MNRAS, 459, 3677
Lemson, G., & Virgo Consortium, T. 2006, ArXiv e-prints [arXiv:astro-ph/0608019]
Lenzen, F., Schindler, S., & Scherzer, O. 2004, A&A, 416, 391
LSST Science Collaboration, Abell, P. A., Allison, J., et al. 2009, ArXiv e-prints [arXiv:0912.0201]
Mallat, S. 2016, Philos. Trans. R. Soc. London Ser. A, 374, 20150203
Marshall, P. J., Lintott, C. J., & Fletcher, L. N. 2015, ARA&A, 53, 247
McKean, J., Jackson, N., Vegetti, S., et al. 2015, Advancing Astrophysics with the Square Kilometre Array (AASKA14), 84
Metcalf, R. B., & Petkova, M. 2014, MNRAS, 445, 1942
More, A., McKean, J. P., Muxlow, T. W. B., et al. 2008, MNRAS, 384, 1701
More, A., Verma, A., Marshall, P. J., et al. 2016, MNRAS, 455, 1191
More, S., van den Bosch, F. C., Cacciato, M., et al. 2011, MNRAS, 410, 210
Nguyen, A., Yosinski, J., & Clune, J. 2014, ArXiv e-prints [arXiv:1412.1897]
Oguri, M., & Marshall, P. J. 2010, MNRAS, 405, 2579
Paraficz, D., Courbin, F., Tramacere, A., et al. 2016, A&A, 592, A75
Pawase, R. S., Courbin, F., Faure, C., Kokotanekova, R., & Meylan, G. 2014, MNRAS, 439, 3392
Petrillo, C. E., Tortora, C., Chatterjee, S., et al. 2017, MNRAS, 472, 1129
Richard, J., Jones, T., Ellis, R., et al. 2011, MNRAS, 413, 643
Rumelhart, D. E., Hinton, G. E., & Williams, R. J. 1986, Nature, 323, 533
Simonyan, K., & Zisserman, A. 2014, ArXiv e-prints [arXiv:1406.2199]
Sonnenfeld, A., Treu, T., Marshall, P. J., et al. 2015, ApJ, 800, 94
Suyu, S. H., Bonvin, V., Courbin, F., et al. 2017, MNRAS, 468, 2590
Treu, T., & Koopmans, L. V. E. 2002a, ApJ, 575, 87
Treu, T., & Koopmans, L. V. E. 2002b, MNRAS, 337, L6
Treu, T., Auger, M. W., Koopmans, L. V. E., et al. 2010, ApJ, 709, 1195