MNRAS **000**, 1–13 (2015) Preprint 10 August 2016 Compiled using MNRAS LaTeX style file v3.0

Star-galaxy Classification Using Deep Convolutional Neural Networks

Edward J. Kim[1]⋆ and Robert J. Brunner[1,2,3,4]
[1]*Department of Physics, University of Illinois, Urbana, IL 61801 USA*
[2]*Department of Astronomy, University of Illinois, Urbana, IL 61801 USA*
[3]*Department of Statistics, University of Illinois, Champaign, IL 61820 USA*
[4]*National Center for Supercomputing Applications, Urbana, IL 61801 USA*

Accepted XXX. Received YYY; in original form ZZZ

ABSTRACT
Most existing star-galaxy classifiers use the reduced summary information from catalogs, requiring careful feature extraction and selection. The latest advances in machine learning that use deep convolutional neural networks allow a machine to automatically learn the features directly from data, minimizing the need for input from human experts. We present a star-galaxy classification framework that uses deep convolutional neural networks (ConvNets) directly on the reduced, calibrated pixel values. Using data from the Sloan Digital Sky Survey (SDSS) and the Canada-France-Hawaii Telescope Lensing Survey (CFHTLenS), we demonstrate that ConvNets are able to produce accurate and well-calibrated probabilistic classifications that are competitive with conventional machine learning techniques. Future advances in deep learning may bring more success with current and forthcoming photometric surveys, such as the Dark Energy Survey (DES) and the Large Synoptic Survey Telescope (LSST), because deep neural networks require very little, manual feature engineering.

Key words: methods: data analysis – techniques: image processing – methods: statistical – surveys – stars: statistics – galaxies:statistics.

1 INTRODUCTION

Currently ongoing and forthcoming large-scale photometric surveys, such as the Dark Energy Survey (DES) and the Large Synoptic Survey Telescope (LSST), aim to collect photometric data for hundreds of millions to billions of stars and galaxies. Due to the sheer volume of data, it is not possible for human experts to manually classify them, and the separation of photometric catalogs into stars and galaxies has to be automated. Furthermore, any classification approach must be probabilistic in nature. A fully probabilistic classifier enables a user to adopt probability cuts to obtain a pure sample for population studies, or to optimize the allocation of observing time by selecting objects for follow-up. Ideally, however, the probability estimates themselves would be retained for all sources and used in subsequent analyses to improve or enhance a particular measurement (Ross et al. 2011; Seo et al. 2012).

With machine learning, we can use algorithms to automatically create accurate source catalogs with well-calibrated posterior probabilities. Machine learning techniques have been a popular tool in many areas of astronomy (Ball et al. 2007; Ball et al. 2008; Banerji et al. 2010; Carrasco Kind & Brunner 2013, 2014a,b; Ivezić et al. 2014; Kamdar et al. 2016a,b). Artificial neural networks were first applied to the problem of star-galaxy classification in the work of Odewahn et al. (1992), and they have become a core part of the astronomical image processing software `SExtractor` (Bertin & Arnouts 1996). Other successfully implemented examples of applying machine learning to the star-galaxy classification problem include decision trees (Weir et al. 1995; Suchkov et al. 2005; Ball et al. 2006; Sevilla-Noarbe & Etayo-Sotos 2015), Support Vector Machines (Fadely et al. 2012), and classifier combination strategies (Kim et al. 2015).

Almost all star-galaxy classifiers published in the literature use the reduced summary information available from astronomical catalogs. Constructing catalogs requires careful engineering and considerable domain expertise to transform the reduced, calibrated pixel values that comprise an image into suitable features, such as magnitudes or shape information of an object. In a branch of machine learning called *deep learning* (LeCun et al. 2015), features are not designed by human experts; they are learned directly from data by deep neural networks. Deep learning methods learn multiple levels of features by transforming the feature at one level into a more abstract feature at a higher level. For example, when an array of pixel values is used as input to a deep learning method, the features in the first layer might represent locations and orientations of edges. The second layer could assemble particular arrangements of edges into more complex shapes, and subsequent layers would detect objects as combinations of low-level features. These multiple layers of abstraction progressively amplify aspects of the input that

⋆ jkim575@illinois.edu

© 2015 The Authors

are important for classification tasks. Deep learning has been applied successfully to galaxy morphological classification in Sloan Digital Sky Survey (SDSS; Dieleman et al. 2015) and Cosmic Assembly Near-infrared Deep Extragalactic Legacy Survey (CANDELS; Huertas-Company et al. 2015) and to photometric redshift estimation (Hoyle 2015), but it has not yet been applied to the problem of source classification.

In this paper, we present a star-galaxy classification framework that uses a convolutional neural network (ConvNet) model directly on the images from the SDSS and the Canada-France-Hawaii Telescope Lensing Survey (CFHTLenS). We compare its performance with a standard machine learning technique that uses the reduced summary information from catalogs, and we demonstrate that our ConvNet model is able to produce accurate and well-calibrated probabilistic classifications with very little feature engineering by hand. In Section 2, we describe the data sets used in this paper and the pre-processing steps for preparing the image data sets. We provide a brief overview of deep learning and ConvNets in Section 3, and discuss our strategy for preventing overfitting in Section 4. In Section 5, we describe a state-of-the-art tree-based machine learning algorithm, to which the performance of our ConvNet model is compared. We present the main results of our ConvNet model in Section 6, and we outline our conclusions in Section 7.

2 DATA

To demonstrate the performance of our ConvNet model, we use photometric and spectroscopic data sets with different characteristics and compositions. In this section, we briefly describe these data sets and the image pre-processing steps for retrieving cutout images.

2.1 Sloan Digital Sky Survey

The Sloan Digital Sky Survey (SDSS; York et al. 2000) phases I–III obtained photometric data in five bands, u, g, r, i, and z, covering 14,555 square degrees, more than one-third of the entire sky. The resulting catalog contains photometry of over 300 million stars and galaxies with a limiting magnitude of $r \approx 22$, making the SDSS one of the largest sky surveys ever undertaken. The SDSS also conducted an expansive spectroscopic follow-up of more than three million stars and galaxies (Eisenstein et al. 2011). In this paper, we use a subset of the photometric and spectroscopic data contained within the Data Release 12 (DR12; Alam et al. 2015), which is publicly available through the online CasJobs server[1] (Li & Thakar 2008).

Using the CasJobs server, we randomly select a total of 65,000 sources, which are either stars or galaxies. We exclude some bad photometric observations as follows. We consider only objects ~~whose r-band magnitude is less than 40; there are~~ with no warning flags in the spectroscopic measurement (`zWarning = 0`); the half-light radius in the r band is less than 30 arc seconds as measured by the exponential and de Vaucouleurs light profiles; the error on the spectroscopic redshift measurement is less than 0.1; and the spectroscopic redshift is less than 2.

To create training images, we obtain the image FITS files for SDSS fields containing these objects in five photometric bands: u, g, r, i, and z. We use the astrometry information in the FITS headers in the `Montage`[2] software to align each image to the reference (r-band) image. We then use `SExtractor` to find the pixel positions of the 65,000 objects we have selected, and to center each object in the cutout image. Magnitudes in the SDSS photometric catalog are expressed as inverse hyperbolic sine magnitudes (also known as luptitudes; Lupton et al. 1999), and we follow the SDSS convention and convert all flux values to luptitudes. Finally, in order to account for the effect of Galactic dust, extinction corrections in magnitudes are applied following Schlegel et al. (1998). In the end, we have cutout images of size 48×48 pixels with luptitude values in each pixel. We note that we have experimented with increasing the pixel dimensions to 60×60 and 72×72 pixels, but do not find noticeable improvement in the performance of our model.

In the end, we have 17,344 stars and 47,656 galaxies available for the training and testing processes. The apparent magnitudes range from $10.7 < r < 23.1$, and the galaxies in this sample have a mean redshift of $z \sim 0.36$. ~~The $ugriz$ color-color space of the sample is plotted in Figure ?.~~ We randomly split the objects into training, held-out validation, and blind test sets of size 40,000, 10,000, and 15,000, respectively. We note that cross-validation is often avoided in deep learning in favor of hold-out validation, since cross-validation is computationally expensive. We also note that we perform a blind test, and the test set is not used in any way to train or calibrate the algorithms. The first two panels of Figure 8 show the number of objects and the fraction of stars in the test set as functions of r-band magnitude. Similarly, Figure 10 shows the number of objects and the fraction of stars in the test set as functions of $g-r$ color. The normalized kernel density estimate distributions for the training and validation sets are almost identical to those of the test set, and they are nearly indistinguishable when overlapped. We do not show the distributions for the training and validation sets in Figures 8 and 10 to avoid cluttering the plots.

2.2 Canada-France-Hawaii Telescope Lensing Survey

We also use photometric data from the Canada-France-Hawaii Telescope Lensing Survey (CFHTLenS[3]; Heymans et al. 2012; Erben et al. 2013; Hildebrandt et al. 2012). This catalog consists of more than twenty five million objects with a limiting magnitude of $i_{\mathrm{AB}} \approx 25.5$. It covers a total of 154 square degrees in the four fields (named W1, W2, W3, and W4) of the CFHT Legacy Survey (CFHTLS; Gwyn 2012) observed in the five photometric bands: u, g, r, i, and z.

We have cross-matched reliable spectroscopic galaxies from the Deep Extragalactic Evolutionary Probe Phase 2 (DEEP2; Davis et al. 2003; Newman et al. 2013), the Sloan Digital Sky Survey Data Release 10 (Alam et al. 2015, SDSS-DR10), the VIsible imaging Multi-Object Spectrograph (VIMOS) Very Large Telescope (VLT) Deep Survey (VVDS; Le Fèvre et al. 2005; Garilli et al. 2008), and the VIMOS Public Extragalactic Redshift Survey (VIPERS; Garilli et al. 2014). We have selected only sources with very secure redshifts and no bad flags (quality flags -1, 3, and 4 for DEEP2; quality flag 0 for SDSS; quality flags 3, 4, 23, and 24 for VIPERS and VVDS).

We obtain FITS images for each 1 square degree CFHTLenS pointing that contains objects with spectroscopic labels. We create cutout images of size 96×96 pixels by using a similar method to

[1] http://skyserver.sdss.org/casjobs/

[2] http://montage.ipac.caltech.edu/

[3] http://www.cfhtlens.org/

that described in Section 2.1. Finally, images are downscaled to 48×48 pixels to reduce the computational cost.

In the end, we have 8,545 stars and 57,843 galaxies available for the training and testing processes. The apparent magnitudes range from $13.9 < r < 25.6$, and the galaxies in this sample have a mean redshift of $z \sim 0.59$. ~~The *ugriz* color-color space of the sample is plotted in Figure ?.~~ We randomly split the objects into training, held-out validation, and blind test sets of size 40,000, 10,000, and 13,278, respectively. Figures 2 and 4 show the distribution of objects in the test set as functions of i-band magnitude and $g-r$ color. We do not show the distributions for the training and validation sets, since the normalized kernel density estimate distributions for the training and validation sets are almost identical to those of the test set.

3 DEEP LEARNING

3.1 Neural Networks

~~A neuron in the human brain receives signals from other neurons through synaptic connections. If the combination of these signals exceeds a certain threshold, the neuron will fire and send a signal to other neurons. Intelligence is believed to be the collective effect of approximately 10^{11} neurons firing.~~ An artificial neuron in most artificial neural networks is represented as a mathematical function that models a biological neural structure (Aggarwal 2014). A schematic representation is shown in Figure 1a. Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ be a vector of inputs to a given neuron, $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$ be a vector of weights, and b be the bias. Then, the output of the neuron is

$$y = \sigma(\boldsymbol{w} \cdot \boldsymbol{x} + b), \tag{1}$$

where σ is the activation function (or *non-linearity*). ~~The sigmoid function, $\sigma(x) = 1/(1+e^{-x})$, has frequently been used as an activation function, but it has recently fallen out of favor, and the hyperbolic tangent function, $\sigma(x) = \tanh(x)$, is generally used instead.~~ The most popular non-linearity at present ~~, however,~~ is the rectified linear unit (ReLU; Nair & Hinton 2010), $\sigma(x) = \max(0, x)$. ReLUs generally allow much faster training of deep neural networks with many layers. However, ReLU units can sometimes result in dead neurons whose output is always zero. To mitigate this problem, we use leaky ReLUs (Maas et al. 2013) that have a small, non-zero slope in the negative region,

$$\sigma(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0.01x & \text{if } x < 0. \end{cases} \tag{2}$$

Many deep learning models use feedforward neural network architectures with multiple layers, where each neuron in one layer is connected to the neurons of the subsequent layer. A schematic representation is shown in Figure 1b. All layers except the input and output layers are conveniently called hidden layers. ~~With multiple hidden layers, deep neural networks can model complex nonlinear relationships, and they have had great success in multimedia tasks where humans traditionally outperformed computers, such as speech recognition, language processing, and image classification.~~

The goal of learning is to find a set of weights and biases such that, given N samples, the output from the network $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)$ approximates the desired output $\hat{\boldsymbol{y}} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_N)$ as closely as possible for all input $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$. We can formulate this as the minimization of a loss function $L(\boldsymbol{y}, \hat{\boldsymbol{y}})$ over the training data. In this work, we use *cross-entropy*,

$$L(y_j, \hat{y}_j) = -\hat{y}_j \log_2 y_j - (1-\hat{y}_j)\log_2(1-y_j), \tag{3}$$

where $\hat{y}_j$ is the actual truth value (e.g., 0 or 1) of the j-th data, and y_j is the probability prediction made by the model. We compute the loss function by taking the average of all cross-entropies in the sample. Thus, given N samples, the loss function becomes

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\frac{1}{N}\sum_{j=1}^{N} \hat{y}_j \log_2 y_j + (1-\hat{y}_j)\log_2(1-\hat{y}_j). \tag{4}$$

To find the weights $\boldsymbol{w}$ and biases $\boldsymbol{b}$ which minimize the loss, we use a technique called *gradient descent*, where we use the following rules to update the parameters in each layer l:

$$\begin{aligned} \boldsymbol{w}_l \rightarrow \boldsymbol{w}'_l &= \boldsymbol{w}_l - \eta \frac{\partial L}{\partial \boldsymbol{w}_l} \\ \boldsymbol{b}_l \rightarrow \boldsymbol{b}'_l &= \boldsymbol{b}_l - \eta \frac{\partial L}{\partial \boldsymbol{b}_l}, \end{aligned} \tag{5}$$

where η is a small, positive number known as the *learning rate*. The gradients can be computed using the backpropagation procedure (Rumelhart et al. 1988). A common approach to speed up training is to split the training data into mini-batches (LeCun et al. 1998a). In mini-batch gradient descent, instead of computing the gradients in Equation 5 for the entire training data, we only compute the gradient of randomly chosen training examples at each step. As training examples are usually correlated, the gradient computed from each mini-batch is a good approximation of the overall gradient (Bottou 1998). As a result, mini-batch gradient descent results in much faster convergence. However, there is a trade-off: the lower the batch size is, the lower the convergence rate will be; the higher the batch size is, the longer it will take to compute the gradient at each step (Bousquet & Bottou 2008). Thus, a moderate batch size, combined with a decaying learning rate, is generally used in practice.

We define an *epoch* as a single, complete pass through the training data, and full training usually requires many epochs. At the end of each epoch, we evaluate the loss function on the validation set, and the model that minimizes the validation loss is chosen as the best model.

~~The gradients in Equation 5 can be computed using the *backpropagation* procedure, which is nothing more than an application of the chain rule for derivatives. The derivative of the loss function with respect to the input of a unit can be computed by working backwards from the gradient with respect to the output of the unit. At the output layer, the derivative of the loss function with respect to the output is given by~~
~~$\frac{\partial L}{\partial y_i} = y_i - \hat{y}_i \; \frac{\partial L}{\partial x_i} = \frac{\partial L}{\partial y_i}\frac{\partial y_i}{\partial x_i}$.~~

~~At each hidden layer, the derivative of the loss function with respect to the output of each unit is a weighted sum of the derivatives with respect to the inputs of the subsequent layer, $\frac{\partial L}{\partial y_j} = \sum_i w_{ji}\frac{\partial L}{\partial x_i} \; \frac{\partial L}{\partial x_j} = \frac{\partial L}{\partial y_j}\frac{\partial y_j}{\partial x_j}$.~~

~~Finally, once the gradients $\partial L/\partial x_j$ have been computed, it is straightforward to compute the gradients for the weight, $\frac{\partial L}{\partial w_{jk}} = y_k \frac{\partial L}{\partial x_j}$.~~

3.2 Convolutional Neural Networks

The convolutional neural network (ConvNet; Fukushima 1980; LeCun et al. 1998b) is a type of deep, feedforward neural network that has recently become a popular approach in the computer vision

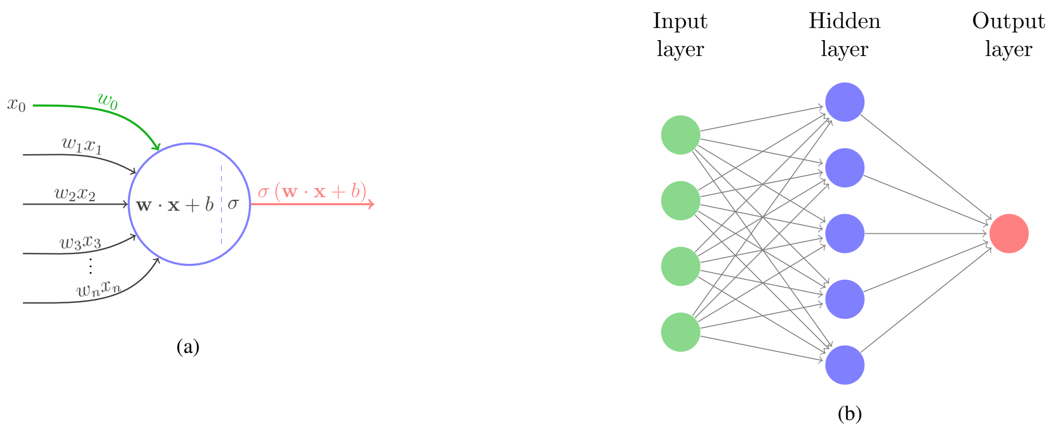

Figure 1. (a) A mathematical model of a biological neuron. (b) A schematic diagram of a neural network with one hidden layer.

community. In a typical ConvNet, the first few stages are composed of two types of layers: convolutional layers and pooling layers.

The input to a convolutional layer is an image, and the output channels of each layer are called *feature maps*. To produce output feature maps, we convolve each feature map with a set of weights called *filters*, and apply a non-linearity such as ReLU to the weighted sum of these convolutions. Different feature maps use different sets of filters, but all neurons in a feature map share the same set of filters. Mathematically, we replace the dot product in Equation 1 with a sum of convolutions. Thus, the k-th feature map is given by

$$y^k = \sigma\left(\sum_m \boldsymbol{w}_m^k * \boldsymbol{x}_m + b^k\right), \tag{6}$$

where we sum over the set of input feature maps, $*$ is the convolution operator, and $\boldsymbol{w}_m^k$ represent the filters.

Typically, a pooling layer computes the maximum of a local 2×2 patch in a feature map. Since the pooling layer aggregates the activations of neighboring units from the previous layer, it reduces the dimensionality of the feature maps and makes the model invariant to small shifts and distortions. Two or more layers of convolution and pooling are stacked, followed by more convolutional and fully-connected layers.

3.3 Neural Network Architecture

We present the overall architecture of our ConvNet model in Table 1. The network consists of eleven trainable layers. The first convolutional layer filters the $5 \times 44 \times 44$ input image (i.e., 44×44 images in five bands *ugriz*) with 32 square filters of size $5 \times 5 \times 5$. We have also experimented with using only three bands *gri* (for three channels of RGB) and four bands *ugri* and *griz* (corresponding to RGBA), and using only colors, e.g., $u-g$, $g-r$, $r-i$, and/or $i-z$, but we find that using magnitudes in all five bands *ugriz* yields the best performance.

The leaky ReLU non-linearity is applied to the output of the first convolutional layer (and all subsequent layers), and the second convolutional layer filters it with 32 filters of $32 \times 3 \times 3$. In the second convolutional layer (and all subsequent convolutional layers), we pad the input with zeros spatially on the border (i.e., the zero-padding is 1 pixel for 3×3 convolutional layers) such that the spatial resolution is preserved after convolution. Max-pooling with filters of size 2×2 follows the second convolutional layer. A stack of six additional convolutional layers, all with filters of size 3×3, is followed by three fully-connected layers. The first two fully-connected layers have 2048 channels each, and the third performs binary classification.

The output of the final fully-connected layer is fed to a *softmax* function. The softmax function is given by

$$P(G \mid \boldsymbol{x}) = \frac{e^{\boldsymbol{x} \cdot \boldsymbol{w}_G}}{\sum_i e^{\boldsymbol{x} \cdot \boldsymbol{w}_i}}, \tag{7}$$

where we sum over the different possible values of the class label (i.e., star or galaxy), and interpret its output as the posterior probability that an object is a galaxy (or a star). We note that we could also try to solve a regression problem, e.g., by normalizing the output values that the network produces for each class. However, we find that solving a regression problem instead of using the softmax function results in significantly worse performance.

We have performed a manual search to explore more than 200 combinations of different architectures and hyperparameters to find an architecture that minimizes the loss function (Equation 4) on the validation set of the SDSS data. The architecture described in this section provides the best performance on the SDSS validation set. To test how this model performs across different, related data, we use the same architecture on the CFHTLenS data set.

The architecture of Krizhevsky et al. (2012) uses relatively large receptive fields (11×11) in the first convolutional layers. Zeiler & Fergus (2014) and Dieleman et al. (2015) also use large receptive fields of 7×7 and 6×6 in the first convolution layer, respectively. However, we find that using a receptive field larger than 5×5 in the first convolutional layer leads to worse performance. This result is in agreement with Simonyan & Zisserman (2014), which features an extremely homogeneous architecture that only performs 3×3 convolutions. Using a large receptive field instead of a stack of multiple 3×3 convolutions leads to a shallower network, and it is often preferable to increase the depth by using smaller receptive fields. However, we find that replacing the first layer with a stack of two 3×3 convolutional layers increases the validation error, but we still follow Simonyan & Zisserman (2014) and add many 3×3 convolutions (with zero-padding) in the remaining layers.

The choice of momentum, learning rate, and initial weights is crucial for achieving high predictive performance and speeding up the learning process (Sutskever et al. 2013). To train our models, we use mini-batch gradient descent with a batch size of 128 and Nesterov momentum (Bengio et al. 2013) of 0.9. We initialize the learning rate η at 0.003 for all layers and decrease it linearly with the number of epochs from 0.003 to 0.0001 over 750 epochs. We

Table 1. Summary of ConvNet architecture and hyperparameters. Note that pooling layers have no learnable parameters.

type	filters	filter size	padding	non-linearity	initial weights	initial biases
convolutional	32	5×5	-	leaky ReLU	orthogonal	0.1
convolutional	32	3×3	1	leaky ReLU	orthogonal	0.1
pooling	-	2×2	-	-	-	-
convolutional	64	3×3	1	leaky ReLU	orthogonal	0.1
convolutional	64	3×3	1	leaky ReLU	orthogonal	0.1
convolutional	64	3×3	1	leaky ReLU	orthogonal	0.1
pooling	-	2×2	-	-	-	-
convolutional	128	3×3	1	leaky ReLU	orthogonal	0.1
convolutional	128	3×3	1	leaky ReLU	orthogonal	0.1
convolutional	128	3×3	1	leaky ReLU	orthogonal	0.1
pooling	-	2×2	-	-	-	-
fully-connected	2048	-	-	leaky ReLU	orthogonal	0.01
fully-connected	2048	-	-	leaky ReLU	orthogonal	0.01
fully-connected	2	-	-	softmax	orthogonal	0.01

also initialize the weights in each layer with random orthogonal initial conditions (Saxe et al. 2013). We use slightly positive values ($b = 0.01$ or 0.1) for all biases. We find initializing biases to a small constant value helps eliminate dead neurons by ensuring that all ReLU neurons fire in the beginning.

To implement our model, we use Python and the Lasagne library (Dieleman & et al. 2015), which is built on top of Theano (Theano Development Team 2016). The Theano library simplifies the use of GPU for computation, and using the GPU allows about an order of magnitude faster training than using just the CPU. We note that training our network takes about forty hours on an NVIDIA Tesla K40 GPU. In the interest of scientific reproducibility, we make all our code available at https://github.com/EdwardJKim/dl4astro.

4 REDUCING OVERFITTING

Our convolution neural network has 11×10^6 learnable parameters, while there are only 4×10^4 images in the training set. As a result, the model is very likely to *overfit* without regularization. In this section, we describe the techniques we used to minimize overfitting.

4.1 Data Augmentation

One common method to combat overfitting is to artificially increase the number of training data by using label-preserving transformations (Krizhevsky et al. 2012; Dieleman et al. 2015, 2016). Each image is transformed as follows:

- Rotation: Rotating an image does not change whether the object is a star or a galaxy. We exploit this rotational symmetry and randomly rotate each image by a multiple of 90°.
- Reflection: We flip each image horizontally with a probability of 0.5 to exploit mirror symmetry.
- Translation: We also have translational symmetry in the images. Given an image of size 48×48 pixels, we extract a random contiguous crop of size 44×44. Each cropping is equivalent to randomly shifting a 44×44 image by up to 4 pixels vertically and/or horizontally.
- Gaussian noise: We introduce random Gaussian noise to each pixel values by using a similar method to Krizhevsky et al. (2012).

In addition to artificially increasing the size of the data set, these data augmentation schemes make the resulting model more invariant to rotation, reflection, translation, and small noise in the pixel values. We also note that the data augmentation steps add almost no computational cost, as they are performed on the CPU while the GPU is training the ConvNets on images.

4.2 Dropout

We use a regularization technique called dropout (Hinton et al. 2012) in the fully-connected layers. Dropout consists of randomly setting to zero the output of each hidden neuron of the previous layer with probability 0.5. The weights of the remaining neurons are multiplied by 0.5 to preserve the scale of input values to the next layer. Since a neuron can be removed at any time, it cannot rely on the presence of other neurons in the same layer and is forced to learn more robust features.

4.3 Model Combination

To make final classifications, we use our ConvNet model to make 64 sets of predictions for 64 transformations of the input images: 4 rotations, 4 horizontal translations, and 4 vertical translations (with random horizontal reflections). Although we use an identical network architecture for all transformations, we consider each set of predictions as separate results from different models. Finally, we use a model combination technique known as Bayesian Model Combination (BMC; Monteith et al. 2011), which uses Bayesian principles to generate an ensemble combination of different models. Although the data augmentation step in Section 4.1 should make our ConvNet model invariant to these types of transformations, we find that applying BMC still results in a significant increase in performance. Although a complete description of BMC is beyond the scope of this paper, we provide a brief description. For more details, we refer the reader to Monteith et al. (2011).

~~The posterior probability that a source is a galaxy is given by $P(G|\boldsymbol{x},\boldsymbol{D},\boldsymbol{M},\boldsymbol{E}) = \sum_{e\in\boldsymbol{E}} P(G|\boldsymbol{x},\boldsymbol{M},e)P(e|\boldsymbol{D})$, where $\boldsymbol{x}$ is the input, $\boldsymbol{D}$ is the data set, $\boldsymbol{M}$ is the set of models (i.e., different transformations of input images), and e is an element in the ensemble space $\boldsymbol{E}$ of possible model combinations. By Bayes' Theorem, the posterior probability of e given $\boldsymbol{D}$ is given by $P(e|\boldsymbol{D}) = \frac{P(e)}{P(\boldsymbol{D})} \prod_{d\in\boldsymbol{D}} P(d|e) \propto P(e) \prod_{d\in\boldsymbol{D}} P(d|e)$. Here, $P(e)$ is the prior probability of e, which we assume to be~~

~~uniform. The product of $P(d|e)$ is over all individual data d in the training data $\boldsymbol{D}$, and $P(\boldsymbol{D})$ is merely a normalization factor and not important. For probabilistic classifiers, we can directly use the probabilistic predictions and write Equation **??** as $P(e|\boldsymbol{D}) \propto P(e)\prod_{i=1}^{N} \hat{y}_i y_i + (1-\hat{y}_i)(1-y_i)$,~~

~~Although the space $\boldsymbol{E}$ of potential model combinations is in principle infinite, we can produce a reasonable finite set of potential model combinations by using sampling techniques. In our implementation, the weights of each combination of the base classifiers is obtained by sampling from a Dirichlet distribution. We first set all alpha values of a Dirichlet distribution to unity. We then sample this distribution q times to obtain q sets of weights. For each combination, we assume a uniform prior and calculate $P(e|\boldsymbol{D})$ using Equation **??**. We select the combination with the highest $P(e|\boldsymbol{D})$, and update the alpha values by adding the weights of the most probable combination to the current alpha values. The next q sets of weights are drawn using the updated alpha values.~~

~~We continue the sampling process until we reach a predefined number of combinations, and we finally use Equation **??** to compute the posterior probability that a source is a star (or a galaxy). In this paper, we use a q value of three, and we consider a total of 1,000 model combinations.~~

5 TREES FOR PROBABILISTIC CLASSIFICATIONS

To compare the performance of ConvNets with machine learning algorithms that use standard photometric features, we use a machine learning framework called Trees for Probabilistic Classifications (TPC). TPC is a parallel, supervised machine learning algorithm that uses prediction trees and random forest techniques (Breiman et al. 1984; Breiman 2001) to produce a star-galaxy classification. A complete description of TPC is beyond the scope of this paper, and we refer the reader to Carrasco Kind & Brunner (2013) and Kim et al. (2015) for more details.

~~TPC is a part of a publicly available software package called MLZ~~

~~TPC uses classification trees, a type of prediction tree that is designed to provide a classification or predict a discrete category. Prediction trees are built by asking a sequence of questions that recursively split the data into branches until a terminal leaf is created that meets a stopping criterion (e.g., a minimum leaf size). The optimal split dimension is decided by choosing the attribute that maximizes the *Information Gain* (I_G), which is defined as~~

~~$I_G(D_{\text{node}}, X) = I_d(D_{\text{node}}) - \sum_{x \in \text{values}(X)} \frac{|D_{\text{node},x}|}{|D_{\text{node}}|} I_d\left(D_{\text{node},x}\right)$,~~

~~where D_{node} is the training data in a given node, X is one of the possible dimensions (e.g., magnitudes or colors) along which the node is split, and x are the possible values of a specific dimension X. $|D_{\text{node}}|$ and $|D_{\text{node},x}|$ are the size of the total training data and the number of objects in a given subset x within the current node, respectively. I_d is the impurity degree index, and TPC can calculate I_d from any of the three standard different impurity indices: *information entropy*, *Gini impurity*, and *classification error*. In this work, we use the information entropy, which is defined similarly to the thermodynamic entropy:~~

~~$I_d(D) = -f_g \log_2 f_g - \left(1 - f_g\right)\log_2\left(1 - f_g\right)$,~~

~~where f_g is the fraction of galaxies in the training data. At each node in our tree, we scan all dimensions to identify the split point that maximizes the information gain as defined by Equation **??**, and select the attribute that maximizes the overall impurity index.~~

Table 2. The definition of the classification performance metrics.

Metric	Meaning
AUC	Area under the Receiver Operating Curve
MSE	Mean squared error
c_g	Galaxy completeness
p_g	Galaxy purity
c_s	Star completeness
p_s	Star purity
$p_g(c_g = x)$	Galaxy purity at x galaxy completeness
$c_s(p_s = x)$	Star completeness at x star purity
CAL	Calibration error with overlapping binning
$\mid \Delta N_g \mid / N_g$	Absolute error in number of galaxies

~~In a technique called random forest, we create bootstrap samples (i.e., N randomly selected objects with replacement) from the input training data by sampling repeatedly from the magnitudes and colors using their measurement errors. We use these bootstrap samples to construct multiple, uncorrelated prediction trees whose individual predictions are aggregated to produce a star-galaxy classification for each source.~~

We train two TPC models on the SDSS data set by using different sets of attributes. The first model, which we denote TPC$_{\text{phot}}$, is trained with only nine photometric attributes: the extinction-corrected model magnitudes in five bands (u, g, r, i, z) and their corresponding colors ($u-g$, $g-r$, $r-i$, $i-z$). The second model, which we denote TPC$_{\text{morph}}$, is trained with the concentration parameter in each band in addition to the magnitudes and colors, for a total of fourteen dimensions. The concentration is defined as the difference between the ~~cmodel and~~ PSF magnitude (psfMag) and the composite model magnitude (cModelMag), i.e., concentration ≡ psfMag − cModelMag. The SDSS pipeline uses a parametric method based on the concentration, an object is classified as a galaxy if concentration > 0.145. We find that the concentration is an excellent morphological feature for star-galaxy separation, and including more morphological features does not show noticeable improvement in performance. The concentration is a good example of carefully handcrafted feature extraction; we show in Section 6 that ConvNets do not require such feature engineering.

We also train two models on the CFHTLenS data set. TPC$_{\text{phot}}$ is trained with the five magnitudes and their corresponding colors: u, g, r, i, z, $u-g$, $g-r$, $r-i$, and $i-z$. Since the CFHTLenS catalog does not provide the concentration parameter, TPC$_{\text{morph}}$ uses `SExtractor`'s `FLUX_RADIUS` (the half-light radius), `A_WORLD` (the semi-major axis), and `B_WORLD` (the semi-minor axis) for morphological features, in addition to the five magnitudes and their corresponding colors, for a total of twelve dimensions.

~~We note that TPC is able to utilize parallelization, and parallelized on 64 cores of Xeon E5 2.7 GHz processors, it takes 23 minutes to train 500 trees, for a total of 25 CPU hours.~~

6 RESULTS AND DISCUSSION

In this section, we first describe the performance metrics that were used for evaluating the models. We then present the classification performance of our ConvNet model on the CFHTLenS and SDSS data sets, and compare it with the performance of TPC.

6.1 Classification Metrics

Probabilistic classification models can be considered as functions that output a probability estimate of each source to be in one of the classes (e.g., a star or a galaxy). The simplest way to use the probability output is to convert it into a class label by using a threshold (a probability cut). Although many studies often choose the threshold by setting it to a fixed value, such as $P_{\rm cut} = 0.5$ (e.g., Henrion et al. 2011; Fadely et al. 2012), choosing 0.5 as a threshold is not the best choice for an unbalanced data set, where galaxies outnumber stars. Furthermore, setting a fixed threshold ignores the operating condition (e.g., science requirements, stellar distribution, misclassification costs) where the model will be applied.

6.1.1 Receiver Operating Characteristic Curve

When we have no information about the operating condition when evaluating the performance of classifiers, there are effective tools such as the Receiver Operating Characteristic (ROC) curve (Swets, Dawes & Monahan 2000). An ROC curve is a graphical plot that illustrates the true positive rate versus the false positive rate of a binary classifier as its classification threshold is varied. The Area Under the Curve (AUC) summarizes the curve information in a single number, and can be used as an assessment of the overall performance.

6.1.2 Completeness and Purity

In astronomical applications, the operating condition usually translates to the completeness and purity requirements of the star or galaxy sample. We define the galaxy *completeness* c_g (also known as recall or sensitivity) as the fraction of the number of true galaxies classified as galaxies out of the total number of true galaxies,

$$c_g = \frac{N_g}{N_g + M_g}, \tag{8}$$

where N_g is the number of true galaxies classified as galaxies, and M_g is the number of true galaxies classified as stars. We define the galaxy *purity* p_g (also known as precision or positive predictive value) as the fraction of the number of true galaxies classified as galaxies out of the total number of objects classified as galaxies,

$$p_g = \frac{N_g}{N_g + M_s}, \tag{9}$$

where M_s is the number of true stars classified as galaxies. Star completeness and purity are defined in a similar manner.

One of the advantages of a probabilistic classification is that the threshold can be adjusted to produce a more complete but less pure sample, or a less complete but more pure one. We can compare the performance of different classification techniques by assuming an arbitrary operating condition. For example, weak lensing science measurements of the DES require $c_g > 0.960$ and $p_g > 0.778$ to control both the statistical and systematic errors on the cosmological parameters, and $c_s > 0.250$ and $p_s > 0.970$ for stellar Point Spread Function (PSF) calibration (Soumagnac et al. 2015). Although these values will likely be different for the science cases of the CFHTLenS data, we adopt these values to compare the classification performance at a reasonable operating condition. Thus, we compute p_g at $c_g = 0.960$ and c_s at $p_s = 0.970$.

6.1.3 Mean Squared Error

We also use the mean squared error (MSE; also known as the Brier score (Brier 1950)) as a performance metric. We define MSE as

$$\mathrm{MSE} = \frac{1}{N}\sum_{j=1}^{N}\left(y_j - \hat{y}_j\right)^2, \tag{10}$$

The MSE can be considered as both a score function that quantifies how well a set of probabilistic predictions is calibrated, or a loss function.

6.1.4 Calibration Error

A fully probabilistic classifier predicts not only the class label, but also its confidence level on the prediction. In a well-calibrated classifier, the posterior class probability estimates should coincide with the proportion of objects that truly belong to a certain class. Probability *calibration curves* (or reliability curves; DeGroot & Fienberg 1983) are often used to display this relationship, where we bin the probability estimates and plot the fraction of positive examples versus the predicted probability in each bin (see Figures 5 and 11).

The problem with a binning approach is either too few or too many bins can distort the evaluation of calibration performance. Thus, we adopt a calibration measure based on overlapping binning (Caruana & Niculescu-Mizil 2004). We order the predicted values $P_{\rm class}$ and put the first 1,000 elements in the first bin. We calculate the true probability $P_{\rm gal}$ by counting the true galaxies in this bin. The calibration error for this bin is $\mid P_{\rm gal} - P_{\rm class} \mid$. We then repeat this for the second bin (2 to 1,001), the third bin (3 to 1,002), and so on, and average the binned calibration errors. Thus, the overall calibration error is given by

$$\mathrm{CAL} = \frac{1}{N-s}\sum_{b=1}^{N-s}\sum_{j=b}^{b+s-1}\left|P_{{\rm class},j} - \frac{\sum_{j=b}^{b+s-1} P_{{\rm gal},j}}{s}\right|, \tag{11}$$

where $s = 1000$ is the bin length, which is chosen approximately equal to the number of objects in the testing set divided by the number of bins used in the calibration curve, i.e., $s \approx N/10$.

6.1.5 Number of galaxies

Ideally, the probabilistic output of a classifier would be used in subsequent scientific analyses. For example, one can weight each object by the probability that it is a galaxy when measuring autocorrelation functions of luminous galaxies (Ross et al. 2011). In other words, given a well-calibrated classifier, instead of counting each galaxy equally, a galaxy could be counted as $P_{\rm class}$, the probability estimate. This should in principle remove the contamination effect of stars. For a perfect classifier, we can count the total number of galaxies in the sample by summing the values of $P_{\rm class}$. Thus, we measure the reliability of classifier output by the absolute error in the estimation of number of galaxies,

$$\frac{\mid \Delta N_g \mid}{N_g} = \frac{\left|N_g - \sum_{j=1}^{N} P_{{\rm class},j}\right|}{N_g}. \tag{12}$$

6.2 CFHTLenS

As described in Section 3.2, we train our ConvNet model by monitoring its performance on the validation set. Once we have finished training the model, we evaluate its performance on the blind test set. We also use the same training and validation sets to train and

Table 3. A summary of the classification performance metrics as applied to the CFHTLenS data. The definition of the metrics is summarized in Table 2. The bold entries highlight the best performance values within each column.

classifier	AUC	MSE	$p_g(c_g = 0.96)$	$c_s(p_s = 0.97)$	CAL	$\|\Delta N_g\|/N_g$
ConvNet	**0.9948**	0.0112	**0.9972**	0.8971	**0.0197**	**0.0029**
$\text{TPC}_{\text{morph}}$	0.9924	**0.0109**	0.9963	**0.9268**	0.0245	0.0056
TPC_{phot}	0.9876	0.0189	0.9927	0.8044	0.0266	0.0101

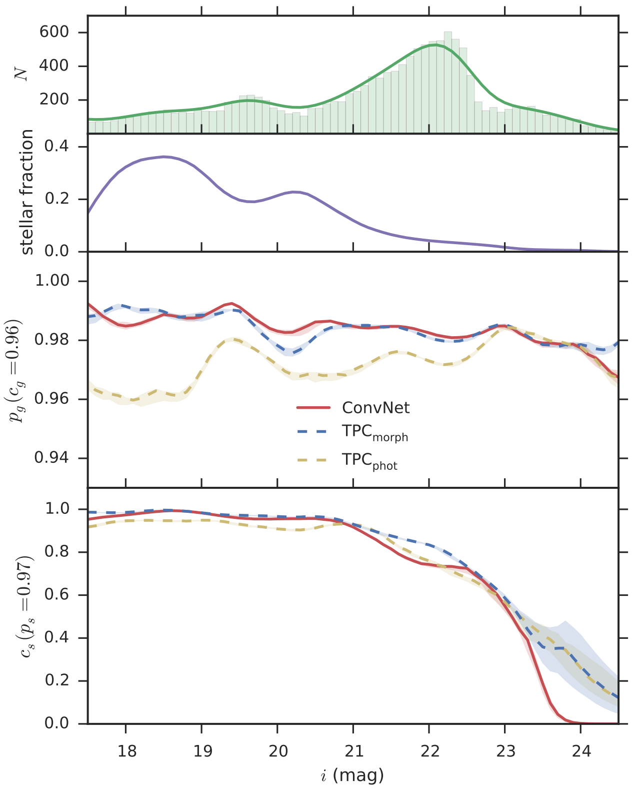

Figure 2. Galaxy purity and star completeness values as functions of the i-band magnitude (differential counts) as estimated by kernel density estimation (KDE) in the CFHTLenS data set. The top panel shows the histogram with a bin size of 0.1 mag and the KDE for objects in the test set. The second panel shows the fraction of stars estimated by KDE as a function of magnitude. The bottom two panels compare the galaxy purity and star completeness values for ConvNet (red solid line), $\text{TPC}_{\text{morph}}$ (blue dashed line), and TPC_{phot} (yellow dashed line) as functions of magnitude. The 1σ confidence bands are estimated by bootstrap sampling.

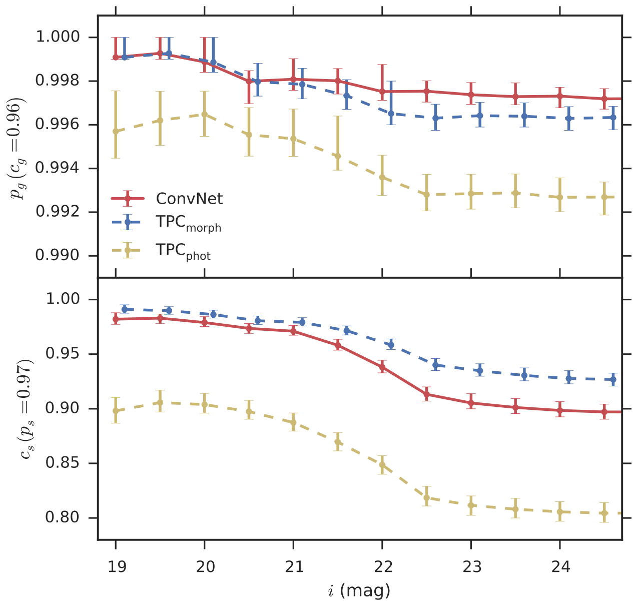

Figure 3. Galaxy purity and star completeness as functions of the i-band magnitude (integrated counts) in the CFHTLenS data set. The upper panel compares the galaxy purity values for ConvNet (red solid line), $\text{TPC}_{\text{morph}}$ (blue dashed line), and TPC_{phot} (yellow dashed line). The lower panel compares the star completeness values. The 1σ error bars are computed following the method of Paterno (2003) to avoid the unphysical errors of binomial or Poisson statistics.

tune the hyperparameters of $\text{TPC}_{\text{morph}}$ and TPC_{phot}, and perform classifications on the same test set to compare their performance with that of ConvNet.

We present in Table 3 a summary of the results obtained by applying ConvNet, $\text{TPC}_{\text{morph}}$, and TPC_{phot} on the test set of the CFHTLenS data. The bold entries highlight the best technique for any particular metric. ConvNet outperforms $\text{TPC}_{\text{morph}}$ in four metrics (AUC, p_g, CAL, and $|\Delta N_g|/N_g$), while $\text{TPC}_{\text{morph}}$ performs better in two metrics (MSE and c_g). It is not surprising that TPC_{phot}, which is trained on only magnitudes and colors, performs significantly worse than both ConvNet and TPC_{phot}. As Figure **??** shows, there is significant overlap between stars and galaxies in color-color space. Thus, magnitudes and colors alone are not sufficient to separate stars from galaxies, and morphology is critical in separating stars from galaxies. ConvNet is able to learn the morphological features automatically from the images, and the performance of ConvNet is therefore comparable to that of $\text{TPC}_{\text{morph}}$, which is trained on both morphological and photometric attributes.

In Figure 2, we compare the galaxy purity and star completeness values for ConvNet, $\text{TPC}_{\text{morph}}$, and TPC_{phot}, as a function of i-band magnitude for the differential counts. We use kernel density estimation (KDE; Silverman 1986) with a Gaussian kernel. As the first panel shows, KDE is able to smooth the fluctuations in the distribution without binning. While ConvNet shows a slightly better performance than $\text{TPC}_{\text{morph}}$ in galaxy purity, ConvNet performs slightly worse than $\text{TPC}_{\text{morph}}$ in star completeness. Again, TPC_{phot} performs significantly worse than both ConvNet and $\text{TPC}_{\text{morph}}$, and this suggests that ConvNets are able to learn the shape information automatically from the images. We note that, at these operating conditions ($c_g = 0.96$ or $p_s = 0.97$), both ConvNet and $\text{TPC}_{\text{morph}}$ outperform the star-galaxy classification provided by the CFHTLenS pipeline (Hildebrandt et al. 2012) over all magnitudes.

In Figure 3, we show the overall galaxy purity and star completeness values as a function of i-band magnitude for the integrated counts. ConvNet is able to maintain a galaxy purity of 0.9972 up to $i \sim 24.5$, while the galaxy purity of $\text{TPC}_{\text{morph}}$ drops to 0.9963. However, $\text{TPC}_{\text{morph}}$ performs better than ConvNet in terms

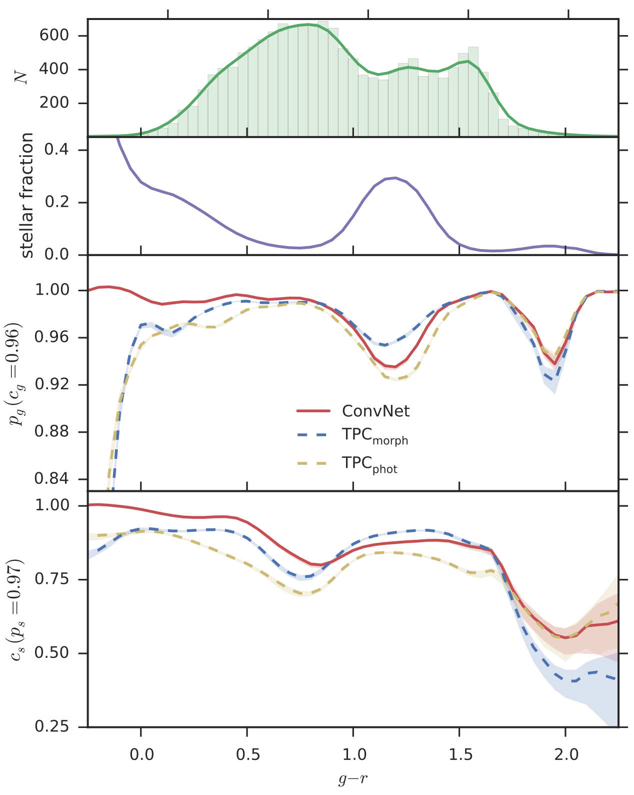

Figure 4. Similar to Figure 2 but as a function of $g-r$ color. The bin size of histogram in the top panel is 0.05.

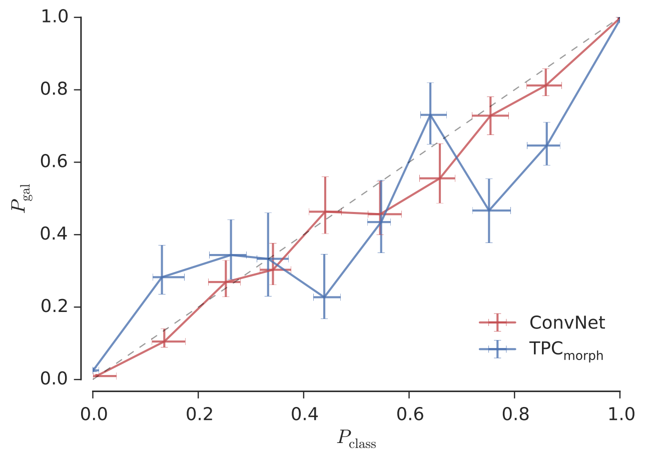

Figure 5. The calibration curves for ConvNet (red) and TPC$_{\rm morph}$ (blue) as applied to the CFHTLenS data set. $P_{\rm gal}$ is the fraction of objects that are galaxies, and $P_{\rm class}$ is the probabilistic outputs generated by the classifiers. The dashed line displays the relationship $P_{\rm gal} = P_{\rm conv}$. The 1σ error bars are computed following the method of Paterno (2003).

of star completeness, maintaining a purity of 0.9252 up to $i \sim 24.5$, while ConvNet drops to 0.8966.

We also show the galaxy purity and star completeness values as functions of $g-r$ color in Figure 4. TPC$_{\rm morph}$ provides slightly better completeness and purity than ConvNet between $0.8 \lesssim g-r \lesssim 1.6$ while ConvNet outperforms TPC$_{\rm morph}$ in the remaining regions.

Figure 5 shows the calibration curves that compare $P_{\rm gal}$, the fraction of objects that are galaxies (as determined from their spectra), to $P_{\rm class}$, the probabilistic outputs produced by ConvNet and TPC$_{\rm morph}$. The calibration curve for our ConvNet model is nearly diagonal, which implies that ConvNet is well-calibrated and we can treat its probabilistic output as the probability that an object is a galaxy. In contrast, the calibration curve for the probabilistic output of TPC$_{\rm morph}$ is apparently not as well-calibrated as ConvNet. These calibration curves visually confirm the results in Table 3 that the calibration error of ConvNet is about 20% lower than that of TPC$_{\rm morph}$. While probabilistic predictions can be further calibrated by using, e.g., isotonic calibration (Zadrozny & Elkan 2001), we do not consider additional probability calibration in this work.

It is informative to visualize how an input image activates the neurons in the convolutional layers. Figures 6 and 7 show the activations of the network when images of a galaxy and a star are fed into the network. The size of feature maps decreases with depth, and layers near the input layer have fewer filters while the later layers have more. The low-level features, e.g., edges or blobs, of the input images are still recognizable in the first convolutional layer. Subsequent layers use these low-level features to detect higher-level features, and the final layer is a classifier that uses these high-level features. Thus, by performing hierarchical abstraction from low-level to high-level features, ConvNets are able to utilize shape information in the classification process.

6.3 SDSS

We have also trained and tested our ConvNet model on the SDSS data set, and we present in Table 4 the same six metrics for ConvNet, TPC$_{\rm morph}$, and TPC$_{\rm phot}$. The bold entries highlight the best technique for any particular metrics. In contrast with the CFHTLenS data set in Section 6.2, it is apparent that TPC$_{\rm morph}$ outperforms ConvNet in all metrics except CAL. Both ConvNet and TPC$_{\rm morph}$ still outperform TPC$_{\rm phot}$ in all six metrics by a significant amount, as magnitudes and colors alone are not sufficient to separate stars from galaxies. Although ConvNet performs worse than TPC$_{\rm morph}$ on the SDSS data, its performance is much closer to TPC$_{\rm morph}$, as ConvNet is able to learn the shape information automatically from the images.

In Figure 8, we compare the galaxy purity and star completeness values for ConvNet, TPC$_{\rm morph}$, and TPC$_{\rm phot}$ as a function of r-band magnitude for the differential counts in the SDSS data. We note that TPC$_{\rm morph}$ outperforms the star-galaxy classifier used by the SDSS pipeline (i.e., an object is classified as a galaxy if concentration > 0.145) over all magnitudes. We do not show the SDSS classifications to avoid cluttering the plots. While ConvNet shows a similar but slightly worse performance than TPC$_{\rm morph}$, the galaxy purity and star completeness values of ConvNet begin to drop at faint magnitudes $i \lesssim 21$. Again, TPC$_{\rm phot}$ performs significantly worse than both ConvNet and TPC at bright magnitudes. One reason that ConvNet fails to outperform TPC$_{\rm phot}$, especially at faint magnitudes, might be its over-reliance on morphological features. Near a survey's limit, the measurement uncertainties generally increase, and morphology is not a reliable metric for star-galaxy classification. Another possibility is that data augmentation has a negative effect at faint magnitudes, as the network may get confused by additional examples of faint galaxies that look like point sources. Data augmentation however is indispensable, since it improves the overall performance greatly.

In Figure 9, we show the overall galaxy purity and star completeness values as a function of magnitude for the integrated counts. ConvNet is able to maintain a galaxy purity of 0.9915 up

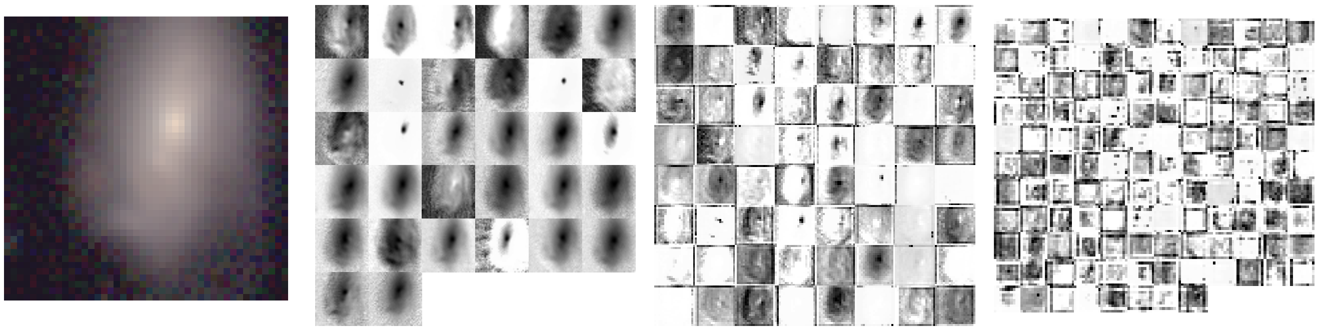
(a) Input (5 bands×44×44)
(b) Layer 1 (32 maps×40×40)
(c) Layer 3 (64 maps×20×20)
(d) Layer 6 (128 maps×10×10)

Figure 6. (a) A sample 44×44 RGB image of a galaxy in the CFHTLenS data set. The RGB image is created by mapping R → *i* band magnitude, G → *r* band magnitude, and B → *g* band magnitude. (b) Activations on the first convolutional layer when a 5×44×44 image is fed into the network. (c) Activations on the third convolutional layer. (d) Activations on the sixth convolutional layer. Each image in (a), (b), and (c) is a feature map corresponding to the output for one of the learned features.

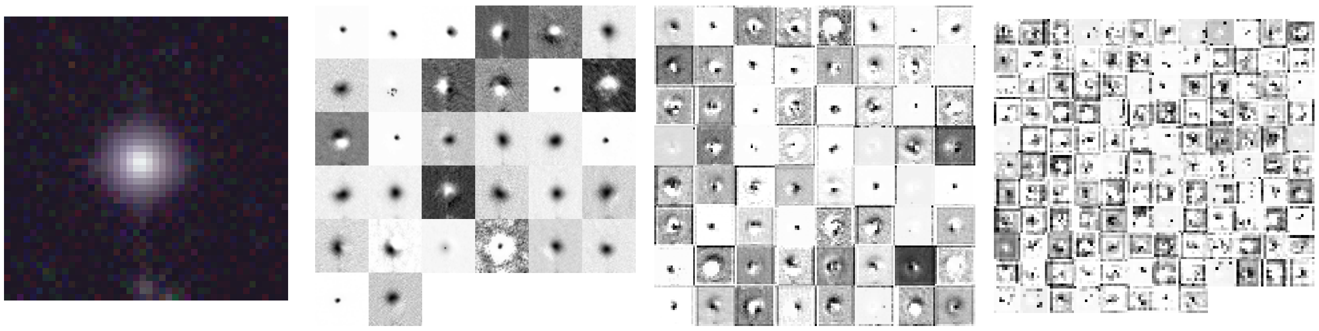
(a) Input (5 bands×44×44)
(b) Layer 1 (32 maps×40×40)
(c) Layer 3 (64 maps×20×20)
(d) Layer 6 (128 maps×10×10)

Figure 7. Similar to Figure 6 but for a star in the CFHTLenS data set.

Table 4. A summary of the classification performance metrics as applied to the SDSS data.

classifier	AUC	MSE	$p_g(c_g = 0.96)$	$c_s(p_s = 0.97)$	CAL	$\lvert\Delta N_g\rvert/N_g$
ConvNet	0.9952	0.0182	0.9915	0.9500	**0.0243**	0.0157
TPC$_{\rm morph}$	**0.9967**	**0.0099**	**0.9977**	**0.9810**	0.0254	**0.0044**
TPC$_{\rm phot}$	0.9886	0.0283	0.9819	0.8879	0.0316	0.0160

to $i \sim 22.5$, while TPC$_{\rm morph}$ provides a galaxy purity of 0.9977. TPC$_{\rm morph}$ also outperforms ConvNet in terms of star completeness, maintaining a purity of 0.9810 up to $i \sim 22.5$, while the star completeness of ConvNet drops to 0.9500.

We also show the galaxy purity and star completeness values as a function of $g-r$ color in Figure 10. ConvNet performs slightly better than TPC$_{\rm morph}$ in both galaxy completeness and star purity between $0.7 \lesssim g-r \lesssim 2.0$, where the stellar fraction is relatively low. On the other hand, both TPC$_{\rm morph}$ and TPC$_{\rm phot}$ outperform ConvNet in the region $g-r \lesssim 0.8$ where the stellar fraction is higher.

Figure 11 shows the calibration curves of ConvNet and TPC$_{\rm morph}$. The calibration curve of ConvNet in Figure 11 is not as well-calibrated as the calibration curve in Figure 5, where the same ConvNet model was applied to the CFHTLenS data set. However, ConvNet may still be better calibrated than TPC$_{\rm morph}$, even when it is applied to the SDSS data set. Although it is not straightforward to compare the two calibration curves by visual inspection, Table 4 shows that the CAL metric of ConvNet is lower than that of TPC$_{\rm morph}$.

7 CONCLUSIONS

We have presented a convolutional neural network for classifying stars and galaxies in the SDSS and CFHTLenS photometric images. For the CFHTLenS data set, the network is able to provide a classification that is as accurate as a random forest algorithm (TPC), while the probability estimates of our ConvNet model appear to be better calibrated. When the same network architecture is applied to the SDSS data set, the network fails to outperform TPC, but the probabilities are still slightly better calibrated. The major advantage of ConvNets is that useful features are learned automatically from images, while traditional machine learning algorithms require feature engineering as a separate process to produce accurate classifications.

ConvNets have recently achieved record-breaking results in many image classification tasks (LeCun et al. 2015) and have been quickly and widely adopted by the computer vision community. One of the main reasons for the success is that ConvNets are general-purpose algorithms that are applicable to a variety of problems without the need for designing a feature extractor. The lack of

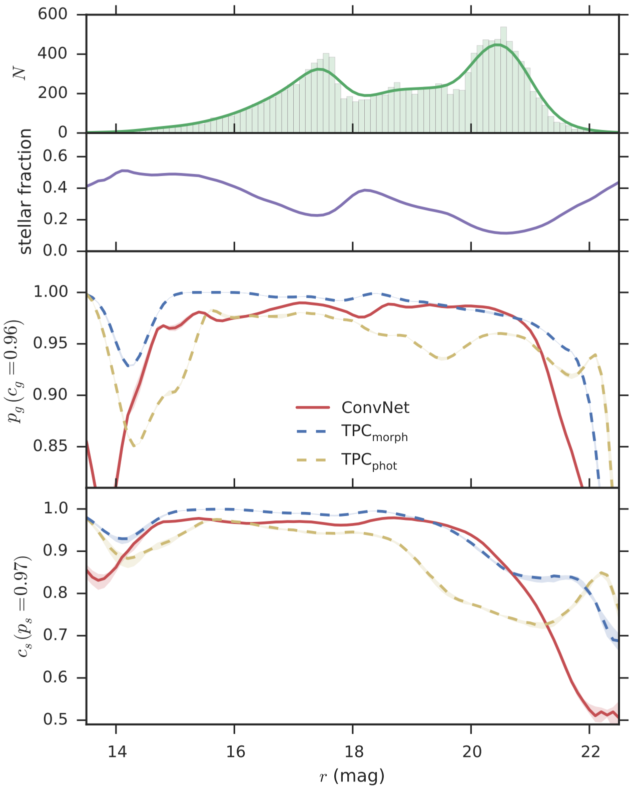

Figure 8. Galaxy purity and star completeness as function of the r-band magnitude for the differential counts in the SDSS data set.

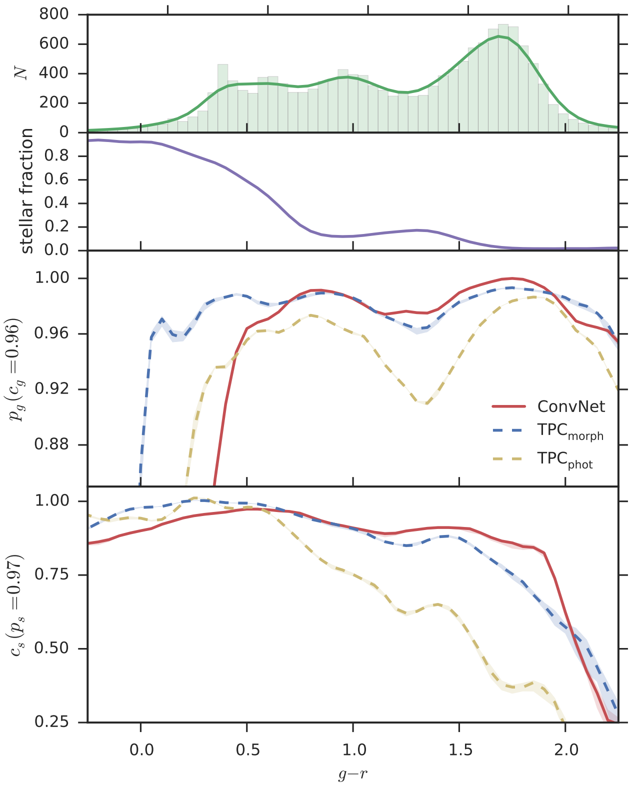

Figure 10. Similar to Figure 8 but as a function of $g - r$ color. The bin size of histogram in the top panel is 0.05.

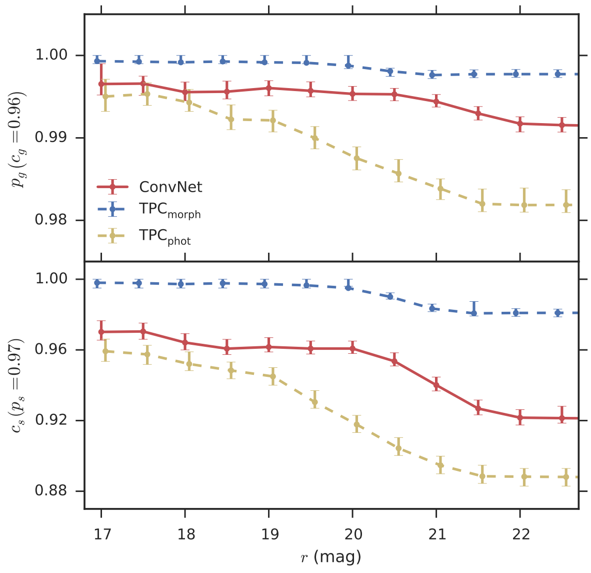

Figure 9. Galaxy purity and star completeness as functions of the r-band magnitude for the integrated counts in the SDSS data set.

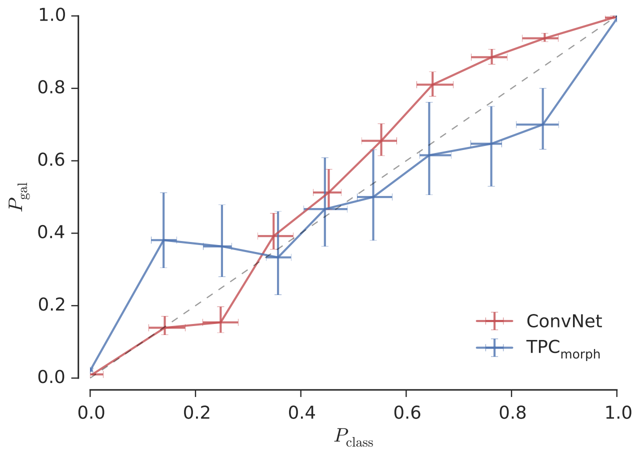

Figure 11. Calibration curves for ConvNet (red) and TPC$_{\text{morph}}$ (blue) as applied to the SDSS data set.

requirement for feature extraction is a huge advantage, e.g., when the task is to classify 1,000 classes in the ImageNet data set (Russakovsky et al. 2015), as a good feature extractor for identifying images of cats would be of little use for classifying sailboats, and it is impractical to design a separate feature extractor for each class. However, when there already exists a good feature extractor for the problem at hand, e.g., the concentration parameter, the weight-averaged `spread_model` parameter from the Dark Energy Survey (Desai et al. 2012; Crocce et al. 2016), or even the `SExtractor` software, conventional machine learning algorithms that have been shown to be effective, such as TPC (Carrasco Kind & Brunner 2013; Kim et al. 2015), remain a viable option. As the "no free lunch" theorem (Wolpert 1996) states, there is no one model that works for every problem. For the CFHTLenS data set, our ConvNet model outperforms TPC. Since the SDSS catalog provides the concentration parameter that is highly optimized for star-galaxy classification, TPC works better for SDSS.

Although we used various techniques to combat overfitting, it is possible that our ConvNet model has overfit the data. Overfitting could explain why our ConvNet model with maximal information fails to significantly outperform a standard machine learning algorithm that uses the reduced summary information from catalogs. The most effective way to prevent overfitting would be to simply collect more training images with spectroscopic follow-up, as the performance of ConvNets generally improves with more training data. However, spectroscopic observations are expensive and time-consuming, and it is unclear if sufficient training data will be available in future photometric surveys. If enough training data become available in DES or LSST, ConvNets become an attractive option because it can be applied immediately on reduced, calibrated images to produce well-calibrated posterior probabilities. We also note that using more training images will require the use of multi-GPU systems, which was beyond the scope of the present work.

Deep learning is a rapidly developing field, and recent developments include improved network architectures. For future work, we plan to train more ConvNet variants, such as the Inception Module (Szegedy et al. 2015) and Residual Network (He et al. 2015). To improve the predictive performance, we have combined the predictions of different models across multiple transformations of the input images (Section 4.3). To further improve the performance, we could also train several networks with different architectures and combine the models. For example, the winning solution of Dieleman et al. (2015) for the Galaxy Zoo challenge was based on a ConvNet model, and it required averaging many sets of predictions from models with different neural network architectures.

It is also likely that the performance will be improved not only by training multiple network architectures, but also by combining them with different star-galaxy classifiers. In Kim et al. (2015), we combined a purely morphological classifier, a supervised machine learning method (TPC), an unsupervised machine learning method based on self-organizing maps, and a hierarchical Bayesian template fitting method, and demonstrated that our combination technique improves the overall performance over any individual classification method. ConvNets could be included as a different machine learning paradigm in this classifier combination framework to produce further improvements in predictive performance.

ACKNOWLEDGEMENTS

The authors thank the referees for a careful reading of the manuscript and comments that improved this work. We acknowledge support from the National Science Foundation Grant No. AST-1313415. RJB acknowledges support as an Associate within the Center for Advanced Study at the University of Illinois.

This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1053575.

This work has made use of the Theano library (http://deeplearning.net/software/theano/), the Lasagne library (http://lasagne.readthedocs.io/), and the Nolearn library (http://pythonhosted.org/nolearn/).

This work is based on observations obtained with MegaPrime/MegaCam, a joint project of CFHT and CEA/DAPNIA, at the Canada-France-Hawaii Telescope (CFHT) which is operated by the National Research Council (NRC) of Canada, the Institut National des Sciences de l'Univers of the Centre National de la Recherche Scientifique (CNRS) of France, and the University of Hawaii. This research used the facilities of the Canadian Astronomy Data Centre operated by the National Research Council of Canada with the support of the Canadian Space Agency. CFHTLenS data processing was made possible thanks to significant computing support from the NSERC Research Tools and Instruments grant program.

Funding for SDSS-III has been provided by the Alfred P. Sloan Foundation, the Participating Institutions, the National Science Foundation, and the U.S. Department of Energy Office of Science. The SDSS-III web site is http://www.sdss3.org/.

SDSS-III is managed by the Astrophysical Research Consortium for the Participating Institutions of the SDSS-III Collaboration including the University of Arizona, the Brazilian Participation Group, Brookhaven National Laboratory, Carnegie Mellon University, University of Florida, the French Participation Group, the German Participation Group, Harvard University, the Instituto de Astrofisica de Canarias, the Michigan State/Notre Dame/JINA Participation Group, Johns Hopkins University, Lawrence Berkeley National Laboratory, Max Planck Institute for Astrophysics, Max Planck Institute for Extraterrestrial Physics, New Mexico State University, New York University, Ohio State University, Pennsylvania State University, University of Portsmouth, Princeton University, the Spanish Participation Group, University of Tokyo, University of Utah, Vanderbilt University, University of Virginia, University of Washington, and Yale University.

REFERENCES

Aggarwal C. C., 2014, Data classification: algorithms and applications. CRC Press
Alam S., et al., 2015, ApJS, 219, 12
Ball N. M., Brunner R. J., Myers A. D., Tcheng D., 2006, ApJ, 650, 497
Ball N. M., Brunner R. J., Myers A. D., Strand N. E., Alberts S. L., Tcheng D., Llorà X., 2007, ApJ, 663, 774
Ball N. M., Brunner R. J., Myers A. D., Strand N. E., Alberts S. L., Tcheng D., 2008, ApJ, 683, 12
Banerji M., et al., 2010, MNRAS, 406, 342
Bengio Y., Boulanger-Lewandowski N., Pascanu R., 2013, in Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on. pp 8624–8628
Bertin E., Arnouts S., 1996, A&AS, 117, 393
Bottou L., 1998, On-line learning in neural networks, 17, 142
Bousquet O., Bottou L., 2008, in Advances in neural information processing systems. pp 161–168
Breiman L., 2001, Machine learning, 45, 5
Breiman L., Friedman J., Stone C. J., Olshen R. A., 1984, Classification and regression trees. CRC press
Brier G. W., 1950, Monthly weather review, 78, 1
Carrasco Kind M., Brunner R. J., 2013, MNRAS, 432, 1483
Carrasco Kind M., Brunner R. J., 2014a, MNRAS, 438, 3409
Carrasco Kind M., Brunner R. J., 2014b, MNRAS, 442, 3380
Caruana R., Niculescu-Mizil A., 2004, in Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. pp 69–78
Crocce M., et al., 2016, MNRAS, 455, 4301
Davis M., et al., 2003, Astronomical Telescopes and Instrumentation, pp 161–172
DeGroot M. H., Fienberg S. E., 1983, The statistician, pp 12–22

Desai S., et al., 2012, ApJ, 757, 83
Dieleman S., et al. 2015, Lasagne: First release., doi:10.5281/zenodo.27878
Dieleman S., Willett K. W., Dambre J., 2015, MNRAS, 450, 1441
Dieleman S., De Fauw J., Kavukcuoglu K., 2016, arXiv preprint arXiv:1602.02660
Eisenstein D. J., et al., 2011, AJ, 142, 72
Erben T., et al., 2013, MNRAS, p. stt928
Fadely R., Hogg D. W., Willman B., 2012, ApJ, 760, 15
Fukushima K., 1980, Biological cybernetics, 36, 193
Garilli B., et al., 2008, A&A, 486, 683
Garilli B., et al., 2014, A&A, 562, A23
Gwyn S. D., 2012, AJ, 143, 38
He K., Zhang X., Ren S., Sun J., 2015, arXiv preprint arXiv:1512.03385
Henrion M., Mortlock D. J., Hand D. J., Gandy A., 2011, MNRAS, 412, 2286
Heymans C., et al., 2012, MNRAS, 427, 146
Hildebrandt H., et al., 2012, MNRAS, 421, 2355
Hinton G. E., Srivastava N., Krizhevsky A., Sutskever I., Salakhutdinov R. R., 2012, arXiv preprint arXiv:1207.0580
Hoyle B., 2015, arXiv preprint arXiv:1504.07255
Huertas-Company M., et al., 2015, ApJS, 221, 8
Ivezić Ž., Connolly A. J., VanderPlas J. T., Gray A., 2014, Statistics, Data Mining, and Machine Learning in Astronomy: A Practical Python Guide for the Analysis of Survey Data. Princeton University Press
Kamdar H. M., Turk M. J., Brunner R. J., 2016a, MNRAS, 455, 642
Kamdar H. M., Turk M. J., Brunner R. J., 2016b, MNRAS, 457, 1162
Kim E. J., Brunner R. J., Kind M. C., 2015, MNRAS, 453, 507
Krizhevsky A., Sutskever I., Hinton G. E., 2012, in Advances in neural information processing systems. pp 1097–1105
Le Fèvre O., et al., 2005, A&A, 439, 845
LeCun Y. A., Bottou L., Orr G. B., Müller K.-R., 1998a, in , Neural networks: Tricks of the trade. Springer, pp 9–50
LeCun Y., Bottou L., Bengio Y., Haffner P., 1998b, Proceedings of the IEEE, 86, 2278
LeCun Y., Bengio Y., Hinton G., 2015, Nature, 521, 436
Li N., Thakar A. R., 2008, Computing in Science & Engineering, 10, 18
Lupton R. H., Gunn J. E., Szalay A. S., 1999, AJ, 118, 1406
Maas A. L., Hannun A. Y., Ng A. Y., 2013, in Proc. ICML.
Monteith K., Carroll J. L., Seppi K., Martinez T., 2011, in Neural Networks (IJCNN), The 2011 International Joint Conference on. pp 2657–2663
Nair V., Hinton G. E., 2010, in Proceedings of the 27th International Conference on Machine Learning (ICML-10). pp 807–814
Newman J. A., et al., 2013, ApJS, 208, 5
Odewahn S. C., Stockwell E. B., Pennington R. L., Humphreys R. M., Zumach W. A., 1992, AJ, 103, 318
Paterno M., 2003, Calculating Efficiencies and Their Uncertainties, `http://home.fnal.gov/~paterno/images/effic.pdf`
Ross A. J., et al., 2011, MNRAS, 417, 1350
Rumelhart D. E., Hinton G. E., Williams R. J., 1988, Cognitive modeling, 5, 1
Russakovsky O., et al., 2015, International Journal of Computer Vision, 115, 211
Saxe A. M., McClelland J. L., Ganguli S., 2013, arXiv preprint arXiv:1312.6120
Schlegel D. J., Finkbeiner D. P., Davis M., 1998, ApJ, 500, 525
Seo H.-J., et al., 2012, ApJ, 761, 13
Sevilla-Noarbe I., Etayo-Sotos P., 2015, Astronomy and Computing, 11, 64
Silverman B. W., 1986, CRC press, 26
Simonyan K., Zisserman A., 2014, arXiv preprint arXiv:1409.1556
Soumagnac M. T., et al., 2015, MNRAS, 450, 666
Suchkov A. A., Hanisch R. J., Margon B., 2005, AJ, 130, 2439
Sutskever I., Martens J., Dahl G., Hinton G., 2013, in Proceedings of the 30th international conference on machine learning (ICML-13). pp 1139–1147
Swets J. A., Dawes R. M., Monahan J., 2000, Scientific American, p. 83
Szegedy C., et al., 2015, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp 1–9
Theano Development Team 2016, arXiv e-prints, abs/1605.02688
Weir N., Fayyad U. M., Djorgovski S., 1995, AJ, 109, 2401
Wolpert D. H., 1996, Neural computation, 8, 1341
York D. G., et al., 2000, AJ, 120, 1579
Zadrozny B., Elkan C., 2001, in ICML. pp 609–616
Zeiler M. D., Fergus R., 2014, in , Computer vision–ECCV 2014. Springer, pp 818–833

This paper has been typeset from a TEX/LATEX file prepared by the author.