Manuscript Title: Subtitle

Evelyn Kimbirk,* Travis Beebe,† Noah Hood,‡ Zice Zhao,§ and Michael Hench¶
University of California San Diego
(Group 1)
(Dated: March 24, 2023)

In this paper, we propose a deep learning model for image classification using a custom-designed VGG based convolutional neural network (CNN) architecture. We use a dataset consisting of images belonging to 37 different classes, and train our model using a combination of the binary cross-entropy loss and the Adam optimizer with a learning rate of 0.001. The model is trained on a dataset consisting of 61,578 training images, and 79,975 test images of galaxies. To evaluate the performance of our model, we compare it to a simple baseline method using accuracy and mean squared error (MSE) metrics. Our results show that the VGG6 model performs with a MSE of 0.0131 and accuracy of 0.7228 on the test set. Overall, our proposed VGG6 model provides a promising approach for image classification tasks, achieving high accuracy on a complex dataset with multiple classes. Future work can explore future optimization of the model architecture and hyperparameters to improve its performance even further.

I. INTRODUCTION

Recent advances in astronomical instrumentation and space exploration have provided us with a wealth of data about the universe, but also underscored the challenges of dealing with the sheer amount and complexity of that data. With the upcoming launch of telescopes such as the James Webb Space Telescope and the Nancy Grace Roman Space Telescope, we expect to generate unprecedented amounts of data about galaxies, raising the need for new tools and techniques to analyze and interpret that data.

One promising approach is to use machine learning algorithms to classify galaxies based on their images. This is not a new idea, as demonstrated by the Kaggle Galaxy Zoo challenge held in 2014, which asked participants to sort photos of galaxies into 37 different categories using machine learning algorithms. The challenge called for using Boosted Decision Trees to sort the photos and used a Mean Squared Error Loss between the predicted image category and its true value as the placement method for teams that took this challenge.

In this paper, we use the same dataset as the Kaggle challenge, but tackle the problem in a different way, exploring the effectiveness of VGGs based on a recent advance in machine learning technique known as VGG-16 [1]. Our goal is to demonstrate the potential of VGG algorithms in handling the vast and complex data generated by modern astronomical instruments as well as to provide insights into the strengths and limitations of neural networks in classifying galaxies.

* lkimbirk@ucsd.edu
† tlbeebe@ucsd.edu
‡ nhood@ucsd.edu
§ ziz084@ucsd.edu
¶ mhench@ucsd.edu

II. DATASET

The dataset used in this paper is the Galaxy Zoo Challenge dataset [2], which contains 61,578 training images and 79,975 test images of galaxies. Each image is a color image of size 424x424 pixels and were classified by volunteers according to the flowchart in Fig. 1. The dataset also includes a training solution file with class probabilities for each galaxy image.

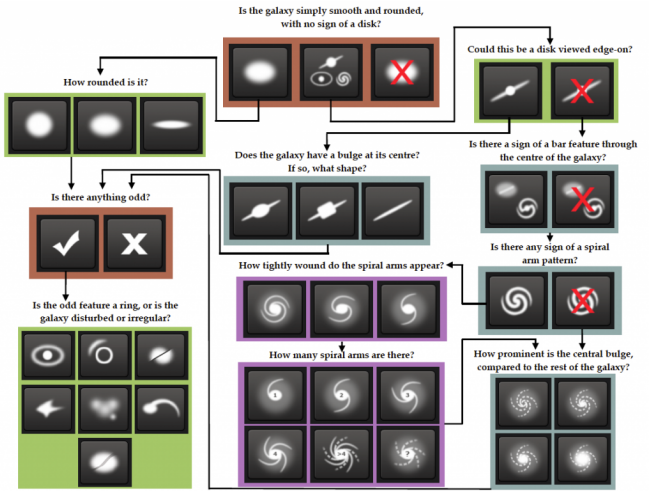

FIG. 1. The flowchart used by volunteers to classify galaxy images for the Galazy Zoo Dataset

To prepare the data for model training, the training images were split into training, validation, and test sets using an 80:20 split. The class probabilities were converted into one-hot encoded labels using the to_categorical () function from the Keras utility library [3].

To preprocess the image data, each image was cropped to the central 128x128 pixel region using the OpenCV library [4]. The galaxy depicted in the dataset are mostly contained within the central 128x128 pixel region, so the

images were cropped to reduce the amount of resources needed to process the images. The pixel values of the images were then normalized to be between 0 and 1 by dividing each pixel value by 255. Finally, the training and validation data were reshaped into a 4D tensor of shape (n_samples, 128, 128. 3), where n_samples is the number of images.

In summary, the dataset was preprocessed by cropping the images to the central region, normalizing the pixel values, and reshaping the data for use in a convolutional neural network.

III. METHODS

The objective of this project was to build a machine learning model to classify images into one of 37 categories. To achieve this, we used the VGG6 architecture, which is a variation of the VGG model with 6 layers. The VGG architecture is known for its simplicity, depth, and strong performance in image classification tasks.

Our VGG-6 model, as seen in Fig. 2, consisted of six convolutional layers with Rectified Linear Unit (ReLU) activation functions, followed by two fully connected layers. The first two convolutional layers had 16 filters each, while the next two had 32 filters each. The fifth layer had 256 neurons, and the output layer had 37 neurons, one for each class. To reduce overfitting, we added dropout layers after the first and fourth convolutional layers and after the fully connected layer. Additionally, we included batch normalization after the fourth convolutional layer.

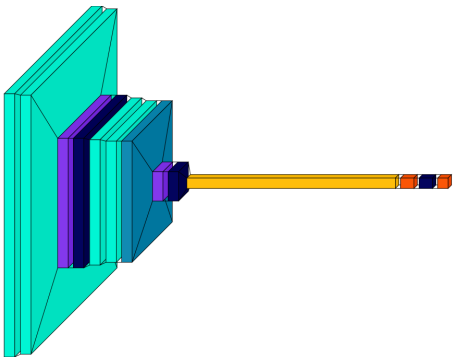

FIG. 2. Model architecture of VGG-6

FIG. 3. Model Legend

The hyperparameters were chosen based on experimentation. The learning rate of the Adam optimizer was left as the default 0.001, which is a popular algorithm for stochastic optimization of gradient descent. The loss function used was binary cross-entropy, which is commonly used for multi-label classification tasks. We also included two additional metrics: mean squared error, for direct comparison against submissions to the Kaggle Galaxy Zoo Challenge, and accuracy, for a general understanding of the performance of the model.

The dataset was stored in a pandas dataframe which was then processed according to the procedure described in the dataset section. The model was then trained for 50 epochs with a batch size of 256.

All of these described methods and more can be found on our project github [5].

IV. RESULTS

In this study, we evaluated the performance of our proposed model using three commonly used metrics: root mean square error (RMSE), accuracy, and area under the receiver operating characteristic (ROC) curve (AUC). RMSE and accuracy were calculated using the model outputs, while the AUC was determined using a test dataset from our sample.

The effectiveness of our model was compared against submissions to the Galaxy Zoo competition, where the top-performing submission achieved a RMSE of 0.07491 on 75% of the provided test data. In contrast, our model achieved an RMSE of 0.11446 on our training data, which would place our performance at the 94th position on the competition leaderboard.

Following training, our model achieved an accuracy of 72.28% on the training data. These results provide valuable insights into the predictive capacity of our proposed model and suggest potential avenues for further model refinement and optimization.

To generate the AUC for all 37 labels, we transformed each label in the test data to a binary value of 1 if the predicted probability was greater than 0.5 and 0 otherwise. Notably, the ROC curve allows for the determination of the model's ability to discriminate between positive and negative classes at varying classification thresholds. Specifically, the ROC curve evaluates the true positive rate (TPR) against the false positive rate (FPR) at different thresholds, providing a graphical depiction of the model's performance across all possible thresholds. As such, each ROC curve provides insights into the model's capacity to effectively classify galaxies into their assigned categories based on the majority of classifications made by human labelers. The code that was used to generate the AUC curves can be found at Lst. 1

Listing 1. AUC Curve Generation

```
from sklearn.metrics import
    roc_curve, auc, RocCurveDisplay

# test labels
for i in range(0, class_test.shape
    [0]):
     for j in range(0, class_test.
         shape[1]):
```

```
6                 if float(class_test[i, j]) >
                      0.5:
7                     class_test[i, j] = 1;
8                 else:
9                     class_test[i, j] = 0
10
11        np.asarray(class_test, dtype = int)
12
13        # plot roc curve for test
              predictions w/ auc
14        for i in range(0, class_test.shape
              [1]):
15            fpr, tpr, thresholds = roc_curve
                  (class_test[:,i], test_pred
                  [:,i], pos_label='1')
16            roc_auc = auc(fpr, tpr)
17            display = RocCurveDisplay(fpr =
                  fpr,tpr = tpr, roc_auc =
                  roc_auc, estimator_name = '
                  test_data')
18            display.plot()
19            plt.savefig('paper/figs/
                  auc_plots/plot_' + str(i) + '
                  .png')
```

A table of the AUC for each label is shown at table I. An AUC of over 95% was achieved in 10 of the labels. 31 of the categories have an AUC of at least 90%. Of the remaining 6 categories, 5 of them have an AUC between 78% and 85%. Finally we have one outlier, label 19, which produced an AUC of 42%. The AUC curve generated is shown at Fig 4, and it is immediately obvious that there was an issue with identifying label 19. The best AUC curves at 99% are shown in Fig 5, Fig 6, Fig 7.

Label	AUC	Label	AUC	Label	AUC	Label	AUC
0	0.94	1	0.94	2	0.94	3	0.99
4	0.94	5	0.93	6	0.90	7	0.94
8	0.80	9	0.97	10	0.94	11	0.93
12	0.78	13	0.90	14	0.90	15	0.96
16	0.92	17	0.95	18	0.95	19	0.42
20	0.82	21	0.96	22	0.96	23	0.93
24	0.94	25	0.97	26	0.99	27	0.99
28	0.93	29	0.94	30	0.91	31	0.85
32	0.93	33	0.95	34	0.98	35	0.98
36	0.85						

TABLE I. All AUCs

The performance of the model shows that it is significantly better than random guessing. Furthermore, the model has shown remarkable results in predicting specific categories that a galaxy image can be classified into. Nevertheless, the model's overall performance still falls short of optimal expectations, indicating there are still areas that need improvement. Additionally, there exists better models that provide superior performance outcomes in the field, which can be used as a benchmark for further development and improvement of this model.

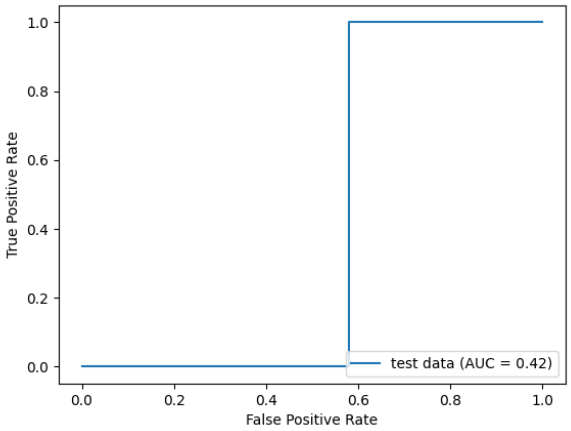

FIG. 4. AUC of Label 19

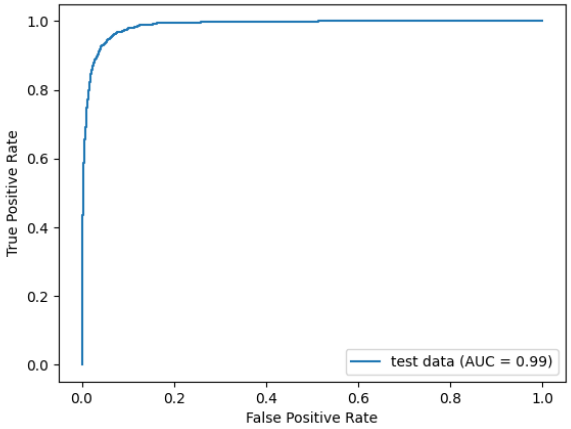

FIG. 5. AUC of Label 3

V. CONCLUSION

In this study, we undertook the task of classifying a large dataset of 79,975 images of galaxies, with a training set comprising 61,578 images across 37 different classes. The VGG6 algorithm was selected for its balance of performance and computational efficiency, and we obtained a classification accuracy of approximately 72% as well as a Mean Squared Error of 0.11446. The accuracy rating is exceptional given the technical limitations we faced during the course of our research. Specifically, we encountered significant instability on DataHub, with frequent kernel crashes, server timeouts, and difficulty in loading code. These limitations significantly impeded our ability to develop and test our models effectively.

Furthermore, the hardware available to use posed a challenge, with the large RAM and GPU requirements of the dataset resulting in heavier preprocessing of data which possibly led to a loss of accuracy as well as a lack

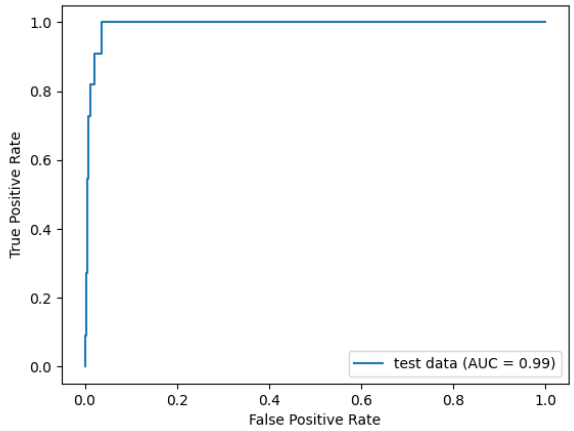

FIG. 6. AUC of Label 26

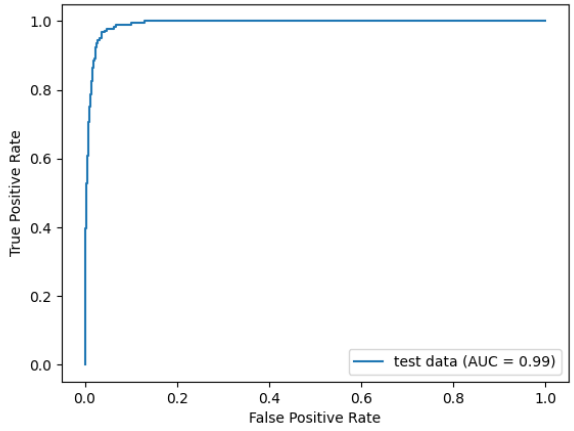

FIG. 7. AUC of Label 27

of VRAM on our RTC 3080 Ti GPU. As such, we were limited to using simpler algorithms such as VGG6, with larger and more complex models being impractical to train given these hardware limitations.

In order to optimize our model for accuracy while minimizing computational complexity, we developed a custom model that was designed to achieve a good balance between performance and computational intensity. Due to resource limitations, we were unable to use dedicated AI computers with multiple GPUs, and thus had to make our model as efficient as possible.

To accomplish this, we employed model compression techniques to reduce the number of classes we needed to consider during training. We discovered that only a small subset of the available classes were present in the majority of the galaxies in our dataset, allowing us to focus our efforts on optimizing the accuracy of our model for those specific classes.

[1] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition (2015), arXiv:1409.1556 [cs.CV].
[2] Galaxy zoo - the galaxy challenge (2014).
[3] Tf.keras.utils tensorflow v2.12.0 (2023).
[4] Opencv library (2023).
[5] Group1 phys139-239 machine learning final project (2023).