The Galaxy Image Classifier

Enmanuel Hernandez

May 3, 2024

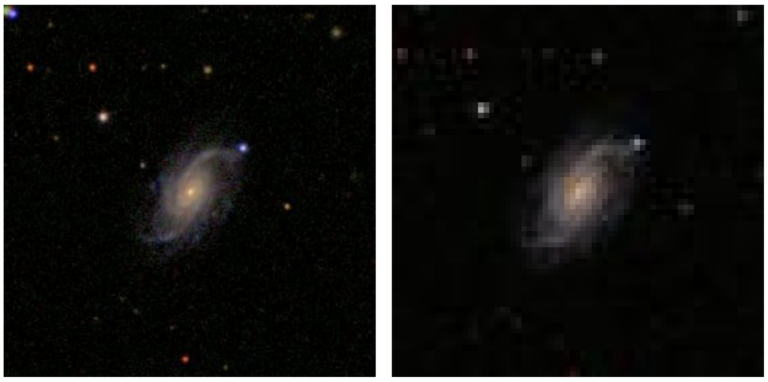


Figure 1: This galaxy image above undergoes a 30 percent reduction in pixelation as part of preprocessing to fit the input requirements of the neural network for the galaxy classification model

1 Introduction

In this study our aim is to categorize galaxies based on their shapes a task typically carried out by astronomers, with care. Galaxies come in forms and structures. Our objective is to create a model that can analyze these traits revealing the history of a galaxys interactions and its present condition. Using a collection of images classified by humans from the Galaxy Zoo project, our model (developed using machine learning neural networks powered by pytorch) will emulate the discerning eye of an astronomer as it sifts through images from the Sloan Digital Sky Survey (SDSS) to identify and classify these objects. Our mission here has two objectives; we aim to automate the classification of galactic shapes but also to search for indications of galactic mergers, which are events in the cosmic cycle where galaxies expand and transform through their gravitational interactions. Mergers transform galaxies, trigger bursts of star formation and fuel the growth of black holes at their centers. By estimating the rate of galaxy mergers based on our classifiers results and comparing it with predictions as well as empirical rates proposed by Lotz et al. we will assess the effectiveness of this model and contribute to ongoing discussions in astronomy.

2 Methods

2.1 Galaxy Zoo Data Analysis

We began by using the Galaxy Zoo classification interface, where we classified galaxies to grasp how it works and to spot any uncertainties, in the process. We inspected galaxies distinguishing between those displaying distinct spiral structures and those with smooth shapes. This hands on approach gave us insight, into the difficulties that automated classification systems could encounter.

Then delveing into the study by Willett (2013) we analyzed the criteria utilized in creating the GZ2 sample from the SDSS DR7 imaging survey. This allowed us to grasp the breadth of the data and the complexities of the GZ2 classification system. Our theoretical exploration was complemented by hands on tasks; manipulating image data using the SDSS ImgCutout tool and conducting calculations to comprehend the scale represented in the images. Subsequently we delved

into Conselices (2014) to emphasize the significance of classification. We linked morphologies with evolution discussing how this classification enlightens us about galaxies history and developmental processes. We also investigated astronomers interest in searching for merging galaxies by referencing studies conducted by Lotz et al. (2011)/ Rodriguez Gomez et al. (2015). The review of literature aided the understanding of mergers role in evolution and traditional methods used for their identification laying a groundwork for evaluating approaches, to identifying mergers.

In direct data handling and analysis process. We assessed the Galaxy Zoo dataset (training images.tar.gz)detailing the number of images and their dimensions, and then plotted a subset of these images. We generated histograms of the label distributions, identifying skewness and multimodality, and plotted prototypical images for each label to understand which features are visually distinct and which may pose challenges to classification.

Listing 1: Galazy Zoo Labels

```
descriptive_labels = {
    'Class1.1': 'galaxy - smooth',
    'Class1.2': 'galaxy - features or disk',
    'Class1.3': 'galaxy - star or artifact',
    'Class2.1': 'disk - edge-on',
    'Class2.2': 'disk - not edge-on',
    'Class3.1': 'bar feature - bar',
    'Class3.2': 'bar feature - no bar',
    'Class4.1': 'spiral arm pattern - spiral',
    'Class4.2': 'spiral arm pattern - no spiral',
    'Class5.1': 'bulge - no bulge',
    'Class5.2': 'bulge - just noticeable',
    'Class5.3': 'bulge - obvious',
    'Class5.4': 'bulge - dominant',
    'Class6.1': 'odd - yes',
    'Class6.2': 'odd - no',
    'Class7.1': 'roundness - rounded',
    'Class7.2': 'roundness - in between',
    'Class7.3': 'roundness - cigar-shaped',
    'Class8.1': 'odd feature - ring',
    'Class8.2': 'odd feature - lens or arc',
    'Class8.3': 'odd feature - disturbed',
    'Class8.4': 'odd feature - irregular',
    'Class8.5': 'odd feature - other',
    'Class8.6': 'odd feature - merger',
    'Class8.7': 'odd feature - dust lane',
    'Class9.1': 'bulge shape - rounded',
    'Class9.2': 'bulge shape - boxy',
    'Class9.3': 'bulge shape - no bulge',
    'Class10.1': 'arms - tight',
    'Class10.2': 'arms - medium',
    'Class10.3': 'arms - loose',
    'Class11.1': 'arm count - 1',
    'Class11.2': 'arm count - 2',
    'Class11.3': 'arm count - 3',
    'Class11.4': 'arm count - 4',
    'Class11.5': 'arm count - more than four',
    'Class11.6': 'arm count - cant tell'
}
```

Then correlation matrix was then made for the labels in our dataset. This analysis helped us to evaluate whether the labels were mutually exclusive as designed and to assess the inter-rater reliability and potential overlaps between different morphological classes.

2.2 Neural Network Model Process

First we calculated memory requirements for image loading and while keeping this into account, we implemented function for image downsizing, where we found how cropping 5 percent off the borders would be a safe bet while knowing a lot of the galaxies in their own image box are large in scale and then downsizing it by a factor of 30 to account for faster processing

Listing 2: Function For Downsizing

```
def downsize_images_in_folder(input_folder, output_folder, target_reduction_factor=30

    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    for image_file in os.listdir(input_folder):
        if image_file.endswith('.jpg') and not image_file.startswith('._'):

            try:
                with Image.open(os.path.join(input_folder, image_file)) as img:
                    border = (img.size[0] * 0.05, img.size[1] * 0.05, img.size[0] * 0
                    img_cropped = ImageOps.crop(img, border)

                    scale_factor = math.sqrt(target_reduction_factor)
                    new_dimensions = (
                        max(1, int(img_cropped.size[0] // scale_factor)),
                        max(1, int(img_cropped.size[1] // scale_factor))
                    )

                    img_resampled = img_cropped.resize(new_dimensions, Image.Resamplin


                    img_resampled.save(os.path.join(output_folder, image_file))
            except Exception as e:
                print(f"Failed␣{image_file}:␣{e}")
```

and then The batch generator function efficiently handles the processing of image data and labels in batches from a dataset for training machine learning models. It calculates the number of batches by dividing the total number of entries in df by batch size, iterating over each batch to load and transform images into tensor format suitable for neural network input. For each batch, it extracts images and corresponding labels, converts them into arrays, and yields these arrays to be used in model training, ensuring that all data handling occurs in memory-efficient batches to avoid overloading the system.

Listing 3: Batch Generator

```
def batch_generator(data_df, image_folder, batch_size):
    num_batches = len(data_df) // batch_size + (len(data_df) % batch_size > 0)

    for i in range(num_batches):
        batch_slice = slice(i * batch_size, (i + 1) * batch_size)
        batch_df = data_df.iloc[batch_slice]

        images = []
        labels = []

        for _, row in batch_df.iterrows():
            image_path = os.path.join(image_folder, row['ImagePath'])
            with Image.open(image_path).convert('RGB') as img:
                img_array = np.array(img)
                img_array = np.transpose(img_array, (2, 0, 1))
                images.append(img_array)
```

```
            label_data = row.drop(['GalaxyID', 'ImagePath']).values.astype(float)
            labels.append(label_data)

        images = np.array(images)
        labels = np.array(labels, dtype=float)

        if np.any(np.isnan(labels)):
            print("NaN values found in labels")


        yield images, labels
```

We also established a baseline model performance using a simple mean-label approach, setting the stage for comparison with more complex models using RMSE loss function defined as

$$L_{\text{RMSE}} = \sqrt{\left\langle (\vec{\ell}_{\text{true}} - \vec{\ell}_{\text{pred}})^2 \right\rangle}$$

$$= \sqrt{\frac{1}{N_{\text{galaxies}} N_{\text{labels}}} \sum_i \sum_j (\ell_{\text{true},ij} - \ell_{\text{pred},ij})^2}.$$

Acknowledging this, we then create the detailed the iterative process of building, training, and refining a Custom Convolutional Neural Network for galaxy classification; The network design includes two layers that use batch normalization, ReLU activation and max pooling to effectively process input images. The initial layer applies 32 filters sized 3x3 with padding to maintain dimensionality while the subsequent layer increases to 64 filters improving the networks ability to capture features. Following convolution a dropout layer prevents overfitting by deactivating 20 percent of neurons. Subsequently connected layers compress the feature maps into a 128 vector. The final output layer utilizes a sigmoid function, for categorizing into 37 classes. Based on sequential set of trials we found that to optimize learning rates and achieve convergence the network would utilize an Adam optimizer compared to the slower SGD method.

Listing 4: Custom Neural Network

```
class CNN(nn.Module):
    def __init__(self, num_labels):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, 3, padding=1)
        self.bn1 = nn.BatchNorm2d(32)
        self.act1 = nn.ReLU()
        self.pool1 = nn.MaxPool2d(2)
        self.conv2 = nn.Conv2d(32, 64, 3, padding=1)
        self.bn2 = nn.BatchNorm2d(64)
        self.act2 = nn.ReLU()
        self.pool2 = nn.MaxPool2d(2)
        self.dropout = nn.Dropout(0.2)
        self.fc1 = nn.Linear(64 * 17 * 17, 128)
        self.act3 = nn.ReLU()
        self.fc2 = nn.Linear(128, num_labels)
        self.act4 = nn.Sigmoid()

    def forward(self, x):
        x = self.pool1(self.bn1(self.act1(self.conv1(x))))
        x = self.pool2(self.bn2(self.act2(self.conv2(x))))
        x = x.view(x.size(0), -1)
        x = self.dropout(x)
        x = self.act3(self.fc1(x))
        x = self.act4(self.fc2(x))
        return x
```

```
# Parameters
num_labels = 37

# Model initialization
model = CNN(num_labels)

# Optimizer and loss function
optimizer = optim.Adam(model.parameters(), lr=0.0001)
loss_fn = nn.MSELoss()  # Using MSE loss for now (will get squared rooted in training
```

Then the function for training the model is designed to teach the network model over a number of rounds and batches utilizing distinct datasets for training and validation to check effectiveness and prevent excessive fitting. During each round the model goes through a learning process where it gains knowledge from the training data by adjusting weights through backpropagation and gradient descent, managing gradients, computing loss with MSE and updating model parameters. After each round of training the model is tested on the validation dataset without updating gradients to assess performance; RMSE loss is calculated for both training and validation results to monitor progress and identify signs of inadequate fitting, as training advances.

Listing 5: NN Training

```
def train_model(model, train_df, valid_df, num_epochs, batch_size):
    train_losses = []
    valid_losses = []
    num_images_seen = 0

    for epoch in range(num_epochs):
        train_gen = batch_generator(train_df, 'new_training_images', batch_size)
        valid_gen = batch_generator(valid_df, 'new_training_images', batch_size)

        model.train()
        total_train_loss = 0
        for images, labels in tqdm(train_gen, desc=f"Epoch {epoch+1}/{num_epochs} - T
            images = torch.tensor(images, dtype=torch.float32)
            labels = torch.tensor(labels, dtype=torch.float32)

            optimizer.zero_grad()
            outputs = model(images)
            mse_loss = loss_fn(outputs, labels)
            mse_loss.backward()
            optimizer.step()

            total_train_loss += mse_loss.item() * images.size(0)

        average_train_loss = np.sqrt(total_train_loss / len(train_df))
        train_losses.append(average_train_loss)
        num_images_seen += 1

        # Validation
        model.eval()
        total_val_loss = 0
        count = 0
        with torch.no_grad():
            for images, labels in tqdm(valid_gen, desc=f"Epoch {epoch+1}/{num_epochs}
                images = torch.tensor(images, dtype=torch.float32)
                labels = torch.tensor(labels, dtype=torch.float32)

                outputs = model(images)
                mse_loss = loss_fn(outputs, labels)
                total_val_loss += mse_loss.item() * images.size(0)
```

```
                count += images.size(0)

        if count == 0:
            print("No data processed in validation.")
        else:
            average_val_loss = np.sqrt(total_val_loss / count)
            valid_losses.append(average_val_loss)
            print(f'Epoch {epoch+1}, Training RMSE: {average_train_loss}, Validation R

    return train_losses, valid_losses, num_images_seen
```

A newer model was then made, utilizing pre-existing architectures like ResNet, optimizing hyperparameters, and employing learning rate schedulers to minimize overfitting and maximize the model's performance.

Listing 6: Resnet NN

```
model_3 = models.resnet18(pretrained=False)

model_3.avgpool = nn.AdaptiveAvgPool2d((1, 1))

model_3.fc = nn.Linear(model_3.fc.in_features, num_labels)

optimizer_3 = optim.Adam(model_3.parameters(), lr=0.001)

loss_fn_3 = nn.MSELoss()
```

Listing 7: Resnet w/ Scheduler

```
def train_model_3(model, train_df, valid_df, num_epochs, batch_size):
    train_losses = []
    valid_losses = []
    num_images_seen = 0

    # Initialize the scheduler without verbose
    scheduler = ReduceLROnPlateau(optimizer_3, 'min', factor=0.1, patience=1, verbose=

    for epoch in range(num_epochs):
    ..... (code process)
    # Scheduler step
        scheduler.step(average_val_loss)
        current_lr = optimizer_3.param_groups[0]['lr']
        print(f"After Epoch {epoch+1}, learning rate is adjusted to: {current_lr}")

    return train_losses, valid_losses, num_images_seen
```

After picking the best model (model3) we then focused on evaluating the model's predictive performance. We compared the true and predicted values for various labels to assess how well the model performed

Listing 8: Model Assesement For Predictions

```
model_3.load_state_dict(torch.load('best_resnet_galaxy_model.pth'))
model_3.eval()


valid_gen = batch_generator(valid_df, 'new_training_images', batch_size=200)
true_labels = []
predicted_labels = []

with torch.no_grad():
    for images, labels in valid_gen:
```

```
        images = torch.tensor(images, dtype=torch.float32)
        labels = torch.tensor(labels, dtype=torch.float32)
        outputs = model_3(images)


        true_labels.extend(labels.numpy())
        predicted_labels.extend(outputs.numpy())

true_labels = np.array(true_labels)
predicted_labels = np.array(predicted_labels)
```

In the final phase of our lab, we used the trained model on a dataset named testimages.tar.gz to figure out the number of galaxy mergers. This process included running the test images through the model and using the probabilities predicted to identify whether each galaxy was involved in a merger. We then computed the percentage of galaxies marked as mergers. Then compared this real world rate, with the merger rate, around z = 0 documented by Lotz et al.

3 Results

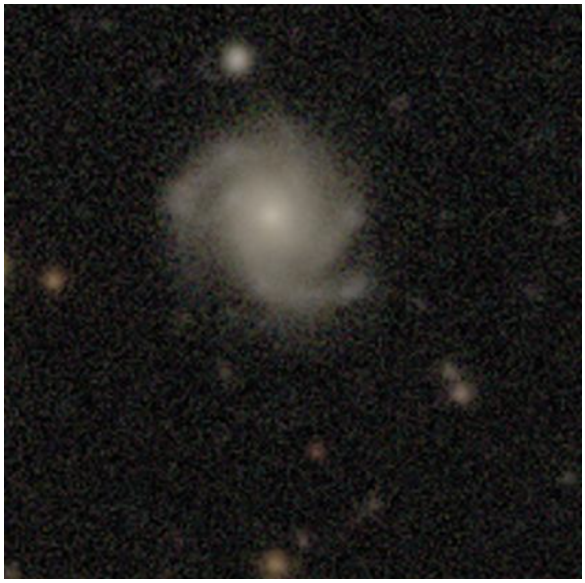


Figure 2: The Image Above is a Galaxy Zoo Project Sample with Clear Structure

Figure 3: The Image Above is a Galaxy Zoo Project Sample with Smooth Structure

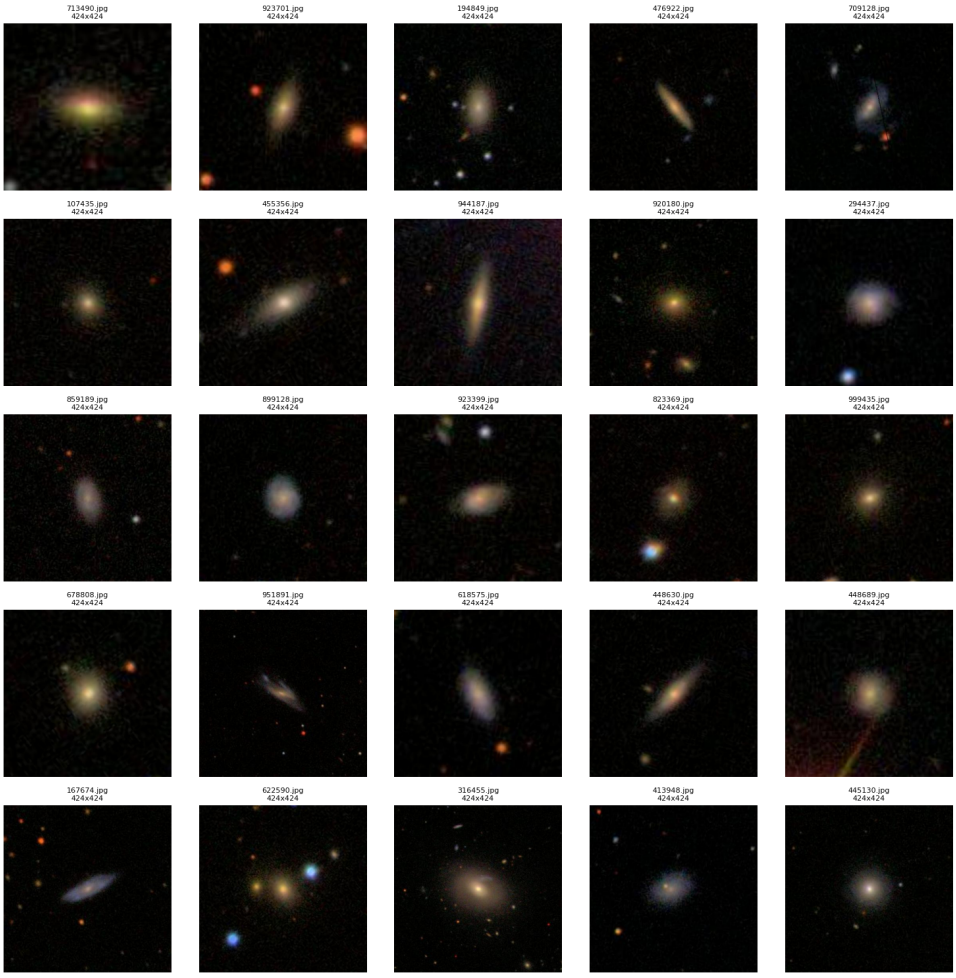


Figure 4: The Images Above are Galaxy Zoo Project Samples with their Dimension Sizes and Galaxy ID

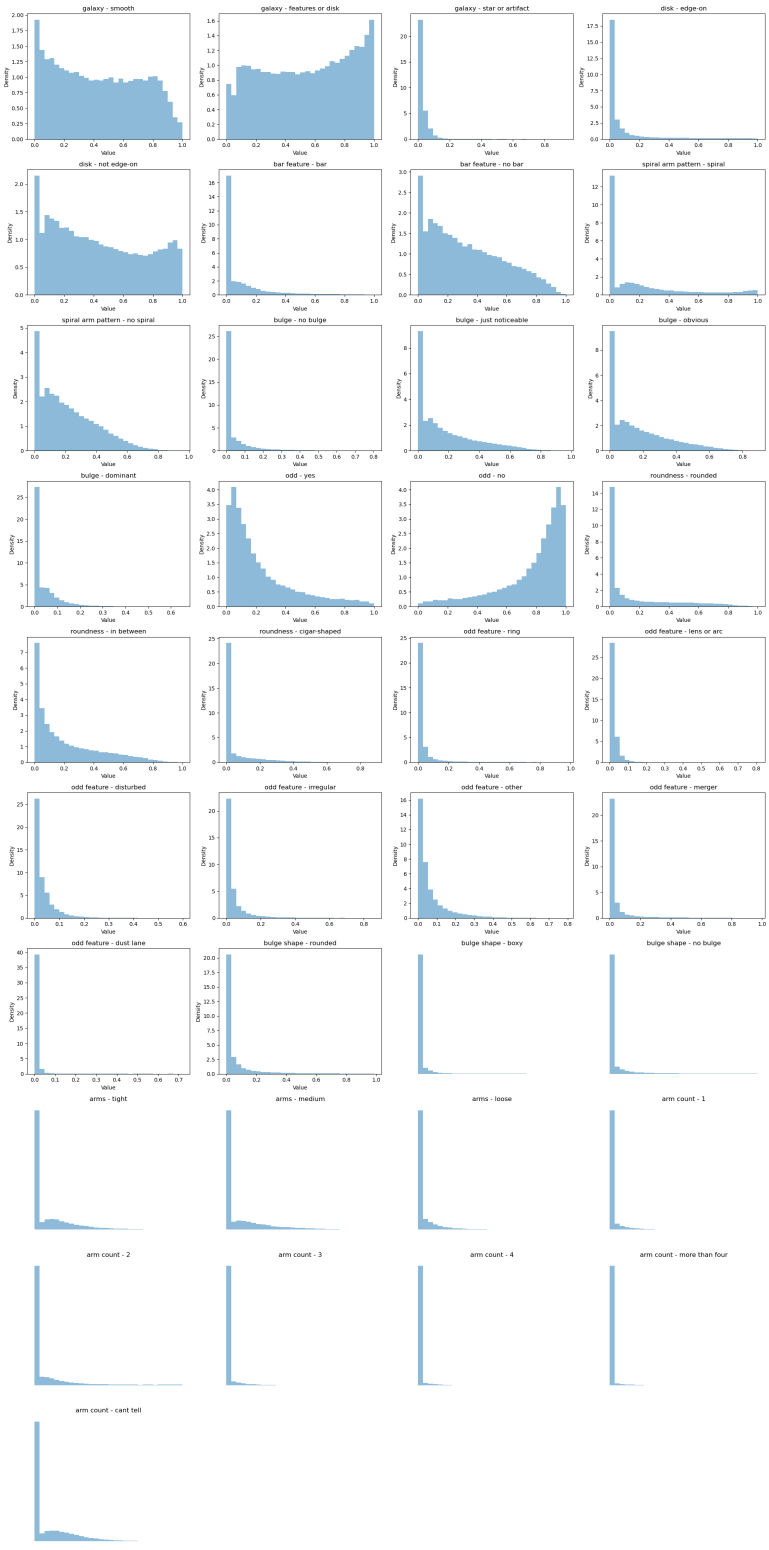


Figure 5: The plot above displays histograms representing the distribution of classification responses for various galaxy morphological features, each derived from the Galaxy Zoo project. Each subplot corresponds to a different class, such as "galaxy - smooth" or "disk - edge-on," showing the density of classifications across the Galaxy Zoo dataset, which provides insights into the typical appearances and frequency of specific galactic features as observed and categorized by volunteers.

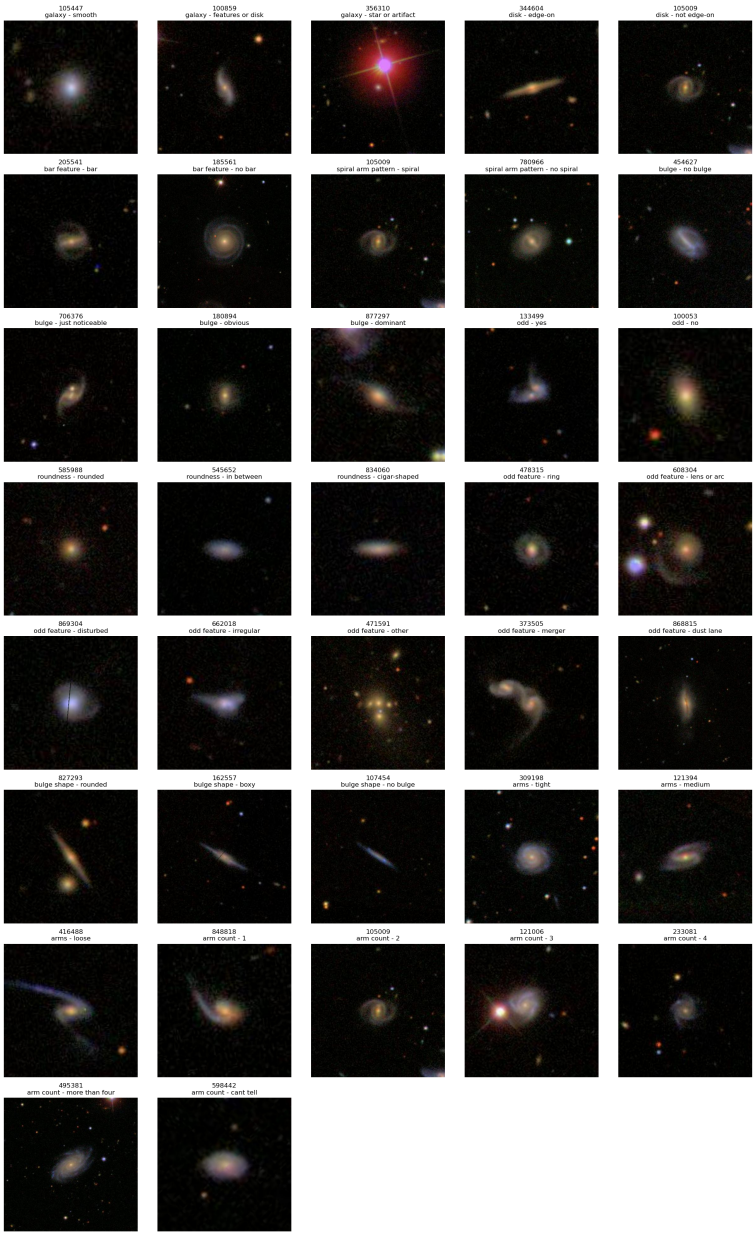


Figure 6: These images represent prototypical examples of various galaxy morphologies identified in the Galaxy Zoo project, each selected as the most representative image for specific classification labels. Each image showcases clear characteristics that define each category

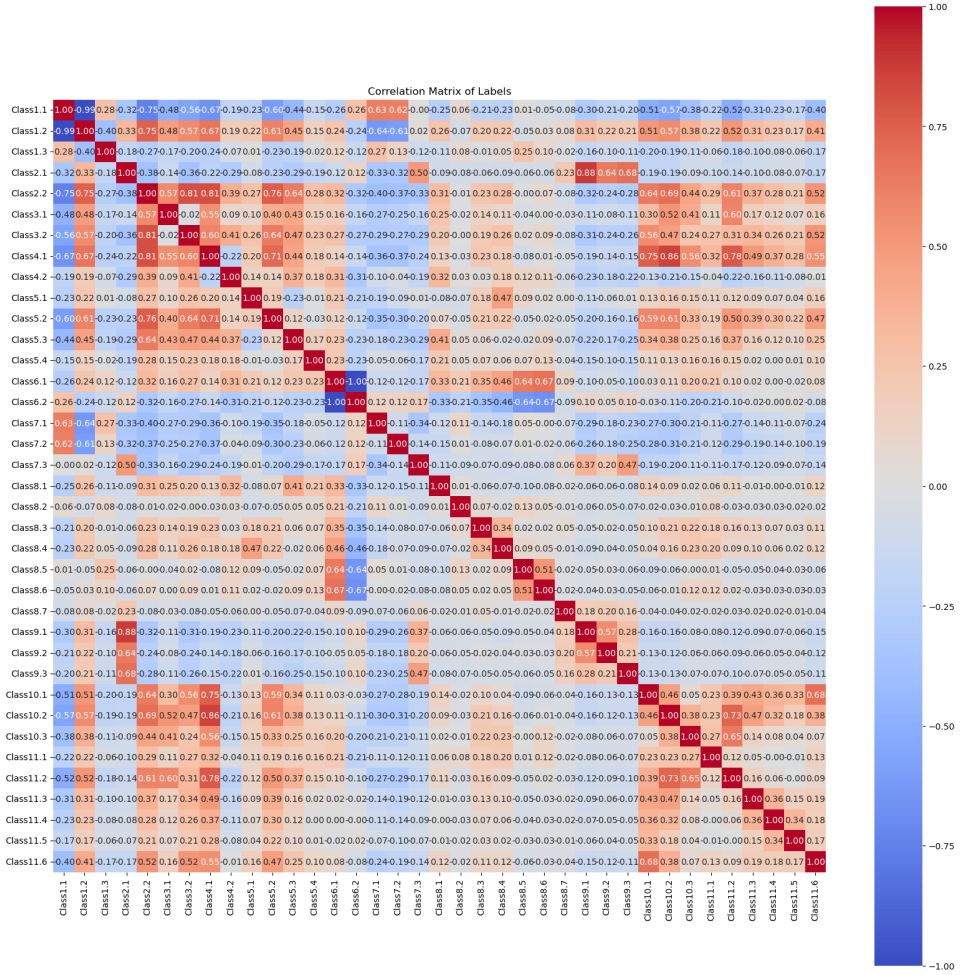


Figure 7: The heatmap illustrates the correlation matrix of various galaxy morphological labels, revealing strong and weak relationships, where some expected negative correlations are surprising by their absence, suggesting complex interdependencies among galaxy features.

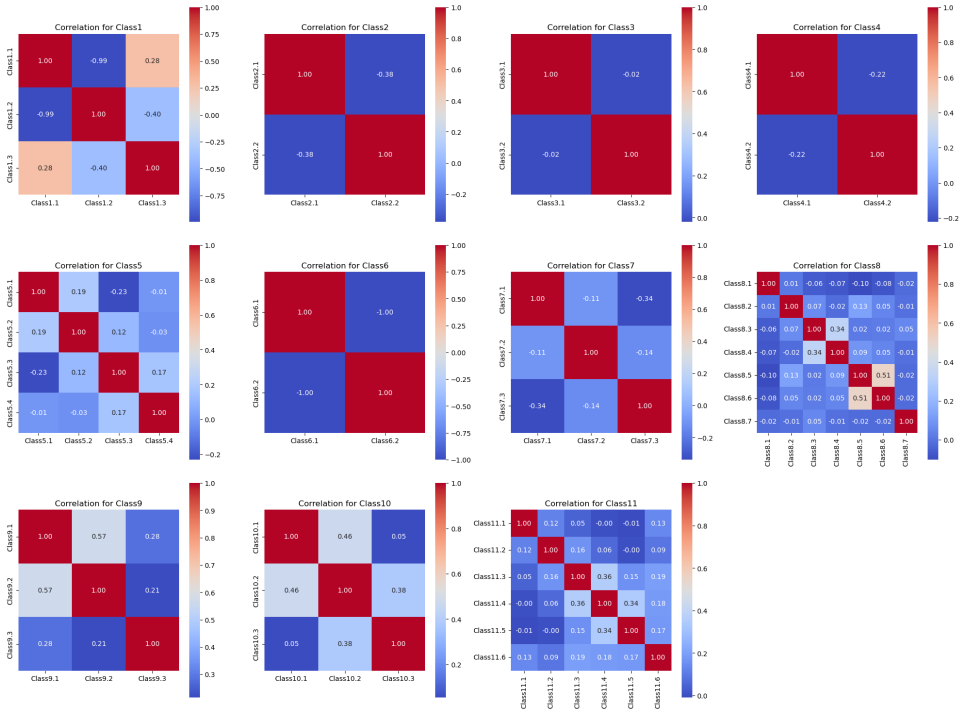


Figure 8: This heatmap illustrates the correlation coefficients among different classification labels within each category of the Galaxy Zoo dataset, highlighting varying degrees of correlation and suggesting areas where label definitions or classifier training may need refinement to improve reliability.

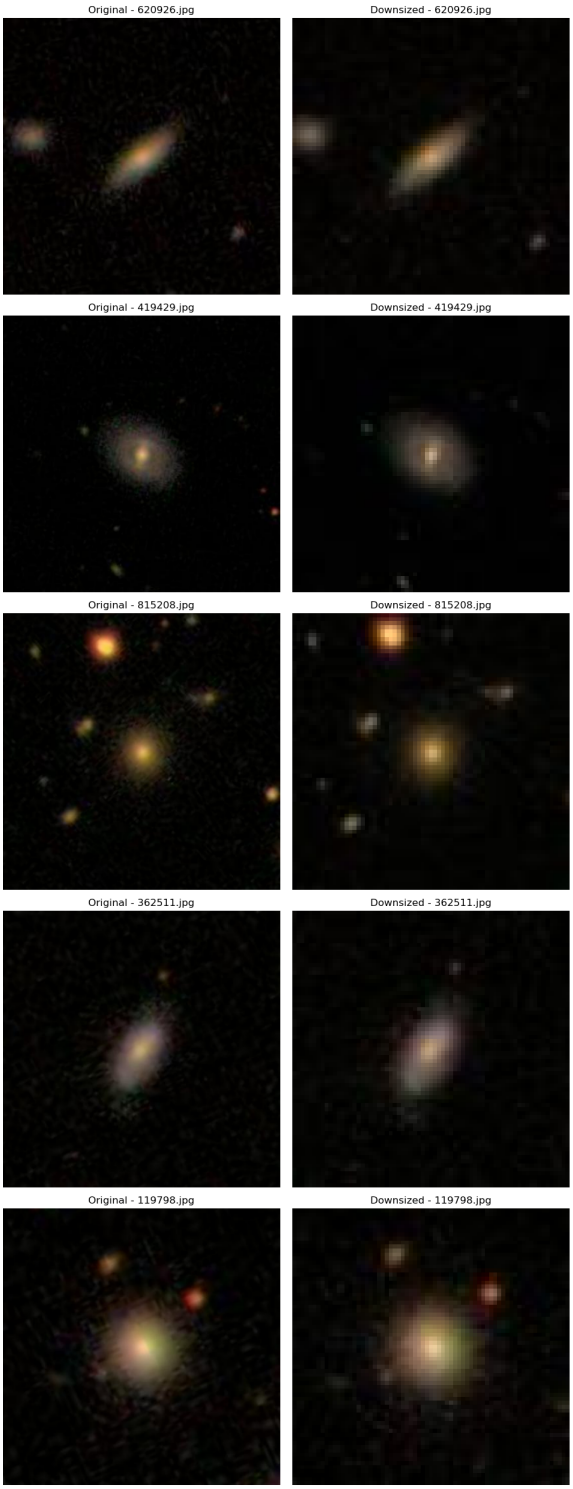


Figure 9: This set of images displays side-by-side comparisons between original and downsized versions of galaxy images, illustrating the effect of image reduction techniques used to prepare data for more efficient processing in galaxy classification models.

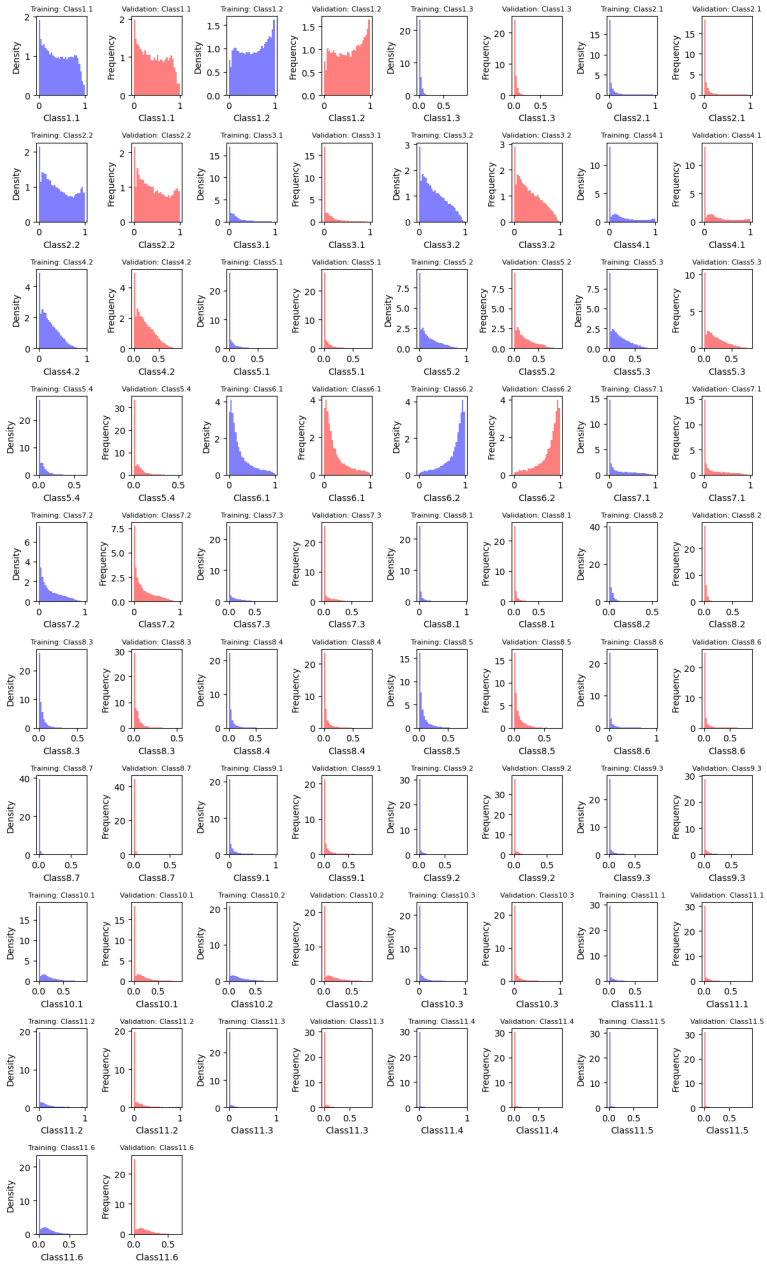


Figure 10: The histograms above compare the distribution of various galaxy morphological features between training and validation datasets, highlighting differences in density and prevalence for each classified label across both subsets of data.

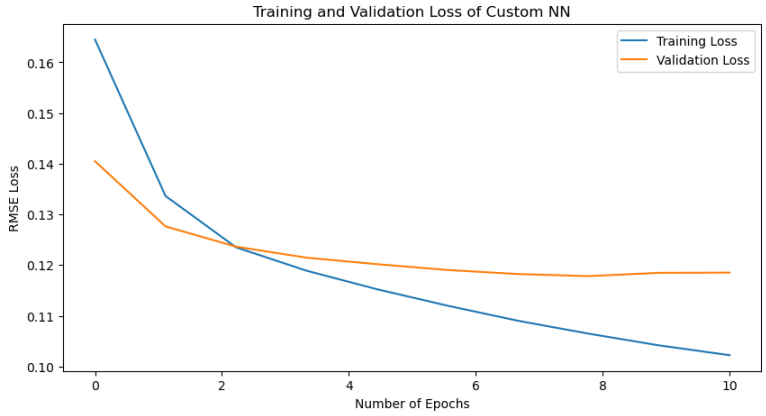


Figure 11: The plot visualizes the training and validation RMSE losses of the custom neural network model trained over multiple epochs, illustrating how the model's performance improves as it learns from a set of galaxy images, with the ultimate goal of enhancing accuracy in galaxy morphology classification.

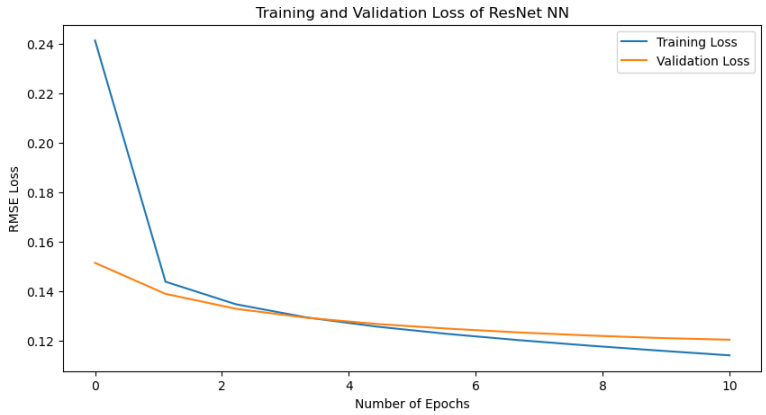


Figure 12: The plot visualizes the training and validation RMSE losses of the out of box neural network model trained over multiple epochs.

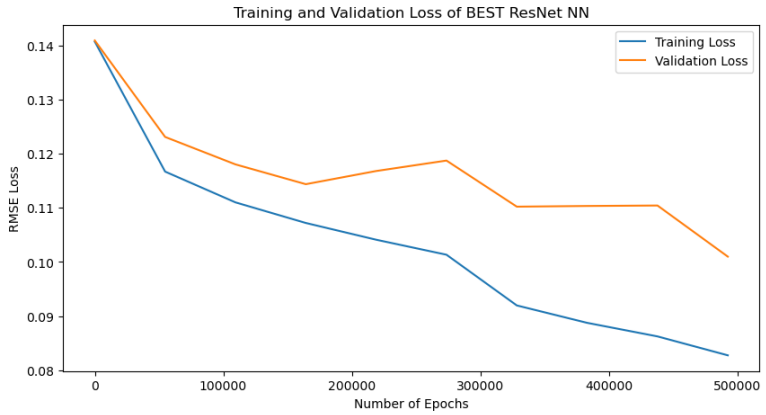


Figure 13: The plot visualizes the training and validation RMSE losses of the out of box augmented neural network model with learning rate scheduler trained over multiple epochs.

Figure 14: The single plot above visualizes the validation RMSE losses of all 3 neural network model trained over multiple epochs.

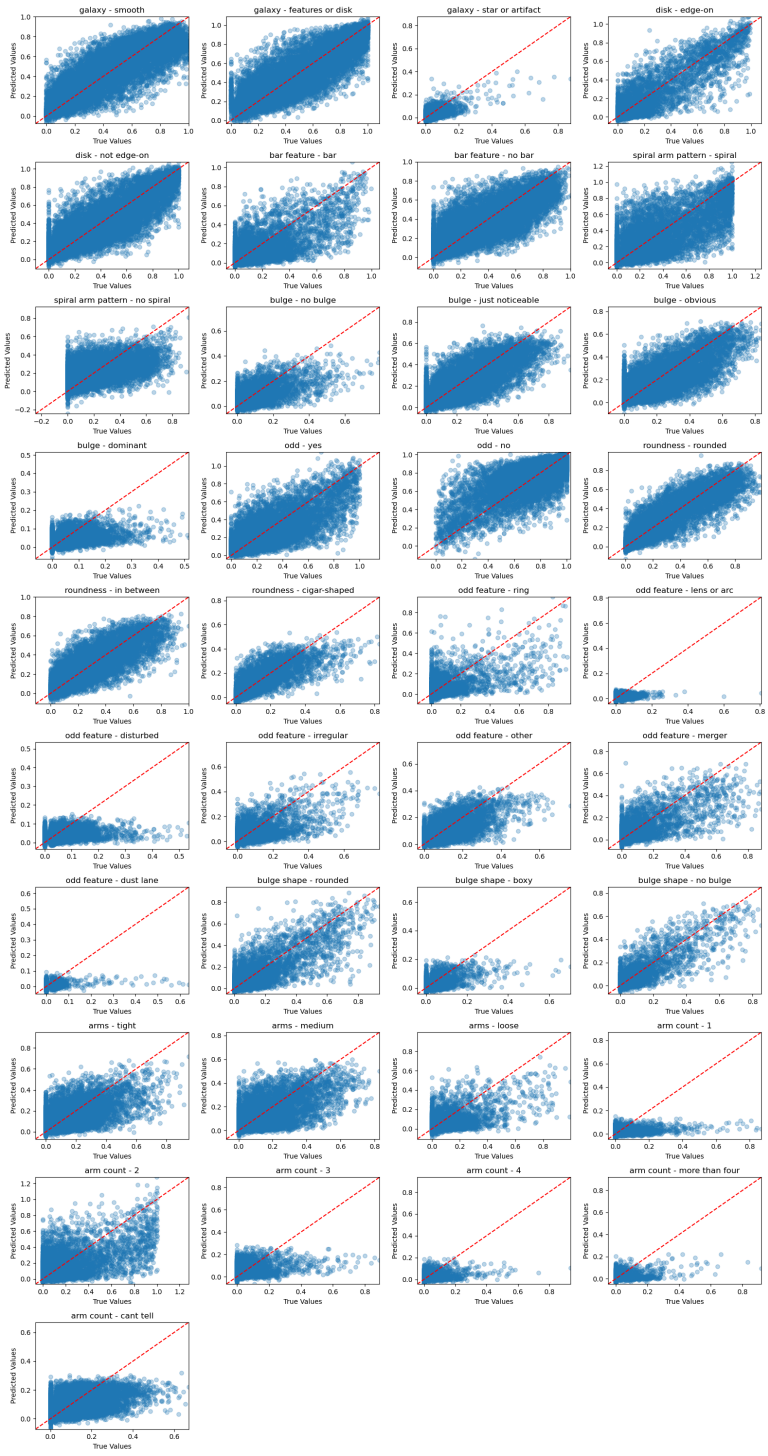


Figure 15: The plot displays the correlation between true and predicted labels for various galaxy morphological features classified by a neural network model. Each subplot represents a different classification label and shows how closely the model's predictions align with actual observations, with the red line indicating perfect prediction accuracy.

Top 5 Actual - smooth

Top 5 Predicted - smooth

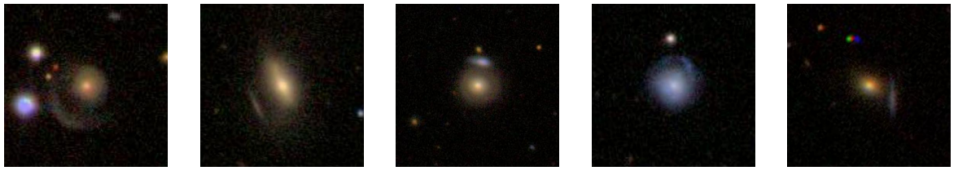
Top 5 Actual - star/artifact

Top 5 Predicted - star/artifact

Top 5 Actual - edge-on disk

Top 5 Predicted - edge-on disk

Top 5 Actual - odd: lens/arc



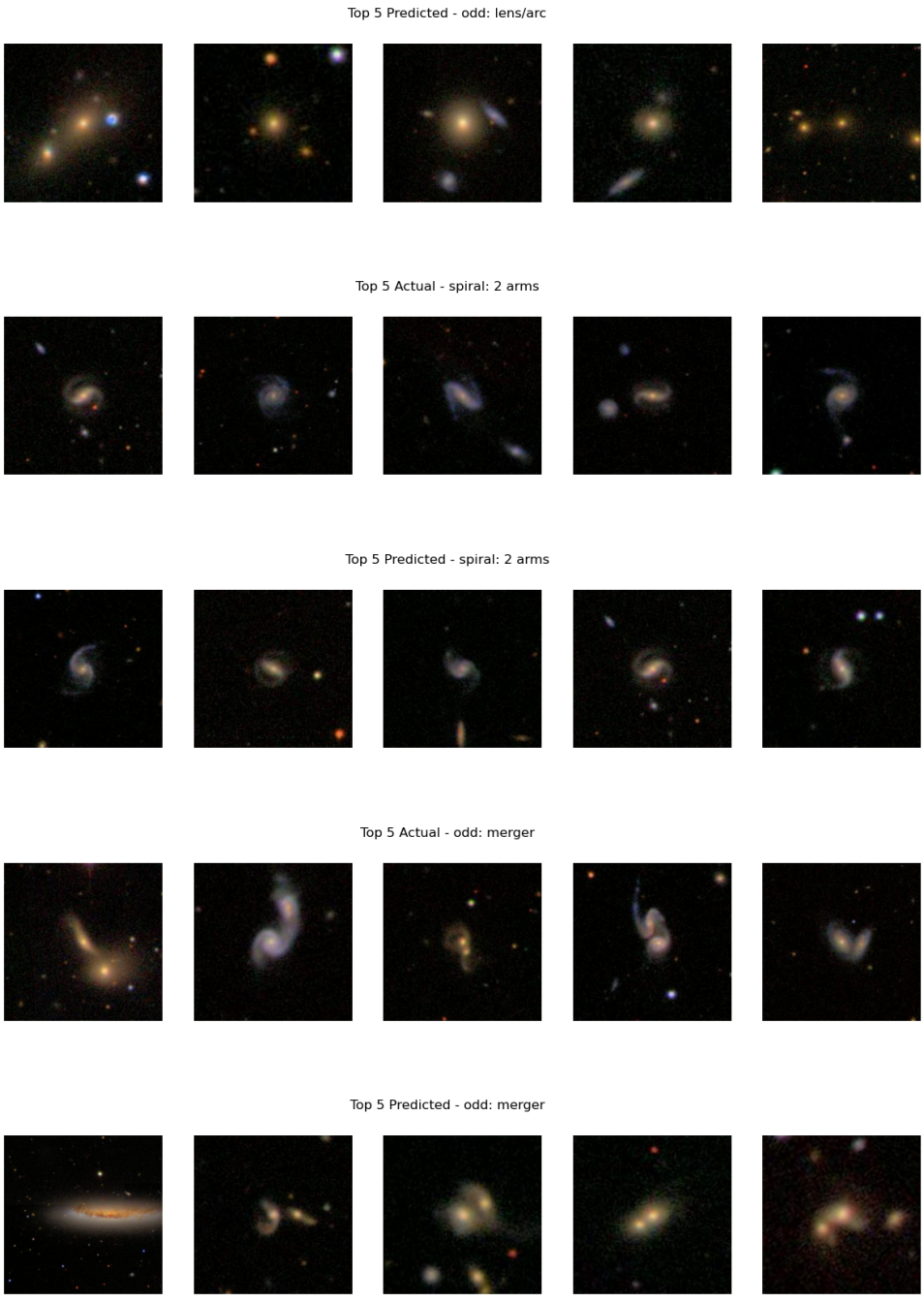


Figure 16: The above showcases top images for various galaxy classification labels, providing a visual comparison between galaxies that were most strongly identified with certain features in actual observations and those predicted by the model

4 Discussion

4.1 Exploring Galaxy Morphologies through GalaxyZoo: An Overview and Classification Challenges

In the Galaxy Zoo classification scheme, galaxies are sorted based on how they look. The "Smooth" group includes galaxies that seem smooth without any features often representing galaxies. The "Features or Disk" category consists of galaxies, with structures, like arms, bulges, bars or other interesting details usually including spiral and irregular galaxies. Lastly the "Star, Artifact or Bad Zoom" category is used when an images quality is affected by stars, artifacts, incorrect scaling or other issues that make it hard to see the galaxy clearly and classify it accurately. For the most of

the classifications, they came out to be pretty ambiguous for the quality of the images or simply having a bad zoom.

4.2 Detailed Analysis of Galaxy Zoo 2, Willett et al. 2013

-The SDSS DR7 Legacy sample, for GZ2 underwent criteria adjustments, such as requiring a half light magnitude brighter than 17.0 in the r band (accounting for Galactic extinction correction) a petroR90 size limit of r>3 arcsec and a spectroscopic redshift range between 0.0005 and 0.25. Any objects identified as saturated, bright or blended without a nodeblend flag were excluded. Initially there were 245,609 galaxies, in the 'original' sample to which an additional 28,174 galaxies were later added and referred to as the 'extra' sample.

-The pictures of galaxies, in GZ2 were made using the SDSS ImgCutout tool. Each image is a 424 × 424 color combination of gri adjusted to a scale of 0.02×petroR90 r arcsec per pixel. Although the pixel scale remains uniform at 0.02×petroR90 r arcsec per pixel the actual size, in kiloparsecs would differ based on the galaxys distance and angular dimensions.

-The classification scheme, in GZ2 uses a decision making process that defines a classification based on the information gathered about a galaxy by an user following the decision tree. The GZ2 decision tree comprises 11 classification tasks with a combined total of 37 responses.

-In GZ1 they used a system to classify galaxies mainly differentiating between spiral types and recognizing mergers. GZ2 on the hand brought in an intricate decision tree that enabled finer distinctions of characteristics within each galaxy.

-Galaxies shown to users were randomly selected from the database. To ensure sufficient classifications per galaxy though, images with low numbers of classifications were shown at a higher rate towards the end of the project.

-When the paper was being written, on average each galaxy, in GZ2 dataset got 44 classifications. -An incremental weighting method was employed to lessen the impact of classifiers. The votes, from users were adjusted according to how they aligned with the majority of classifications. Each users overall alignment was. Weights were allocated so that 95 percent of classifiers received equal treatment while only approximately 1 percent of classifiers exhibiting the least consistency were notably devalued.

-The main reason, behind the observed redshift evolution in the GZ2 sample is related to classification bias. This bias occurs because galaxies that are away often seem smaller and fainter in images, which makes it harder to recognize morphological characteristics. As a result the shifts in type with redshift are not just a result of actual changes, in galaxy properties but also stem from challenges in accurately observing and categorizing these distant galaxies.

-GZ2 addresses this bias by modifying classifications based on a reference point taken from galaxies, at redshifts. This adjustment helps to ensure that classifications accurately represent the characteristics of galaxies across varying distances. In GZ2 classification bias pertains to the errors in categorizing galaxy shapes caused by shifts in galaxy visibility, as redshift changes.

-Galaxies observed at higher redshifts, in GZ2 are often labeled as mergers due to their increased brightness and distinct features that mimic the disordered shapes commonly seen in merging galaxies. Moreover the smaller size and lower clarity of these galaxies at redshifts can pose challenges, in identifying details accurately sometimes resulting in misinterpretations of interactions or irregularities that may not actually exist.

-Every galaxy classification starts with a or no response, to a set of questions regarding its shape and structure. Yet the outcomes are displayed as numbers between 0 and 1 as they indicate the proportion of votes each answer garnered. This method combines the inputs from volunteers to create a likelihood measure for each characteristic mirroring the collective agreement among participants, on the galaxys form.

-GZ2 uses contributions, from a group of volunteers who provide classifications, which are then combined to form probabilistic measures. This method differs from expert created catalogs that rely on classifiers. May not include a confidence based probabilistic approach, for each feature.

4.3 The Importance of Morphological Classification in Understanding Galaxy Evolution, Insights from Conselice, 2014

Astronomers find the categorization of galaxies important because it is key, to grasping how galaxies form and change over time. Conselice (2014) highlights that understanding the structure and

appearance of galaxies is a part of research aiding in recognizing different galaxy types and deducing their evolutionary paths. For example the presence of characteristics like bars spiral arms or bulges can indicate ways in which galaxies form and evolve. Moreover analyzing galaxy shapes across eras helps astronomers track the processes influencing galaxy development, such as mergers, star formation rates and interactions, with their surroundings.

4.4 Search for Merging Galaxies: Methods and Motivations Based on Literature by Lotz et al., 2011 and Rodriguez-Gomez et al., 2015

Astronomers actively seek out merging galaxies because these occurrences play a role, in shaping the evolution of galaxies affecting processes such as star formation, the growth of black holes and changes in their overall structure. They spot mergers by observing positioned galaxy pairs that're likely to merge together or by detecting visual irregularities like tidal tails and asymmetries that indicate recent interactions. Tools like the Gini coefficient/M20, which analyze how light is distributed are also used to identify merging galaxies. Understanding merger rates and their impact, on galaxy evolution requires combining data from detection methods each capturing aspects and stages of galaxy mergers effectively.

4.5 Exploring the Galaxy Zoo Training Set: Image Analysis and Data Characteristics

There are 61,578 JPEG files in the folder, and each image has dimensions of 424 x 424 pixels.

4.6 Distribution Analysis of Galaxy Morphological Labels from the Galaxy Zoo Training Set

There seems to be a lot of skewness within these histograms. When there's a label with a right skew it indicates that the attribute being analyzed is not prevalent, in the majority of galaxies in the sample. On the hand a left skew suggests that the attribute is more common. There doesn't seem to be any evidence of modal distributions, which would imply different subgroups within the sample.

4.7 Visualizing Prototypical Galaxies: Analyzing the Most Representative Images for Each Morphological Label

Certain labels, with distinct characteristics are easier to categorize compared to others. For instance labels like "Galaxy - Smooth," describing galaxies without features or a disk; "Disk - Edge On," showing the galaxys disk from a side view;. Spiral Arm Pattern - Spiral," with well defined spiral arms are relatively simple to classify. On the other hand nuanced labels pose greater challenges. For example identifying a "Bulge - Just Noticeable" can be tricky as it is subtle and may resemble galaxies without a bulge. Detecting a Bar Feature - Bar" or unclear images can be complex. Additionally the "Odd Feature - Other" label includes features that don't neatly fit into predefined categories making classification more intricate. These subtle labels may introduce uncertainties, in the classification process.

4.8 Investigating Label Correlations in Galaxy Zoo Training Set

The correlation matrix shows that distinct labels, like "smooth and featureless" compared to "features or a disk " exhibit the correlation with a coefficient close to 1. Notably Class 6.1 stands out due to its increasing correlations with peculiarities such as rings, mergers or dust lanes, in Classes 8.1 to 8.6. Then for Class 6.2 there seems to be a rise for negative correlations for those specific labels. Additionally there seems to be a huge portion of non negative correlations for Classes 10.1 to 11.6 partaining to arms and their counts.

4.9 Assessing the Reliability of Classifications

Some classes exhibit more mutual exclusivity than others. For instance Class 1 and Class 6 display it as anticipated w/ their values being close to -1. Not all non diagonal values equate to -1,

indicating that they are not all entirely mutually exclusive. Classes such as Class 5 and Class 11 demonstrate relationships among certain labels indicating a necessity to enhance the precision of definitions or the instruction of classifiers for those particular attributes. Inconsistent correlation values within the within the same class suggest inter-rater reliability issues.

4.10 Memory Requirements for Image-Based Galaxy Classification: Estimating Load Sizes

The minimum amount of memory to load each image in the training set would be 31672.22 MB

4.11 Performance in Galaxy Classification: Simple Statistical Model

Training RMSE: 0.1637932709304058, Validation RMSE: 0.16409999918120458

4.12 Designing and Training a Convolutional Neural Network for Galaxy Morphology Classification

The network structure consists of two layers, each followed by batch normalization, ReLU activation and max pooling. The initial convolutional layer applies 32 filters of size 3x3, with padding to maintain dimensions followed by batch normalization and ReLU activation for non linearity and a max pooling layer that reduces the size by half. The second convolutional layer increases the depth to 64 while maintaining the sequence of operations to enhance the networks ability to capture intricate features. Following the layers is a dropout layer that helps prevent overfitting by deactivating a portion of 20 percent input units during training. The network then moves on to layers; for instance self.fc1 = nn.Linear(64 * 17 * 17 128) adjusts the flattened output of pooled feature maps to a vector of size 128 after the image dimensions shrink from 69x69 to 17x17 post two pooling layers. The final layer employs a sigmoid activation function to produce probabilities for each of the 37 labels. Opting for Adam optimizer over SGD was based on Adams capability to adjust learning rates for parameters and its quicker convergence compared to SGDs slower convergence I received intially by it taking many epochs to get a proper result.

4.13 Evaluating the CNN Model for Galaxy Classification, A Parameter Analysis

The model contains a total of 2,391,973 parameters, which is much less than the 234,536,382 total pixels in the training set. The difference between them suggests that the model is not likely to overfit by memorizing the training data considering its complexity compared to the size. Therefore there is no need to be concerned about the number of parameters, in this scenario!

4.14 ResNets for Galaxy Image Classification

-The current CNN includes 2 convolutional layers . A dropout rate of 20 percent is applied before the fully connected layers to mitigate overfitting. Pooling is performed using 2x2 max pooling.

-Residual Networks, also known as ResNets represent a type of Convolutional Neural Network that stands out for its increased depth compared to models. This advancement was made possible by incorporating residual blocks, with shortcut connections as proposed in 2015 He et al. These networks enhance the training process of architectures by enabling gradients to flow efficiently through the network. This approach effectively tackles issues, like vanishing gradients which leads to improved model performance. Unlike CNNs that learn unreferenced functions, ResNets are structured to learn residual functions while considering layer inputs. This design makes it easier to train networks without sacrificing performance quality as deeper models have the potential to capture intricate functions theoretically.

4.15 Implementing and Comparing ResNet and Custom CNN Models

The Resnet model did fairly the same compared to the custom NN in terms of acheiving almost the same RMSE and total trainable parameters here is 11,195,493.

4.16 Optimizing Learning Rates in Neural Network

After adjusting the learning rate using a scheduler the plot showed that the loss stabilized and then started decreasing as the epochs progressed. This suggests that the changes made were successful, in helping the model overcome plateaus and keep improving and acheiving a less than .10 RMSE !

4.17 Assessing Model Accuracy in Galaxy Morphological Classification

In general the model did well in the classification task accurately predicting values, for categories like smooth galaxy and disk Edge on and ect.. However it faced challenges with features such as odd feature - Dust lane and arm count labels where the actual data had some outliers and the model made less precise predictions which will be modified in the future

4.18 Evaluating Extreme Cases in Galaxy Classification: Comparing Actual vs. Predicted Labels

The model usually does a good job closely aligning with the labels, in the validation set particularly for classifications such as "smooth " "edge on disk " and "2 armed spiral," showing strong consistency. Yet there are differences observed in categories like "lens/arc" and "odd: merger," where the predictions sometimes deviate from the actual labels possibly due to the intricacy/infrequency of these characteristics in the validation set.

4.19 Estimating Galaxy Merger Rates Using Neural Network: A Comparing with Theoretical Expectations

When using other imported galaxy samples, the current model predicts a merger (above a .5 threshold) a fraction of 0.033 below the reported z = 0 merger rate of around 0.05, by Lotz et al. This difference may stem from variations in how mergers defined in the criteria for selecting samples/merger rates from the paper are given in units Gyr^{-1}, for the merger fraction. The rates provided show how often mergers occur relative, to the volume and number of galaxies, over time taking into account the Hubble constant h70 in the volume, so the difference in units will be adjusted for this.

5 Conclusion

In summary the research detailed in this paper underscores the importance of using automated methods to classify galaxies in furthering our knowledge of galaxy shapes and changes, over time. Through analyzing data from Galaxy Zoo and utilizing neural network models, including unique designs and ResNet we have enhanced the precision and effectiveness of galaxy classification. Challenges like bias in classification and pinpointing merging galaxies in a manner were dealt with, which then improved our methodology and predictive abilities. This study did not only enhance how we categorize galaxy images but also enriches the discussion in astrophysics by blending computational tools with observational astronomy to unravel the characteristics of galaxy evolution.

References

Galaxy Zoo Project

D. G. York, et al. 2000 ; Sloan Digital Sky Survey arXiv:0006396

Lintott et al. 2008; Galaxy Zoo : Morphologies derived from visual inspection of galaxies from the Sloan Digital Sky Surveyarxiv:0804.4483

Masters et al. 2019;Twelve Years of Galaxy Zoo arXiv:1910.08177

Willett et al. 2013 ; Galaxy Zoo 2: detailed morphological classifications for 304,122 galaxies from the Sloan Digital Sky Survey arXiv:1308.3496

Krizhevsky et al. 2020; ImageNet Classification with Deep Convolutional Neural Networks ImageNet Classification

Conselice, 2014; The Evolution of Galaxy Structure over Cosmic Time arXiv:1403.2783

Rodriguez-Gomez et al, 2015; The merger rate of galaxies in the Illustris Simulation: a comparison with observations and semi-empirical modelsarXiv:1502.01339

Lotz et al., 2011; The Major and Minor Galaxy Merger Rates at z $<$ 1.5 arXiv:1108.2508

He et al. 2015; Deep Residual Learning for Image Recognition arXiv:1512.03385