Galaxy Identification with Convolutional Neural Networks

Ramiz Sohail
University of Michigan
rsohail@umich.edu

Nihal Kurki
University of Michigan
nkurki@umich.edu

Aditya Gandhi
University of Michigan
adgandhi@umich.edu

Vincent Weng
University of Michigan
vweng@umich.edu

Abstract

Our project utilizes CNNs to identify galaxy images. We preprocess a diverse dataset of galaxy images and train a CNN model using high-performance computing. We conduct analysis of our model using visualization techniques. This has the potential to aid astronomers in efficiently identifying and studying galaxies, contributing to our understanding of the universe's structure.

1. Introduction

The study of the universe has been an incredibly fascinating subject for centuries. However, to truly understand it, we need and should categorize and classify different types of galaxies accurately. The task of classifying galaxies is an essential problem for astronomers as it helps them to map out the universe and understand the past, present, and future.

Utilizing computer vision to classify galaxies has become more accessible due to the advancements in deep learning algorithms and the availability of massive datasets such as the Galaxy Zoo dataset [1]. By training a deep learning model on the Galaxy Zoo dataset, we can classify galaxies into various types based on their morphology.

By classifying different types of galaxies, we can learn more about their evolution and understand the physical processes that drive their formation. This, in turn, can help us to develop a better understanding of the universe and our place in it.

Previous research has been conducted to classify galaxies using different approaches. Some studies have used manual classification, while others have used machine learning algorithms. However, the use of deep learning models has shown promise in achieving high accuracy in classifying galaxies based on their make-up.

In our project, we will be using a deep learning model, specifically a convolutional neural network, to classify different types of galaxies based on their morphology. By training the model on the large Galaxy Zoo dataset, we will be able to classify galaxies with a solid degree of accuracy, providing valuable insights into the universe's structure.

There are two relevant studies that have been proposed for galaxy classification. The two related works are shown below:

"Deep Galaxy Image Classification using Convolutional Neural Networks", which proposes a deep learning-based approach that uses convolutional neural networks to classify galaxy images. The CNN model is trained on a large dataset of galaxy images, and it can accurately classify galaxies into different categories based on their visual features such as shape, color, and texture [3].

"Galaxy Morphology Classification using Transfer Learning with Pre-trained Convolutional Neural Networks" presents an approach for galaxy morphology classification using transfer learning with pre-trained convolutional neural networks. The CNN model learns to extract relevant features from the galaxy images, allowing accurate classification into categories such as spiral, elliptical, or irregular based on their morphological characteristics [4].

Our project is essentially built upon these inspirations of existing work and it contributes to the field by leveraging computer vision for galaxy classifications.

Our method involves training a CNN model on a large dataset of galaxy images, allowing the model to learn the features and patterns that distinguish different classes of galaxies. The trained CNN model is then used for predicting the class labels of new galaxy images, making our approach accurate, efficient, and scalable for large-scale galaxy classification tasks. By combining the power of computer vision and deep learning, our model aims to provide a robust and effective solution for identifying galaxies into different classes, contributing to the advancement of our understanding of the universe and its galaxies.

2. Approach

It is important to note that we took inspiration from EECS 442 Homework 4's general data pipeline structure and added a few additional features on top of it [6]. In addition, we also took inspiration from Shibin Paul's Galaxy Morphology Predictor on Kaggle to initially construct our CNN architecture [5].

2.1. Preprocessing

Prior to training the CNN, we preprocessed the galaxy images by cropping them to a consistent resolution (256x256) to reduce computational complexity and

standardize the image sizes. We also converted the images to grayscale, which simplified the input data and reduced the number of channels in the CNN input. The decision to use grayscale images was based on the fact that we were primarily interested in extracting features and curvature information from the galaxy images, rather than relying on color information. Grayscaling helped to focus on these key features while reducing computational time significantly. Image examples are shown below.

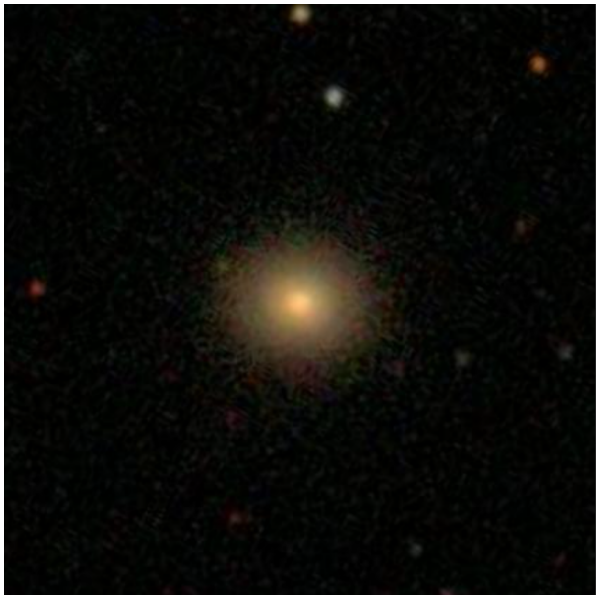

Figure 1: Example image given by GalaxyZoo [1].

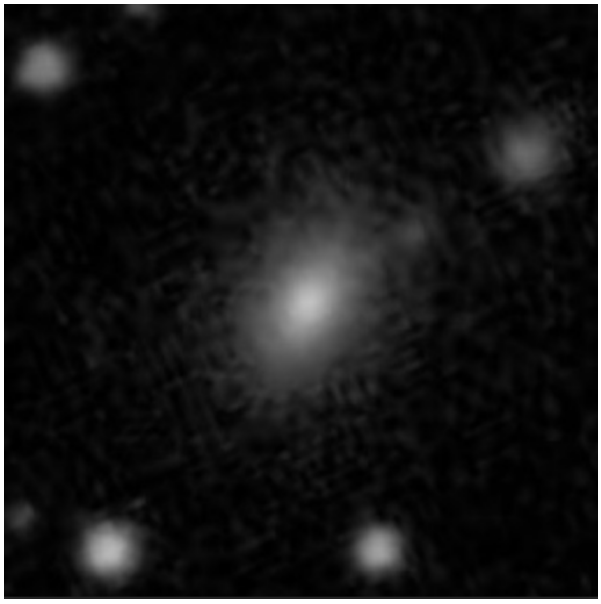

Figure 2: Example of a preprocessed image [1].

2.2. Data Splitting

To evaluate the performance of the CNN model, we split the dataset into training, validation, and testing sets. The training set was used for model training, the validation set was used for hyperparameter tuning and model selection, and the testing set was used for final model evaluation. The dataset was randomly partitioned into these sets to ensure that the distribution of different galaxy classes was maintained in each set, avoiding bias in model evaluation. The split between training and validation sets was controlled by three variables, indicating the proportion of data to be used for training, validation, and testing. This allowed us to control the size of the different subsets based on the specific needs of our model and dataset. This tuning was done by analyzing the validation and testing results.

2.3. CNN

The input to the CNN is expected to be grayscale images with a shape of (N, 1, H, W), where N is the batch size, 1 is the number of input channels (grayscale images have only one channel), and H and W are the height and width of the input images, respectively.

The CNN has four convolutional layers defined as self.conv1, self.conv2, self.conv3, and self.conv4. These layers perform convolutional operations on the input images with different numbers of input and output channels (also known as feature maps) and kernel sizes. The activation function used after each convolutional layer is ReLU (F.relu()).

The CNN has two max pooling layers defined as self.pool1 and self.pool2. These layers perform max pooling operations on the feature maps to reduce their spatial dimensions.

The CNN has two dropout layers defined as self.drop1 and self.drop2. These layers perform dropout regularization during training, randomly setting a fraction of input units to 0 at each update to prevent overfitting.

The CNN has one fully connected layer defined as self.fc1. This layer takes the flattened output from the previous convolutional layers and produces the final output logits for classification. The output size of this layer depends on the number of classes in the classification task (in this case, it is set to 37).

The forward() function implements the forward propagation of the CNN. It takes the input tensor x and passes it through the defined convolutional, pooling, dropout, and fully connected layers in a sequential manner, applying activation functions as needed. The final output logits z are returned as a torch.Tensor object.

2.4. Training

One important thing to note here is that we utilized the training step function from EECS 442 Homework 4 as the basic framework and edited some of the code [6]. During training, we monitored the model's performance on the validation set, adjusting hyperparameters and model architecture as needed to prevent overfitting and underfitting. Once the model was trained, we evaluated its performance on the testing set to assess its accuracy.

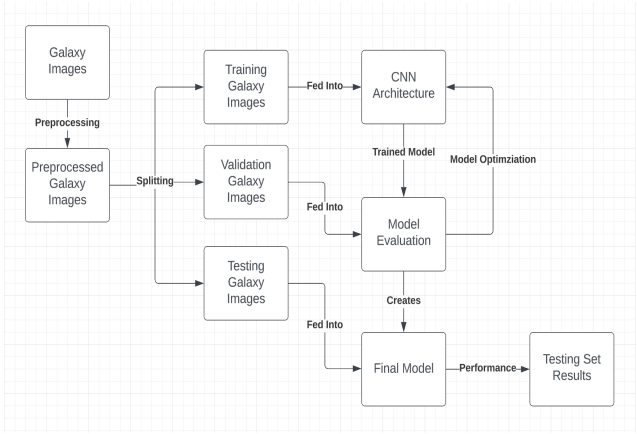

Figure 1: This figure shows the general data pipeline of taking images in, training them through the CNN, and finalizing the final model after model validation. This was made with LucidCharts [2].

3. Experiments

For the validation of the project, it is best to start with what the model is attempting to describe. The model, given a labeled preprocessed image of an item in space, is tasked with providing answers to the following decision tree questions as a (,37) sized vector (37 for each of the 37 questions).

Although the model doesn't know what the questions are asking, it is able to predict why some images provide higher answers to some questions than others. So the input data of a labeled image makes sense as the model is determining what the probability for each of the questions is. And a further goal may be to take these probabilities in the 37 length vector and try to categorize the image based on the decision tree in Figure 4 [1].

Some metrics for us to recognize what success is can be by looking at the squared loss of the prediction and the actual label. The loss we have defined can change, but currently a correct categorization of an image is when 35 or more of the 37 probabilities are within 0.1 of the correct answer. This measure is reasonable because when attempting to classify a galaxy, getting more than 35 of the decision tree questions correct means that the classification is very close to what the real answer is, if not fully correct.

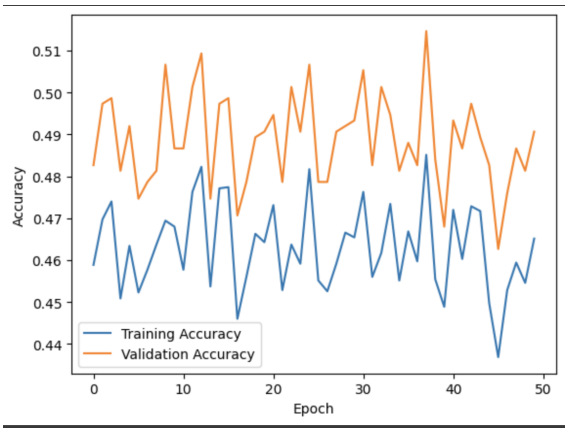

Figure 3: Provides the training and validation accuracy of the CNN model after 50 epochs.

The figure above represents our current qualitative results. Although the accuracy is not very high, this is to be expected with a CNN that cannot train on global trends of an image, rather just the local areas that it sees in every grayscale and cropped image.

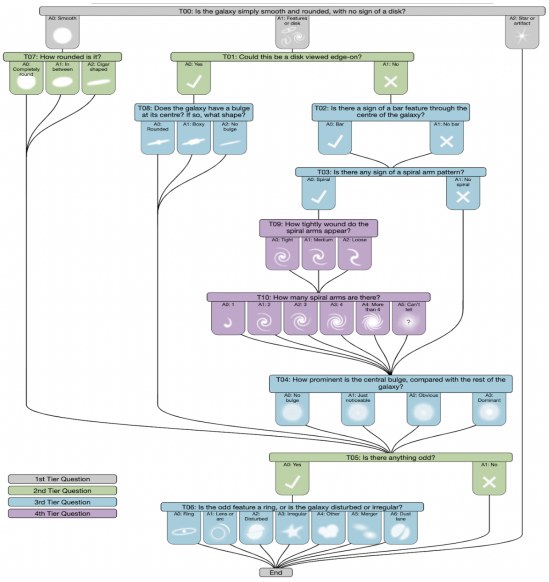

Figure 4: This figure displays the 37 different questions in the galaxy zoo 2 decision tree dataset [1].

And finally, if we were to choose a baseline to compare the model against, such as random chance, we see that the accuracy holds up very well. If we randomly selected a number from 0-1 for each of the 37 questions, we would expect only about 4 to be within 0.1 of the correct answer. That means it would be very difficult for a randomly selected model to get at least 35 within 0.1, the chances are close to 0 of that happening. Therefore, with our model's accuracy of around 45%, we can expect it to

definitely be performing better than random chance.

Finally, after fine-tuning our final model, we tested the model on the testing set and obtained the accuracy performance shown in the graphic below.

```
 Evaluate on test set
100%|██████████| 75/75 [10:13<00:00,  8.18s/it]
 Evaluation accuracy: 0.45866666666666667
CPU times: user 15min 1s, sys: 58.8 s, total: 16min
Wall time: 1h 29min 14s

0.45866666666666667
```

Figure 5: This figure displays the evaluation accuracy on the testing set.

It is important to know that this result is much higher than any random baseline would get, because the chances of randomly picking between 0 and 1 and being within 0.1, for at least 35 out of 37, is near 0.

4. Implementations

Because there have been competitions to classify images in the Galaxy Zoo dataset, we referenced some competition participants to build our model. Specifically, our model was inspired from Shibin Paul's network [5]. His network included a cycle of 2 convolutional layers, an activation function, and a max pooling layer. He then added a cycle of a dropout layer, dense layer, and activation function. After looking at Paul's architecture, we created a model that cycled between a convolutional layer, a ReLU activation function, and then a max pooling layer. We used 4 convolutional layers, and after the third, we started adding dropout layers. We finalized the network with a fully connected layer. All our networks and CNN functions were borrowed from the Pytorch library. After, we began tuning our hyperparameters such as the learning rate, weight decay, loss function and the optimizer.

```
Your network:
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 254, 254]             640
         MaxPool2d-2           [-1, 64, 63, 63]               0
            Conv2d-3          [-1, 256, 59, 59]         409,856
            Conv2d-4           [-1, 16, 57, 57]          36,880
         MaxPool2d-5           [-1, 16, 28, 28]               0
           Dropout-6           [-1, 16, 28, 28]               0
            Conv2d-7            [-1, 8, 26, 26]           1,160
           Dropout-8            [-1, 8, 26, 26]               0
            Linear-9                   [-1, 37]         200,133
================================================================
Total params: 648,669
Trainable params: 648,669
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.25
Forward/backward pass size (MB): 40.91
Params size (MB): 2.47
Estimated Total Size (MB): 43.63
----------------------------------------------------------------
None
```

Figure 6: PyTorch summary of our CNN's model architecture.

Our weight decay was set to $1 * 10^{-5}$ and our learning rate was set to $1 * 10^{-1}$. We used Adam Optimization and Cross Entropy Loss. We specifically used Cross Entropy Loss because it takes the model's prediction and the actual solutions to find their distance and minimize them. Unfortunately, even with tuning, our model still struggled to learn. This is most likely due to the fact that we used only 5000 images as our maximum. For a feature vector of 37 different outputs, 5000 images is not enough to train a model to learn the relationship of the images to all the features listed. However, due to time constraints, we could not train on more than 5000 images. Training on all the images provided would have taken at least a few hours on GPU settings.

[1] 5. References

[2] "Galaxy Zoo Data," *Galaxyzoo.org*. [Online]. Available: https://data.galaxyzoo.org/. [Accessed: 22-Apr-2023].

[3] "Intelligent diagramming," *Lucidchart*. [Online]. Available: https://www.lucidchart.com/pages/. [Accessed: 22-Apr-2023].

[4] Shi, H. "Galaxy Classification with Deep Convolutional Neural Networks." Master's thesis, University of Illinois at Urbana-Champaign, 2016. (34 pages)

[5] A. Ghosh, C.M. Urry, Z. Wang, K. Schawinski, D. Turp, and M.C. Powell, "Galaxy Morphology Classification using Transfer Learning with Pre-trained Convolutional Neural Networks," in Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 112-121. DOI: 10.3847/1538-4357/ab8a47

[6] S. Paul, "Galaxy Morphology Predictor." Galaxy Zoo - The Galaxy Challenge, 2023. [Online]. Available: https://www.kaggle.com/code/shibinjudah/galaxy-morphology-predictor

[7] D. Fouhey, "EECS 442 Computer Vision: Homework 4,", 2023, pp. 1-13. [Online]. Available :https://web.eecs.umich.edu/~fouhey/teaching/EECS442_W23/resources/hw4.pdf