Draft version May 5, 2022
Typeset using LaTeX **twocolumn** style in AASTeX62

ASTR 502 Project - Finding Low Surface Brightness Galaxies with Convolutional Neural Networks

Hayden R. Foote[1]

[1] *Steward Observatory, The University of Arizona, 933 North Cherry Avenue, Tucson, AZ 85721, USA.*

1. INTRODUCTION

Low Surface-Brightness Galaxies (LSBGs) are galaxies with central surface brightnesses lower than the typical sky background, i.e. greater than 22 mag/arcsec^2 in the g-band. Thus, by nature, detecting and determining the properties of LSBGs is quite challenging. For example, LSBGs are easily confused with artifacts such as star-forming regions in larger galaxies, tidal ejecta, small bright sources blended with light from foreground stars, and other low surface-brightness features.

Nevertheless, LSBGs are extremely important systems to study, primarily because they inhabit the low end of the galaxy luminosity function. This makes them some of the most numerous galaxies in the universe, and their abundance and clustering properties can help test our theories of galaxy formation and evolution at small scales (e.g. Kaviraj 2020 and references therein).

Traditionally, selecting samples of LSBGs from survey data has required visual inspection to separate true LSBGs from artifacts (e.g. Greco et al. 2018, Tanoglidis et al. 2021b). However, the data volumes expected from next-gen surveys such as the Legacy Survey of Space and Time (LSST) at the Vera Rubin Observatory require the development of pipelines to automate this process.

In this project, I attempt to reproduce the results of Tanoglidis et al. (2021a), who present a convolutional neural network (CNN) called *DeepShadows* which is designed to separate LSBGs from artifacts. My work is heavily based on this paper, as well as on the code in the associated public repository.[1] My code for this project can be found at https://github.com/hfoote/ASTR502_project_Foote.

This paper is organized as follows: In § 2, I describe the datasets used to train and test the network. In § 3, I explain the *DeepShadows* architecture. In § 4, I describe the training and testing of the network on a large dataset from the Dark Energy Survey (DES). In § 5, I describe a transfer learning task designed to see how well the network generalizes to other surveys. Finally, I conclude in § 6.

Corresponding author: Hayden R. Foote
haydenfoote@email.arizona.edu

[1] https://github.com/dtanoglidis/DeepShadows

2. DATASETS

This work uses two datasets of survey images to train and test the *DeepShadows* CNN. The primary data comes from the DES, while a smaller set from the Hyper Suprime-Cam Subaru Strategic Program (HSC SSP) is used for transfer learning.

2.1. *DES Data*

The main dataset for this project comes from the DES archives. Tanoglidis et al. (2021b) describe the creation of this dataset in more detail, but I provide a short overview here: `SourceExtractor` (Bertin & Arnouts 1996) was used to identify sources in the DES footprint and tabulate each source's photometric properties. LSBG candidates were then selected based on their `SourceExtractor` - derived parameters, with emphasis on their mean surface brightnesses and half-light radii. Actual LSBGs were separated from artifacts within this sample of candidates through visual inspection. The total provided dataset consists of two "master" lists, containing the RA and DEC coordinates of 20,000 LSBGs and 20,000 artifacts.

2.2. *HSC Data*

A smaller, secondary dataset is compiled from the HSC SSP archives as outlined by Greco et al. (2018). This dataset includes the coordinates of 781 LSBGs and 641 artifacts, which were also identified with `SourceExtractor` and then visually classified. Importantly, this dataset comes from an independent survey, and includes different potential biases along the classification pipeline. This dataset is used to test how well the trained *DeepShadows* network generalizes to other surveys.

2.3. *Preprocessing*

The `fetch_data.ipynb` notebook in my project repository downloads the datasets and performs preprocessing. Images from both datasets are downloaded from the DESI Legacy Imaging Surveys Sky Viewer (Dey et al.

2019), using the lists of coordinates provided in the original *DeepShadows* repository. Each image is centered on the LSBG candidate, and is initially 255 × 255 pixels. To reduce storage and memory requirements as well as speed up training, the images are downsized to 64 × 64 during preprocessing. Color information is included in RGB channels, corresponding to the g, r, z bands of the DES or the g, r, i bands of HSC, such that the dimensions of each image are 64 × 64 × 3. Pixel values are re-scaled to lie in [0,1] to improve the network's behavior during training.

The DES dataset is randomly split into a training set of 30,000 images, a validation set of 5,000 images, and a test set of 5,000 images. The HSC images are randomly assigned into a 320-image training set and a 960-image test set. All of these subsets contain equal numbers of artifacts and LSBGs. While Tanoglidis et al. 2021a publish the coordinates for their exact test, validation, and training sets in addition to the master lists, I chose to randomly resample my own datasets from the master coordinate lists. All images are assigned a label of "0" if they contain an artifact, or "1" if they contain a LSBG.

3. NETWORK ARCHITECTURE

The notebook `network_tf.ipynb` in the project repository defines the *DeepShadows* network using `TensorFlow` (Abadi et al. 2015), and performs all training and testing. The network architecture is summarized in Figure 1; see also Table 1 of Tanoglidis et al. 2021a for a detailed breakdown of each layer. The feature extractor consists of three identical blocks, which in turn consist of a convolutional and pooling layer, among regularization steps. Each block begins with a convolutional layer using a 3 × 3 kernel. The number of kernels is designed to double the number of channels, with the exception of the first block, which uses 16 filters on the original 3-channel image. The convolutional layers add one pixel of padding such that the input and output resolutions are the same, use a 1 × 1 stride, and have a ReLU activation function. During training, weight decay in the form of L2 regularization is added to the convolutional layers, with $\lambda = 0.13$.

Following the convolutional layer, each block then performs batch normalization to improve the behavior of the optimization during training. After this, the dimensionality of each channel is reduced by a factor of two via a max pooling layer with a 2 × 2 kernel of 2 × 2 stride. To mitigate overfitting, each block concludes with a dropout layer which randomly suppresses 40% of the block's output. The end result of each block is to double the number of output channels while reducing the feature map resolution by a factor of two.

After three such blocks, the output is flattened, and passed to the classifier, which consists of two dense layers. The first layer drops the length of the feature vector by a factor of four from 4096 to 1028. This layer again uses ReLU activation and weight decay. In this dense layer, the regularization constant is reduced slightly to $\lambda = 0.12$. This layer is followed by an additional dense layer with one output. A sigmoid activation function is used such that the output is the probability in [0,1] that the image contains a LSBG.

4. TRAINING

The primary training for the network is performed on the DES training set, which contains 30,000 images. We reproduce the training methods of Tanoglidis et al. 2021a, namely a batch size of 64, a binary cross-entropy loss function, and the Adadelta optimizer with a base learning rate of 0.1. Additionally, I train for 100 epochs. Tanoglidis et al. 2021a state that the optimization does not improve significantly after this point, though my results (see Figure 2) suggest that the network performance does not improve significantly after ∼50 epochs.

I did not have access to machine with a GPU for training, so I instead trained the network on my personal laptop, which has a 2.3 GHz 4-core intel CPU and 16 GB of RAM. Fortunately, this machine was sufficient to train the network on the entire provided training set in a few hours.

The primary metric I use to assess the network's performance is the accuracy, or simply the number of correct predictions divided by the total number of examples. Since the output is a probability, we need to adopt a threshold probability above which we say the predicted label is 1 (the image is classified as a LSBG) and below which the predicted label is 0 (the image is an artifact). Like Tanoglidis et al. 2021a, I adopt a value of 0.5 for this threshold probability.

Figure 2 shows the progress of the optimization. The network reaches 90% accuracy in less than ten epochs, while the rest of the training only improves the accuracy by a few percent. This suggests that if one is pressed for time or computational resources, they could train *DeepShadows* for significantly fewer epochs with only a small reduction in performance. During training, the optimization progress is assessed on the DES validation set. Importantly, the validation accuracy closely matches the training accuracy throughout the optimization, indicating the network is not overfit.

After training, I use *DeepShadows* to predict the labels of the DES test set of 5,000 images. Figure 3 shows the confusion matrix from this testing step. The overall test accuracy is 91.17%, within the 95% confidence

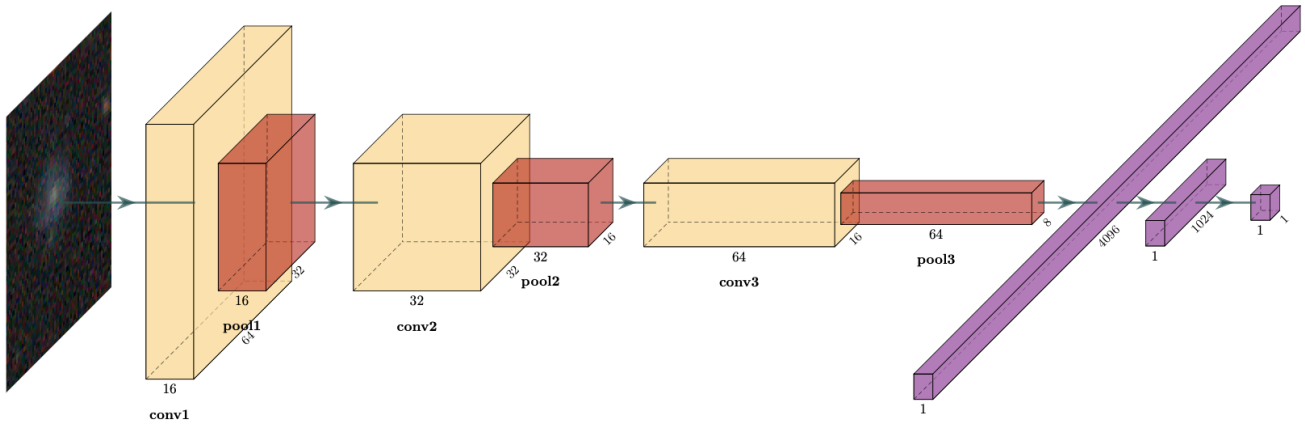

Figure 1. The *DeepShadows* CNN architecture. The input is a 64 × 64 × 3 image. The feature extractor consists of three blocks containing a convolutional layer (yellow) and a max pooling layer (red). Each block increases the number of filters while reducing the feature map resolution. The output of the third block is passed to the classifier (purple), which consists of a flattening layer followed by two dense layers. The output is a single probability that the image contains a LSBG. Reproduced from Figure 1 of Tanoglidis et al. 2021a.

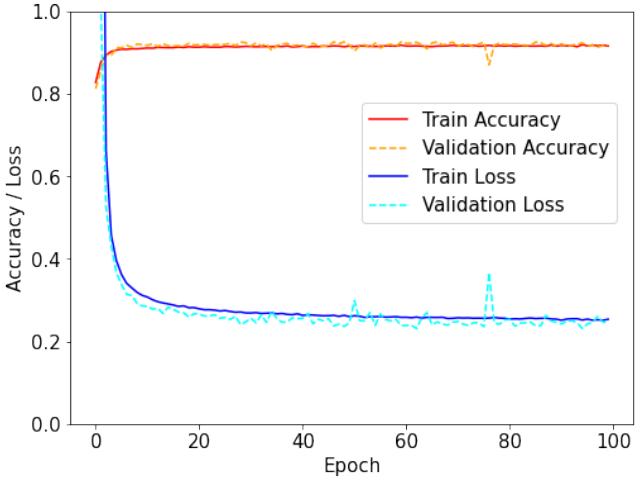

Figure 2. Accuracy and loss as evaluated on the training and validation sets, during training on the DES data. The accuracy is shown in warm colors, while the loss is shown in cool colors. Training set values are shown by solid lines, while validation set values are shown by dashed lines. Network performance improves quickly, reaching close to its final values after 20 epochs, then improving slowly over the remaining training. Training and validation accuracy remain very close throughout training, indicating that the network is not overfit.

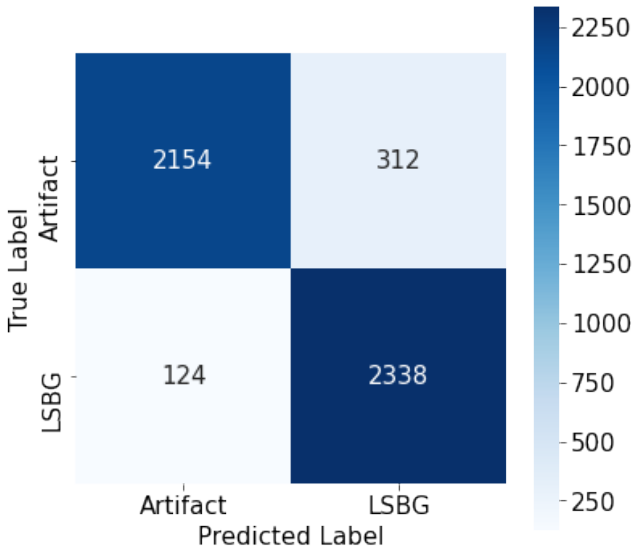

Figure 3. Confusion matrix for the DES test set after training. The overall network accuracy is 91.17%.

interval given in Tanoglidis et al. 2021a. There are more false positives than false negatives. Clearly, when trained on a large, un-augmented, observational dataset, *DeepShadows* is very effective at separating LSBGs from artifacts in data taken from the same survey on which it was trained.

5. TRANSFER LEARNING

The second portion of this project is focused on assessing the extent to which *DeepShadows* can be used on datasets from other surveys with minimal to no retraining. Observational training data is expensive, often requiring many hours of manual classification to apply labels. The data volume of next-gen surveys makes visual classification of images prohibitive, instead requiring that data pipelines be automated. Having access to an effective network that does not require an additional

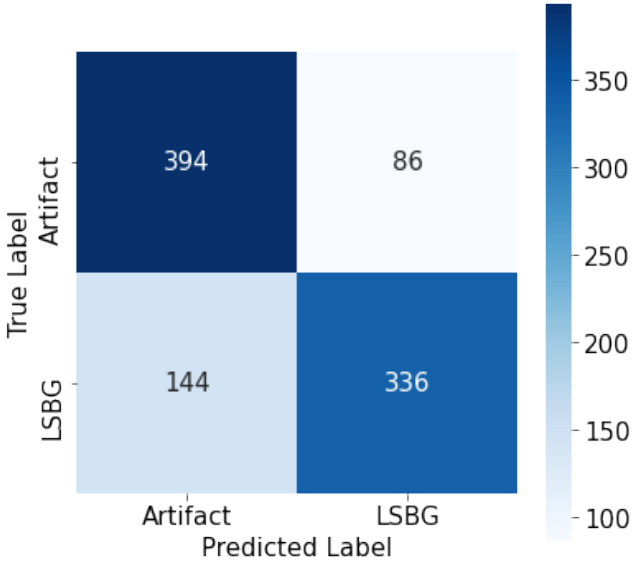

Figure 4. Confusion matrix for the HSC test set BEFORE transfer learning. The overall network accuracy is 76.04%.

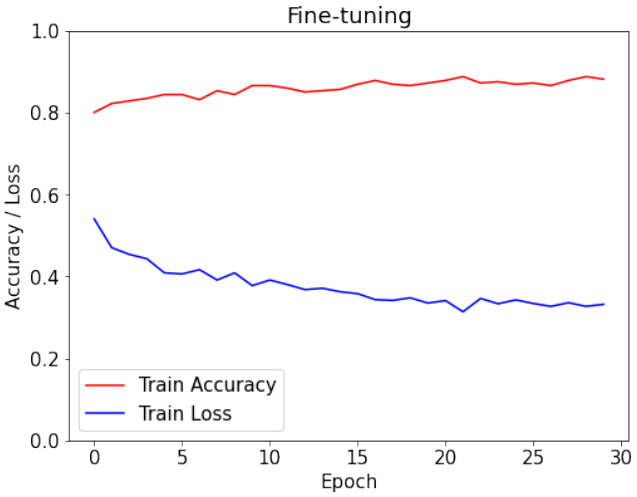

Figure 5. Accuracy and loss as evaluated on the HSC training set during fine-tuning (transfer learning). The accuracy is shown in red, while the loss is shown in blue. Note there is no validation set for the HSC data. Network performance improves throughout training.

expensive training set will be especially useful for object classification in these surveys.

To this end, I first test the trained *DeepShadows* network on the second test dataset from HSC with no further modifications. The confusion matrix from this test is shown in Figure 4. The overall accuracy is 76.04%, much worse than the DES dataset, but still much better than randomly guessing. Additionally, there are now more false negatives than false positives.

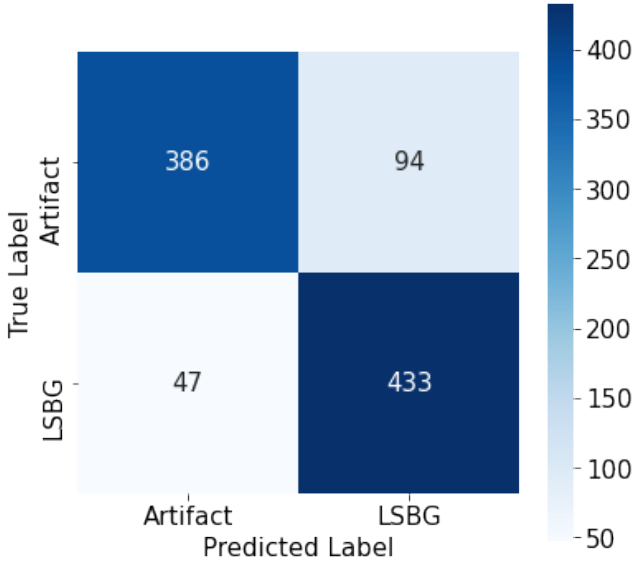

Figure 6. Confusion matrix for the HSC test set AFTER transfer learning. The overall network accuracy is 85.31%.

Following this, I perform transfer learning, i.e. using the HSC training set to fine-tune the network parameters for an additional 30 epochs. In this fine tuning step, I again follow Tanoglidis et al. 2021a, dropping the batch size to 32 and Adadelta's base learning rate to 0.005. Figure 5 shows the progress of the fine-tuning. This time, progress is more steady and the accuracy is still showing gradual improvement when I stop training, suggesting that continuing for more than 30 epochs may continue to improve performance slightly. Fine-tuning on the small, 320-image training set is much less expensive, with 30 epochs taking less than a minute on my machine.

After transfer learning, the network is again tested with the HSC test set. The accuracy improves markedly to 85.31%, but this is now slightly below the 95% confidence interval in Tanoglidis et al. 2021a. Their confidence intervals are derived from bootstrap resampling of the three datasets from the master list of objects, and as I resampled my datasets independently, this could be the source of the discrepancy. The authors do not provide 3-σ intervals, though the 2-σ interval is 0.856 - 0.896, and my result is very close to this range and likely lies within 3-σ. Additionally, it should be noted that my accuracy for the DES data was on the lower end of the confidence interval, so it is possible that this lower-than-average performance was propagated through transfer learning.

6. CONCLUSIONS

Overall, I was able to reproduce the *DeepShadows* network as well as its training and transfer learning on the same datasets that were used to train the origi-

nal network. While my reproduction achieved slightly lower performance than the original network, my results demonstrate the promise of using CNNs to classify objects in nex-gen surveys. Importantly, these results highlight that while training data is expensive, the effort of obtaining a large training set once is well-justified. Once this is done, the network performs very well on data from the same survey, and can be generalized easily to other surveys through inexpensive transfer learning, with only a small drop in performance.

Lastly, I note that Tanoglidis et al. 2021a also include an extensive analysis of two classical machine learning algorithms for the same task. These algorithms, a support vector classifier and a random-forest classifier, use the `SourceExtractor` - derived parameters as inputs rather than the images themselves. *DeepShadows* outperforms both of these classical algorithms, while also being unreliant on `SourceExtractor` for feature extraction. Overall, CNN's show much promise for automating LSBG classification in survey data.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., et al. 2015, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, , , software available from tensorflow.org. https://www.tensorflow.org/

Bertin, E., & Arnouts, S. 1996, A&AS, 117, 393

Dey, A., Schlegel, D. J., Lang, D., et al. 2019, AJ, 157, 168

Greco, J. P., Greene, J. E., Strauss, M. A., et al. 2018, ApJ, 857, 104

Kaviraj, S. 2020, arXiv e-prints, arXiv:2001.01728

Tanoglidis, D., Ćiprijanović, A., & Drlica-Wagner, A. 2021a, Astronomy and Computing, 35, 100469

Tanoglidis, D., Drlica-Wagner, A., Wei, K., et al. 2021b, ApJS, 252, 18