Mon. Not. R. Astron. Soc. **000**, 1–17 (0000) Printed 13 December 2022 (MN LATEX style file v2.2)

Finding quadruply imaged quasars with machine learning. I. Methods

A. Akhazhanov,[1,2⋆] A. More,[3,4] A. Amini,[5] C. Hazlett,[5,6] T. Treu,[7]† S. Birrer,[8] A. Shajib,[9]‡ K. Liao,[10] C. Lemon,[11] A. Agnello, [12] B. Nord,[13,14,9] M. Aguena,[15] S. Allam,[13] F. Andrade-Oliveira,[16,15] J. Annis,[13] D. Brooks,[17] E. Buckley-Geer,[9,13] D. L. Burke,[18,19] A. Carnero Rosell,[15] M. Carrasco Kind,[20,21] J. Carretero,[22] A. Choi,[23] C. Conselice,[24,25] M. Costanzi,[26,27,28] L. N. da Costa,[15,29] M. E. S. Pereira,[30,31] J. De Vicente,[32] S. Desai,[33] J. P. Dietrich,[34] P. Doel,[17] S. Everett,[35] I. Ferrero,[36] D. A. Finley,[13] B. Flaugher,[13] J. Frieman,[13,14] J. García-Bellido,[37] D. W. Gerdes,[38,30] D. Gruen,[39] R. A. Gruendl,[20,21] J. Gschwend,[15,29] G. Gutierrez,[13] S. R. Hinton,[40] D. L. Hollowood,[35] K. Honscheid,[23,41] D. J. James,[42] A. G. Kim,[43] K. Kuehn,[44,45] N. Kuropatkin,[13] O. Lahav,[17] M. Lima,[46,15] H. Lin,[13] M. A. G. Maia,[15,29] M. March,[47] F. Menanteau,[20,21] R. Miquel,[48,22] R. Morgan,[49] A. Palmese,[13,14] F. Paz-Chinchón,[20,50] A. Pieres,[15,29] A. A. Plazas Malagón,[51] E. Sanchez,[32] V. Scarpine,[13] S. Serrano,[52,53] I. Sevilla-Noarbe,[32] M. Smith,[54] M. Soares-Santos,[30] E. Suchyta,[55] M. E. C. Swanson,[20] G. Tarle,[30] C. To,[56,18,19] T. N. Varga,[57,39] and J. Weller[57,39]

(DES Collaboration)

Accepted . Received

ABSTRACT

Strongly lensed quadruply imaged quasars (quads) are extraordinary objects. They are very rare in the sky and yet they provide unique information about a wide range of topics, including the expansion history and the composition of the Universe, the distribution of stars and dark matter in galaxies, the host galaxies of quasars, and the stellar initial mass function. Finding them in astronomical images is a classic "needle in a haystack" problem, as they are outnumbered by other (contaminant) sources by many orders of magnitude. To solve this problem, we develop state-of-the-art deep learning methods and train them on realistic simulated quads based on real images of galaxies taken from the Dark Energy Survey, with realistic source and deflector models, including the chromatic effects of microlensing. The performance of the best methods on a mixture of simulated and real objects is excellent, yielding area under the receiver operating curve in the range 0.86 to 0.89. Recall is close to 100% down to total magnitude $i \sim 21$ indicating high completeness, while precision declines from 85% to 70% in the range $i \sim 17 - 21$. The methods are extremely fast: training on 2 million samples takes 20 hours on a GPU machine, and 10^8 multi-band cutouts can be evaluated per GPU-hour. The speed and performance of the method pave the way to apply it to large samples of astronomical sources, bypassing the need for photometric pre-selection that is likely to be a major cause of incompleteness in current samples of known quads.

Key words: gravitational lensing: strong – methods: statistical – astronomical data bases: catalogs

1 INTRODUCTION

Strong gravitational lenses are extremely valuable sources of information about the Universe. They provide unique information about the expansion rate of the Universe, the properties of distant sources that would be too faint (compact) to be detected (resolved), and about the distribution of mass

⋆ e-mail:gkcalat@ucla.edu
† Packard fellow
‡ NHFP Einstein Fellow

in the Universe (Treu 2010, and references therein). Unfortunately, they are very rare on the sky, because the phenomenon requires the almost perfect alignment of a background source with a foreground deflector.

Quadruply imaged quasars are a very special case of strong lensing. They are especially valuable because of the wealth of information they provide, including, for example, three independent time delays and flux ratios. At the same time, they are especially rare because they require an intrinsically rare source (quasar) to be within the inner caustic of a foreground massive galaxy. Based on the model by Oguri & Marshall (2010), the density of quads in the sky is expected to be of order 10^{-2} deg^{-2} with total flux brighter than $i \sim 20$ (i.e. ~ 400 in the full sky), but only a fraction of those will be resolved and identifiable in ground based wide-field imaging of the kind obtained by the Dark Energy Survey (DES, e.g., Treu et al. 2018). Even though the numbers have improved considerably in the past few years, only ∼60 quads are known across the entire sky at the time of this writing.

There are two main challenges in identifying quads from ground based optical imaging data. The first challenge is the sheer volume of data one has to inspect, considering that there are about $\sim 10^8$ stars and galaxies in the sky brighter than $i \sim 20$ (Annis et al. 2014). The second challenge is that many of the quads are only partially resolved in ground based images and thus difficult to identify and separate from astronomical contaminants. In order to overcome the first challenge, many search teams rely on color preselection to reduce the number of astronomical sources. The second challenge then becomes more manageable with a combination of algorithms applied to the image pixels and visual inspection. In the end, even in the most successful cases, confirmation via spectroscopy or higher resolution imaging (from space or from the ground with adaptive optics) is needed. Considering the cost of spectroscopy and high resolution imaging, most searches so far have focused on obtaining high purity candidate lists with high confirmation rates (e.g. Lemon et al. 2020). The drawback of this process is that many lenses are lost along the way, as evidenced by the low completeness of searches so far.

We present a new machine learning based approach to finding quadruply imaged quasars. Machine learning techniques have been applied with success to lens searches before (Agnello et al. 2015; Williams et al. 2017a; Hezaveh et al. 2017; Petrillo et al. 2017, 2018; Lanusse et al. 2017; Pourrahmani et al. 2018; Schaefer, C. et al. 2018; Avestruz et al. 2019; Madireddy et al. 2019; Cheng et al. 2020; Jacobs et al. 2019a; Jacobs et al. 2019b). While building on the work in this area, our effort differs in two main ways. First, we focus exclusively on quadruply imaged quasars, developing a realistic training set using real astronomical images from the Dark Energy Survey coupled with macro and millilensing models. Second, we avoid any need for image preselection with the goal of running our algorithm on complete flux-limited samples, which in turns requires our method to be extremely fast. This may allow us to recover quads that would have otherwise been lost in preselection steps, while retaining sufficient purity to be cost-effective for follow up. To achieve these goals we apply several new techniques and methodological improvements over previous astrophysical work, such as polar convolutions, the use of multiple networks, and attention masking. In addition to focusing exclusively on quads, tests on validation datasets suggest our method outperforms machine learners used for previous searches for lensed quasars (Agnello et al. 2015; Williams et al. 2017a).

In this first paper of a series we describe the method, the training set, and the results on validating datasets. A follow-up paper will present the results on a search on actual Dark Energy Survey data. The paper is organized as follows. Section 2 provides some background on the machine learning methods. Section 3 describes the training set. Section 4 describes the machine learning methods used. Section 5 describes the application of our machine learning algorithms to the problem and evaluates the performance on validating datasets. A summary is given in Section 6.

2 ELEMENTS OF DEEP LEARNING

In recent decades, machine learning, and particularly deep learning, have demonstrated extraordinary success in tackling a wide range of tasks related to computer vision and natural language processing, benefiting fields ranging from healthcare to the development of self-driving cars, among many others (LeCun et al. 2015; Wang 2016). In this section, we briefly review elements of deep learning that are relevant to detecting rare objects and that form the building blocks of the models employed here. A more thorough introduction to deep learning can be found in Goodfellow et al. (2016).

Deep learning builds on simple artificial neural network (ANN) models dating back to the perceptron algorithm (Rosenblatt 1958). Loosely inspired by biological neural networks, ANNs employ computational units referred to as neurons. A single neuron receives a set of inputs, represented by vector X, either directly from the stimulus (data), or from the outputs of a preceding set of neurons. Each neuron takes a weighted sum of its inputs, $\langle w, X\rangle$, adds a offset (aka "intercept" or "bias") term b, then passes this weighted sum through a function g to arrive at that neuron's activation level, $g(\langle w, X\rangle + b)$. The term b is often absorbed into the weight vector by simply appending a constant 1 to the dimensions of X and letting b be the corresponding weight, allowing the activation level to be written more simply as $g(\langle w, X\rangle)$. When $g(\cdot)$ is chosen to be non-linear, this allows for more complicated models to be built than those represented by conventional (linear) models, particularly as layers are added to the neural network.

The arrangement of neurons in layers is specified by an *architecture* that, for each group of neurons (called layers), determines from which other groups they receive their inputs and to which groups they send their outputs. Layers in which every neuron is connected to every neuron of the next layer are called "dense" (see Figure 1). The first layer of neurons are activated directly by the data. The final layer is the output layer that generates the quantity of interest, for example, the predicted class of the input in a classification task. An ANN with more than a few layers is often called *deep*, in contrast to early-generation ANNs that typically employed only one or two *hidden* layers located between the input and output layers. Adding layers increases the learn-

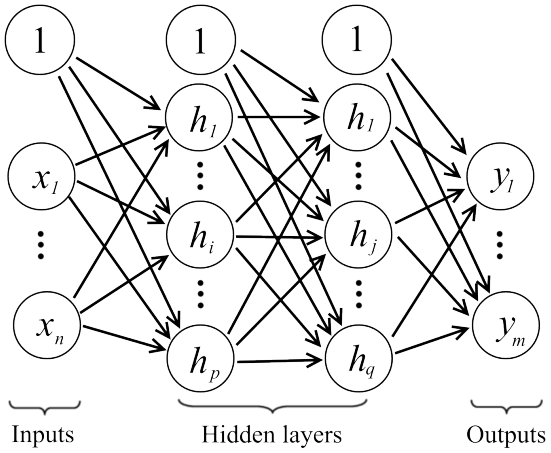

Figure 1. An example of a feed-forward neural network comprised of the input layer, two hidden layers, and the output layer. The information flows in one direction from each layer to the next, and the connections are called *dense* since every neuron in one layer is connected to every neuron in the next one.

ing capacity of the network, but exacerbates the difficulties of fitting or "training" the model.

2.1 Training the model

Neural networks are trained by adjusting the weights so that they optimally convert input data X (e.g. an astronomical image) into the desired output Y (e.g. a prediction of whether the image contains a lensed quasar). During the training, the ANN takes input data X and generates an output, $\hat{Y}$, for each training example, by propagating computations forward through the network: each neuron computes $g(\langle w, X\rangle)$ using the inputs (X) given to it and the current value of the weights (inclusive of the bias term), w. The difference between the generated output, $\hat{Y}$, and the correct answer, Y, is assessed using a loss function, $\mathcal{L}(Y, \hat{Y})$, chosen according to the nature of the problem. Various algorithms are available to determine how the weights of the network should be adjusted in light of each error. Typically these approaches assess how each weight contributes to the error, in a process that works backwards through the model computing the gradient of the loss using the chain rule. The weights can then be updated by some fraction of the value needed to correct the response, a procedure generally referred to as backpropagation as it propagates the error signal backwards through the network. Numerous optimization methods have been proposed with varying performance. One popular choice, used here, is the adaptive momentum (Adam) algorithm (Kingma & Ba 2015), which computes individual, adaptive learning rates for the model parameters considering the first and second moments of the gradients.

Training is usually iterative, with each step using a portion of the training data (called a "mini-batch") and adjusting the weights according to learning rates that control the speed of convergence. The end goal of the optimization is to reach a global minimum, a set of values of the parameters where the loss function is minimized. However, as the number of parameters is typically very large (in the millions for all models here), we generally expect multiple local minima. Stochastic gradient descent avoids getting "stuck" in suboptimal local minima by randomly selecting mini-batches.

Because these models are so flexible, they can lead to "overfitting", where the model learns specific characteristics of the training dataset associated with particular outcomes, but which do not generalize well to unseen data. This results in large errors when the model is applied to new data. Numerous methods help to prevent this. One important approach is to add terms to the loss function that effectively penalize models with more extreme weights. This constrains the model and avoids results that depend heavily on very specific features but that might have turned out very differently in different samples. Such an penalization scheme is known as regularization and can be represented as seeking to minimize

$$\mathcal{L}^*(Y, \hat{Y}) = \mathcal{L}(Y, \hat{Y}) + \lambda\Omega(W), \tag{1}$$

where the scalar λ controls the strength of regularization and Ω defines a regularizing functional. In the past, the first and second norms were frequently chosen as Ω and referred as L_1 and L_2 regularization respectively. Later, orthogonality of the weights was argued to be a desirable property since multiplication by an orthogonal matrix leaves the norm of the input unchanged. This led to orthogonal regularization, $\Omega(W) = \|\langle W, W^{\mathsf{T}}\rangle - I\|_1$, where I is the identity matrix.

2.2 Choice of the activation function

In many practical problems the model must be made to fit a non-linear function between the inputs and outputs, calling for a non-linear choice of activation function. Until recent years, the logistic (or sigmoid) function $g(x) = e^x/(1+e^x)$, hyperbolic tangent $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, softmax $[s(x)]_i = \frac{e^{x_i}}{\sum_k e^{x_k}}$, and linear rectifier $\mathrm{ReLU}(x) = \max(0, x)$ were among the most popular activation functions. However, deeper models require a choice of activation function that protects against the risk of having an error gradient equal to zero for many weights. To this end, functions such as parametric ReLU (PReLU) (He et al. 2015), exponential linear unit (ELU) (Clevert et al. 2016), scaled exponential linear unit (SELU) (Klambauer et al. 2017), and Swish (Ramachandran et al. 2018) have become widely used. These are defined as:

$$\begin{aligned}
\mathrm{PReLU}(x) &= \begin{cases} x, & \text{if } x \geqslant 0, \\ ax, & \text{if } x < 0, \end{cases} \quad (a < 1), \\
\mathrm{ELU}(x) &= \begin{cases} x, & \text{if } x \geqslant 0, \\ a(e^x - 1), & \text{if } x < 0, \end{cases} \\
\mathrm{SELU}(x) &= \beta \cdot \mathrm{ELU}(x), \quad \beta > 1, \\
\mathrm{Swish}(x) &= x \cdot \mathrm{sigmoid}(x) = \frac{x}{1 + e^{-x}}.
\end{aligned} \tag{2}$$

2.3 Convolutional neural networks

While dense ANNs with at least one hidden layer and an appropriate activation function can approximate any function, in practice ANNs can be made far more powerful, with less training data and fewer parameters to tune when they

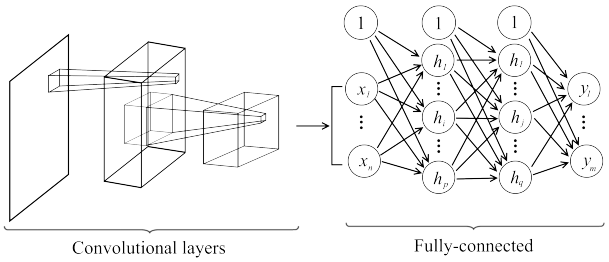

Figure 2. A convolutional neural network consisting of two convolutional layers and three dense layers. The first convolutional layer produces a feature map by sliding a convolution window over the input image. Within each window the layer takes a weighted sum of the corresponding pixels to produce a single output pixel. The output feature map of the second convolutional layer is flattened into a one-dimensional vector, used as an input the following fully-connected ANN.

can be designed to extract the more relevant and informative features from the data. To this end, convolutional neural networks (CNNs) (see Figure 2) have proven extremely powerful in image processing applications.

These networks have one or more layers that, like layers of visual cortex, contain neurons whose activation's summarize key features in designated patches of the input image. Specifically, neurons in a given layer are receptive to a particular area of the input/image, known as their "receptive fields". The convolution or weighted average they perform over their receptive fields summarizes the information in it. The weights used in this convolution, which constitute a kernel or filter, can be preset or learned during training. In CNNs with multiple convolutional layers, each layer takes the previous layer's activation as its input, with wider kernels in subsequent layers, so that the receptive field corresponding to a neuron grows to cover larger receptive fields over the image.

Stacking these convolutional layers, therefore, enables learning representations of the data at different levels of abstraction. This architecture has made deep CNNs very effective in computer vision, sometimes achieving superhuman performance in classification and discrimination tasks. Units within a given layer typically share the same kernels (weights) to reduce the number of learned parameters. Non-trainable convolutional layers are often referred as pooling layers and can be used to apply basic operations such as $\max(\cdot)$ or $\mathrm{mean}(\cdot)$ that corresponds to "MaxPooling" and "AvgPooling" layers respectively.

2.4 Blocks

Blocks of neurons can be designed and connected together to develop powerful network architectures. Here we discuss three types of blocks that have proven valuable in related computer vision problems and that we consider in our own architecture.

The "inception" block, is best known from InceptionNet (also known as GoogLeNet) (Szegedy et al. 2015), and later ResNet (Szegedy et al. 2017). This block is described in Figure 3 A). It concatenates four versions of the processed input data, each processed in parallel: the original image, two convolutional layers that use a sequence of increasingly large receptive fields, and one simpler local-averaged/smoothed version of the image.

The next block type is the residual block. A key problem with deeper networks—and part of the reason they did not emerge in earlier decades of ANNs—is the "vanishing gradient" problem: the updates to the weights computed by backpropogation factor in the gradient at each step, and a long sequence of such factors produces update values close to zero. One way to address this, employed in the ResNet architecture (He et al. 2016) is an "identity shortcut connection", also known as a residual or skip-connection. In this configuration, the input of each learning block is added to the output before propagating to the next one (see Figure 3 B). This makes it easier to propagate information forward and backward without significant alterations and simplifies training of deep models.

The third block type we consider is the dense block in the convolutional setting, as in the DenseNet architecture (Huang et al. 2017). While ResNet and its residual block use element-wise addition, dense convolutional blocks combine processed inputs of one layer (here, a convolutional layer) with lower level data by concatenation instead of addition. Each layer thus receives feature maps from all preceding convolutions, within the same block (see Figure 3 C). Later blocks may then use pooling or other approaches to reduce dimensionality. DenseNet has become widely used in various computer vision problem such as image classification, object detection, and image segmentation due to superior computational efficiency and quality of the learned features.

2.5 Generative modeling and data representation

Some tasks in computer vision are "image-to-image" problems, in which we have one input image and desire to create another related image of the same size. These including denoising (removing artifacts from an input image), segmentation (estimating a set of binary masks that encode different regions on the input image), and detection (locating objects on the input image) tasks. The U-Net is one important architecture for such problems. The model was first popularized in biomedical image segmentation (Ronneberger et al. 2015). In this context it takes input images (e.g. CT scans) and outputs segmentation masks that show regions of interest (e.g. malignant tumors), based on information from a labeled dataset. The U-net architecture is so named because it contains contracting and then expansive paths (see Figure 4 A). The contracting path employs a series of feature extraction blocks followed by one or more "scaling down" layers, such as a pooling layer. The expansive path concatenates the features of the same resolution, fuses the extracted features, and up-scales the image representation. The up-scaling method could be anything from a non-trainable nearest-neighbor interpolation to a trainable deep CNN. Additional skip connections between individual blocks of the contracting and expansive pathways enable easier gradient propagation. Importantly, the symmetry between two paths and the U-shaped architectures lets the network propagate context information to higher resolution layers.

Another valuable class of image-to-image architectures is the autoencoder (AE), which operate on unlabeled data and provide efficient, latent space representations. The ar-

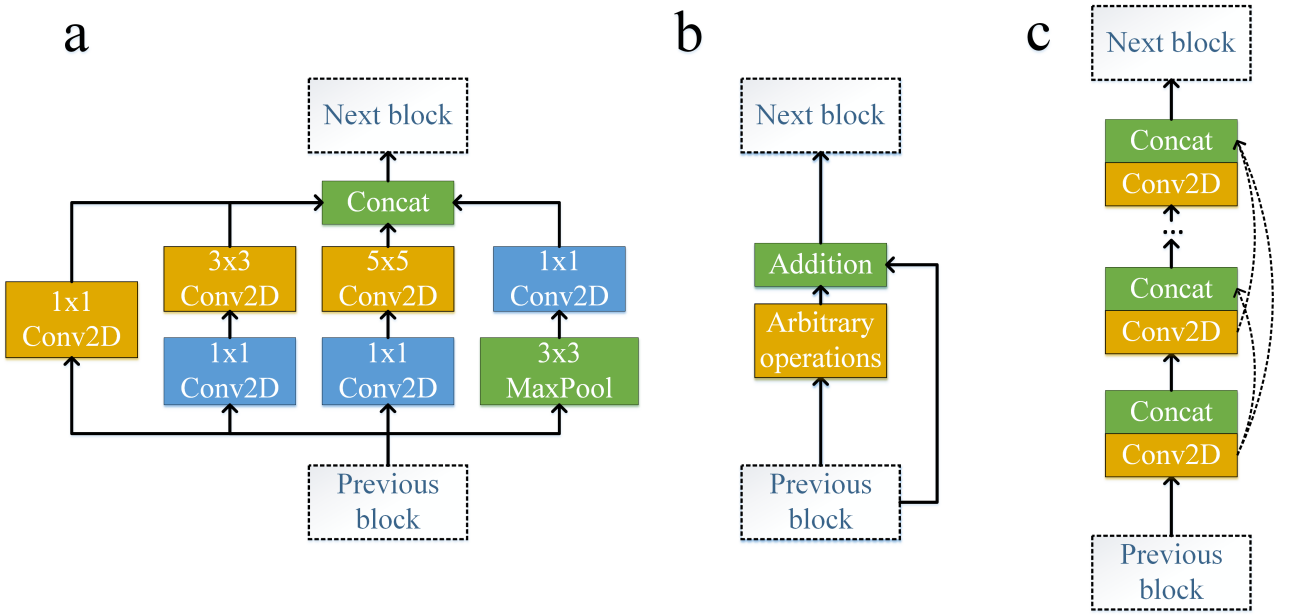

Figure 3. Common computer vision building blocks: (A) inception block, (B) residual block, and (C) dense block. The inception block applies multiple convolutional operations of different window size (shown in orange) and a pooling operation (shown in green) in parallel, instead of being restricted to a single window size, and then concatenates the extracted features together. As concatenation leads to a quickly growing size of the output tensor, 1-by-1 convolutional layers (shown in blue) are used to reduce the dimensionality. The residual block employs a *skip-connection* that adds the input tensor to the output tensor, which mitigates the "vanishing gradient" problem. The dense block similarly uses skip-connections, but concatenates the tensors instead of adding them.

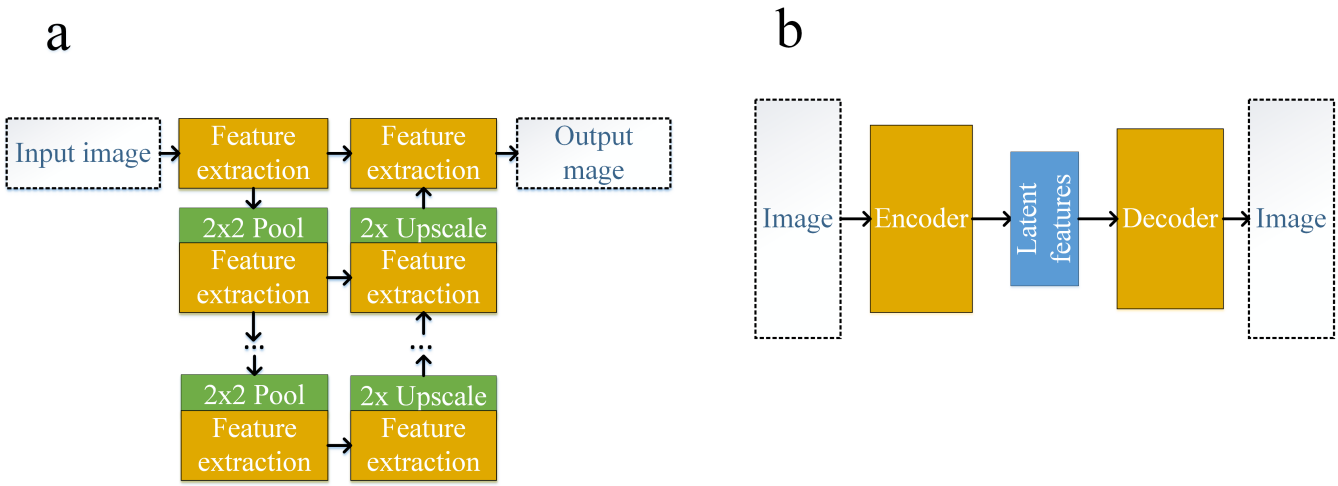

Figure 4. Data representation and generative modeling: (A) U-Net and (B) autoencoder. The U-Net model contains a contracting path and an expansive path that down-scales and up-scales the input image to allow simultaneous feature extraction at higher and lower image resolutions. The autoencoder model has two major parts: an encoder that maps the input image to the latent features, and a decoder that maps the latent features to a reconstruction of the input image. As the dimensionality of the latent features is often smaller than the dimensionality of the input image, the model learns a compressed representation of the data and removes noisy components.

chitecture has two major parts connected sequentially: encoder and decoder (see Figure 4 B). The learning objective is to reconstruct the original image as effectively as possible, as judged by a loss function, while forcing the information to pass through a lower-dimensional middle block, labeled "latent features" above. The activations of neurons in this layer thus offer a lower dimensional representation of the input image, sufficient to recover the best reconstructed image possible. It is thus valuable as an unsupervised means of learning the important features of images.

One limitation of the lower dimensional representations learned by AE is that they can follow an arbitrary distribution, which may lead to situations where samples of the same class have drastically different latent features. Varia-

tional AE (VAE) is often used to alleviate this deficiency (Doersch 2016). VAE differs from AE in two ways: its latent features are selected pseudo-randomly and its loss function is extended by a penalty term. During training, each latent feature is independently drawn from a Gaussian distribution $\mathcal{N}(\mu, \sigma)$, where the parameters μ and σ are taken from the output vector of the encoder $\{\mu, \log(\sigma)\}$. This exposes the decoder to a range of encoding vectors (as opposed to a single vector in AE) forcing it to map neighbouring feature vectors to the same image. The loss function is penalized by a Kullback–Leibler divergence of the latent features for a given input sample and the standard normal distribution. The penalty term ensures that the encoder refrains from producing extreme values and encourages it to evenly distribute around the center of the latent space. This leads to a continuous and orthogonal latent space, a highly desired property for data representation.

3 METHODOLOGY: GENERATION OF TRAINING DATASETS

We generate a training sample of lensed quasars and known sets of contaminants for training of the network. Since only a few tens of true lensed quasars have been confirmed, we cannot use them alone to construct a training set, and must generate simulated lensed quasars based on their well understood physics. Ensuring the realism of these simulated observations is essential to the ultimate generalization of our model to real data. We use a version of SIMCT (More et al. 2016) modified for this purpose. Leaving the details to More et al. (2016), we begin by using the redMaGiC galaxy catalog (Rozo et al. 2016) as our parent galaxy sample. All galaxies from this sample are considered potential lenses. By using the measured redshift and magnitudes and known scaling relations, we estimate the lens mass assuming that the mass density profile follows a singular isothermal ellipsoid. We assume mass follows light to determine the centroid, ellipticity and position angle of the lens. We also include external shear to account for effects due to objects in the immediate environments of the lens galaxy. We draw sources from known luminosity functions with a certain i-band magnitude and redshift. Colors are then extracted from the quasar catalog of Tie et al. (2017) by cross-matching the source i-band magnitude and redshift. Given the lens parameters and source parameters, we calculate the lensing cross-section and determine if a source would be lensed by the potential lensing galaxy such that the multiple images can be well-resolved and above the limiting magnitude.

We further implemented the microlensing magnification effect by stars within the lensing galaxy which can affect the fluxes of the lensed quasars. For a given lens and source, we calculate the positions and fluxes of the lensed quasar images. The microlensing effect increases or decreases the flux of the lensed images as determined by the local convergence (κ), shear (γ) and smooth matter fraction ($s = 1 - \kappa_*/\kappa_{\rm tot}$) as described, e.g., by Vernardos (2019). In order to optimize computing resources, we compute microlensing magnification maps for a large number of fixed values of κ, γ and s and interpolate from this grid to real cases. Stars are assumed to have masses 1 M$\odot$ and the stellar density profile is assumed to follow the de Vaucouleurs profile (de Vaucouleurs 1948). We determine the convergence due to compact (stellar) population κ_* in the image plane following Vernardos (2019). The resulting sample consists of about 28500 simulated lensed images of background quasar which are then added on top of the redmagic galaxies from the DES-Y3 data (Sevilla-Noarbe et al. 2021). We show a few simulated lenses in the top row of Figure 5 with fainter systems on the left end and brighter on the right. There are two examples for each of the faint ($i > 19.0$), intermediate ($18.5 < i < 18.$) and bright ($i < 17.5$) magnitude bins. The image cutouts are 6.75 arcsec on the side where each pixel is 0.27 arcsec wide matching the DES pixel resolution.

For the training set of non-lenses, we use the spectroscopically confirmed stars from the Sloan Digital Sky Survey data and photometrically selected quasars (Tie et al. 2017) and blue cloud galaxies (Williams et al. 2017b). As we are interested in quads, we randomly draw objects from these catalogs and place them randomly around a massive galaxy which could mimic a lensed quad. About 2000 such systems are generated and this sample size is increased by a factor of five by applying rotations. We show examples of these non-lenses in the bottom row of Figure 5. As before, fainter systems are on the left end and brighter on the right. These examples correspond to the same magnitude bins as the simulated lenses in the top row. As part of the non-lens sample, we also include the same redMaGiC galaxies that were used to generate simulated lenses but without any lensing features around them. This resulted in 28,500 of simulated positive examples (lenses) and 28,500 of negative examples (contaminant galaxies).

4 METHODOLOGY: MODEL ARCHITECTURE AND TRAINING

Our modeling methodology involves several steps. First, we pre-process, augment, and split the data for purposes of model training and tuning. Then, we employ unsupervised learning methods to explore the data and aid in feature extraction. Next, we train a series of supervised learning models with a variety of architectures. Finally, we develop an ensemble over the resulting models. In this section we describe each of these steps in turn.

4.1 Data pre-processing and splitting

We first standardize each image so that the pixels in each image (pooled together across all four griz-bands) have zero mean and unit standard deviation. More specifically, let I_{ipg} denote the intensity in griz-band $g \in \{1, 2, 3, 4\}$ of pixel p in image i. Let μ_i and σ_i be the sample mean and standard deviation of $\{I_{ipg}\}_{p,g}$, respectively. We normalize each pixel by computing

$$(I_{igp} - \mu_i)/(\sigma_i + \varepsilon) \tag{3}$$

across all i, g and p, where $\varepsilon > 0$ is a small perturbation introduced for numerical stability. We refer to this procedure as *instance normalization*.

The resulting images are augmented with random rotations to ensure a rotation invariant result. We then split the data into training, validation, and testing subsets. This

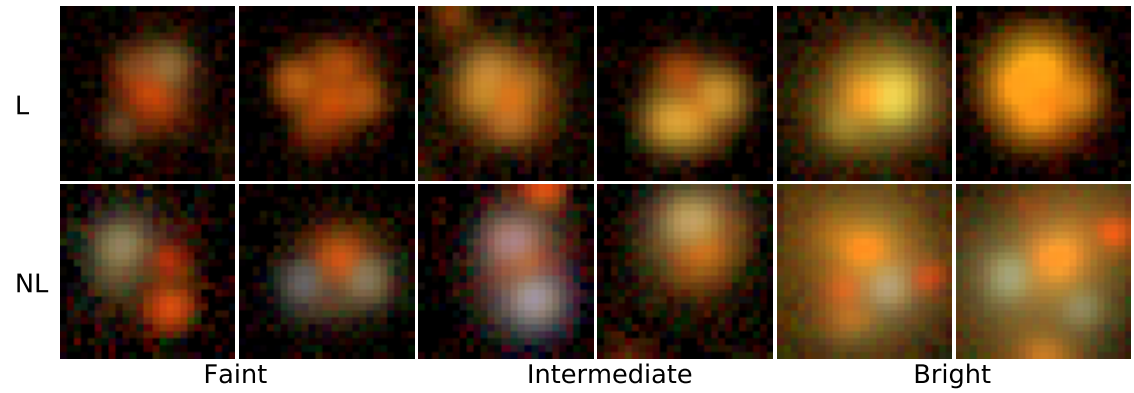

Figure 5. Training sample comprising of lenses (top) and non-lenses (bottom). The left-most two columns show objects selected from the fainter sample ($i > 19.0$), middle two columns show objects from the intermediate sample ($18.5 < i < 18$) and the right-most two columns show the objects from the brighter sample ($i < 17.5$). The images are 6.75 arcsec on the side.

Table 1. Details of various samples used in our analysis.

Sample Name	Description	Sample size	
		Original	Augmented
Simulated lenses	RedMagic deflectors + simulated lensed quasars	28,500	1,254,000
Simulated non-lenses	RedMagic deflectors + contaminants	2,000	128,000
RedMaGiC non-lenses	RedMagic galaxies	28,500	1,140,000
Augmented data	Augmented simulated lenses and non-lenses, and RedMaGiC non-lenses	-	2,522,000
Training dataset	40% of the augmented data	-	1,008,800
Validation dataset	40% of the augmented data	-	1,008,800
Test dataset	20% of the augmented data	-	504,400
Confirmed real systems	Spectroscopically confirmed true positives and negatives from STRIDES	119	128
Mixed dataset	Random samples from the test dataset and additionally augmented confirmed real systems	-	1216

produces roughly 10^6 images in each of the training and validation sets and 5×10^5 in the test set.

The models described below are trained on the training set, with hyperparameters optimized by minimizing the prediction loss on the validation dataset. After training and choosing hyperparameters, the training and validation data are combined to be re-used for training. The resulting model is then evaluated for its performance on the testing set.

4.2 Unsupervised learning

To aid in constructing features that would facilitate the supervised learning process, we begin with an unsupervised exploratory process on a subset of 100,000 images sampled from augmented data. We first decompose the dataset using a Gaussian kernel principal components analysis (kPCA). Explaining 95% of the variance in the dataset requires 8 components. Plotting just the first two principal components (Figure 6A) shows little evidence of the separability of these two classes. Fortunately, however, the higher-dimensional information does suggest the classes are reasonably separable. This is visualized using the two-dimensional t-distributed stochastic neighbor embedding (tSNE) of the images in Figure 6B, which uses the relative similarity of lens images with each other as compared to non-lens images to infer the possibility of successfully distinguishing these objects in a sufficiently rich non-linear feature space.

Next, Figure 7A depicts the architecture of VAE which we use to generate an orthogonal, lower-dimensional latent space characterization. The encoding CNN (left side of the VAE) contains two blocks, each with two convolutional layers followed by normalization and dropout layers. The purpose of the first convolutional block is to extract low-level features using 5×5 kernels. The second convolutional block further expands the effective receptive field using 9×9 kernels to extract higher order features. Note that this configuration makes the effective receptive field of the last convolutional layer of 25×25, covering the entire input image. The drop-out layer is used to reduce overfitting by dropping a portion of the randomly selected inputs.

The subsequent block consists of two parallel dense layers that process the input tensor and map it to the latent space of dimension p. The upper layer applies the sigmoid activation to its output to estimate a *feature-selecting vector* $\mathrm{sigmoid}\big(W_{upper}z\big)$ for each training sample. By multiplying it with the latent space representation of an image z produced by the output of the lower layer, $W_{lower}z$, we locally suppress irrelevant latent features of each sample:

$$(W_{lower}z) \odot \mathrm{sigmoid}\big(W_{upper}z\big),$$

where W_{lower} and W_{upper} are $p \times q$ matrices and z is a flattened image of size $q \times 1$. The output dense layer, shown in pink in Figure 7A, linearly combines the remaining features to estimate the means and standard deviations. Finally, as

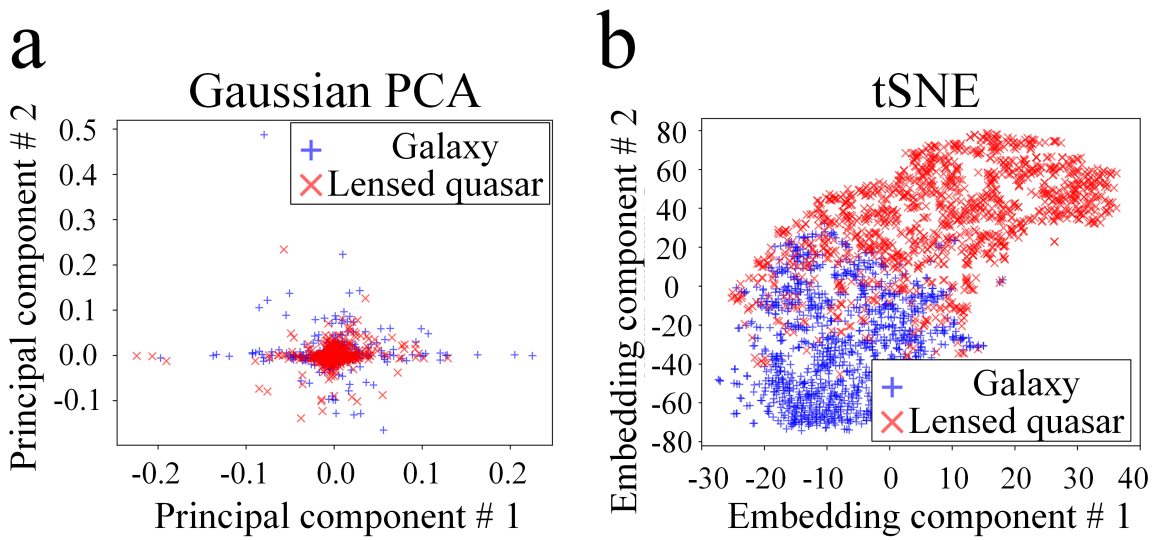

Figure 6. Visualization of a subset of the data consisting of galaxies (blue) and lensed quasars (red) after dimensionality reduction: (A) the first two components of the Gaussian Principal Component Analysis (PCA) and (B) two-dimensional t-distributed stochastic neighbor embedding (tSNE). PCA shows little evidence of the separability of lenses from non-lenses. The two-dimensional tSNE, however, suggest that the classes are reasonably separable in a higher dimensional space.

shown on the right half, the process is repeated in reverse to reconstruct an image from the compressed features.

To explore the result, Figure 7B shows how the two classes are distributed along four selected latent features. As VAE gives a mean (μ) and a standard deviation (σ) for each latent feature, the aforementioned figure depicts empirical distribution of the means of the latent features. We observed that although many features such as feature 1 do not discriminate contaminant galaxies from lensed quasars, some of the features such as feature 4 do separate the two classes well. To better understand the meaning of the latent features, we plot some of the weights of the lower dense layer in Figure 7C. The plots depict the underlying modes that naturally span the dataset including those that prevail among lensed quasars. To see how dimensionality of the latent space impacts the explained variance along individual bands (g, r, i, z) or their combination (griz), we apply linear PCA on the features of 1024-dimensional VAE models that were trained to reconstruct a single band or the entire image. The results in Figure 7 D suggest that 16-dimensional space covers 95% of the variance in griz-bands (around 12 latent variables for R), while 128 features correspond to 99% of the variance (around 64 latent features in R). This shows that VAE models can learn meaningful features and describe the dataset with a 128 latent variables. We later employ an encoding portion of the VAE models for feature extraction and subsequent image classification.

4.3 Supervised learning

This section provides details for the construction and fitting of several neural network architectures, before describing how they are combined in an ensemble.

The unsupervised learning explored in Section 4.2 indicates that lensed quasars have distinctive geometric features that could be utilized in their detection. We frame lens detection as a binary classification problem in which "one" corresponds to a sample with a lensed quasar and "zero" corresponds to anything else. A classifier takes an image as an input and outputs a binary label. In practice, the output layer has two units, whose activations we can refer to as z_0 and z_1. To these we apply the softmax function to obtain quantities we interpret as probabilities, i.e., $\hat{y}_1 = \mathbb{P}(Y = \text{“lens”}) = \exp(z_1)/(\exp(z_0) + \exp(z_1))$, with $\hat{y}_0 = \mathbb{P}(Y = \text{“not lens”}) = 1 - \mathbb{P}(Y = \text{“lens”})$. We found empirically that training with the softmax activation results in a better generalization compared to training with the sigmoid activation.

For the loss function, we employ the binary cross-entropy, also known as the negative log-likelihood, between a class label $y \in Y$ and the predicted probabilities $(\hat{y}_1, \hat{y}_0)$,

$$\mathcal{L}(y, \hat{y}_1, \hat{y}_0) = -y \cdot \log(\hat{y}_1) - (1 - y) \cdot \log(\hat{y}_0).$$

Moreover, binary cross-entropy leads to a more consistent gradient propagation as the log-terms mitigate exponential behavior due to gradient saturation for extreme values.

Instead of the conventional batch normalization, we use instance normalization introduced in (3). While instance normalization makes the samples of both classes statistically indistinguishable in terms of the pixelwise mean and variance, it has several important advantages. First, it limits the range of the values that tensors can take, preventing saturation of hyperbolic activation functions, and improving gradient propagation. Second, it improves generalization by avoiding overfitting on synthetic data, as the gap between the simulated and the real objects is often the major issue in problems that rely on statistical models trained on synthetically-generated data. In particular, it reduces the effect of the outliers that could cause significant covariate shift in hidden layers of the ANN.

To penalize overfitting, we extensively use dropout lay-

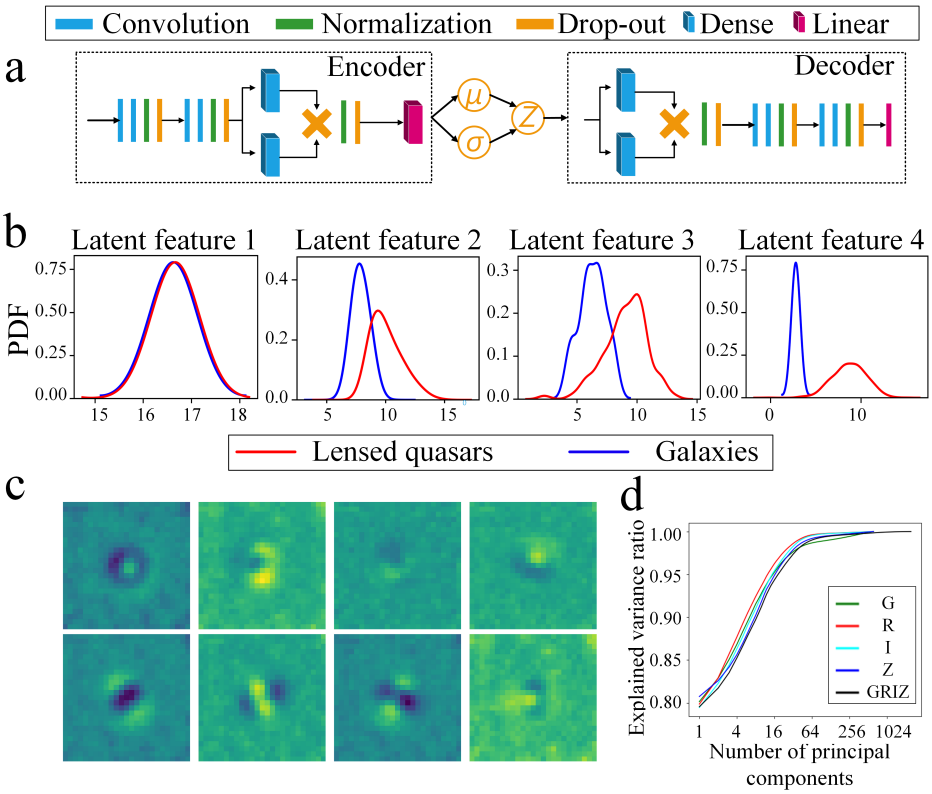

Figure 7. Unsupervised exploration of the latent space of the dataset: (A) a VAE model used for learning the mapping from images to the means and standard deviations of the latent variables; (B) empirical distributions of selected latent variables showing that despite nearly identical statistical properties in galaxies and lensed quasars across the majority of learned latent variables, some of them describe features that can be used to distinguish between these two classes; (C) a visualization of a few learned features of the lower dense layer of the VAE model generated by putting the learned weights into square shapes of the same size as the output of the previous convolutional layer; (D) empirical relationship between the explained variance and the number of principal components of the latent variables of 1024-dimensional VAE suggests that a 128-dimensional latent space can be sufficient to describe the dataset.

ers. The exact order of the hidden layers, activation functions, normalization layer, and dropout layers is typically optimized for every problem. We empirically found that the combination of activation layers followed by normalization and dropout layers achieve a better bias–variance trade-off. To additionally limit overfitting, we introduce zero-mean Gaussian noise at the input of each model, which adds random noise to the input images at every training step. We set the noise covariance matrix to $\sigma^2 I$, where I is the identity matrix and $\sigma^2 = 0.062$, with the value determined by hyper-parameter optimization. Note that this is equivalent to additional data augmentation performed simultaneously with training.

Models are trained using the Adam algorithm (Kingma & Ba 2015). Adam uses adaptive learning rates for every model parameter, enabling faster convergence. We also employ the scheduled (staircase exponential) and triggered decrease of the learning rate. The latter changes the step size when the loss rate of decay is below a certain threshold.

To further combat overfitting, we employ early stopping based on the validation loss. More specifically, after every iteration on the training set, we evaluate the loss function on the validation set which includes data not used for optimizing model parameters. We stop once the validation, rather than the training, loss has converged.

Besides trainable parameters, each model has hyperparameters, a small set of values that define model architecture and training dynamics. These include model depth (number of hidden layers), model width (number of convolutional filters and dense units), model resolution (size of convolutional kernels), and the choice of the activation function, regularization strength, and learning rate. We use the recently proposed Hyperband algorithm (Li et al. 2018), an efficient bandit-based method for hyperparameter optimization. It allocates computational resources to as many configurations as possible and throws out those that show poor performance over time until a single configuration remains. This method maximizes the number of tested configurations and results in a more efficient resource utilization compared to the grid search, random search, or Bayesian optimization.

4.3.1 Prior methods

Numerous prior works proposed using the traditional computer vision models such as InceptionNet, ResNet, and DenseNet for lens searches. These architectures have demonstrated exceptional performance on traditional computer vision problems, but the vast number of trainable parameters they require can result in either poor generalization (overfitting) or weak convergence (underfitting). Pre-training these

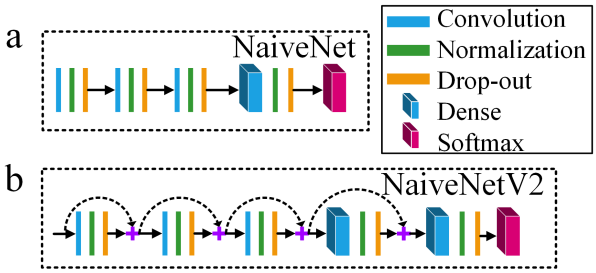

Figure 8. Optimized instances of generalized CNN architectures: (A) NaiveNet based on three convolutional blocks and two dense layers and (B) NaiveNetV2 that uses residual connections and adds an additional dense layer to improve the performance.

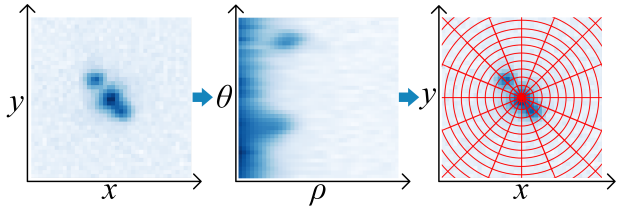

Figure 9. Visualization of the proposed polar convolution operation: each feature map gets converted to higher resolution polar coordinates where a regular rectangular convolution is applied and then translated back to Cartesian coordinates. Effectively, the operation has a convolutional window of an angular sector of annuli of different radius (shown in red) with smaller window size at the center of the image.

models on the ImageNet, a large dataset of ordinary images collected for object recognition, degrades the performance further. We hypothesize that this is due to drastic differences in visual features between the lensed quasars and ImageNet samples.

Fortunately, several earlier projects have sought to detect lenses using various deep CNN architectures. We describe these here before proposing two novel architectures. Previously proposed CNN models include CMU DeepLens (Lanusse et al. 2017), LensFlow (Pourrahmani et al. 2018), CNNS (Jacobs et al. 2019b), and several others (Hezaveh et al. 2017; Schaefer, C. et al. 2018; Avestruz et al. 2019). We reproduce and retrain LensFlow and CNNS on our data for comparison. LensFlow is based on a classical architecture comprised of three convolutional layers, maximum pooling layer and four dense layers all connected in series. Drop-out layers in-between the dense layers mitigate over-fitting. Unusually, LensFlow applies hyperbolic tangent to the outputs of the convolutional layers and ReLu activation function to the output of the dense layers. CNNS combines four convolutional layers into blocks of two layers each with three subsequent dense layers. Each convolutional block is followed by pooling operations, halving the dimensionality of the feature maps the output. The activation function is ReLu, and drop-outs are again used to mitigate over-fitting.

4.3.2 NaiveNet

Inspired by these early examples, we first propose a CNN involving several convolutional layers followed by dense layers, which we refer to as NaiveNet, illustrated in Figure 8. The number of layers, number of dense units (or convolutional kernels) within each layer, and size of the convolutional kernels respectively, are optimized together with other hyperparameters. We intentionally avoid specific constraints on the architecture and, instead, optimize performance through a hyperparameter search. Next, we introduce NaiveNetV2, replacing convolutional and dense blocks with their residual alternatives. Each residual path has two consecutive layers: the first one with a chosen non-linearity and the second one with a linear activation function.

4.3.3 Polar convolution

Next, we propose an alternative approach to feature extraction. Since we can ensure positioning of the objects at the center of the griz images and given that the gravitational field originates from the center of mass of the objects, a polar coordinate system is a natural choice. We define the corresponding 2D polar convolution similarly to its rectangular counterpart:

$$\big(W * T(z)\big)_{\rho,\theta} = \sum_i \sum_j W_{i,j} \cdot T(z)_{\rho+i,\theta+j} \tag{4}$$

where $T(\cdot)$ translates an input tensor from Cartesian to a polar coordinate system. It first transforms a fixed size rectangular tensor z into a scatter field of points z^* with polar coordinates $\{\rho, \theta\} = \{\sqrt{(x^2+y^2)}, \arctan(y/x)\}$. To avoid loss of information due to image cropping at the corners and blurring between pixels, it estimates a smooth rectangular tensor s using biharmonic spline interpolation:

$$\begin{cases} s_i = \sum_j \alpha_j g(i,j) \\ g(i,j) = \|p_i - p_j\|^2 (\log \|p_i - p_j\| - 1) \end{cases} \tag{5}$$

where g is Green's function, $p_i = [x_i, y_i]$ is a vector pointing at the corresponding scatter point with value z_i^*, and the weight vector α can be found by solving $g \cdot \alpha = z^*$. An illustration of the proposed operation is shown in Figure 9. Beyond fitting the underlying geometrical structure on the images, polar convolution yields a heterogeneous receptive field size and a particularly sparse connectivity matrix by focusing at the central part of the image, which is useful as the corners of the images are expected to be less informative, with more noise. Similar ideas were previously proposed in several works including Polar Transformer Network (Esteves et al. 2018), Polar Coordinate CNN (Jiang & Mei 2019), and Cylindrical CNN (Kim et al. 2020). Our approach, however, was developed for learning features suitable for finding lensed quasars.

4.3.4 Visual attention masking

We train an additional U-Net-like model that produces a binary mask representing regions of lensed quasars (named as AttnCNN in Figure 10 D). Similarly to U-Net, AttnCNN is a fully convolutional network with contracting and upsampling branches, but it does not output a segmentation mask. Instead, it learns the mapping from multi-channel input image (griz) to an attention mask of a smaller size, with a single channel, and values between 0 and 1. The goal of

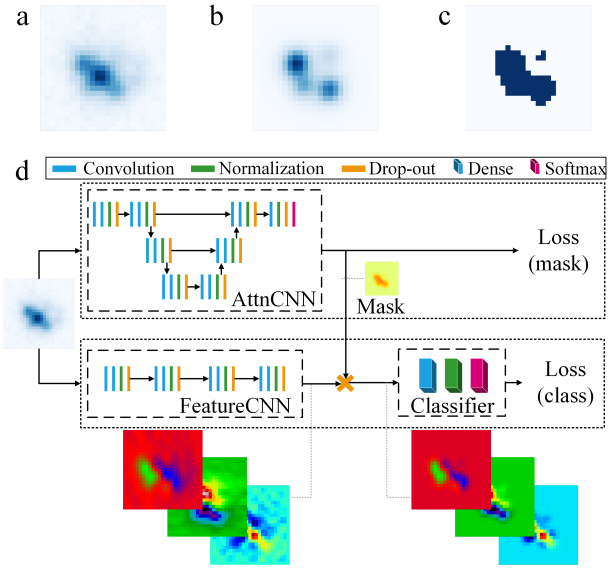

Figure 10. An example of the proposed attention masking for feature extraction: (A) a sample of the original input image; (B) the corresponding lensed quasar blended in the original image; (C) a binary mask used as a label for training the attention CNN model. The mask is produced by setting a threshold on intensity of the lensed quasar image. (D) The proposed AttnNet architecture is comprised of three models: the AttnCNN trained to segment regions containing lensed quasars, the FeatureCNN that extracts features from the original image and then applies the estimated binary mask to remove noisy components, and a binary classifier trained to identify lensed quasars.

this auxiliary masking network is to guide feature extraction by bringing attention to regions likely to have important features. We found empirically that thresholding image pixels with 22.5% of their maximum intensity creates meaningful binary masks as the brightest regions are the most informative. We tried several loss functions for training the AttnCNN and found that mean absolute error leads to the most stable results. It was beneficial to train the entire assembly simultaneously in an end-to-end fashion. At each optimization step i (batch size of n), we update the parameters of the AttnCNN model $\Psi(\cdot)$ to minimize the discrepancy between its outputs $\Psi(X_i)$ and the binary masks M_i:

$$\mathcal{L}_i^{\text{AttnCNN}} = \frac{1}{n}\sum_{k=1}^{n}|\Psi(X_i)_k - M_{ik}|.$$

At the same time, we independently update parameters of the feature extracting model $\Lambda(\cdot)$ (named as FeatureCNN in the figure) and the binary classification model $\Pi(\cdot)$ by backpropagating the gradient of the cross-entropy loss $\mathcal{L}_i\Big(\Pi\big[\Psi(X_i)\odot\Lambda(X_i)\big], Y_i\Big)$. This way AttnCNN directly affects the performance of the feature extracting CNN and the classifier, which forces FeatureCNN to adjust its learned parameters.

4.4 Argus: ensemble of models

One of the challenges we encountered was overfitting stemming from the gap between the simulated and real data. Particularly, we observed a significantly lower performance during testing on real samples generated from the DES, despite the imposed regularization and optimal synthetic testing results. Given the limited size of the real dataset, we found it impractical to directly bridge the gap by finding a projection between real and synthetic samples. Moreover, since the confirmed objects do not represent the entire range of possible lenses, additional training on them would further amplify the bias.

Instead, a more practical and efficient method to improve generalization is model stacking (sometimes called blending), which combines the outputs of pre-trained models. We employ this technique by blending classifier models and VAEs (see Figure 11). The upper branch maps the hidden feature spaces of the AttnNet and NaiveNetV2 based on rectangular and polar convolutions to another 128-dimensional space. It employs two dense layers with ReLu non-linearity to fuse a 512-dimensional vector from the AttnNet with a pair of 128-dimensional vectors from the rectangular and polar NaiveNetV2: $\Phi_s : \mathbb{R}^{512+128+128} \mapsto \mathbb{R}^{128}$. The lower branch mixes the latent spaces of VAE models that were pre-trained to reconstruct specific components on the images including lensed quasars (VAE_{LQ}), contaminant galaxies (VAE_G), underlying source quasars (VAE_Q), and deflectors (VAE_D) that represent the strength of the gravitational field of the lensing galaxy. We found that reconstruction on the quasars and deflectors achieved the best results when conditioned on the output of the VAE_{LQ} and VAE_G respectively. This is implemented by stacking the second order VAE model on top of the decoder and using the mean component of the latent features of the first encoder. We choose a 128-dimensional latent space since it is able to converge to a global minimum of the loss function and it was earlier suggested by the results of PCA to keep the explained variance ratio above 99%. However, because galaxies and lensed quasars have an identical distribution along the majority of the latent space axes, we apply a stronger dimensionality reduction: $\Phi_u : \mathbb{R}^{512+512} \mapsto \mathbb{R}^{64}$. Note that the standard deviations estimated by the encoding CNN models carry information about the noise in the images and the associated confidence levels. Since μ and σ often have different scales and distributions we first blend them separately and then combine new intermediate feature spaces. The output layer linearly combines the features of the upper and lower branches to classify the input samples.

To train the entire end-to-end model we split the training dataset into two halves. We train each individual component (supervised and unsupervised) on the first half. Then, after stacking, we train the blending dense layers using the second half of the data. This scheme allows us to minimize additional overfitting as the blending layers learn how to combine the feature spaces of the pre-trained models on previously unseen samples. Moreover, to foster generalization we use normalization and dropout layers as well as combination of L_1 and L_2 ("elastic net") regularization. The corresponding hyperparameters of the model and its components are selected using the Hyperband search described earlier. After training Argus, we investigated the importance of the extracted features from each component of the ensemble by conducting an ablation study. We disconnected one of the components at a time and evaluated model predictions using AUROC score on the *mixed dataset* as de-

Table 2. Classification performance (AUROC).

Model & parameters	Simulated objects	Past candidates	Mixed dataset
CNNS (8.8M)	0.889	0.580	0.853
LensFlow (8.8M)	0.901	0.593	0.855
NaiveNet (10.2M)	0.901	0.605	0.862
NaiveNetV2 (2.4M)	0.914	0.684	0.863
NaiveNetV2 polar (2.4M)	0.937	0.582	0.864
AttnNet (5.8M)	0.940	0.774	0.878
Argus (17.7M)	0.997	0.650	0.894

scribed in the next section. Performance severely degraded by dropping any component, excepting NaiveNetV2: 3.86% drop without VAEs, 6.28% drop without AttnNet, 3.48% drop without polar NaiveNetV2 and 0.23% drop without NaiveNetV2. In addition, we observed no significant differences in the distribution of the weights of the dense layers that take the outputs of different components, which makes us believe that each part is equally important and benefits the overall performance.

5 RESULTS

5.1 Training and testing setup

After training these models we tested their performance on three datasets that were not used during the training. First, the *simulated objects* is a conventional "testing" dataset that was sampled from the data generated for model training as described in Section 3. It contains around 250,000 lensed quasars and 250,000 contaminant galaxies. Second, the *past candidates* dataset contains 20 confirmed lensed quasars and 108 confirmed non-lenses that were previously proposed as candidates for spectroscopic follow-up within the STRIDES collaboration (as collated by one of us, C.L.). Finally, the *mixed dataset* contains 1216 objects randomly sampled from the *simulated objects* and *past candidates*. We augmented the samples from *past candidates* via random rotations and mirroring to achieve a roughly 1:1 ratio.

We trained each model on a computer with a single GPU (Nvidia GeForce GTX 1080 Ti). A full training cycle of a single model with hyperparameter optimization took between 20 to 180 hours depending on the model. In addition to our proposed models, we reproduced two previously reported models, CNNS and LensFlow, retraining these on our data. In doing so we followed the design choices as reported, but adjusted learning rates and the corresponding parameters (decay rate, learning rate schedule, etc.) to maximize performance.

5.2 Performance

As a first performance metric we consider the area under the receiving operating curve (AUROC). This can be interpreted as the probability that the model ranks a random positive example more highly than a random negative example.

Table 2 shows the results. Both existing models, CNNS and LensFlow, had approximately 8.8 million trainable parameters and demonstrated AUROC values of roughly 0.85 on the mixed data, comparable to our initial NaiveNet model. The enhanced version with residual skip-connection, NaiveNetV2, achieved an AUROC of 0.86 on the mixed dataset while reducing the number of parameters to 2.4 million.

Attention masking (AttnNet), however, led to a substantial improvement in performance with an AUROC of 0.940 on the simulated objects and 0.774 on the past candidates. AttnNet was the most conservative in detecting lenses and had the highest AUROC on the past candidates dataset, which is driven by the built-in feature selection and stronger orthogonal regularization. Finally, the proposed ensemble model (Argus) reached a nearly perfect AUROC score of 0.997 on the simulated data, but had a slightly lower score on the past candidates compared to AttnNet. On the mixed data, Argus achieved an AUROC of 0.894, the highest among all models.

ROC curves of the AttnNet and Argus models on the "mixed dataset" are shown in Figure 12. The Argus ensemble achieves a nearly perfect true positive rate while keeping the false positive rate under 0.3. The AttnNet demonstrates true positive rate of about 0.8 for the same false positive rate under 0.3.

Figure 13 depicts how precision (the fraction of true lenses among the predicted lenses) and recall (the fraction of correctly identified lenses among true lenses) of two best performing models change with AB magnitude of the whole system (obtained by summing the flux over the entire cutout; for comparison, the 10-σ limiting magnitude of the data for point sources is 23.34). To produce these plots we grouped the samples of the mixed dataset into five bins of equal width in AB magnitude and evaluated the metrics for each group. We found that as AB magnitude increases the proposed models start suffering from a noticeable drop in precision. This is expected because of the declining signal to noise ratio of the images and false positives become more difficult to identify as such. Remarkably, however, recall is uniformly close to 100% suggesting that our method should achieve high completeness even when the signal to noise ratio is low.

Figure 14 shows objects misclassified as lenses (false positives) by the Argus Ensemble model. Those in the top row were also selected by previous searches and were considered credible enough to warrant telescope time for investigation, so it is not catastrophic that they mislead Argus. Those in the bottom row were simulated false positives and indeed look like plausible lenses to a human classifier, so it does not seem catastrophic that they mislead Argus as well. At some level, a small number of false positive is inevitable due to the finite amount of information in the imaging data used for the search. Those false positives will need additional data to be resolved (spectroscopy and/or higher resolution images). We discuss the practical implications of this and other limitations to be kept in mind when applying this method to search for quads in the next section.

5.3 Practical Considerations

Considering that telescope time is costly and limited, we wish to estimate how effectively lists of candidates produced by Argus or AttnNet can prioritize search time. The two key aspects of performance in this context are *precision*—the

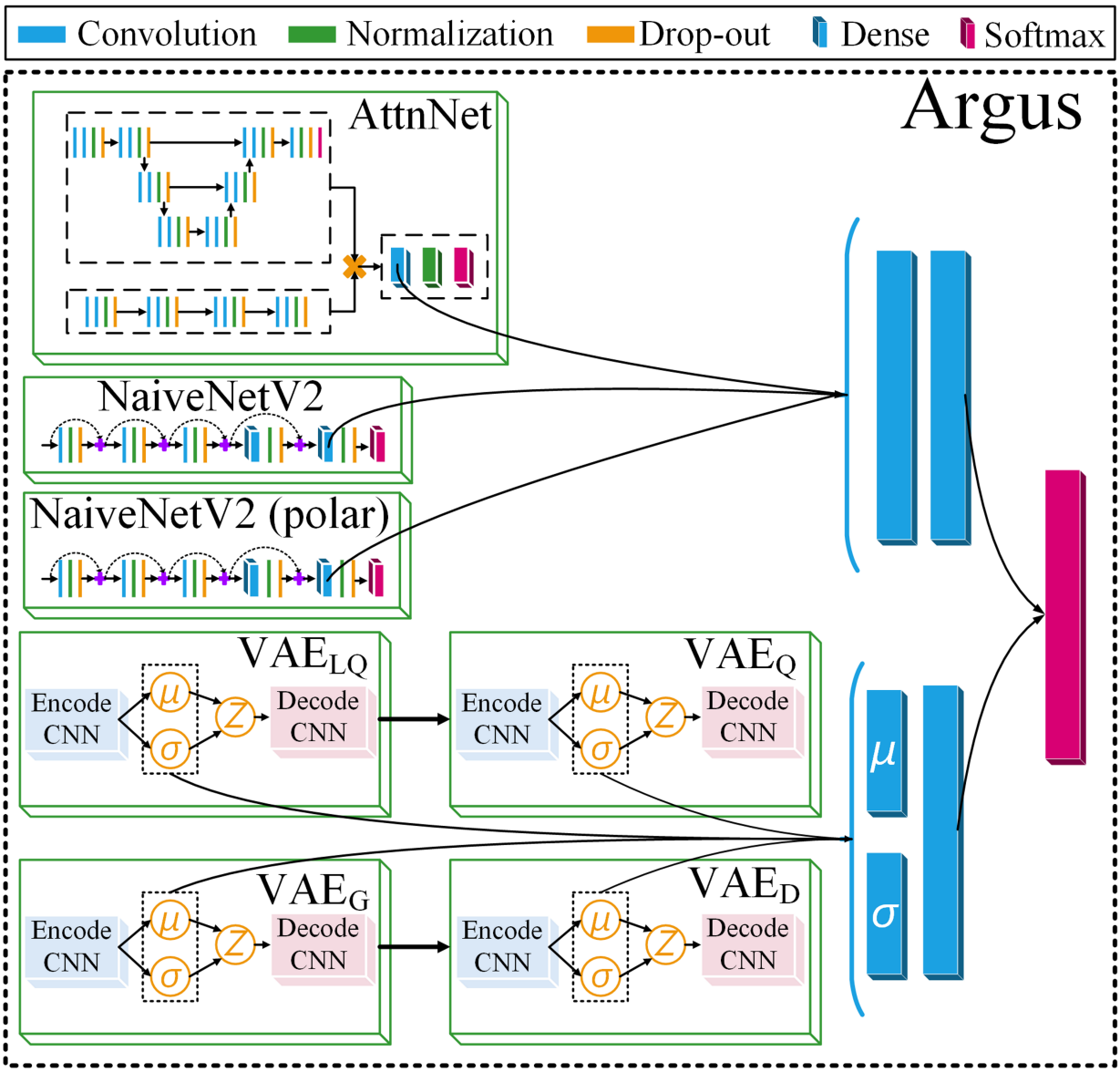

Figure 11. Argus: an end-to-end ensemble of deep neural networks. The proposed method includes feature extraction elements of pre-trained models: a 512-dimensional output from the first dense layer of AttnNet, and a 128-dimensional vector from the second dense layer of NaiveNetV2. These features are concatenated together and then reduced to 128-dimensional space through two dense layers. In addition, the model takes 128-dimensional latent features of four serially-connected VAE models trained on reconstructing images of lensed quasars (LQ), galaxies (G), delensed quasars (Q), and deflector fields (D) from the blended input image. As the encoders produce both means and standard deviations of these features, they are first processed independently and then combined into a 64-dimensional feature vector. The final output is produced by a dense layer that classifies samples based on 192 features.

proportion of candidates examined that prove to be lenses—and *recall*, the fraction of true lenses that are identified in a given catalog. Figure 15 shows the estimated precision and recall as we adjust the number of selected candidates ranked by the models. Both Argus and AttnNet look promising in the sense that over a wide range of the number of candidates that might feasibly be considered, the precision rate remains quite high, well over 80% until the number of candidates begins to approach the actual number of lenses in this sample (400). Naturally, the recall can at best grow linearly with a slope of 1, and the observed recall is not far below this. If these results generalize to images outside of those used in this paper, this plot would suggest that a search using our methods would be very efficient, yielding a complete sample of quadruply imaged quasars with a relatively modest fraction of contaminants. A follow-up paper will put our methods to further test by applying them to new DES datasets.

6 SUMMARY

In this work, we proposed and trained novel deep learning-based methods for detecting quadruply lensed quasars. We investigated several ANN architectures and compared them with previously proposed LensFlow and CNNS models.

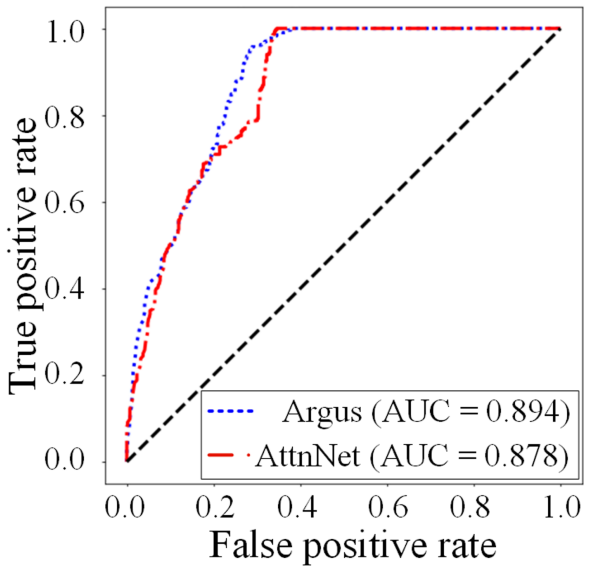

Figure 12. Receiver operating characteristic (ROC) curves of the best performing models on the *mixed dataset*, which show the possible trade-offs between the probability of a correct detection (true positive rate) and the probability of false detection (false positive rate). The Argus ensemble achieves a nearly perfect true positive rate while keeping the false positive rate under 0.3. The AttnNet demonstrates true positive rate of about 0.8 for the same false positive rate under 0.3.

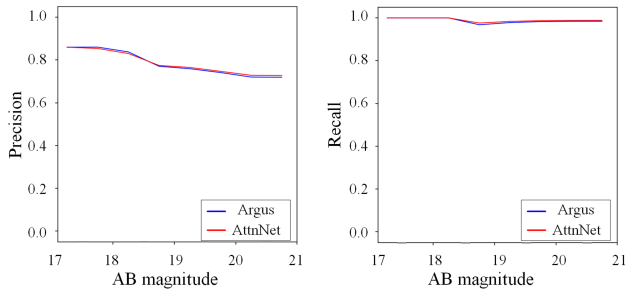

Figure 13. Precision and recall as a function of the AB magnitude of the whole system (obtained by summing up the flux in the cutout) estimated on the *mixed dataset*. For comparison, the 10-σ limiting magnitude of the data for point sources is 23.34.

Overall, our newly developed tools and in particular the ensemble model improve on the previously proposed algorithms (Agnello et al. 2015; Williams et al. 2017a; Hezaveh et al. 2017; Petrillo et al. 2017, 2018; Lanusse et al. 2017; Pourrahmani et al. 2018; Schaefer, C. et al. 2018; Avestruz et al. 2019; Madireddy et al. 2019; Cheng et al. 2020; Jacobs et al. 2019a; Jacobs et al. 2019b) in terms of precision and recall, although prior work was not tailored to the detection of quadruply imaged quasars, and therefore the comparison is not direct.

A major strength of our method is its ability to process 10^8 objects in a single day on an ordinary hardware setup with a single GPU. This makes it suitable for discovering lenses across wide areas from surveys such as DES with minimal pre-screening of candidates, reducing the risk of discarding lenses. In a follow-up paper we will carry out such a search.

Two particularly novel and powerful elements of our approach are the use of polar convolution, which is well tuned to detecting circularly-arranged features at different scales, and attention masking. In the future, yet other ideas for feature extraction approaches, likely motivated by expert knowledge, may further improve performance.

Finally, the challenges that remain are not only algorithmic, but in the acquisition of training data, real or simulated. That said, one area in which further progress could be made is in taking account of additional data, such as bands beyond griz, or using time series rather than single epoch images. Availability of larger sets of unlabeled and labeled training data might enable use of more sophisticated and accurate models including deep CNN models such as ResNet, DenseNet, and EfficientNet (Tan & Le 2019), as well as novel Vision Transformer architectures (Dosovitskiy et al. 2020). Another promising direction for future research includes use of ranking models that are routinely trained on large volumes of data in the domain of recommender systems, because the problem of detecting extremely rare objects such as gravitational lenses practically comes down to ranking of a set of candidates with respect to the probability of being a lens. For example, a promising solution could be an ensemble of CNN and transformer models trained on several tasks such as classification, ranking, prediction, and so on. However, use of complex ensembles requires thorough analysis of the extracted features and the development of methods to understand and interpret predictions made by each component, which could be another interesting direction for future research.

DATA AVAILABILITY STATEMENT

All data and code required to reproduce these results and apply the trained model will be made publicly available upon publication at `https://des.ncsa.illinois.edu/releases/other/paper-data`.

ACKNOWLEDGMENTS

We are grateful to Paul Schechter for numerous insightful conversations on lens finding and training sets. TT acknowledges support from the National Science Foundation through grant NSF-AST-1906976, from NASA through grants HST-GO-15320 and HST-GO-15652, and from the Packard Foundation through a Packard Research Fellowship.

Funding for the DES Projects has been provided by the U.S. Department of Energy, the U.S. National Science Foundation, the Ministry of Science and Education of Spain, the Science and Technology Facilities Council of the United Kingdom, the Higher Education Funding Council for England, the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign, the Kavli Institute of Cosmological Physics at the University of Chicago, the Center for Cosmology and Astro-Particle Physics at the Ohio State University, the Mitchell Institute for Fundamental Physics and Astronomy at Texas A&M

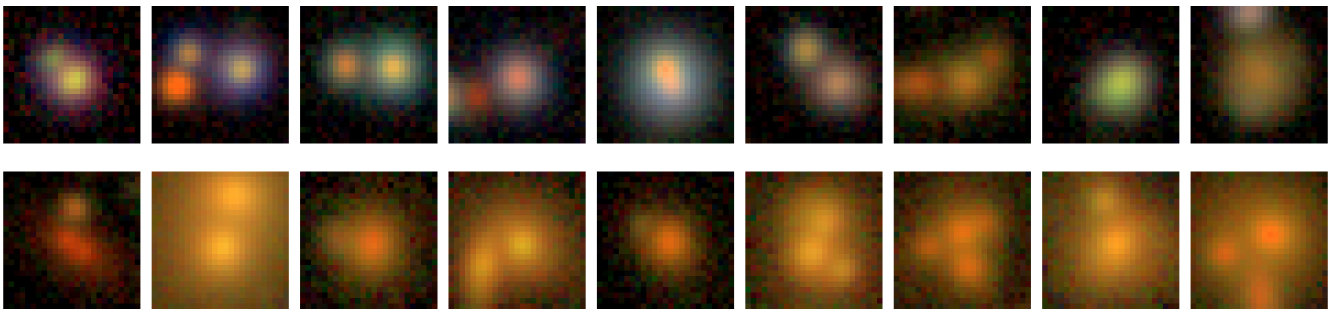

Figure 14. Examples of false positives found by the Argus ensemble model. The false positives identified by Argus from a sample of lenses found by previous searches in the DES data are shown in the top row. The false positives identified by Argus from the simulated sample are shown in the bottom row.

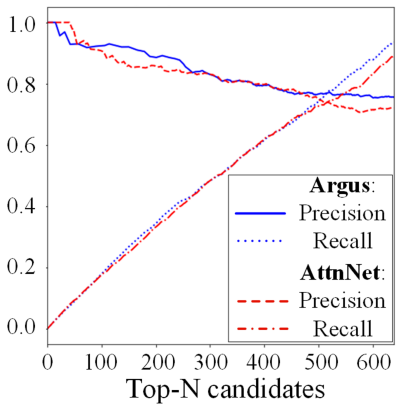

Figure 15. Precision and recall at n highest-ranked candidates in the *mixed dataset*. The plots were generated by taking n candidates with the highest scores estimated by the models as predicted *positives* and taking the remaining objects as predicted *negatives*.

University, Financiadora de Estudos e Projetos, Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro, Conselho Nacional de Desenvolvimento Científico e Tecnológico and the Ministério da Ciência, Tecnologia e Inovação, the Deutsche Forschungsgemeinschaft and the Collaborating Institutions in the Dark Energy Survey.

The Collaborating Institutions are Argonne National Laboratory, the University of California at Santa Cruz, the University of Cambridge, Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas-Madrid, the University of Chicago, University College London, the DES-Brazil Consortium, the University of Edinburgh, the Eidgenössische Technische Hochschule (ETH) Zürich, Fermi National Accelerator Laboratory, the University of Illinois at Urbana-Champaign, the Institut de Ciències de l'Espai (IEEC/CSIC), the Institut de Física d'Altes Energies, Lawrence Berkeley National Laboratory, the Ludwig-Maximilians Universität München and the associated Excellence Cluster Universe, the University of Michigan, NSF's NOIRLab, the University of Nottingham, The Ohio State University, the University of Pennsylvania, the University of Portsmouth, SLAC National Accelerator Laboratory, Stanford University, the University of Sussex, Texas A&M University, and the OzDES Membership Consortium.

Based in part on observations at Cerro Tololo Inter-American Observatory at NSF's NOIRLab (NOIRLab Prop. ID 2012B-0001; PI: J. Frieman), which is managed by the Association of Universities for Research in Astronomy (AURA) under a cooperative agreement with the National Science Foundation.

The DES data management system is supported by the National Science Foundation under Grant Numbers AST-1138766 and AST-1536171. The DES participants from Spanish institutions are partially supported by MICINN under grants ESP2017-89838, PGC2018-094773, PGC2018-102021, SEV-2016-0588, SEV-2016-0597, and MDM-2015-0509, some of which include ERDF funds from the European Union. IFAE is partially funded by the CERCA program of the Generalitat de Catalunya. Research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Program (FP7/2007-2013) including ERC grant agreements 240672, 291329, and 306478. We acknowledge support from the Brazilian Instituto Nacional de Ciência e Tecnologia (INCT) do e-Universo (CNPq grant 465376/2014-2).

This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.

REFERENCES

Agnello A., Kelly B. C., Treu T., Marshall P. J., 2015, MNRAS, 448, 1446

Annis J., et al., 2014, ApJ, 794, 120

Avestruz C., Li N., Zhu H., Lightman M., Collett T. E., Luo W., 2019, The Astrophysical Journal, 877, 58

Cheng T.-Y., Li N., Conselice C. J., Aragon-Salamanca A., Dye S., Metcalf R. B., 2020, Monthly Notices of the Royal Astronomical Society, 494, 3750

Clevert D.-A., Unterthiner T., Hochreiter S., 2016, in Bengio Y., LeCun Y., eds, 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings. `http://arxiv.org/abs/1511.07289`

Doersch C., 2016, Tutorial on Variational Autoencoders (arXiv:1606.05908)
Dosovitskiy A., et al., 2020, CoRR, abs/2010.11929
Esteves C., Allen-Blanchette C., Zhou X., Daniilidis K., 2018, International Conference on Learning Representations
Goodfellow I., Bengio Y., Courville A., 2016, Deep Learning. MIT Press
He K., Zhang X., Ren S., Sun J., 2015, in Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV). ICCV 15. IEEE Computer Society, USA, p. 1026–1034, doi:10.1109/ICCV.2015.123, https://doi.org/10.1109/ICCV.2015.123
He K., Zhang X., Ren S., Sun J., 2016, in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
Hezaveh Y. D., Levasseur L. P., Marshall P. J., 2017, Nature, 548, 555
Huang G., Liu Z., Van Der Maaten L., Weinberger K. Q., 2017, in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp 2261–2269, doi:10.1109/CVPR.2017.243
Jacobs C., et al., 2019a, ApJS, 243, 17
Jacobs C., et al., 2019b, The Astrophysical Journal Supplement Series, 243, 17
Jiang R., Mei S., 2019, in 2019 IEEE International Conference on Image Processing (ICIP). pp 355–359, doi:10.1109/ICIP.2019.8802940
Kim J., Jung W., Kim H., Lee J., 2020, ArXiv, abs/2007.10588
Kingma D. P., Ba J., 2015, in Bengio Y., LeCun Y., eds, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. http://arxiv.org/abs/1412.6980
Klambauer G., Unterthiner T., Mayr A., Hochreiter S., 2017, in Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS 17. Curran Associates Inc., Red Hook, NY, USA, p. 972–981
Lanusse F., Ma Q., Li N., Collett T. E., Li C.-L., Ravanbakhsh S., Mandelbaum R., Poczos B., 2017, Monthly Notices of the Royal Astronomical Society, 473, 3895
LeCun Y., Bengio Y., Hinton G., 2015, Nature, 521, 436
Lemon C., et al., 2020, MNRAS, 494, 3491
Li L., Jamieson K., DeSalvo G., Rostamizadeh A., Talwalkar A., 2018, Journal of Machine Learning Research, 18, 1
Madireddy S., Li N., Ramachandra N., Butler J., Balaprakash P., Habib S., Heitmann K., 2019, A Modular Deep Learning Pipeline for Galaxy-Scale Strong Gravitational Lens Detection and Modeling (arXiv:1911.03867)
More A., et al., 2016, MNRAS, 455, 1191
Oguri M., Marshall P. J., 2010, MNRAS, 405, 2579
Petrillo C. E., et al., 2017, Monthly Notices of the Royal Astronomical Society, 472, 1129
Petrillo C. E., et al., 2018, Monthly Notices of the Royal Astronomical Society, 482, 807
Pourrahmani M., Nayyeri H., Cooray A., 2018, The Astrophysical Journal, 856, 68
Ramachandran P., Zoph B., Le Q., 2018. https://arxiv.org/pdf/1710.05941.pdf
Ronneberger O., Fischer P., Brox T., 2015, in Navab N., Hornegger J., Wells W. M., Frangi A. F., eds, Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. Springer International Publishing, Cham, pp 234–241
Rosenblatt F., 1958, Psychological Review, 65, 386
Rozo E., et al., 2016, Monthly Notices of the Royal Astronomical Society, 461, 1431
Schaefer, C. Geiger, M. Kuntzer, T. Kneib, J.-P. 2018, A&A, 611, A2
Sevilla-Noarbe I., et al., 2021, ApJS, 254, 24
Szegedy C., et al., 2015, in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp 1–9, doi:10.1109/CVPR.2015.7298594
Szegedy C., Ioffe S., Vanhoucke V., Alemi A. A., 2017, in Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence. AAAI 17. AAAI Press, pp 4278–4284, doi:10.5555/3298023.3298188
Tan M., Le Q. V., 2019, in Chaudhuri K., Salakhutdinov R., eds, Proceedings of Machine Learning Research Vol. 97, Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA. PMLR, pp 6105–6114, http://proceedings.mlr.press/v97/tan19a.html
Tie S. S., et al., 2017, The Astronomical Journal, 153, 107
Treu T., 2010, ARA&A, 48, 87
Treu T., et al., 2018, MNRAS, 481, 1041
Vernardos G., 2019, MNRAS, 483, 5583
Wang G., 2016, IEEE Access, 4, 8914
Williams P., Agnello A., Treu T., 2017a, MNRAS, 466, 3088
Williams P., Agnello A., Treu T., 2017b, MNRAS, 466, 3088
de Vaucouleurs G., 1948, Annales d'Astrophysique, 11, 247

APPENDIX A: AFFILIATIONS

[1] Department of Electrical and Computer Engineering, University of California, Los Angeles, CA 90095, USA
[2] Department of Electrical and Computer Engineering, Nazarbayev University, Nur-Sultan, Kazakhstan
[3] The Inter-University Centre for Astronomy and Astrophysics (IUCAA), Post Bag 4, Ganeshkhind, Pune 411007, India
[4] Kavli IPMU (WPI), UTIAS, The University of Tokyo, Kashiwa, Chiba 277-8583, Japan
[5] Department of Statistics, University of California, Los Angeles, CA 90095, USA
[6] Department of Political Science, University of California, Los Angeles, CA 90095, USA
[7] Department of Physics and Astronomy, PAB, 430 Portola Plaza, Box 951547, Los Angeles, CA 90095-1547, USA
[8] Kavli Institute for Particle Astrophysics and Cosmology and Department of Physics, Stanford University, Stanford, CA 94305, USA
[9] Department of Astronomy & Astrophysics, University of Chicago, Chicago, IL 60637, USA
[10] School of Physics and Technology, Wuhan University, Wuhan 430072, China
[11] Laboratoire d'Astrophysique, Ecole Polytechnique Fédérale de Lausanne (EPFL), Observatoire de Sauverny, CH-1290 Versoix, Switzerland

[12] DARK, Niels Bohr Institute, Jagtvej 128, Copenhagen (DK)
[13] Fermi National Accelerator Laboratory, P. O. Box 500, Batavia, IL 60510, USA
[14] Kavli Institute for Cosmological Physics, University of Chicago, Chicago, IL 60637, USA
[15] Laboratório Interinstitucional de e-Astronomia - LIneA, Rua Gal. José Cristino 77, Rio de Janeiro, RJ - 20921-400, Brazil
[16] Instituto de Física Teórica, Universidade Estadual Paulista, São Paulo, Brazil
[17] Department of Physics & Astronomy, University College London, Gower Street, London, WC1E 6BT, UK
[18] Kavli Institute for Particle Astrophysics & Cosmology, P. O. Box 2450, Stanford University, Stanford, CA 94305, USA
[19] SLAC National Accelerator Laboratory, Menlo Park, CA 94025, USA
[20] Center for Astrophysical Surveys, National Center for Supercomputing Applications, 1205 West Clark St., Urbana, IL 61801, USA
[21] Department of Astronomy, University of Illinois at Urbana-Champaign, 1002 W. Green Street, Urbana, IL 61801, USA
[22] Institut de Física d'Altes Energies (IFAE), The Barcelona Institute of Science and Technology, Campus UAB, 08193 Bellaterra (Barcelona) Spain
[23] Center for Cosmology and Astro-Particle Physics, The Ohio State University, Columbus, OH 43210, USA
[24] Jodrell Bank Center for Astrophysics, School of Physics and Astronomy, University of Manchester, Oxford Road, Manchester, M13 9PL, UK
[25] University of Nottingham, School of Physics and Astronomy, Nottingham NG7 2RD, UK
[26] Astronomy Unit, Department of Physics, University of Trieste, via Tiepolo 11, I-34131 Trieste, Italy
[27] INAF-Osservatorio Astronomico di Trieste, via G. B. Tiepolo 11, I-34143 Trieste, Italy
[28] Institute for Fundamental Physics of the Universe, Via Beirut 2, 34014 Trieste, Italy
[29] Observatório Nacional, Rua Gal. José Cristino 77, Rio de Janeiro, RJ - 20921-400, Brazil
[30] Department of Physics, University of Michigan, Ann Arbor, MI 48109, USA
[31] Hamburger Sternwarte, Universität Hamburg, Gojenbergsweg 112, 21029 Hamburg, Germany
[32] Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT), Madrid, Spain
[33] Department of Physics, IIT Hyderabad, Kandi, Telangana 502285, India
[34] Faculty of Physics, Ludwig-Maximilians-Universität, Scheinerstr. 1, 81679 Munich, Germany
[35] Santa Cruz Institute for Particle Physics, Santa Cruz, CA 95064, USA
[36] Institute of Theoretical Astrophysics, University of Oslo. P.O. Box 1029 Blindern, NO-0315 Oslo, Norway
[37] Instituto de Fisica Teorica UAM/CSIC, Universidad Autonoma de Madrid, 28049 Madrid, Spain
[38] Department of Astronomy, University of Michigan, Ann Arbor, MI 48109, USA
[39] Universitäts-Sternwarte, Fakultät für Physik, Ludwig-Maximilians Universität München, Scheinerstr. 1, 81679 München, Germany
[40] School of Mathematics and Physics, University of Queensland, Brisbane, QLD 4072, Australia
[41] Department of Physics, The Ohio State University, Columbus, OH 43210, USA
[42] Center for Astrophysics | Harvard & Smithsonian, 60 Garden Street, Cambridge, MA 02138, USA
[43] Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, USA
[44] Australian Astronomical Optics, Macquarie University, North Ryde, NSW 2113, Australia
[45] Lowell Observatory, 1400 Mars Hill Rd, Flagstaff, AZ 86001, USA
[46] Departamento de Física Matemática, Instituto de Física, Universidade de São Paulo, CP 66318, São Paulo, SP, 05314-970, Brazil
[47] Department of Physics and Astronomy, University of Pennsylvania, Philadelphia, PA 19104, USA
[48] Institució Catalana de Recerca i Estudis Avançats, E-08010 Barcelona, Spain
[49] Physics Department, 2320 Chamberlin Hall, University of Wisconsin-Madison, 1150 University Avenue Madison, WI 53706-1390
[50] Institute of Astronomy, University of Cambridge, Madingley Road, Cambridge CB3 0HA, UK
[51] Department of Astrophysical Sciences, Princeton University, Peyton Hall, Princeton, NJ 08544, USA
[52] Institut d'Estudis Espacials de Catalunya (IEEC), 08034 Barcelona, Spain
[53] Institute of Space Sciences (ICE, CSIC), Campus UAB, Carrer de Can Magrans, s/n, 08193 Barcelona, Spain
[54] School of Physics and Astronomy, University of Southampton, Southampton, SO17 1BJ, UK
[55] Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831
[56] Department of Physics, Stanford University, 382 Via Pueblo Mall, Stanford, CA 94305, USA
[57] Max Planck Institute for Extraterrestrial Physics, Giessenbachstrasse, 85748 Garching, Germany