The Astrophysical Journal, 856:68 (11pp), 2018 March 20 https://doi.org/10.3847/1538-4357/aaae6a

LensFlow: A Convolutional Neural Network in Search of Strong Gravitational Lenses

Milad Pourrahmani[1], Hooshang Nayyeri, and Asantha Cooray
Department of Physics and Astronomy, University of California Irvine, Irvine, CA, USA
Received 2017 May 23; revised 2018 January 31; accepted 2018 February 7; published 2018 March 26

Abstract

In this work, we present our machine learning classification algorithm for identifying strong gravitational lenses from wide-area surveys using convolutional neural networks; LENSFLOW. We train and test the algorithm using a wide variety of strong gravitational lens configurations from simulations of lensing events. Images are processed through multiple convolutional layers that extract feature maps necessary to assign a lens probability to each image. LENSFLOW provides a ranking scheme for all sources that could be used to identify potential gravitational lens candidates by significantly reducing the number of images that have to be visually inspected. We apply our algorithm to the *HST*/ACS i-band observations of the COSMOS field and present our sample of identified lensing candidates. The developed machine learning algorithm is more computationally efficient and complimentary to classical lens identification algorithms and is ideal for discovering such events across wide areas from current and future surveys such as LSST and *WFIRST*.

Key words: gravitational lensing: strong – methods: data analysis – techniques: image processing

1. Introduction

Gravitational lensing, a prediction of Einstein's general theory of relativity, is a very powerful tool in cosmological studies. It has been used extensively to understand various aspects of galaxy formation and evolution (e.g., Refsdal & Bondi 1964; Blandford & Narayan 1992; Postman et al. 2012; Atek et al. 2015; Nayyeri et al. 2016). This involves accurate cosmological parameter estimation (Treu 2010), studies of dark matter distribution from weak gravitational lensing events (Kaiser & Squires 1993; Velander et al. 2014), black hole physics (Peng et al. 2006), and searches for the most distant galaxies (Coe et al. 2012; Oesch et al. 2015), among others.

One of the main goals of observational cosmology is to constrain the main cosmological parameters that dictate the evolution of the universe (Tegmark et al. 2004; Komatsu et al. 2009; Weinberg et al. 2013). Strong gravitational lensing has been utilized over the past few years to estimate and constrain these cosmological parameters (Broadhurst et al. 2005; Suyu et al. 2013, 2014; Agnello et al. 2017; Goobar et al. 2017; More et al. 2017). This is achieved through accurate lens modeling of such events and comparing the model predictions with observations (such as with observations of lensing induced time delays (Eigenbrod et al. 2005; Treu 2010; Suyu et al. 2014; Rodney et al. 2016; Treu & Marshall 2016). In a recent study, for example, Suyu et al. (2013) used combined WMAP, Keck, and *HST* data on gravitational time delays in two lensed sources to constrain the Hubble constant within 4% in a ΛCDM cosmological framework.

One of the key aspects of gravitational lensing is its use as natural telescopes through boosting the observed signal and increasing the spatial resolution (Treu 2010). This is quite advantageous in searches for distant and/or faint objects at moderate observing costs and has been utilized extensively in various surveys in searches for such objects, the identification of which would not have been possible without it (Bolton et al. 2006; Heymans et al. 2012). Given that the number of identified lenses for different classes of galaxies rises sufficiently due to better lens finding algorithms, one could study the intrinsic properties of distant galaxies from such searches to understand the physics of star formation and mass assembly (Fu et al. 2012; Timmons et al. 2016; Nayyeri et al. 2017; Wilson et al. 2017). In the past few years, deep diffraction limited observations have also taken advantage of gravitational lensing to extend the faint end of the luminosity function of galaxies by a few orders of magnitude (Atek et al. 2015) to produce the deepest images of the sky ever taken across multiple bands. Strong gravitational lensing events have been observed extensively in such surveys as galaxy–galaxy lensing in field surveys such as the Cosmic Assembly Near-infrared Deep Extragalactic Legacy Survey (CANDELS; Grogin et al. 2011; Koekemoer et al. 2011) and the Cosmological Evolution Survey (COSMOS; Capak et al. 2007; Scoville et al. 2007) or as cluster lensing from observations of nearby massive clusters (Postman et al. 2012; Treu et al. 2015; Lotz et al. 2017) with the *Hubble* Space Telescope leading to the identification of the first generations of galaxies (out to $z \sim 11$; Oesch et al. 2015) and studies of galaxy formation and evolution at the epoch of reionization. This was, in fact, one of the main motivations behind *Hubble* cluster lensing studies such as Cluster Lensing and Supernova Survey with Hubble (CLASH; Postman et al. 2012) and the *Hubble* Frontier Fields (Lotz et al. 2017). The power of gravitational lensing could also be used in the detection of low surface brightness emission from extended objects such as millimeter and radio emissions from dust and molecular gas at $z \sim 2-3$ as observed with ALMA used to study the physics of the cold ISM (Spilker et al. 2016).

Strongly lensed galaxies are normally targeted and identified from dedicated surveys (Bolton et al. 2006). Traditionally, these lens identifications are either catalog-based, in which lensing events are identified by looking for objects in a lensing configuration, or pixel based, with the search starting from a set of pixels. These lensing searches are normally computationally challenging in that individual pixels are constantly compared with adjacent ones and they could be biased toward a given population and/or brighter objects. Recent far-infrared wide area observations (such as those with *Herschel*) advanced searches for

[1] Corresponding author.

lensed galaxies by adopting a simple efficient selection technique of lensed candidates through observations of excessive flux in the far-infrared (as an indication of strong lensing events supported by number count distributions; Wardlow et al. 2012; Nayyeri et al. 2016). However, such surveys are also biased toward populations of red dusty star-forming galaxies (missing any blue lenses) and are not always available across the full sky (the *Herschel* surveys that were targeted had ∼0.2–0.4 deg^{-2} lensing events, much lower than expected from optical surveys). Given that tests of cosmological models require simple unbiased selection functions, it is important to have a complete unbiased catalog of lensing events.

We have entered the era of big data astronomy. Sky surveys such as the LSST, *Euclid*, and *WFIRST* will produce more imaging data than humans can ever analyze by eye. The challenges of designing such surveys are no longer merely instrumentational, but they also demand powerful data analysis and classification tools that can identify astronomical objects autonomously. Fortunately, computer vision has drastically improved in the last decade making autonomous astronomy possible. The past couple of years has been the most exciting era in the field of machine learning (ML). Researchers from both the public and the private sectors have achieved landmarks in developing image recognition/classification techniques. One of the most exciting recent events in the ML community was the release of TENSORFLOW by Google, a parallel processing platform designed for development of fast deep learning algorithms (Abadi et al. 2016). Packages and software, such as MATHEMATICA, TENSORFLOW, CAFFE, and others, alongside cheaper and more powerful Graphics Processing Units (GPUs) have enabled researchers to develop very complex and fast classification algorithms. Among these deep learning programs, ConvNets have deservingly received a lot of attention in many fields of science and industry in the past few years (Krizhevsky et al. 2012). Complex ConvNets such as GOOGLENET and ALEXNET, which are publicly available, have achieved superhuman performance on the task of image classification. Google's TENSORFLOW has made it possible to easily develop parallelized deep learning algorithms, which, if integrated with Google's Tensor Processing Units (TPUs), could address the data mining challenges in the field of astronomy. In this work, we introduce and image classification algorithm, LENSFLOW, which is a ConvNet that can be used to search for strong gravitational lenses with the final version of the code publicly available on Github.[2]

This paper is organized as follow. In Section 2, we will explain the principal concepts underlying neural networks, supervised learning, and ConvNets. Before feeding the images to a ConvNet, they must be normalized and should be enhanced. The details data extraction and normalization are discussed in Section 3.1. As discussed in Section 3.3, we explain the architecture of LENSFLOW and its pretraining process on CIFAR data (Krizhevsky & Hinton 2009). In Sections 3.4 and 3.5, we discuss training LENSFLOW on COSMOS data and show its performance on recovering known tracer lenses. In Section 3.6, we share a set of new lenses found by LENSFLOW and we conclude our results in Section 4. Throughout this paper, we assume a standard cosmology with $H_0 = 70\ \mathrm{km s^{-1}\ Mpc^{-1}}$, $\Omega_m = 0.3$, and $\Omega_\Lambda = 0.7$. Magnitudes are in the AB system where $m_{\mathrm{AB}} = 23.9 - 2.5 \times \log(f_\nu/1\mu\mathrm{Jy})$ (Oke & Gunn 1983).

[2] https://github.com/Miladiouss/LensFlow

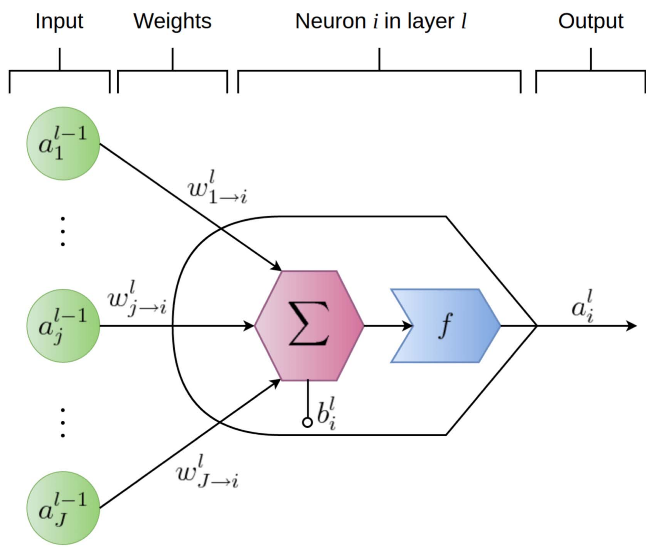

Figure 1. Schematic representation of an artificial neuron. The weighted sum of the neurons in the previous layer (green circles), plus the internal bias of the neuron, are mapped as the output of the neuron by an activation function. This model is captured by Equation (1). During the learning process, weights and biases of the neurons will be adjusted to achieve the desired network output.

2. Deep Learning Algorithms

Artificial neural networks are inspired by biological neurons. Just like biological neurons, artificial neurons receive input signals and send out an output signal to other neurons (see Figure 1). The synaptic connections between neurons are known as weights and the output of a neuron is known as its activation. To reduce the computational time and simplify neural network models, neurons are placed in consecutive layers rather than having a connection with every other neuron. This neural network setup is known as the Multi-layer Perception where neurons from one layer cannot talk to each other or to the neurons in arbitrary layers; they may only send their signal to the neurons in the succeeding layer. A neuron receives the weighted sum of the activation of all the neurons in the previous layer, adds an internal parameter known as the bias, and maps this sum to a value computed by an activation function (e.g., sigmoid, hyperbolic tangent, rectilinear, softmax). This model can be stated mathematically by the following equation:

$$a_i^l = f\left(\sum_j a_j^{l-1} w_{j\to i}^l + b_i^l\right). \tag{1}$$

Here, a_i^l is the activation of the neuron in hand (i.e., the ith neuron in the lth layer), f is the activation function of this neuron, a_j^{l-1} is the activation of the neuron j in layer $l-1$ (the previous layer), $w_{j\to i}^l$ is the synaptic weight connecting the ith neuron in layer l to the jth neuron in layer $l-1$, and b_i^l is the bias of the neuron to adjust its activation sensitivity. The first layer, i.e., the input layer, in a deep learning neural net acts as a sensory layer, analogous to the retina. As it gets analyzed, the information from the input layer travels through multiple layers until it reaches the final layer, called the classification layer. Each class of images corresponds to a classifying neuron. In our case, we have a neuron corresponding to non-lens and another to lens images. The neuron with the highest output determines which class an input image is placed in.

A neural net learns how to classify images by adjusting the weights between its neurons and the biases within them, having one goal in mind: minimizing the loss function $C(\boldsymbol{x}, \boldsymbol{y})$. The loss function, sometimes called the cost function, can take many forms but it has to capture the misfiring of the classification neurons, i.e., the deviation between the target class versus the predicted class. This is why such algorithms are known as supervised learning algorithms, in contrast to unsupervised techniques. A common choice for the loss function is the cross-entropy loss function with the following form (Nielsen 2016):

$$C(\boldsymbol{x}, \boldsymbol{y}) = \textstyle\sum_{j=\text{non-lens, lens}} y_j \ln a_j^L + (1 - y_j)\ln(1 - a_j^L). \quad (2)$$

a_j^L is the activation of neurons in the final (classifying) layer, $\boldsymbol{x}$ is the input data in the vector form, and $\boldsymbol{y}$ represents the desired activations of the two classifying neurons. Of course, this function depends on the architecture of the neural net, weights, and biases, but they have not been expressed explicitly. As an example, if an image is a lens, its target output has to be (0.0, 1.0), meaning the activation of the non-lens neuron should be zero and the activation of the lens neuron should be unity. During the training process of a neural net, images from a training data set are presented to the network and the weights and the biases are adjusted to minimize the loss function for those images. The parameter space is massive and a change in one of the parameters of a neuron will affect the activation of a series of neurons in other layers. The first challenge is solved by minimizing algorithms such as the stochastic gradient descent (SGD) and the second one is solved via back-propagation. Since optimizing over the whole training set at once is not possible, because the training data would not fit in memory, stochastic optimization algorithms provide guaranteed convergence even if the gradients are evaluated on a randomly (stochastically) selected subset (batch) of the training data set. They provide a practical way to optimize a model over extremely large data sets. Batches yield noisy approximations to the true gradient, and larger batches can better approximate this quantity while if the batch size is too small, that approximation would be too poor and the algorithm may never converge in practice.

ConvNets are a class of neural networks with multiple convolutional layers. A convolutional layer consists of a set of convolving neurons (on the order of 10 neurons) that can be connected to a small rectangular region of an image. The set of weights of a convolving neuron is known as a filter and is subject to change as the ConvNet learns. A filter scans an entire image by striding (convolving with specified steps) over the image and assembling its output into an image, which is known as a feature map. Feature maps contain information such as texture and edges. See Figure 2 as an example of a set of filters in a LENSFLOW convolutional layer. A few examples of feature maps have been shown in Figure 3. These feature maps are bundled together as an image with the same number of channels as the number of feature maps. In this image, we have selected three feature maps and represent them with different colors to display what the neural network sees as the image passes through the layers.

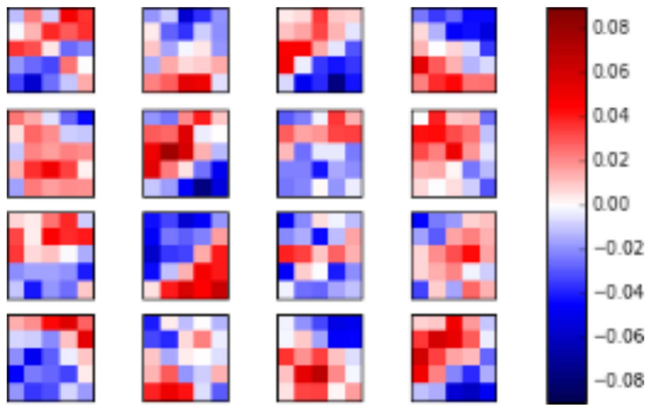

Figure 2. Examples of filters used in a convolutional layer. The pixels in each box represent the weights of a convolving neuron that are connected to a 5×5 region input image. As these filters convolve over the entire input image, they generate feature maps. Red pixels have a positive contribution and blue pixels have a negative contribution toward the activation of the convolving neuron. These filters are helpful for edge and texture recognition.

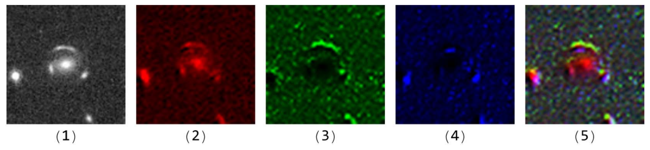

Figure 3. Examples of a convolutional layer feature maps. An image of a normalized physical lens has been shown in (1). ((2)–(4)) Three outputs of the second convolution layer of LENSFLOW. (5) The superposition of these three maps. As can be seen in (5), these feature maps create a contrast between the upper arc and the foreground source, making it possible for the fully connected layers to decide whether an imaged is a lens or a non-lens.

3. Methodology

This section covers the image normalization process, simulation of gravitational lenses and training and testing data set creation, architecture of LENSFLOW, its pretraining on CIFAR data set, and its training on COSMOS data in two sequential steps: course and fine classification phases.

3.1. Data Extraction and Normalization

We used *HST*/ACS i-band observations in the full COSMOS field to search for candidate gravitationally lensed sources. In order to prepare the survey data for the neural network, we created 200×200 pixel cutouts around sources identified by SEXTRACTOR, which corresponds to roughly 3×3 square arcseconds. We ignored sources that extended less than 200 pixels total (not to be confused with our cutout size) and were not 1.5σ brighter than the background, totaling 236,000 images. These images were then downsampled to 100×100 pixels to speed up the training and scanning process.

Before inputting the images to LENSFLOW, we normalized and enhanced them, which is a necessary step to ensure a stable training and prevents activation saturation issues. For deep learning purposes, there are different methods of image normalization to choose from. This is due to the fact that raw image pixels come in a wide range of values. As we will discuss in the next section, when lenses are produced by superposing simulated arcs on top of actual sources, it is crucially important to ensure that superposed images

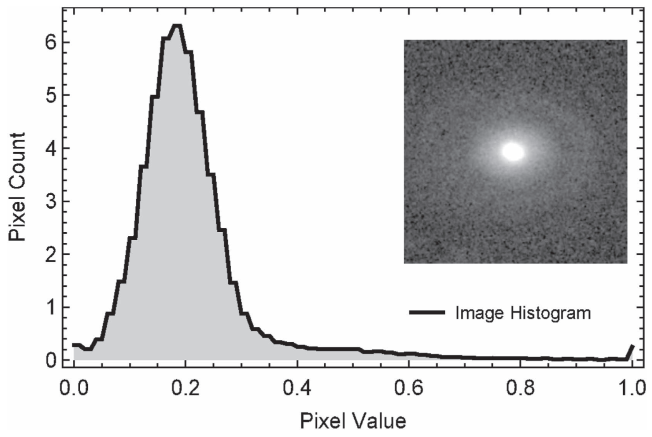

Figure 4. Template image histogram for image normalization. The histograms of all sources were transformed to match this template histogram, which was obtained by adjusting the brightness and the contrast and by performing gamma correction for a known dim lens image displayed above. This transformation not only normalizes images, but it will also enhance the contrast between the arcs and the foreground source.

are renormalized after the superposition process. If lens images are not renormalized, the net will become sensitive to the total sum of the pixels and achieve a meaningless perfect classification on the training and test data sets with no application for searching for real lens images.

There are different methods of image normalization used that often involve shifting the mean, normalizing the standard deviation, and bounding the pixels between two fixed values. Though sufficient for classification of daily object photos, we did not find these methods helpful to our algorithm since astronomical images require gamma correction to adjust the image contrast. Gamma correction introduces a nonlinearity according to the following equation:

$$p_{\text{out}} = A p_{\text{in}}^{\gamma}, \tag{3}$$

where A and γ are constants and p_{in} and p_{out} are the initial and corrected pixel values. However, applying the same gamma function to different sources is not practical. For instance, the arcs in some lens images might get enhanced, while they are obscured by the foreground in other images. Similar problems happen when cutting off bright and dim pixels. To overcome these issue, we have selected one dim real lens image and have adjusted its brightness, its contrast, and have performed gamma correction so the foreground source and the arcs are clearly separated and visible, while keeping all pixel values between 0 and 1. The image histogram (the histogram of pixel values) of this modified lens image was extracted (See Figure 4) and was used as a template histogram to transform the image histogram of all the extracted sources. This method not only enhances the arcs for all the previously known lenses from COSMOS, but it also automates the process of gamma correction and normalization since all pixel values fall between 0 and 1 and their mean and their standard deviation are roughly the same.

3.2. Lens Simulation

In order to train a neural network, typically a few thousand examples are needed per class. Since the number of known

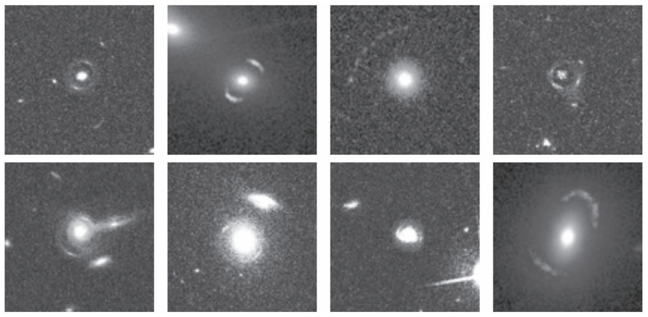

Figure 5. Examples of simulated lenses.

lenses are far more limited than the required number, these lenses have to be simulated. For these simulations, we used LENSTOOL (Jullo et al. 2007) to generate image plane models of lensing systems using realistic models of randomly selected elliptical galaxies within the COSMOS field as deflectors and coadded these to the selected elliptical galaxies to generate the training set. Here, we focus on elliptical galaxies as foreground deflectors. Although known examples of spiral galaxy lensing exist (Calanog et al. 2014), most galaxy–galaxy lensing events occur around massive elliptical galaxies as foreground deflectors. We generated a training set of 200 galaxies using this method. Figure 5 shows several examples from the generated training set used by LENSFLOW.

As discussed in the previous section, simulated lenses must be renormalized to prevent the net from classifying lenses based on the total sum of the pixels since without normalization, the total pixel sum of lens images will always be higher than non-lens images.

The number of generated lenses is still very low for training purposes. To overcome this, we can use image augmentation to artificially boost the number of training examples. We do this by rotating and reflecting images. In more detail, we use eight transformations that come from eight elements of the symmetry group of the square, namely: 0°, 90°, 180°, and 270° rotations, and horizontal, vertical, diagonal, and anti-diagonal reflections. In addition to rotation and to reflection, we take eight 90 × 90 pixel cutouts at random positions and rescale them to 100 × 100 pixels. We will refer to these two processes as image augmentation.

3.3. Architecture of LENSFLOW ConvNet and Pretraining (Phase 0)

The architecture of the data determines the dimensionality of the ConvNet layers. We use 1 × 100 × 100 images, where 1 indicates the number of color channels.[3] Classifying lenses with multiple color bands will be easier and more accurate since foreground and background sources have a color contrast. However, we have chosen to use one color channel so our algorithm can be sensitive to geometry rather than color contrast in order to expand its applicability to a wider range of bands as well as eliminating its need for multiband images when unavailable. The results of galaxy lens identification from wide-area surveys with color information will appear in a future work (M. Pourrahmani et al. 2018 in preparation). As we see in Figure 6 and Table 1, after normalizing these

[3] In this paper and in our code, we have adapted the $N \times C \times H \times W$ where N, C, H, and W stand for number of input images in a batch, number of color or feature channels, height, and width respectively.

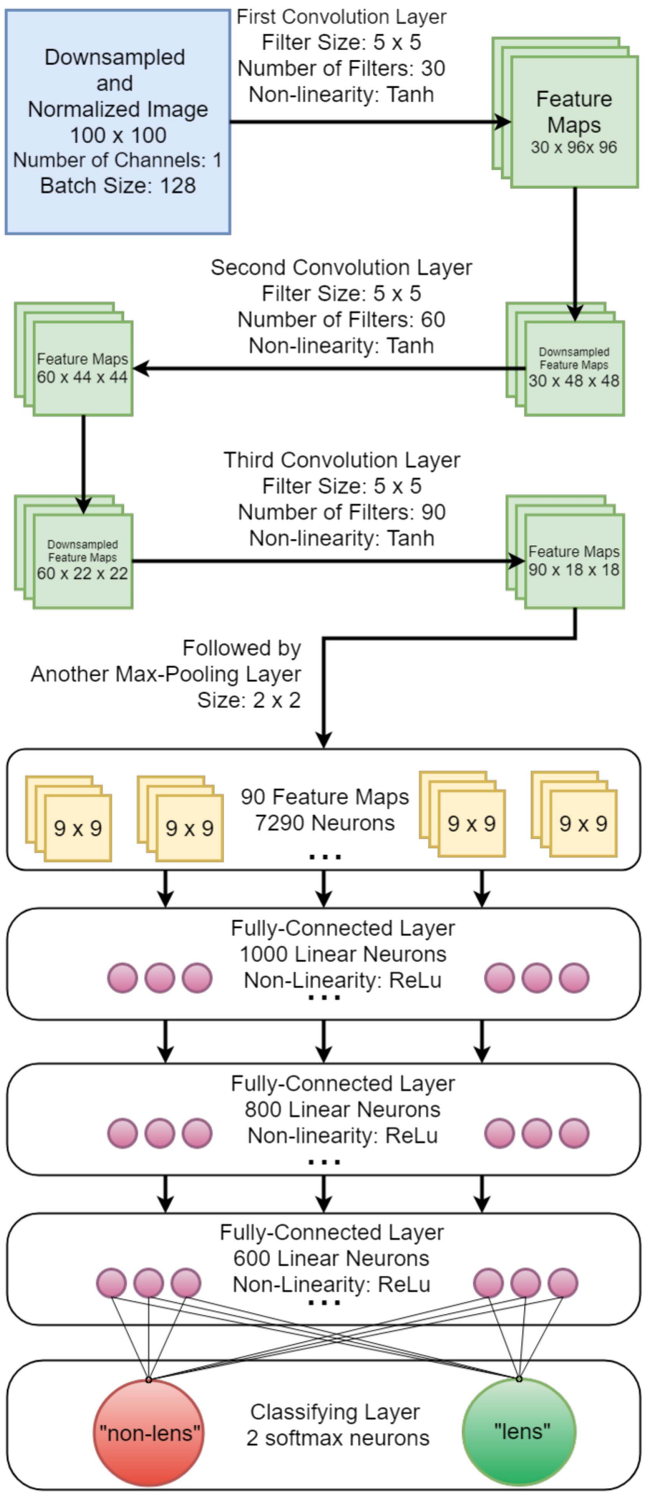

Figure 6. Representation of data flow through the ConvNet layers. The data is downsampled and fed to three convolutional-max-pooling layers. The data is then flattened into an array and is fed to three fully connected linear layers, which are then connected to the classifying layers consisting of a linear layer with two neurons followed by a sofmax layer. The convolution layers are responsible for feature extraction and the fully connected layers learn the difference between non-lens and lens images.

single-channeled images, LENSFLOW applies an average-pooling of a kernel of size 5 × 5 and a stride of 1 without padding. The hyperbolic tangent function introduces non-linearity to the convolution layer. The common choice is often

Table 1
Tabulated Architecture of the LENSFLOW ConvNet

Layer	Type	Data Dimensionality
	input	1 × 100 × 100
1	convolution	30 × 96 × 96
2	tanh	30 × 96 × 96
3	pooling	30 × 48 × 48
4	convolution	60 × 44 × 44
5	tanh	60 × 44 × 44
6	pooling	60 × 22 × 22
7	convolution	90 × 18 × 18
8	tanh	90 × 18 × 18
9	pooling	90 × 9 × 9
10	flatten	7290
11	linear	1000
12	ReLU	1000
13	dropout	1000
14	linear	800
15	ReLU	800
16	dropout	800
17	linear	600
18	ReLU	600
19	dropout	600
20	linear	2
21	softmax	2
	output	2

a rectified linear unit (ReLU), which sets pixels smaller than a self-learned threshold to zero. ReLU is ideal for edge detection but since astronomical images do not have hard edges, an smoother function like the hyperbolic tangent is suited. The output of this layer is fed to a pooling layer with a kernel of size 2 × 2 and a stride of 1, outputting the largest pixel value as it convolves its input for all channels. The result of this layer is a set of 30 downsampled (48 × 48) feature maps. The next two convolution layers are identical to the first set described above except that the second convolution layer has 60 and the last convolution layer has 90 filters. The output of the last convolution layer is a set of 90 × 9 × 9 feature maps, which are flattened from a tensor to a 1D vector with 7290 rows, which is fed to the fully connected layers. The first fully connected layer has 1000 linear neurons (identity function as f in Equation (1)). This layer is followed by a dropout layer where the output of 50% of the neurons is set to zero. Dropout layers prevent overfitting in early stages of training. Two more linear layers of size 800 and 600 follow this layer. Finally, all inputs are fed to a linear layer of size two with a softmax nonlinearity. These two layers act as a classifying layer where the softmax layer converts the output of the linear layer to probabilities:

$$\sigma_c(\boldsymbol{Z}) = \frac{e^{Z_c}}{e^{Z_{\text{non-lens}}} + e^{Z_{\text{lens}}}}. \tag{4}$$

Here, each component of $\boldsymbol{Z}$ is the output of the last linear layer (i.e., output of Layer 20) with two components. c specifies whether we are talking about the neuron corresponding to lens or non-lens images. The softmax function ensures the output sums to one and when used with the cross-entropy loss function, these outputs are interpreted as class probabilities. We use these probabilities to rank images from most probable lens candidates to least probable.

To optimize our ConvNet, we have chosen a cross-entropy function as our loss function, which we minimize using the Adam Optimizer. This adaptive optimizer algorithm computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients for the loss function (Kingma & Ba 2014). During the training phase, 64 non-lens and 64 lens images were placed in a batch of 128 images. This combination technique will prevent under- or over-representation of classes even if the training size for different classes contains a different number of examples.

A common practice for training a neural net on a smaller data set is to first pretrain it on a different but larger data set and later, retrain it on the main but smaller data set in hand, while keeping the weights and biases in the convolution layers stiff or fixed by dampening the learning rate. This is valid since these layers are used for general feature extraction. On the other hand, these parameters are relaxed in the fully connected layers where the main task of classification takes place. This technique is known as transfer learning. We have selected two classes from the CIFAR data set, a famous data set used for testing computer vision algorithms. After reducing the images to grayscale and changing their size to 100 × 100 pixels, we applied the image normalization explained above. Before training, it is important to properly initialize the weights and biases. While setting all biases to zero, we use the Xavier method (Glorot & Bengio 2010) where the neuron weights in layer L are sampled from a normal distribution with a mean of zero and a variance of $2/(N^L + N^{L+1})$, where N^L is the size of layer L. Xavier weight initialization addresses the vanishing gradient problem and prevents the existence of overly strong or overly weak weights resulting in a steady signal throughout the network. After initializing, an initial learning rate of $8.\times10^{-4}$ was chosen and the network was trained for a total of eight rounds. The number of iteration rounds is determined by the deviation of the loss function for training and test data sets. It is an indication of overfitting if the average loss function for the training data set drops while it remains the same or increases for the test data set. In simpler terms, overfitting means the net is memorizing the training data set rather than learning generalizable feature extraction and classification. Similarly, we determine the number of iteration rounds for other training phases discussed in the following two subsections. Pretraining is crucial for our data set without which no learning occurs. We suspect that having soft edges as well as a central dominant object are the main causes of the trouble here, preventing the network from learning edge detection and picking up on arc-like features. Techniques such as reducing the brightness of central pixels were tested that triggered the learning process even though with a poor performance. However, by using a pretrained net, the need for masking the central bright pixels was eliminated and a much better performance was achieved.

3.4. Coarse Classification Phase (Phase 1)

The training data set for this phase consists of 3200 lenses created by augmenting the 200 simulated lenses and randomly selecting 3200 images from COSMOS. It is possible that these randomly selected images contain actual lenses, but a few misclassified examples will not affect the training process in a noticeable way. To generate a validation and a testing data set, we have selected 52 out of 67 discovered lens candidates by Faure et al. (2008) and have applied image augmentation (rotations and reflections only) to increase their number to 464, which were accompanied by 464 randomly selected images from COSMOS labeled as non-lens images. Among the lens candidates that were not selected, three were larger than 3 arcseconds in diameter, one did not have an i-band image, and the rest did not have any arc features in the i-band and were classified as lenses by Faure et al. (2008) by mainly relying on other bands.

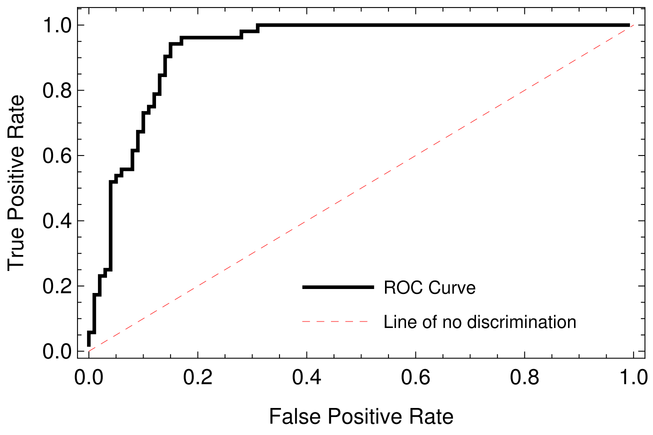

Figure 7. Receiver operating characteristic (ROC) diagram. The black curve shows the trend of the true positive rate verses the false positive rate of LensFlow, while the red dashed curve shows an untrained classifier.

Since the convolution layers of the pretrained net (layers 1 to 10 in Table 1) are used for feature extraction and are transferable from one data set to another, during the training process on COSMOS data, the learning rate of those layers were reduced to 10 %. On the other hand, the main purpose of the fully connected layers are to classify based on the extracted features and are not transferable. Hence, their learning rate was unaltered during the training process.

After 20 training epochs, we conducted two main performance tests. The first test is obtaining the receiver operating characteristic curve, i.e., plotting the ROC curve, which is a standard measure of the performance of a classifier. As plotted in Figure 7, the horizontal and the vertical axes indicate the false positive rate (FPR) and the true positive rate (TPR) respectively. The ROC curve is obtained by evaluating FPR and TPR for different classification thresholds (i.e., the minimum lens probability for an image to be categorized as lens). Even though ROC curves are very useful for most classifiers, we suggest a different performance measure that is more appropriate for the field of astronomy where thousands or millions of images have to be scanned to identify desired sources. In our case, after training the net, we have placed the 52 selected lenses as tracers among the entire 236,000 source images extracted from COSMOS. The assigned lens probability by this net has been used to rank the images from the most likely lens candidates to the least likely. The number of recovered tracer lenses as a function of relative ranks (i.e., rank of an image divided by the total number of images) are plotted in Figure 8. We will refer to this curve as the tracer rank curve (TRC). We see that 100% of tracer lenses fall in the top 6% of the sorted images. A TRC can be quantified by one number, which we refer to as the ranking performance. We define the ranking performance as the area between a TRC (the solid black line in Figure 8) and the line of no discrimination (the dashed red line in the same figure) divided by the TRC for a perfect classifier (i.e., approximately one-half when the number of scanned images is much larger than the number of tracer images). Therefore, a ranking performance of 1 corresponds to

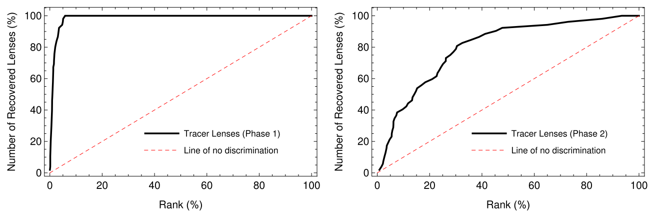

Figure 8. Normalized ranking of tracer lenses by LENSFLOW. Ranking performance is defined as the area between the black tracer rank curve (TRC) and the dashed red line of no discrimination divided by the area between a perfect TRC and the line of no discrimination (approximately one-half for large data sets). Left: 100% of the tracer lenses are in the top 6%. The ranking performance of Phase 1 is 0.97. Right: during Phase 2, the ConvNet was trained on the top 6% images from Phase 1. The ranking of the tracer lenses has been shown. 80% of the tracer lenses are in the top 30%. The ranking performance of Phase 2 is 0.60.

placing all tracer lenses in the highest ranks, while a ranking performance of 0 corresponds to dispersing tracer lenses among all images, which is a sign of no learning. A negative ranking performance, on the other hand, means the classifier is systematically misclassifying images. The ranking performance of our ConvNet during this phase is 0.97 which is quite good for such a simple ConvNet applied on a data set with one color channel. However, placing all the tracer lenses in the top 6% means lenses have to be recovered among 14,000 images. Even though this is a massive reduction from 236,000, examining 14,000 images by eye is not very practical and would be impossible for larger surveys. For this reason, we have introduced another phase that specializes in finding lenses among the remaining 14,000 images by retraining our net on the top 6% images, as discussed in the following subsection.

3.5. Fine Classification Phase (Phase 2)

In order to further reduce the number of images that have to be examined by eye, we have constructed a data set by randomly selecting 3200 images from the remaining COSMOS images from Phase 1. The same 3200 augmented simulated lens images were added to complete the data set. The ConvNet was trained over 25 iterations and the remaining images from Phase 1 were scanned and ranked using this net. The first 300 images were examined, which included some lenses but mostly artifacts, spiral galaxies, and satellite galaxies (see Figure 9). Lens images were removed and the remaining were augmented and added to the training data set for retraining to eliminate most probable false classifications. The TRC for this net is plotted in the right panel of Figure 8 with a ranking performance of 0.60. The same net with the same methodology could do extremely better if images had color information, which is not present in our data. Other ways to improve the results is to create a more diverse and larger data set, which is very time consuming. Using more complex nets such as GoogLeNet with perception models might improve the results, which we will investigate in a future study. The results of Phase 1 show that simple and fast deep learning algorithms, such as ours, are sufficient enough to reduce the data by a factor of 17 since both training and scanning are more time consuming with complex nets such as GoogLeNet.

3.6. Search Phase (Phase 3)

This phase is identical to the previous phase with the exception of including augmented tracer lenses in the training data set in order to increase the size of the data set to improve the chances of

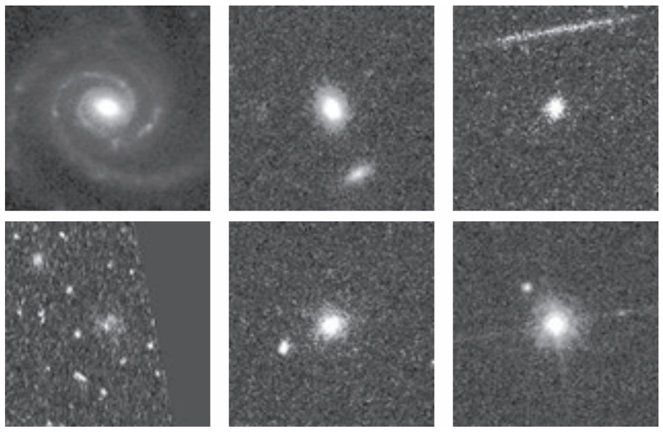

Figure 9. Examples of common misclassified images. These images were used to retrain LENSFLOW to improve its ranking performance.

finding new lens candidates. The previous phase was necessary to obtain the maximum number of training epochs and to test the performance of LENSFLOW. After training this ConvNet, we examined 2000 images and identified 46 new lens candidates that were not mentioned in Faure et al. (2008; the examination process took roughly 20 minutes). These lens candidates are shown in Figure 10 and their coordinates are listed in Table 2 (see Appendix B). Classification algorithms like ours can benefit from Citizen Science projects such as the SPACE WARPS project (Marshall et al. 2015; More et al. 2016), where volunteers are presented with real and simulated gravitational lenses in order to obtain a measure of their classification performance, while identifying new lenses. Citizen Science projects such as SPACE WARPS can help with classification of images with high lensing probabilities assigned by automated classifiers.

4. Discussion

Non-ML computer algorithms have been previously used for finding gravitationally lensed arcs (Lenzen et al. 2004; Alard 2006; More et al. 2012) and rings (Gavazzi et al. 2014). As discussed in More et al. (2012) and Gavazzi et al. (2014), RINGFINDER is an algorithm that uses color information and ARCFINDER detects arc-like pixels. In more detail, ARCFINDER starts by polishing the images by convolving a smoothing kernel. For each pixel, an estimator of elongation is calculated by taking the ratio between the sum of the flux of a few pixels along the horizontal line and the maximum value of a few nearby pixels along the vertical line, which passes through the pixel in hand. This process is repeated for all pixels and those with smaller than an specified elongation threshold are set to zero to create a sharp arc map. An arc map that satisfies thresholds on the arc properties such as the size and surface brightness will be selected as an arc candidate for further visual inspection. Such techniques can be used as complementary methods to deep learning. Currently, both techniques may suffer from many false positive detections, which commonly include tidally interacting galaxies, artifacts, and ring and spiral galaxies. The hope is that with more developed training data sets, deep learning algorithms can resolve such false positive cases (see Figure 9 and Section 3.5).

Other researchers (Jacobs et al. 2017; Lanusse et al. 2018; Petrillo et al. 2017) also find deep learning to be a suitable solution for finding gravitational lenses. Lanusse et al. (2018) use residual ConvNets with 46 layers. Residual ConvNets are

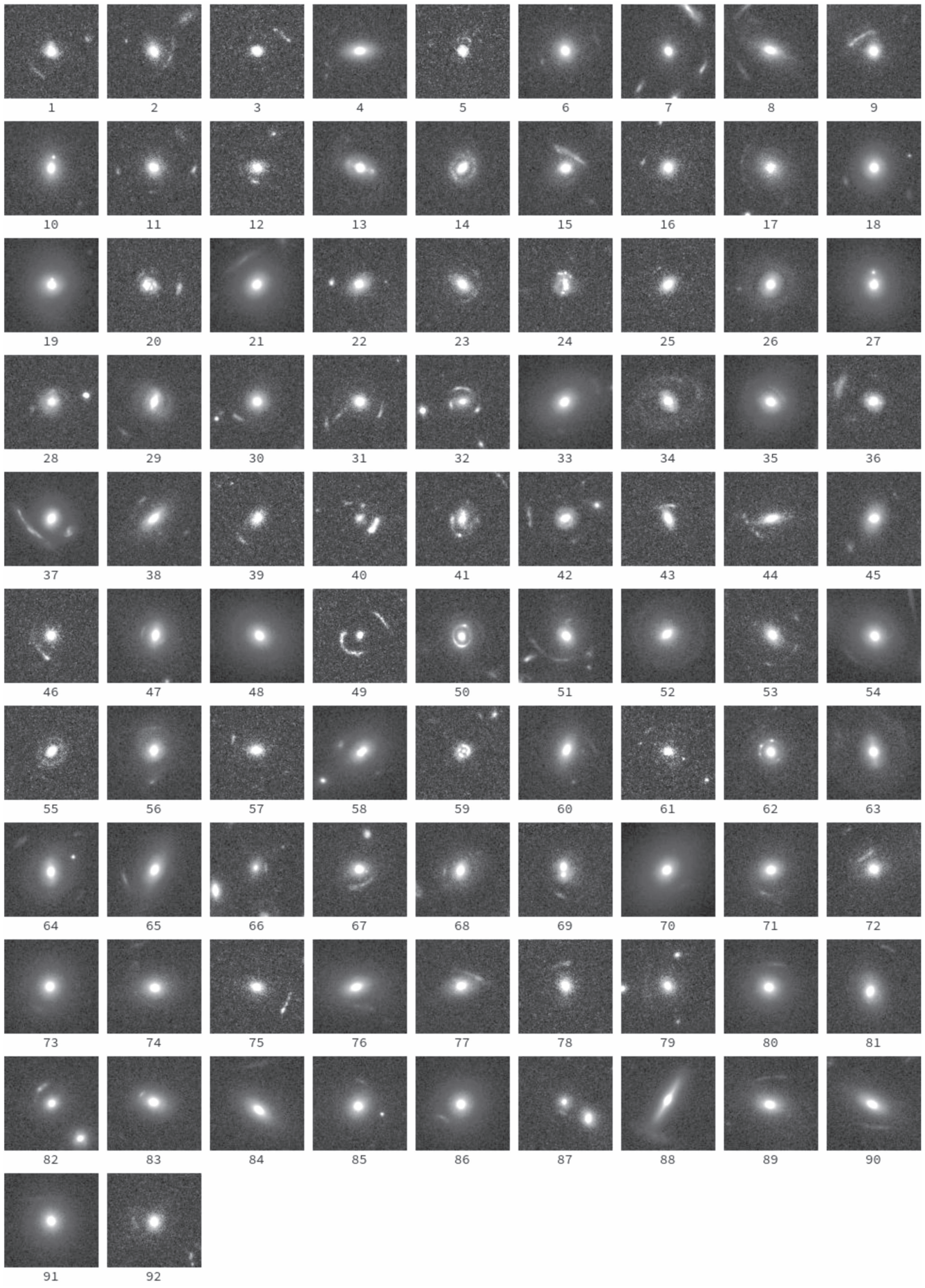

Figure 10. Identified COSMSOS lens candidates by LENSFLOW. These candidates are cataloged in Table 2.

Table 2
Catalog of Identified Lenses by LENSFLOW

Lens	R.A. (deg)	Decl. (deg)	Einstein Radius (arcsec)	Magnitude (AB)	Absolute Rank In 236,000	Average Grade[a] A/B/C
1	+149.545323	+1.614164	1.82	22.16	1211	B
2	+150.440339	+1.754854	1.36	21.90	204	A
3	+150.180910	+1.714817	1.90	22.22	383	C
4	+150.066345	+1.772114	2.11	19.85	7	C
5	+149.489741	+1.736721	1.08	21.18	1744	C
6	+150.646190	+1.840283	1.91	20.28	1018	B
7	+150.091078	+1.935850	2.53	21.13	136	B
8	+149.632091	+1.882368	2.38	20.41	260	B
9	+150.670701	+2.091367	1.30	21.19	458	B
10	+149.894802	+2.109357	0.72	20.44	845	B
11	+149.856184	+2.112118	2.33	21.61	1321	B
12	+150.549644	+2.140845	0.91	22.61	584	B
13	+150.259607	+2.209858	0.88	20.16	1275	B
14	+150.117743	+2.266765	0.86	20.34	1597	B
15	+149.730719	+2.147258	1.82	20.62	21	B
16	+149.644851	+2.135518	1.74	21.55	189	B
17	+150.656039	+2.447838	1.51	20.81	308	A
18	+150.411971	+2.308876	1.86	19.93	1069	B
19	+150.095108	+2.300498	0.43	18.57	1593	C
20	+150.085701	+2.297656	0.91	21.86	300	B
21	+150.085616	+2.364097	1.79	18.72	506	B
22	+150.106308	+2.432955	1.82	21.34	1674	C
23	+149.961446	+2.349389	1.16	21.60	208	B
24	+149.628310	+2.354862	0.88	21.82	1820	B
25	+149.722715	+2.428631	1.19	21.60	359	C
26	+150.571395	+2.506658	1.55	20.18	66	C
27	+150.624611	+2.540319	0.88	18.97	910	C
28	+150.694125	+2.547939	1.87	21.52	292	C
29	+150.317117	+2.531471	2.71	20.62	1451	B
30	+150.141624	+2.464563	1.65	20.82	86	B
31	+150.063881	+2.605824	1.47	21.50	979	B
32	+149.878942	+2.574346	0.97	21.78	573	A
33	+149.542116	+2.495012	2.23	19.03	1505	B
34	+150.747020	+2.666027	1.96	21.36	153	C
35	+150.548642	+2.766168	0.84	18.91	1595	B
36	+150.329391	+2.671669	2.44	21.72	1027	B
37	+150.284903	+2.674951	1.53	19.08	321	A
38	+150.250216	+2.763947	1.60	20.72	1515	B
39	+150.101284	+2.703268	1.74	22.58	932	B
40	+150.217548	+2.659542	1.11	23.18	1567	C
41	+149.855932	+2.650953	0.87	21.97	1345	B
42	+149.621847	+2.733148	2.33	21.31	1319	B
43	+150.644507	+2.808898	1.02	21.94	1098	B
44	+150.443480	+2.847808	1.55	21.66	1442	C
45	+150.104053	+2.844371	2.24	20.66	222	B
46	+149.611769	+2.809775	2.21	22.29	328	B
47[b]	+150.052500	+2.337500	0.90	19.28	2470	A
48[b]	+150.057917	+2.380278	1.65	18.89	910	B
49[b]	+150.076667	+2.645833	0.40	23.60	392	A
50[b]	+150.159167	+2.692500	0.74	20.39	13610	A
51[b]	+150.198333	+1.839722	0.70	20.65	2916	A
52[b]	+150.205000	+1.857778	2.22	19.61	693	C
53[b]	+150.210833	+2.816944	1.90	21.72	5333	A
54[b]	+150.236250	+2.207222	1.20	18.70	4041	B
55[b]	+150.352083	+1.855833	0.84	22.43	5125	B
56[b]	+150.570000	+2.498611	1.96	19.98	3521	B
57[b]	+150.614583	+2.080833	1.62	21.94	1495	C
58[b]	+150.725000	+2.241667	1.54	18.85	5783	B
59[b]	+149.494167	+2.256944	0.35	22.27	3868	B
60[b]	+149.737500	+1.996944	2.15	20.05	1164	C
61[b]	+149.811250	+2.205278	1.86	23.25	4700	C
62[b]	+149.840417	+2.110556	0.80	20.34	9218	A
63[b]	+149.949167	+2.797778	2.55	19.83	1	C
64[b]	+150.040417	+2.415278	2.63	19.49	8071	B

Table 2
(Continued)

Lens	R.A. (deg)	Decl. (deg)	Einstein Radius (arcsec)	Magnitude (AB)	Absolute Rank In 236,000	Average Grade[a] A/B/C
65[b]	+150.196250	+2.491944	2.00	19.56	13476	A
66[b]	+150.211667	+2.065833	0.66	22.31	1197	B
67[b]	+150.232083	+1.639167	1.05	20.86	616	A
68[b]	+150.272083	+2.758611	1.00	20.38	3204	B
69[b]	+150.334167	+1.764167	1.28	21.31	1300	B
70[b]	+150.450417	+2.390278	1.43	18.81	216	C
71[b]	+150.535417	+2.239444	1.59	20.06	8647	B
72[b]	+150.584167	+2.393056	1.04	20.99	283	B
73[b]	+150.587917	+2.577778	1.57	19.35	9564	C
74[b]	+150.650000	+2.801944	1.27	20.15	3865	B
75[b]	+149.450000	+1.923333	2.21	22.15	1236	A
76[b]	+149.461250	+1.938611	0.73	19.19	9936	B
77[b]	+149.467083	+2.349167	0.98	20.27	5965	B
78[b]	+149.475417	+1.997778	1.33	22.23	204	B
79[b]	+149.523333	+2.070278	1.61	23.00	3764	B
80[b]	+149.528333	+1.969167	1.50	19.14	2442	B
81[b]	+149.624583	+1.626111	2.97	19.95	12066	B
82[b]	+149.629167	+1.725556	1.17	21.06	2092	A
83[b]	+149.672500	+2.779444	0.93	19.65	177	B
84[b]	+149.733750	+2.798611	2.97	19.54	5951	C
85[b]	+149.870833	+1.764722	1.56	19.99	200	B
86[b]	+149.879583	+2.041389	1.48	19.20	894	B
87[b]	+149.883333	+2.171667	0.68	21.92	51	B
88[b]	+149.902917	+2.605833	2.40	19.14	7344	C
89[b]	+149.912917	+2.512222	0.70	20.08	93	B
90[b]	+149.918333	+2.548056	1.53	19.45	186	B
91[b]	+149.929583	+2.471111	1.64	19.43	2684	C
92[b]	+149.998750	+2.063333	1.20	21.62	44	B

Notes. The first column corresponds to the image number in Figure 10.
[a] Grade A corresponds to images that are clearly a strong gravitational lens. Grade B lenses correspond to images that are most likely a lens, but there is a chance they could also be artifacts, noise, structures in elliptical galaxies, satellite galaxies, tidally interacting galaxies, etc. Grade C lenses consist of images that are most likely not a lens, but there is a chance they might be gravitationally lensed.
[b] These marked lenses were previously cataloged by Faure et al. (2008).

modified ConvNets that do not suffer from layer saturation like ordinary ConvNets do. After adding more than 50 layers, the accuracy of ordinary ConvNets no longer improves and the training becomes more challenging. He et al. (2016) were able to overcome this issue by providing residual maps in between layers, which has been employed by Lanusse et al. (2018). They have simulated LSST mock observations in a single band and have trained and tested their network on these images. Jacobs et al. (2017) have trained their ConvNet using multiple color bands and have applied it to the Canada–France–Hawaii Telescope Legacy Survey. Petrillo et al. (2017) have searched for lenses in Kilo Degree Survey by training their ConvNet on cataloged luminous red galaxies.

In our independently developed work, we focus on the morphology of the lenses and only rely on one color band, similar to Petrillo et al. (2017) and Lanusse et al. (2018). Our lens simulation method is very similar to that of Petrillo et al. (2017), as we both merge simulated arcs with real images of galaxies to preserve the complexity of the physical data. In contrast to others, we do not discriminate against different sources found in the COSMOS field. That is, artifacts, stars, and other sources have been included in our training data set so LENSFLOW can be directly applied to fields without a need for a catalog with galaxy type information. The deepness of our ConvNet is comparable to Petrillo et al. (2017) and Jacobs et al. (2017) but it is shallower than Lanusse et al. (2018). As mentioned in Jacobs et al. (2017), the morphology of lenses are much simpler than the morphology of daily objects and human faces, which extremely deep ConvNets are developed for. However, the cost to performance ratio of ConvNets with varying deepness has not been studied yet. The effectiveness of deeper ConvNets cannot be compared between ours (and Petrillo et al. 2017) and Lanusse et al. (2018) since this work did not apply their algorithm to physical data. However, Lanusse et al. (2018) studied the change in the performance of their ConvNet by varying the Einstein radii and signal-to-noise ratio of their lenses.

A catalog of the strong gravitational lenses in the COSMOS field has previously been generated (Faure et al. 2008) by looking at early-type bright galaxies in the redshift range of $0.2 \leqslant z \leqslant 1.0$ in specific environments and by visually inspecting and cataloging 60 high- and low-quality lens candidates. In contrast, we have examined 236,000 sources in the *HST*/ACS i-band of the COSMOS field. In this paper, we reported our sample of gravitational lenses and presented an introduction to neural networks including ConvNets. Furthermore, we laid out the procedure for constructing simulated images for training and testing LENSFLOW. The architecture of LENSFLOW and its performance on test data constructed from real lenses were also presented. Finally, we used LENSFLOW to identify new lens candidates using *HST* data. Scanning all of the *HST*/ACS images in the COSMOS field roughly took 140 seconds on one GPU with 3840 NVIDIA CUDA cores. This corresponds to scanning 1.7 thousand 100 × 100-pixel images per second (or equivalently 4.2 $\text{arcmin}^2\ \text{s}^{-1}$ for the *Hubble ACS* camera). This speed is suitable for all-sky surveys and the computation time can be reduced further by increasing the number of employed GPUs.

We wish to thank the referee for reading the original manuscript and providing useful feedback. Financial support for this paper was provided by NSF grant AST-1313319 and GAANN P200A150121. We are also thankful for the donated GPU by NVIDIA Grant Program. Figures 1 and 6 were generated using http://www.draw.io. The backbone of our algorithm was initially inspired by Hvass Laboratories on Github (Pedersen 2016). Figure 2 was plotted using his code as well. We would like to thank Dr. Emre Neftci, Dr. Charless Fowlkes, Dr. Pierre Baldi for their valuable feedback. We would also like to thank Noah Ghotbi and Aroosa Ansari for their assistance.

Appendix A
Remarks on the LENSFLOW Code

We have developed LENSFLOW (Pourrahmani et al. 2018) in the Wolram Language (a.k.a MATHEMATICA), using its image processing and state-of-the-art ML functionalities and it can be accessed on Github (see footnote 2). The main notebook is called LENSFLOW, which contains all the deep learning portions of the code. Other notebooks are used for lens simulation, image normalization, etc. If the user does not have access to MATHEMATICA, they can download CDF Player for free to view the code. We will also provide a PDF of the main notebook with documentation alongside the code. From a practical perspective, it is important to store the images as JPEG files or other compressed formats since non-compressed image formants such as FITS occupy a significantly larger memory and storage volume and loading these images to memory will require much longer time. Training LENSFLOW during each phase on a GPU with 3840 NVIDIA CUDA cores takes less than 5 minutes for the training data set discussed in this paper. MATHEMATICA uses MXNET, so a trained network can be easily transferred to other languages. We initially developed our algorithm using TENSORFLOW, later, with the adoption of KERAS in Python 3.5.2 in Jupyter notebooks. These codes will be provided as extras. Even though they are not polished or fully developed, the codes is briefly documented in the Jupyter notebooks and may contain useful functions for data curation, helping the user to go from tiles to cutouts around extracted sources or automatically generating random arcs using LENSTOOL.

Appendix B
Identified and Recovered Lenses

LENSFLOW was able to identify 92 lenses in the COSMOS field, 46 of which were new and the rest were previously reported in Faure et al. (2008). The coordinates, Einstein radii, magnitudes of the brightest object, the LENSFLOW rankings of the lens among 236,000 images, and their grades are reported in Table 2. The corresponding images are shown in Figure 10.

ORCID iDs

Milad Pourrahmani https://orcid.org/0000-0003-3351-5986
Hooshang Nayyeri https://orcid.org/0000-0001-8242-9983
Asantha Cooray https://orcid.org/0000-0002-3892-0190

References

Abadi, M., Agarwal, A., Barham, P., et al. 2016, arXiv:1603.04467
Agnello, A., Lin, H., Buckley-Geer, L., et al. 2017, MNRAS, 472, 4038
Alard, C. 2006, arXiv:astro-ph/0606757
Atek, H., Richard, J., Kneib, J.-P., et al. 2015, ApJ, 800, 18
Blandford, R., & Narayan, R. 1992, ARA&A, 30, 311
Bolton, A. S., Burles, S., Koopmans, L. V., Treu, T., & Moustakas, L. A. 2006, ApJ, 638, 703
Broadhurst, T., Benítez, N., Coe, D., et al. 2005, ApJ, 621, 53
Calanog, J. A., Fu, H., Cooray, A., et al. 2014, ApJ, 797, 138
Capak, P., Aussel, H., Ajiki, M., et al. 2007, ApJS, 172, 99
Coe, D., Zitrin, A., Carrasco, M., et al. 2012, ApJ, 762, 32
Eigenbrod, A., Courbin, F., Vuissoz, C., et al. 2005, A&A, 436, 25
Faure, C., Kneib, J.-P., Covone, G., et al. 2008, ApJS, 176, 19
Fu, H., Jullo, E., Cooray, A., et al. 2012, ApJ, 753, 134
Gavazzi, R., Marshall, P. J., Treu, T., & Sonnenfeld, A. 2014, ApJ, 785, 144
Glorot, X., & Bengio, Y. 2010, in Proc. Machine Learning Research 9, Proc. 13th Int. Conf. Artificial Intelligence and Statistics, ed. Y. W. Teh & M. Titterington (Chia Laguna Resort, Sardinia: PMLR), 249
Goobar, A., Amanullah, R., Kulkarni, S. R., et al. 2017, Sci, 356, 291
Grogin, N. A., Kocevski, D. D., Faber, S., et al. 2011, ApJS, 197, 35
He, K., Zhang, X., Ren, S., & Sun, J. 2016, in Proc. IEEE Conf. Computer Vision and Pattern Recognition, ed. R. Bajcsy (Washington, DC: IEEE), 770
Heymans, C., Van Waerbeke, L., Miller, L., et al. 2012, MNRAS, 427, 146
Jacobs, C., Glazebrook, K., Collett, T., More, A., & McCarthy, C. 2017, MNRAS, 471, 167
Jullo, E., Kneib, J.-P., Limousin, M., et al. 2007, NJPh, 9, 447
Kaiser, N., & Squires, G. 1993, ApJ, 404, 441
Kingma, D. P., & Ba, J. 2014, arXiv:1412.6980
Koekemoer, A. M., Faber, S., Ferguson, H. C., et al. 2011, ApJS, 197, 36
Komatsu, E., Dunkley, J., Nolta, M., et al. 2009, ApJS, 180, 330
Krizhevsky, A., & Hinton, G. 2009, Learning Multiple Layers of Features from Tiny Images (State College, PA: Citeseer)
Krizhevsky, A., Sutskever, I., & Hinton, G. E. 2012, in Proc. Advances Neural Information Processing Systems 25 Conf., ed. F. Pereira et al. (La Jolla, CA: NIPS Foundation), https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks
Lanusse, F., Ma, Q., Li, N., et al. 2018, MNRAS, 473, 3895
Lenzen, F., Schindler, S., & Scherzer, O. 2004, A&A, 416, 391
Lotz, J., Koekemoer, A., Coe, D., et al. 2017, ApJ, 837, 97
Marshall, P. J., Lintott, C. J., & Fletcher, L. N. 2015, ARA&A, 53, 247
More, A., Cabanac, R., More, S., et al. 2012, ApJ, 749, 38
More, A., Suyu, S. H., Oguri, M., More, S., & Lee, C.-H. 2017, ApJL, 835, L25
More, A., Verma, A., Marshall, P. J., et al. 2016, MNRAS, 455, 1191
Nayyeri, H., Cooray, A., Jullo, E., et al. 2017, ApJ, 844, 82
Nayyeri, H., Keele, M., Cooray, A., et al. 2016, ApJ, 823, 17
Nielsen, M. 2016, Neural Networks and Deep Learning, http://neuralnetworksanddeeplearning.com/chap3.html
Oesch, P., Bouwens, R., Illingworth, G., et al. 2015, ApJ, 808, 104
Oke, J., & Gunn, J. 1983, ApJ, 266, 713
Pedersen, M. E. H. 2016, Hvass-Labs, https://github.com/Hvass-Labs/TensorFlow-Tutorials
Peng, C. Y., Impey, C. D., Rix, H.-W., et al. 2006, ApJ, 649, 616
Petrillo, C. E., Tortora, C., Chatterjee, S., et al. 2017, MNRAS, 472, 1129
Postman, M., Coe, D., Benítez, N., et al. 2012, ApJS, 199, 25
Pourrahmani, M., Nayyeri, H., & Cooray, A. 2018, LensFlow, Zenodo, doi:10.5281/zenodo.1163024
Refsdal, S., & Bondi, H. 1964, MNRAS, 128, 295
Rodney, S. A., Strolger, L.-G., Kelly, P. L., et al. 2016, ApJ, 820, 50
Scoville, N., Abraham, R., Aussel, H., et al. 2007, ApJS, 172, 38
Spilker, J. S., Marrone, D. P., Aravena, M., et al. 2016, ApJ, 826, 112
Suyu, S., Auger, M., Hilbert, S., et al. 2013, ApJ, 766, 70
Suyu, S., Treu, T., Hilbert, S., et al. 2014, ApJL, 788, L35
Tegmark, M., Strauss, M. A., Blanton, M. R., et al. 2004, PhRvD, 69, 103501
Timmons, N., Cooray, A., Riechers, D. A., et al. 2016, ApJ, 829, 21
Treu, T. 2010, ARA&A, 48, 87
Treu, T., & Marshall, P. J. 2016, A&ARv, 24, 11
Treu, T., Schmidt, K., Brammer, G., et al. 2015, ApJ, 812, 114
Velander, M., van Uitert, E., Hoekstra, H., et al. 2014, MNRAS, 437, 2111
Wardlow, J. L., Cooray, A., De Bernardis, F., et al. 2012, ApJ, 762, 59
Weinberg, D. H., Mortonson, M. J., Eisenstein, D. J., et al. 2013, PhR, 530, 87
Wilson, D., Cooray, A., Nayyeri, H., et al. 2017, ApJ, 848, 30