Draft version May 3, 2024
Typeset using LaTeX **twocolumn** style in AASTeX63

Astronomy 128: Astro Data Science Lab
Lab 3: Galaxy Image Classification and the Galaxy Merger Rate

Hien Nguyen[1, 2]

[1] *Department of Astronomy, University of California, Berkeley, CA 94720-3411, USA*
[2] *Department of Physics, University of California, Berkeley, CA 94720-7300, USA*

ABSTRACT

Three convolutional neural network (CNN) image classification models (conventional custom CNN, ResNet50, optimized ResNet50) were trained with galaxy images from the Sloan Digital Sky Survey (SDSS) and classification labels from the Galaxy Zoo 2 project. Average MSE (RMSE) validation losses of 0.012 (0.11), 0.0067 (0.082), and 0.0092 (0.096) were achieved with mean validation accuracy 0.75, 0.73, and 0.78, respectively. The fraction of merger galaxies at redshift $z = 0$ was estimated to be 0.026, on the basis of a merger probability $\geq$ 30%.

Keywords: convolutional neural network, morphological classification, galaxy merger rate

1. INTRODUCTION

The purpose of this lab is to build a model that can morphologically classify images of galaxies. Ideally, the model should be able to perform all morphological identification that a human can, albeit at a much faster rate. I will be using galaxy images from the Sloan Digital Sky Survey (SDSS); see York et al. (2000), with training classification data from the Galaxy Zoo project; see Lintott et al. (2008), Willett et al. (2013), and Masters & Galaxy Zoo Team (2020). Throughout the lab, I reference Willett et al. (2013) for their morphological classification scheme to generate data. Krizhevsky et al. (2020) goes into details about the use of convolution neural networks in image classifications.

2. BACKGROUND

Galaxy Zoo is a galaxy classification project that enlist the help of volunteers to answer a series of questions about the morphology of a galaxy based on images from SDSS data release 7. There are a total of 304,122 galaxies in Galaxy Zoo 2 (GZ2). Cuts were applied to the DR7 imaging survey to include the closest, brightest, and largest system with clear morphological features that can be identified and classified Willett et al. (2013). Each image in GZ2 is 424×424 pixels scaled to 42.4 arcseconds per pixel. The classification scheme in GZ2 is a decision trees of 11 classification tasks or questions, with a total of 37 possible responses or labels. This is a change from GZ1, where there was no implemented debiasing and decision tree. Images are shown to classifiers at random for most of the duration of GZ2. However, to accurately characterize the likelihood of classification, images with low numbers of classifications were shown at a higher rate toward the end of GZ2. The project has over 4×10^7 classifications, with each classifications being presented as probabilities of a galaxy containing a morphological feature. There are biases that need to be addressed when it comes to galaxy classification, especially with image classification. An example is classification bias, or "the change in observed morphology fractions as a function of redshift *independent of any true evolution in galaxy properties*". This bias stems from the fact that faraway galaxies on average are smaller and dimmer than nearby galaxies, so their finer features are more difficult to classify. Correction for classification bias involves adjusting the vote fractions based on the galaxies' *physical* rather than*observed* parameters. Another bias that arises from redshift is the angular bias for high redshift galaxies. These faraway galaxies will appear to have a small angular separation even if they are not nearby neighbors of each other, causing them to be misclassified as mergers. Additionally, the influence of unreliable classifications is reduced through an iterative weighting scheme.

An important application of morphological classification of galaxies is in the study of the evolution of galaxies Conselice (2014). Morphological classification also contribute to the quantification of galaxy merger rate, which enables modification to galaxy formation models in order to bring them closer in agreement to the observed distribution of galaxy morphologies Rodriguez-Gomez et al. (2015).

3. INSPECTING TRAINING SET IMAGES AND CLASSIFICATION LABELS

Corresponding author: Hien Nguyen
hien24199@berkeley.edu

Galaxy images used as testing and training sets, and the training set classifications were downloaded from Professor Dan Weisz's Box. The training and testing set images were downloaded as `.tar` files and opened using `tarfile.open()`. All images in the files were then extracted with python's built in `extractall()`. There are a total of 61,578 images in the training set.

Figure 1 shows 25 random RGB images of galaxies from the training set. The dimensions of each image is 424 pixels × 424 pixels.

3.1. *Distribution of Training Labels*

Normalized histograms for each of the 37 labels from the training set are plotted in figure 2. Descriptive labels are shown in table 1. Morphological features that are easily identifiable have multi-modal distribution in their labels, i.e. there are multiple peaks in the distribution. In this case, these multi-modal labels are: smoothness, features or disk, and disk not viewed head-on. The rest of the features are skewed, meaning their distributions have long tails due to the presence of outliers far from the mean. The skewed distributions suggest that these features are not as easy to classify.

3.2. *Identifying Prototypes of Training Labels*

The prototype of a label is an image for which the label has the highest value. Identification of prototype lets us visually determine which labels are easy to classify and which are subtle and more likely to cause problems in the classification model. I used `numpy.argmax()` to obtain the index of the largest label value. Then I pulled out galaxy IDs that correspond to these indices and plot their images, as shown in figure 3.

3.3. *Classification Label Correlation Matrices*

The correlation between labels i and j can be defined as

$$\rho_{ij} = \frac{\langle ij \rangle - \langle i \rangle \langle j \rangle}{\sigma_i \sigma_j} \tag{1}$$

where

$$\sigma_i \equiv \sqrt{\langle i^2 \rangle - \langle i \rangle^2} \tag{2}$$

$$\sigma_j \equiv \sqrt{\langle j^2 \rangle - \langle j \rangle^2}\,. \tag{3}$$

The correlation matrix $\boldsymbol{\rho}$ is then a 37 × 37 matrix. A heat map of the correlation matrix is shown in figure 4. Some labels are expected to have no correlation. For example, a smooth and featureless galaxy (Class 1.1, see table 1) cannot have features or a disk (Class 1.2). The matrix elements ρ_{ij} corresponding to this correlation should be zero. However, the heat map in figure 4 indicate that uncorrelated labels have nonzero ρ_{ij} values (color values of off diagonal elements are not zero). The submatrices heatmap in figure 5 reveal more details for each classes. Comparing to table 1, it is evident that classes with binary classifications (i.e. yes or no) exhibit proper correlation heat maps (classes 2, 3, 4, and 6). The heatmaps for the other classes have nonzero values on the off-diagonals, as mentioned previously. This means that the exists uncertainties for certain classifications in the training set, which is to be expected due to biases such as classification bias.

Table 1. A table of descriptive labels for galaxy morphology, referenced from Table 2 in Willett et al. (2013). The classification tasks and their corresponding questions are given to any who voluntary participate in the classification of galaxies by the Galaxy Zoo project; see GalaxyZoo/Classify.

Class	Questions	Responses
01	Is the galaxy smooth and rounded, with no sign of a disk?	Smooth Features or disk Star or artifact
02	Could this be a disk viewed edge-on?	Yes No
03	Is there a sign of a bar feature through the center of the galaxy?	Yes No
04	Is there any sign of a spiral arm pattern?	Yes No
05	How prominent is the central bulge, compared with the rest of the galaxy	No bulge Just noticeable Obvious Dominant
06	Is there anything odd?	Yes No
07	How rounded is it?	Completely round In between Cigar-shaped
08	Is the odd feature a ring, or is the galaxy disturbed, or irregular?	Ring Lens or arc Disturbed Irregular Other Merger Dust lane
09	Does the galaxy have a bulge at its center? If so, what shape?	Rounded Boxy No bulge
10	How tightly wound do the spiral arms appear	Tight Medium Lose
11	How many spiral arms are there?	1 2 3 4 More than 4 Can't tell

3.4. *Normalized Label Distributions of Training and Validation Set Images*

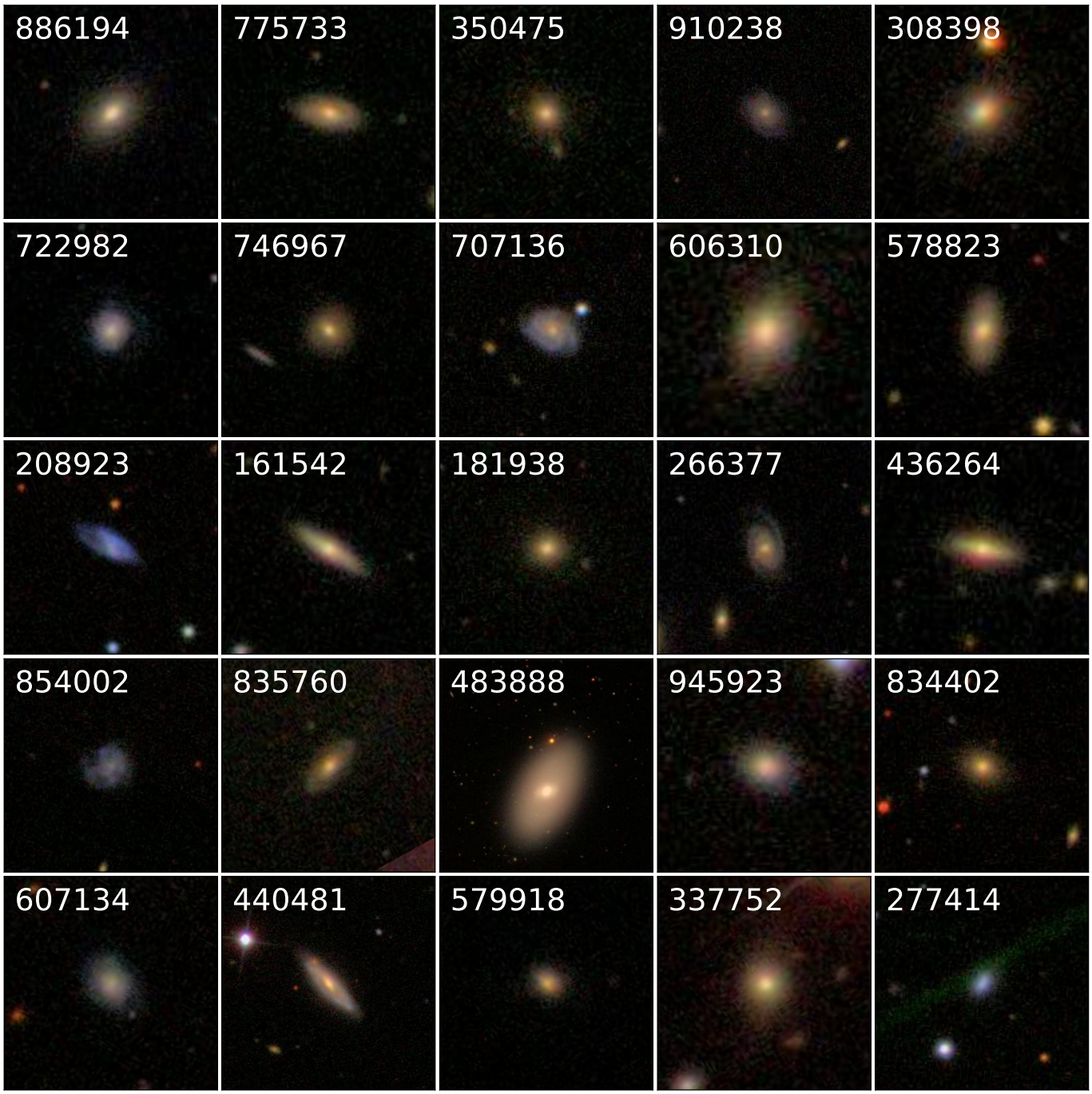

Figure 1. 25 random RGB images of galaxies from the training set. The dimensions of each image is 424 pixels × 424 pixels. Each image is labeled by a galaxy id.

Images in the training set were randomly shuffled and split into a training set and a validation set using `train_test_split` from `sklearn`. The training set consists of 80% of the images or 49,262 images, and the validation set contains 20% of the images or 12,316 images.

Figure 6 compares the normalized distribution of training and validation labels. Overall, there are minimal systematic differences between the two sets of images.

4. REDUCING COMPUTATIONAL & MEMORY COSTS OF CLASSIFICATION

Training of the classification model involves repeated comparison of predictions for each training set image to its known labels. The size of a random image is roughly 540 kilobytes (kb). For 61,578 images in the training

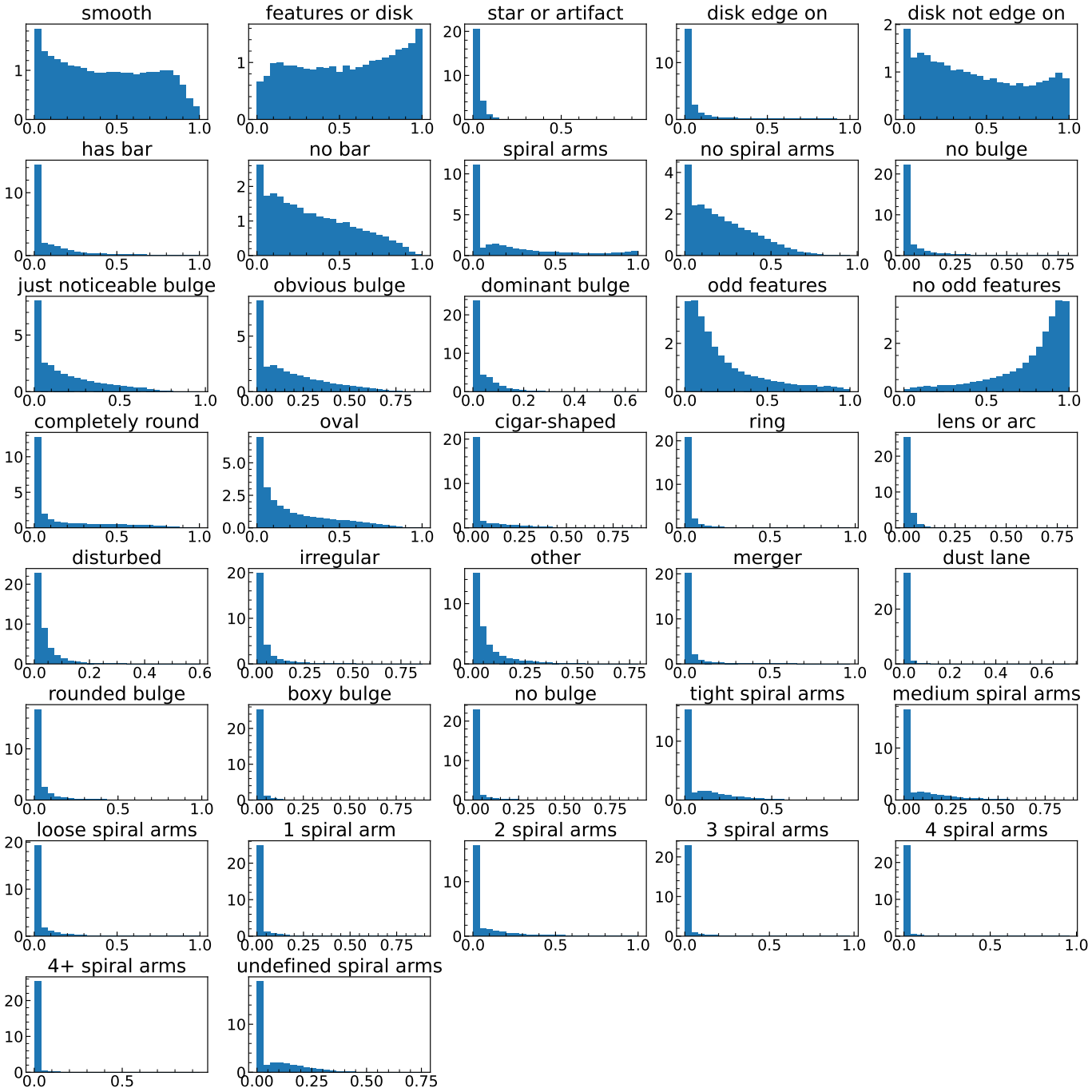

Figure 2. A plot of normalized histograms to show the distribution for each of the 37 training labels. The bin size was set to 25. See table 1 for details about the descriptive labels.

set, that is about 33 gigabytes (gb) of data that needs to be loaded into memory during the training process.

4.1. *Downsizing Training Set Images*

A way to reduce the computational cost of classification is to downsize the images in the training set. Downsizing involved a combination of cropping and rescaling with the Python Imaging Library (PIL). Each image was cropped to remove empty spaces at the edges. The number of pixels was reduced from 424×424 to 64×64. The size of each image was reduced from 540 kb to 12 kb, with the total size of the training set reduced to 0.77 gb. An example image is shown in figure 7 for galaxy 391898.

4.2. *Reading Images in Batches with Generator Functions*

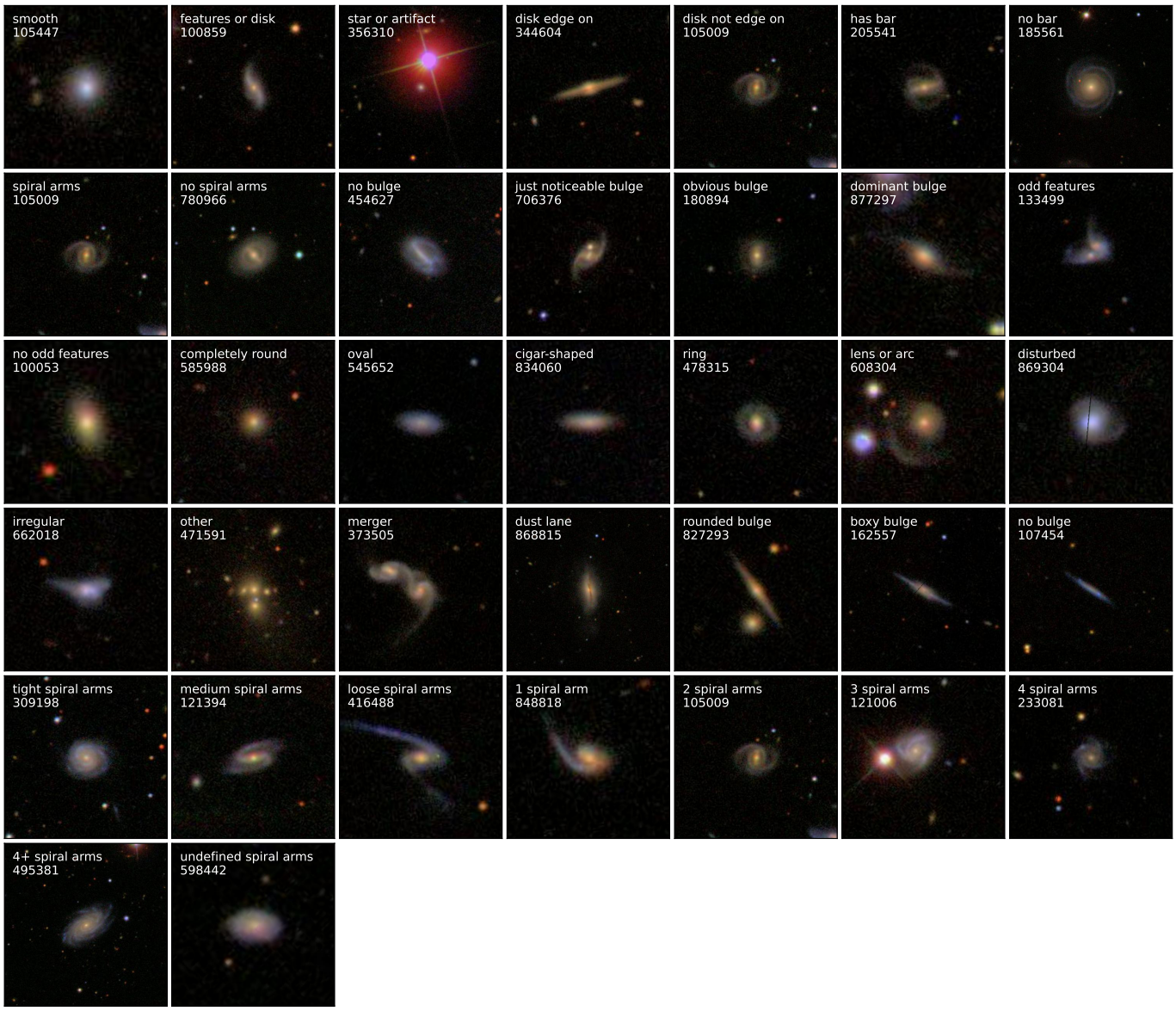

Figure 3. A plot of prototypes or galaxy images that best represent the morphology classification for the 37 labels. Each image has their corresponding descriptive label (see table 1) and galaxy id shown.

To not blow up my computer by overloading the amount of memory available, training set images are read in batches using generator functions. Unlike regular functions with return statements, generator functions with yield statements do not save objects to memory. A batch was defined to have 128 images. Additionally, generator functions were used to save downsized training set and testing set images to further increase efficiency during read in. Note that the training set was split into training and validation sets prior to downsizing.

5. CONVOLUTIONAL NEURAL NETWORK

Convolutional neural networks (CNNs) operate by sliding square filters or kernels over an image grid, learning features and patterns through trainable weights. These weights are updated during training via *backpropagation*, where the network's output is compared to the true labels, and errors are propagated through the layers to minimize the RMSE loss function.

I used the Adaptive Moment Estimation (Adam) optimization algorithm, which employs the method of *stochastic gradient descent*. Adam introduces noise to the parameter space during optimization, which can help prevent the model from getting stuck in local minima and expedite convergence. Additionally, it dynamically adjusts the learning rate for each trainable weight, aiding in efficient weight updates.

The batch size determines the number of images passed through the network before weight updates oc-

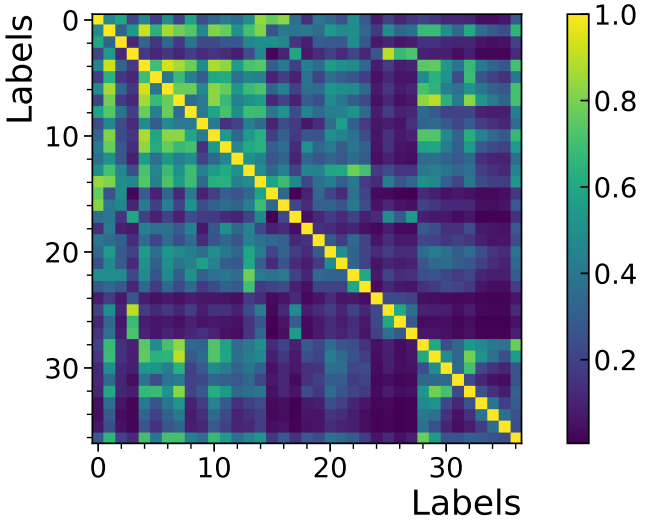

Figure 4. A heat map of the correlation matrix elements ρ_{ij}. The color values denote the strength of correlation.

cur. Larger batch sizes generally improve the accuracy of gradient estimation but at the cost of increased computation time. Conversely, smaller batch sizes reduce computation time but may sacrifice performance. With the models used in this lab, a batch size of 128 was utilized.

To enhance the training process, various callback functions were incorporated. The ReduceLROnPlateau function dynamically adjusts the learning rate if the validation loss fails to improve over a defined number of epochs. The CSVLogger callback records the model's loss and accuracy history during training, facilitating performance evaluation. Finally, an early stopping mechanism halts training when the validation loss ceases to improve within a specified number of epochs.

Three CNN models were used in the lab: a custom CNN model, a ResNet50 preset, and another ResNet50 with custom optimization architecture. All models were implemented using Tensorflow Keras.

5.1. *Simple Mean Label Model*

A simple point of reference for the performance of the neural network models is a model that compares the mean value of a label to an image's true label. The loss function used for this simple model and the neural net models is the *root mean squared error*, defined as

$$L_{\rm RMSE} \equiv \sqrt{\left\langle (\ell_{\rm true} - \ell_{\rm pred})^2 \right\rangle}$$
$$= \sqrt{\frac{1}{N_{\rm galaxies} N_{\rm labels}} \sum_{i}^{N_{\rm galaxies}} \sum_{j}^{N_{\rm labels}} (\ell_{{\rm true},ij} - \ell_{{\rm pred},ij})^2}$$

where $\ell_{\rm pred}$ and $\ell_{\rm true}$ are tensors of the predicted and true labels for each image. The RMSE losses for the training and validation sets with this simple model are both $\tilde{0}.16$. Ideally, the neural net models should perform much better than this with RMSE loss < 0.16. Note that loss values from Keras are Mean Squared Error (MSE) values, so it is expected for the models to have MSE values < 0.0256.

5.2. *Custom CNN*

The custom CNN architecture comprises three blocks of convolutional layers. Each block consists of convolutional layers with 32, 64, and 128 filters or kernels, respectively. The kernels in the first two layers are 3×3 grids, resulting in 9 trainable weights per kernel. Given three color channels (RGB), each of the first two layers possesses $32 \times 9 \times 3 = 896$ trainable parameters. In the third layer, kernels are 6×6 grids, with 36 trainable weights.

To address covariate drift and prevent overfitting, batch normalization is applied in the first convolutional layer. This normalization process ensures that outputs are scaled to fall within the range of -1 and 1. Subsequently, the outputs pass through a rectified linear unit (ReLU) activation function, ensuring only positive values are returned. Following each of the first two convolutional layers is a MaxPooling2D layer, which downsamples the output by taking the maximum value over a 2×22×2 window or pool size, reducing computational cost and training time. Additionally, each block concludes with a dropout layer, deactivating 20% of neurons to mitigate overfitting.

After the three convolution blocks, three fully connected dense layers are employed, with output shapes of 128, 64, and 37, respectively. Outputs from the convolutional blocks are flattened to reduce dimensions before being passed to the dense layers. The first dense layer consists of 128 output neurons, and the second has 64. Due to the significant increase in trainable parameters with dense layers, dropout layers are reintroduced after each to deactivate 20% of neurons. The final dense layer maps the preceding layer's outputs to the 37 labels. A sigmoid activation function is applied at the end of the dense block to confine the final values between 0 and 1, representing the probability of each label.

The model was set to train for 100 epochs with 200 steps per epoch. The earlystop callback function monitors the validation accuracy with a minimum delta of 0.001 and patience of 20 epochs. The reduce learning rate scheduler also monitors validation accuracy. It reduces the learning rate by a factor of 0.5 when the validation accuracy stagnates. The patience is set to be 6 epochs, and the minimum learning rate is set to 10^{-13}. For a summary of the custom CNN model architecture, see table 2.

The total number of parameters in this custom CNN model is roughly 1.6 million, with only 64 untrainable parameters as shown in table 2. Since the training set images were downscaled to $64 \times 64 = 4096$ pixels, for a

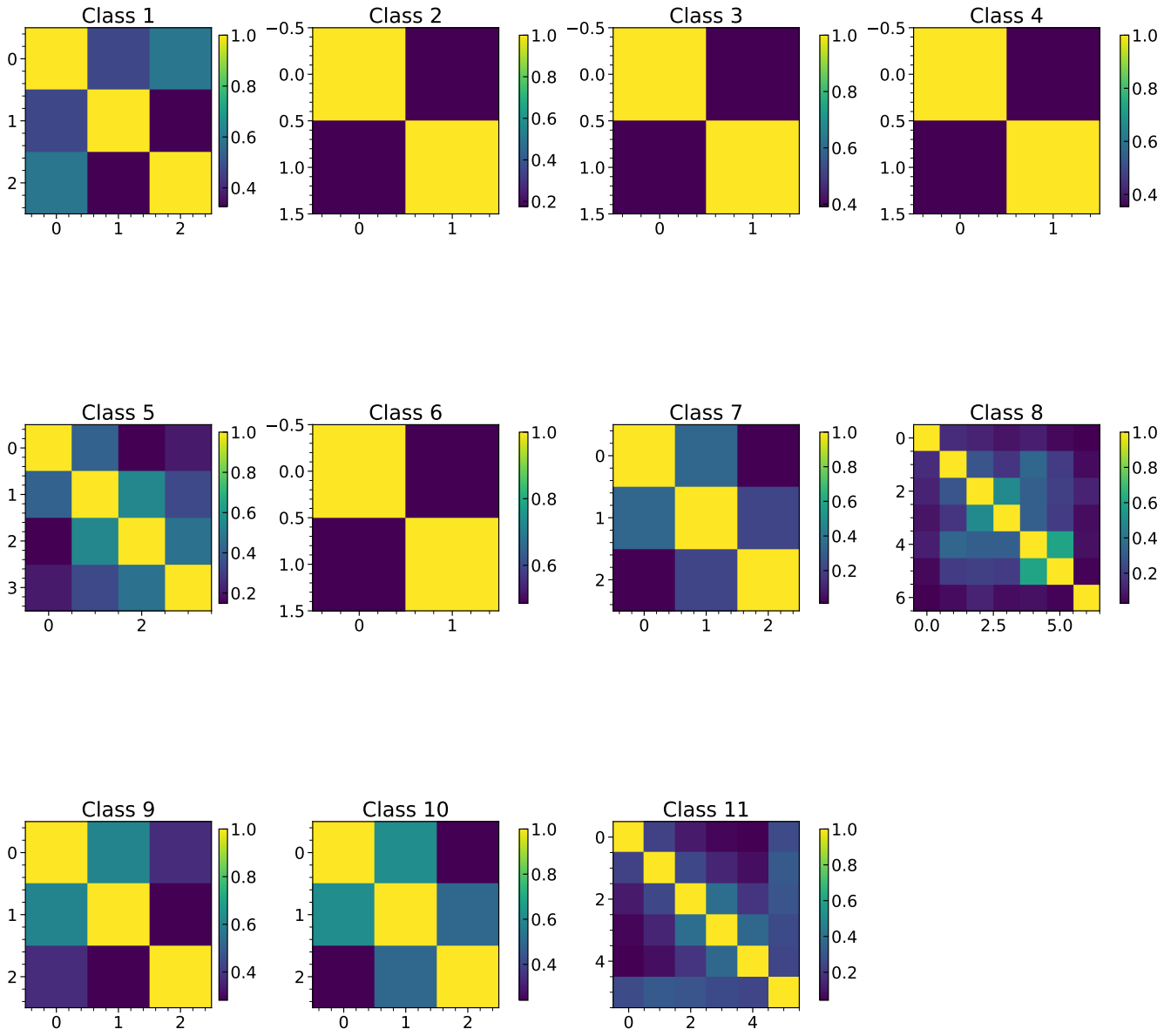

Figure 5. Heat maps of the correlation submatrices, one for each class (see table 1).

total of 252,223,488 pixels in the entire training set. The number of free parameter is thus less than 1% ($\tilde{0}.6\%$) of the number of pixels. This indicates that the model is capable of learning about important features in the training set images with a relatively small number of free parameters compared to the total information contained in the images. At the same time, there is a concern that a small relative number of free parameters limit the model's capacity to properly learn to predict labels from the training set.

5.3. *ResNet50 Preset*

The Residual Network (ResNet) is a specialized CNN tailored for image classification. In this implementation, the preset model ResNet50 from Keras is utilized. As implied by its name, ResNet50 comprises 50 layers, detailed extensively in He et al. (2015). ResNet addresses the vanishing gradient issue encountered when augmenting conventional CNNs with more layers. As additional layers are added, the backpropagated errors diminish to

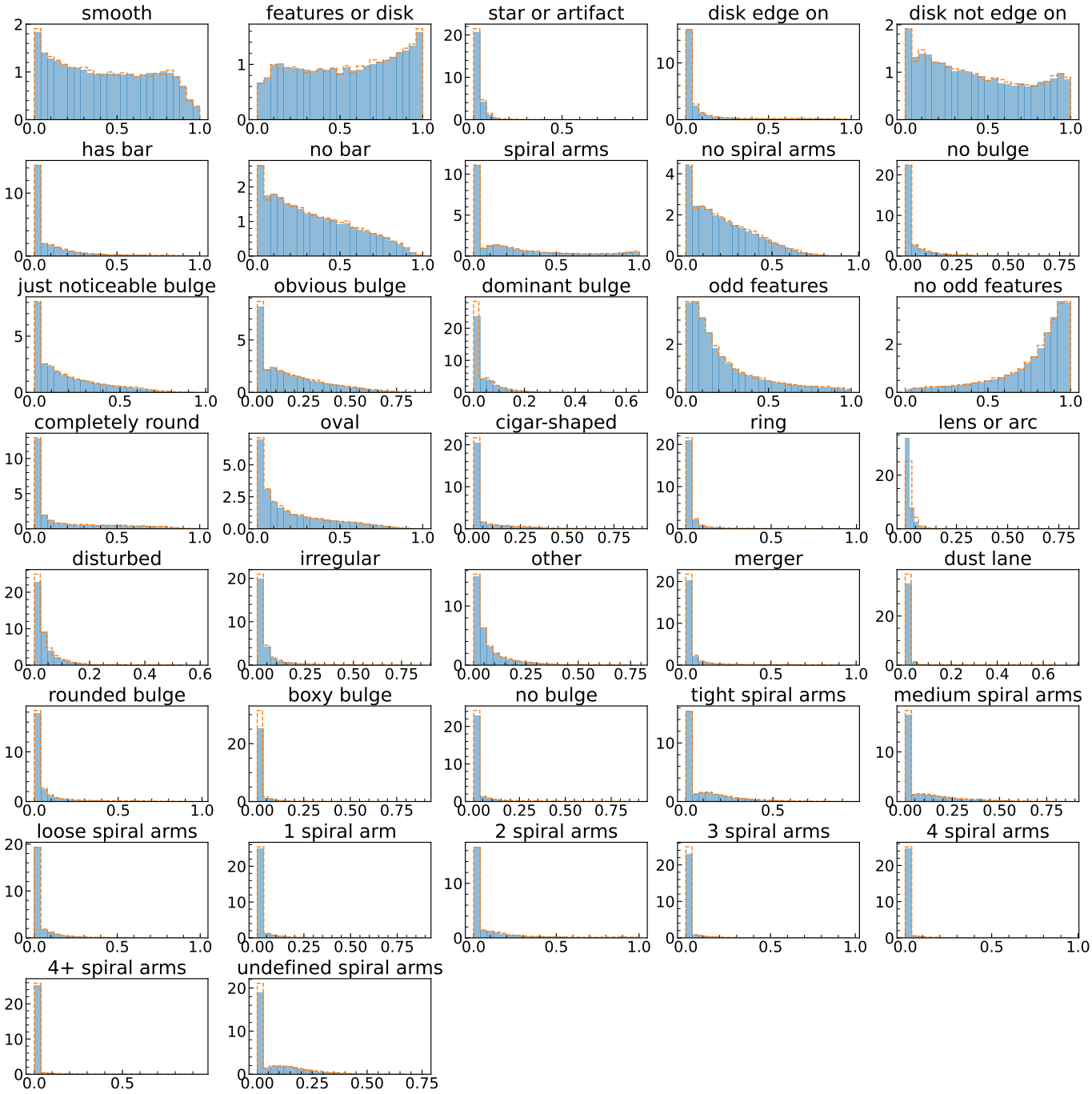

Figure 6. Normalized distributions of labels from the training and validation sets of galaxy images. The solid light blue histograms are the training set labels and the dashed orange line histograms are the validation set labels.

the extent that trainable weights can no longer be updated.

ResNet incorporates skip connections within blocks, facilitating direct connections between input and output. These skip connections, manifested as identity mappings, enable the input to traverse directly to the output alongside the convolutional layer mappings. Consequently, the input is compared with the output, stabilizing the error gradient and ensuring continuous model convergence, thus averting premature termination of the training process.

The ResNet50 model has been pre-trained on a ImageNet dataset comprising 14 million images of everyday objects. Pre-trained weight from this previous training will be applied to our galaxy images. The training process for image classification models, regardless of the specific task, follows a similar trajectory, given the universality of feature and shape detection processes.

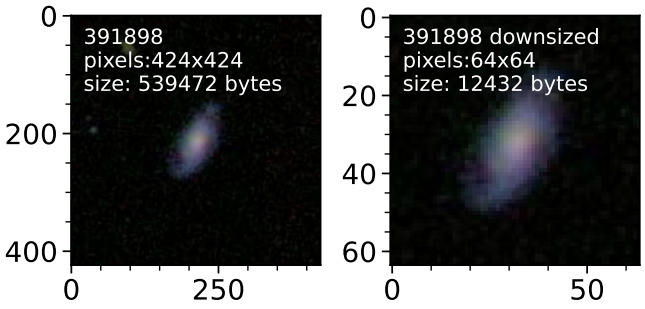

Figure 7. An example of a downsized training set image for galaxy 391898.

Table 2. A summary of the custom CNN model, detailing the layer type, output shape, and number of parameters in each of the 6 blocks. There are a total of 1,652,453 parameters, of which 1,652,389 are trainable parameters with only 64 non-trainable parameters.

Layer type	Output Shape	# Parameters
Conv2D	(None, 62, 62, 32)	896
BatchNormalization	(None, 62, 62, 32)	128
Activation (relu)	(None, 62, 62, 32)	0
MaxPooling2D	(None, 31, 31, 32)	0
Dropout	(None, 31, 31, 32)	0
Conv2D	(None, 29, 29, 64)	18496
Activation (relu)	(None, 29, 29, 64)	0
MaxPooling2D	(None, 14, 14, 64)	0
Dropout	(None, 14, 14, 64)	0
Conv2D	(None, 9, 9, 128)	295040
Activation (relu)	(None, 9, 9, 128)	0
Dropout	(None, 9, 9, 128)	0
Flatten	(None, 10368)	0
Dense	(None, 128)	1327232
Dropout	(None, 128	0
Dense	(None, 64)	8256
Dropout	(None, 64)	0
Dense	(None, 37)	2405
Activation (sigmoid)	(None, 37)	0

Table 3. A summary of the default ResNet50 preset in Keras, with additional layers governing outputs. The total number of paramters is 23,890,853. There are 23,837,733 trainable parameters, and 53,120 non-trainable parameters.

Layer type	Output shape	# Parameters
resnet50	(None, 2, 2, 2048)	23587712
Flatten	(None, 8192)	0
Dense	(None, 37)	303141
Activation (sigmoid)	(None, 37)	0

The outputs of ResNet50 were flattened, as with the custom CNN model, before being passed through a single dense layer that maps the output to the 37 labels. The sigmoid activation function was once again implemented to ensure that the output is represented as a probability. Everything else about the architecture is the same as described in section 5.2. Additionally, the learning rate of the optimizer was adjusted with the `ReduceLROnPlateau` scheduler every time the validation loss plateaus. This implementation of ResNet50 has 23 million trainable parameters, as discussed in table 3. In the following section I attempt to resolve overfitting (see section 5.5) with the implementation of data augmentation and a re-weight process that accounts for label dependencies.

5.4. *Optimized ResNet50 Preset*

To further optimize ResNet50 in addition to the `ReduceLROnPlateau` scheduler, I implement a data augmentation scheme to arbitrarily increase the size of the training set. I also utilized a re-weighing scheme that takes into account the dependencies between labels.

5.4.1. *Data Augmentation*

Essentially, the size of the training set images was artificially increased by exploiting rotational and reflection symmetry. The labels of a galaxy should not change whether its image is rotated or mirrored. Therefore, the read in process was modified to rotate an image by a random angle $\theta \in [0, 360]$ deg before the image is downized (cropped and rescaled). With this data augmentation, the ResNet model should not see the same image twice even after many epochs of training.

5.4.2. *Labels Dependencies*

As discussed in more details in Lintott et al. (2008), Masters & Galaxy Zoo Team (2020), and Willett et al. (2013), not all 37 labels are not independent. The probability value of a label depends on the probability assigned to its parent class (see table 1). Consider how a volunteer would go through the classification process of Galaxy Zoo. If a certain percentage of users identify a galaxy to have spiral arms (class 4.1), then the probabilities of labels for the number of spiral arms in class 11 would be dependent on the initial percentage of users that identify the galaxy to have spiral arms. The purpose of this weighing scheme is to ensure that a high-confidence classification farther down the decision tree requires a high-confidence at the earlier corresponding node(s) of the tree.

To include this dependency information in the model, I used Keras' `get_custom_objects` module to define a custom activation function that takes in the outputs and returned output probabilities weighted by the label dependencies. The 37 labels were grouped into parent classes with class slices (reference table 1):

```
class_slices = [
    slice(0,3), slice(3,5), slice(5,7),
    slice(7,9), slice(9,13), slice(13,15),
    slice(15,18), slice(18,25), slice(25,28),
    slice(28,31), slice(31,37)
]
```

A normalization mask was implemented to ensure each row of the output probabilities sums to 1:

```
# initialize normalization mask
norm_mask = np.zeros(
    (37,37),
    dtype='float32'
)
# small value to address division by zero
epsilon=1e-12

# fill normalization mask with values
for s in class_slices:
    norm_mask[s,s] = 1.0

# normalize output probabilities
normalization = tf.matmul(
    outputs,
    tf.convert_to_tensor(norm_mask)
) + epsilon

outputs_norm = outputs / normalization
```

The sets of labels and their corresponding dependencies are detailed in table 4. For example, class slice(3,5) include indices 3 and 4, corresponding to classes 2.1 and 2.2 with labels being 'observed edge-on disk' and 'no edge-on disk'. These subclasses/labels depend on class 1.2 'features or disk' earlier in the classification tree, so they need to be rescaled with probability value from this 'parent' class. A weight tensor that encode label dependencies was constructed as follows

```
# class 1 & 6 independent
one = tf.divide(
    outputs_norm[:,1],
    outputs_norm[:,1]
)

# rescaling classes
c1 = output_norm[:,1] # class 1.2
c4 = output_norm[:,4] # class 2.2
c0 = output_norm[:,0] # class 1.1
c13 = output_norm[:,13] # class 6.1
c3 = output_norm[:,3] # class 2.1
c7 = output_norm[:,7] # class 4.1

# rescaling tensor
weighted_tensor = tf.stack(
    [
        one, one, one, #class1
        c1, c1, #class2
        c4, c4, #class3
        c4, c4, #class4
        c4, c4, c4, c4, #class5
        one, one, #class6
        c0, c0 , c0, #class7
        c13, c13, c13, c13, #class8
        c13, c13, c13,
        c3, c3, c3, #class9
        c7, c7, c7, #class10
        c7, c7, c7, c7, c7, c7 #class11
    ],
    axis=1
)
```

Classes 1 and 6 are not dependent on other labels so their scale factor is 1, i.e. their output probabilities do not need to be re-weighted. Class 2 is dependent on class 1.2. Classes 3, 4, and 5 are dependent on class 2.2, and so on. The normalized outputs are then multiplied with the weight tensor to get new output values that account for label dependencies. The re-weighting function described above is used as an activation function that takes in probability values from a sigmoid activation function at the end of a dense block.

In addition to the data augmentation and label dependencies re-weighting described above, the second ResNet50 model also consists of 2 additional dense layers alongside relu activation functions, and 2 dropout layers that switch of 20% of the neurons. A summary of the optimized ResNet50 model is tabulated in table 5.

5.5. *CNN Models Performance Comparison*

Observe from figures 8, 9, and table 6 that the custom CNN model is the best performing out of the 3 models, achieving an MSE loss of 0.012 (RMSE 0.11). This is better than the MSE and RMSE values from the simple mean label model in section 5.1, which is good. The training and validation loss are also very close to each other (first panel of figure 8), with minimal overfitting. However, further improvements to the model could very well bring the RMSE loss value down below 0.10.

The second panel of figure 8 compares the training and validation loss of the default ResNet50. The earlystop callback function ended the training process around 30 epochs, unlike the custom CNN model which continued training for 60 epochs in total. Despite this, it is evident with the validation loss curve that ResNet50 had converged in 30 epochs. The average MSE validation loss of 0.011 is an 8% improvement in performance over the custom CNN model, at least within the 30 epochs (see figure 9). The issue is that the ResNet50 model is massively overtraining or overfitting, as shown by the large gap between the training and validation loss curves. The model performed well with training set images, as shown by the continued decrease in training loss, but performed poorly when it came to new data in the validation set.

Table 4. A table outlining the reweight scheme to account for label dependencies. The left 3 columns are for labels that need their output probability to be rescaled based on the probability of labels higher up in the classification tree. The right 3 columns are for the labels being used as rescaling factors, i.e. they are the labels earlier in the classification tree that lower labels depend on. The indices are labelled starting from 0 for 37 labels. Refer to table 1 for the full set of labels and their descriptions.

Classes	Indices	Labels	Rescale class	Label	Index
2.1, 2.2	3, 4	edge-on disk: yes/no	1.2	features or disk	1
3.1, 3.2	5, 6	bar: yes/no	2.2	no edge-on disk	4
4.1, 4.2	7, 8	spiral arms: yes/no			
5.1, 5.2, 5.3, 5.4	9, 10, 11, 12	bulge prominance: no, just noticeable, obvious, dominant			
7.1, 7.2, 7.3	15, 16, 17	roundness: completely, in between, cigar-shaped	1.1	smooth	0
8.1, 8.2, 8.3,	18, 19, 20,	ring, lens or arc, disturbed	6.1	odd	13
8.4, 8.5, 8.6,	21, 22, 23,	irregular, other, merger			
8.7	24	dust lane			
9.1, 9.2, 9.3	25, 26, 27	bulge shape: rounded, boxy, no	2.1	edge-on	3
10.1, 10.2, 10.3	28, 29, 30	arms spacing: tight, medium, loose	4.1	spiral	7
11.1, 11.2, 11.3,	31, 32, 33,	# arms: 1, 2, 3, 4, >4, ?			
11.4, 11.5, 11.6	34, 35, 36				

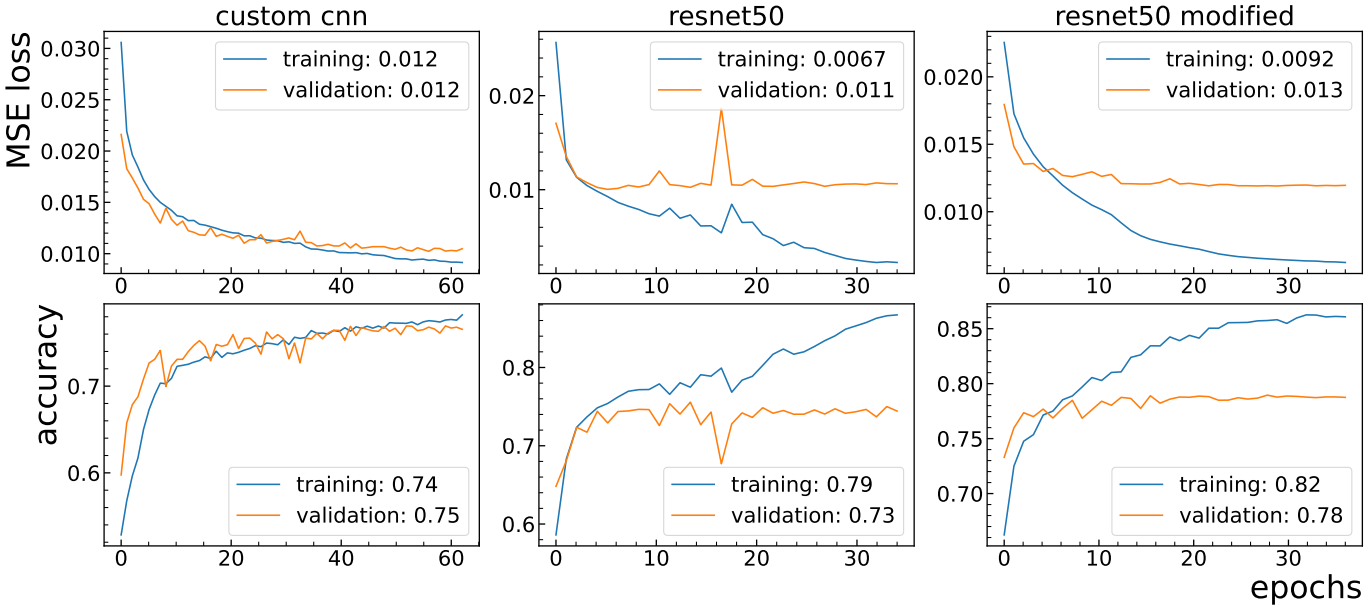

Figure 8. Comparison plots for the accuracy and MSE loss of the custom CNN model, the ResNet50 preset model, and the optimized ResNet50 preset model. See table 6 for details.

Table 5. A summary of the optimized ResNet50 preset in Keras, with additional layers governing outputs. The total number of paramters is 25,250,149. There are 25,197,029 trainable parameters, and 53,120 non-trainable parameters.

Layer type	Output shape	# Parameters
resnet50	(None, 2, 2, 2048)	23587712
Flatten	(None, 8192)	0
Dense	(None, 200)	1638600
Dropout	(None, 200)	0
Dense	(None, 100)	20100
Dropout	(None, 100)	0
Dense	(None, 37)	3737
Activation (sigmoid)	(None, 37)	0
Activation (re-weight)	(None, 37)	0

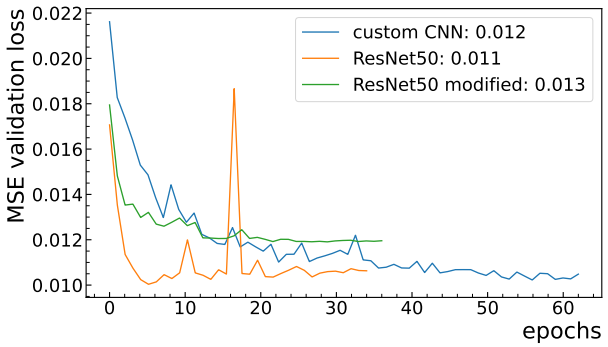

Figure 9. Comparison plots for MSE validation loss between the custom CNN model, the preset ResNet50 model, and the optimized preset ResNet50 model. See table 6 for details.

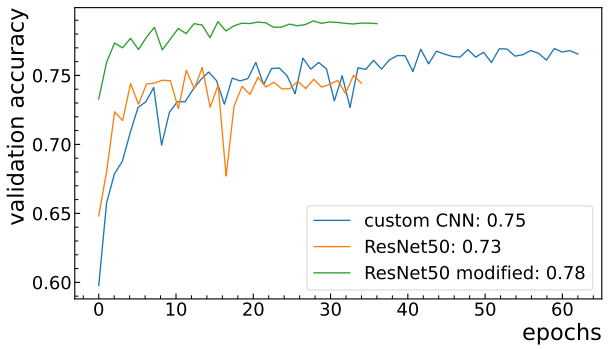

Figure 10. Comparison plots for validation accuracy between the custom CNN model, the preset ResNet50 model, and the optimized preset ResNet50 model. See table 6 for details.

A possible source of overfitting is the large number of trainable parameters, 20 times that of the custom CNN model. This likely caused the ResNet50 model to fixate on irrelevant features that do not characterize the galaxy images.

The optimized ResNet50 model turned out to be the worst performing out of the three when it comes to validation loss, as shown by its average MSE validation loss being the largest, it overtraining more than the default ResNet50, and its validation loss curve being above the other two at convergence. However, this model has the highest validation accuracy, likely due to the additional optimization schemes that were implemented. Consequently, the optimized ResNet50 model should be able to predict galaxy labels from images with high accuracy.

While both ResNet50 models suffer from overtraining, the training loss continuous decrease (see figure 8) also indicate that if the overtraining was resolved both models should continue to train for more than 30 epochs.

6. RESULTS

In this section I use all three models to predict values for all 37 labels in the validation set. I will also compare "true" and "predicted" labels of the top 5 prototypes for a selected set of labels. Finally, I run the models on a second set of testing images to estimate the fraction of images that are galaxy mergers.

6.1. *Label Prediction*

Figures 11, 12, and 13 show the scatter comparison between "true" and "predicted" labels. All three models have dense scatter around 0 or 1, indicating that the models are able to make confident label predictions. Some labels have wide scatter distribution around 0.5, meaning there are relatively large uncertainties for these labels. Linear correlation between predicted and true labels can also be observed, showing that the two are in good agreement.

6.2. *Top 5 Label Prototypes*

For the following 7 labels:

- smooth
- star/artifact
- edge-on disk
- ring
- lens/arc
- 2 spiral arms
- merger

I plotted the images of the 5 objects in the validation set that are the most extreme example of that label. In essence, these objects are the 5 prototypes of that label

Table 6. A table of the average accuracy and MSE loss in the training and validation set for the custom CNN model, the preset ResNet50 model, and the optimized preset ResNet50 model.

Model	training loss	validation loss	training accuracy	validation accuracy
CNN custom	0.012	0.012	0.74	0.75
ResNet50	0.0067	0.011	0.79	0.73
ResNet50 optimized	0.0092	0.013	0.82	0.78

with the highest probability assigned to them. Figure 14 show the prototypes for the true labels in the validation set, while figures 15, 16, and 17 show the prototypes for the predicted labels from each model. From a visual inspection, the models perform as well as the human eye. There are no prototypes that are incorrectly labelled, and all prototypes are good visual representation of their label features.

6.3. *Estimation of Galaxy Merger Fraction*

To estimate the galaxy merger rate at a redshift of $z = 0$, I apply the best performing (lowest validation loss) custom CNN model to the testing set consisting of 79,975 images. All of the galaxies in this data set are approximately around this redshift distance. The custom CNN model trained on 49,262 images from the training set, so this is a typical application where the model is expected to make prediction on a data set much larger than the one it was trained on. A probability cutoff of 30% was applied to the merger classification label. Nine random images with merger label probability greater than 30% are shown in figure 18. Applying this confidence level in the merger probability returned 2110 images, thus yielding a merger fraction of

$$\frac{2110}{79,975} \approx 0.026 \tag{4}$$

which is in good agreement with the observed merger rate in the upper right panel of figure 13 in Lotz et al. (2011). Note that the results in that paper is a rate of number of mergers per unit time, whereas the merger fraction estimated here is a number fraction of images that are mergers.

7. CONCLUSIONS

The CNN models used in this lab were able to converge on average RMSE validation losses ranging from 0.08 to 0.1, which is lower than the RMSE of 0.16 from the simple mean label value model in section 5.1. The models were able to produce confident predictions in galaxy labels and identify images that are good visual representations of their labels (figures 15, 16, 17). Application of the custom CNN to a dataset larger than its training dataset yielded a reasonable merger fraction that is in agreement with the merger rate observed by Lotz et al. (2011). There are still improvements that can be made to the ResNet50 models to resolve the overtraining issue. But for the purpose of this lab, the models performed well.

ACKNOWLEDGEMENTS

Sincerest thanks to Professor Dan Weisz, and graduate student instructors Anna Pusack and Olivia Aspegren for guidance and instruction on the conceptual and technical components throughout this lab. Thank you to Dr. Richard Bonventre for mentoring me in the use of Tensorflow Keras. Thank you to Talia Saarinen, Bradley Arias, Jason Wong, and Andrew Goh for their feedback and support.

Morphological classifications from the Galaxy Zoo project were used in this lab. Galaxy Zoo: Cosmic Dawn is partly supported by the ESCAPE project, which aims to bring together the astronomy, astroparticle and particle physics communities to support open science, according to FAIR (Findable, Accessible, Interoperable and Reuseable) principles.

REFERENCES

Conselice, C. J. 2014, ARA&A, 52, 291, doi: 10.1146/annurev-astro-081913-040037

He, K., Zhang, X., Ren, S., & Sun, J. 2015, arXiv e-prints, arXiv:1512.03385, doi: 10.48550/arXiv.1512.03385

Krizhevsky, A., Sutskever, I., & Hinton, G. E. 2020

Lintott, C. J., Schawinski, K., Slosar, A., et al. 2008, MNRAS, 389, 1179, doi: 10.1111/j.1365-2966.2008.13689.x

Lotz, J. M., Jonsson, P., Cox, T. J., et al. 2011, ApJ, 742, 103, doi: 10.1088/0004-637X/742/2/103

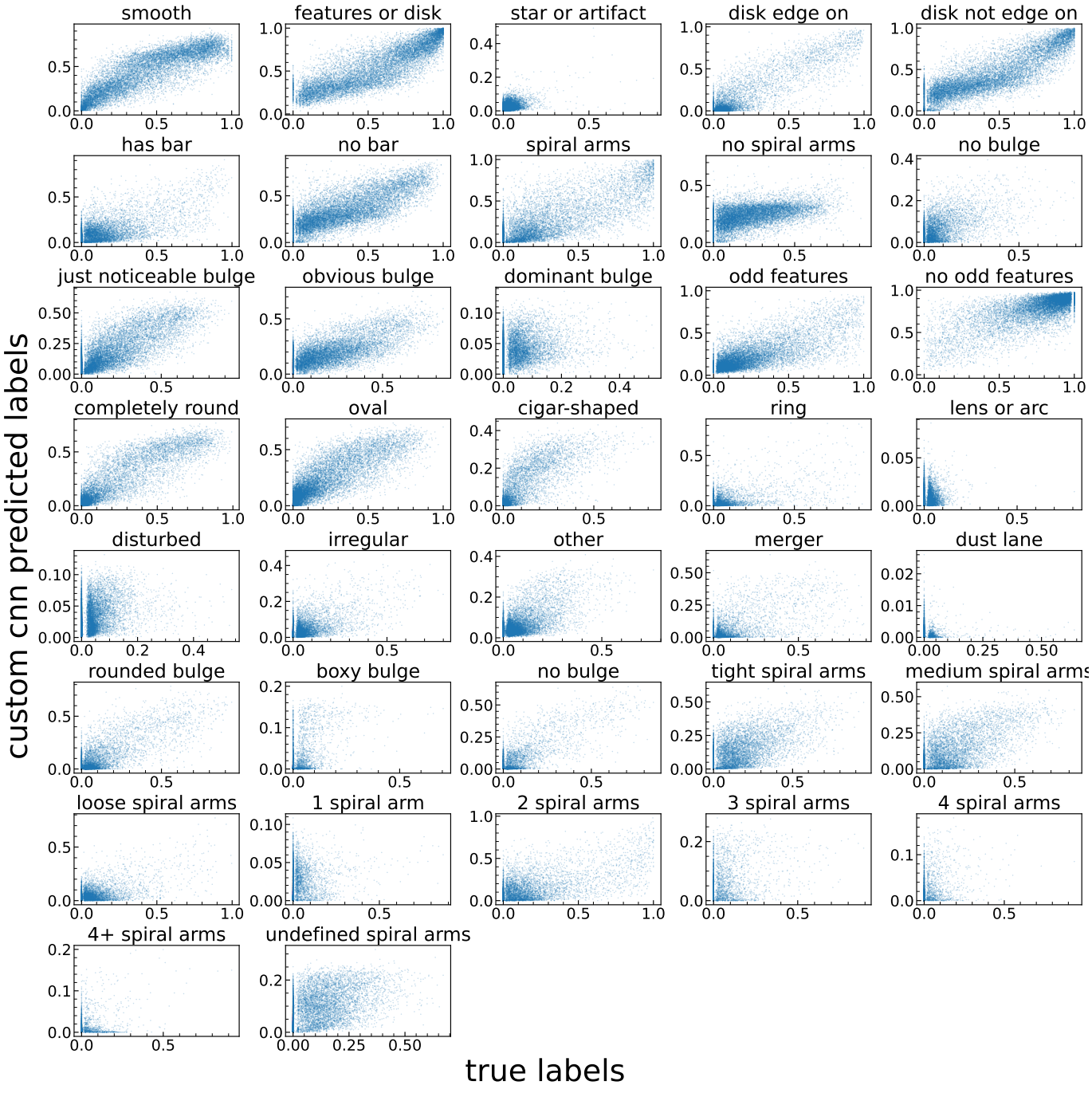

Figure 11. Scatter plots comparing the "true" and "predicted" values of all 37 labels for the validation set, with the custom CNN model.

Masters, K. L., & Galaxy Zoo Team. 2020, in Galactic Dynamics in the Era of Large Surveys, ed. M. Valluri & J. A. Sellwood, Vol. 353, 205–212, doi: 10.1017/S1743921319008615

Rodriguez-Gomez, V., Genel, S., Vogelsberger, M., et al. 2015, MNRAS, 449, 49, doi: 10.1093/mnras/stv264

Willett, K. W., Lintott, C. J., Bamford, S. P., et al. 2013, MNRAS, 435, 2835, doi: 10.1093/mnras/stt1458

York, D. G., Adelman, J., Anderson, John E., J., et al. 2000, AJ, 120, 1579, doi: 10.1086/301513

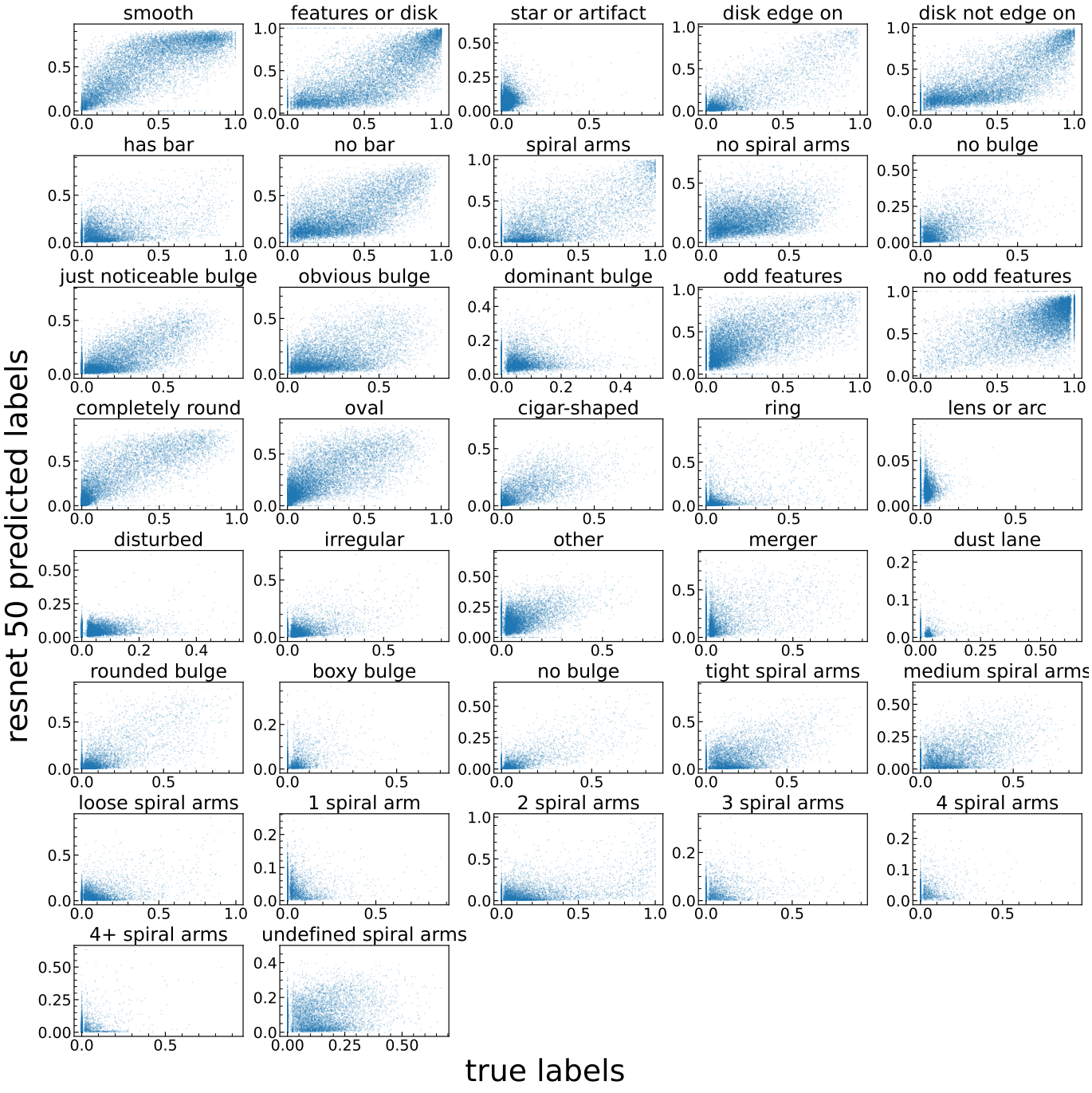

Figure 12. Scatter plots comparing the "true" and "predicted" values of all 37 labels for the validation set, with the default ResNet50 model.

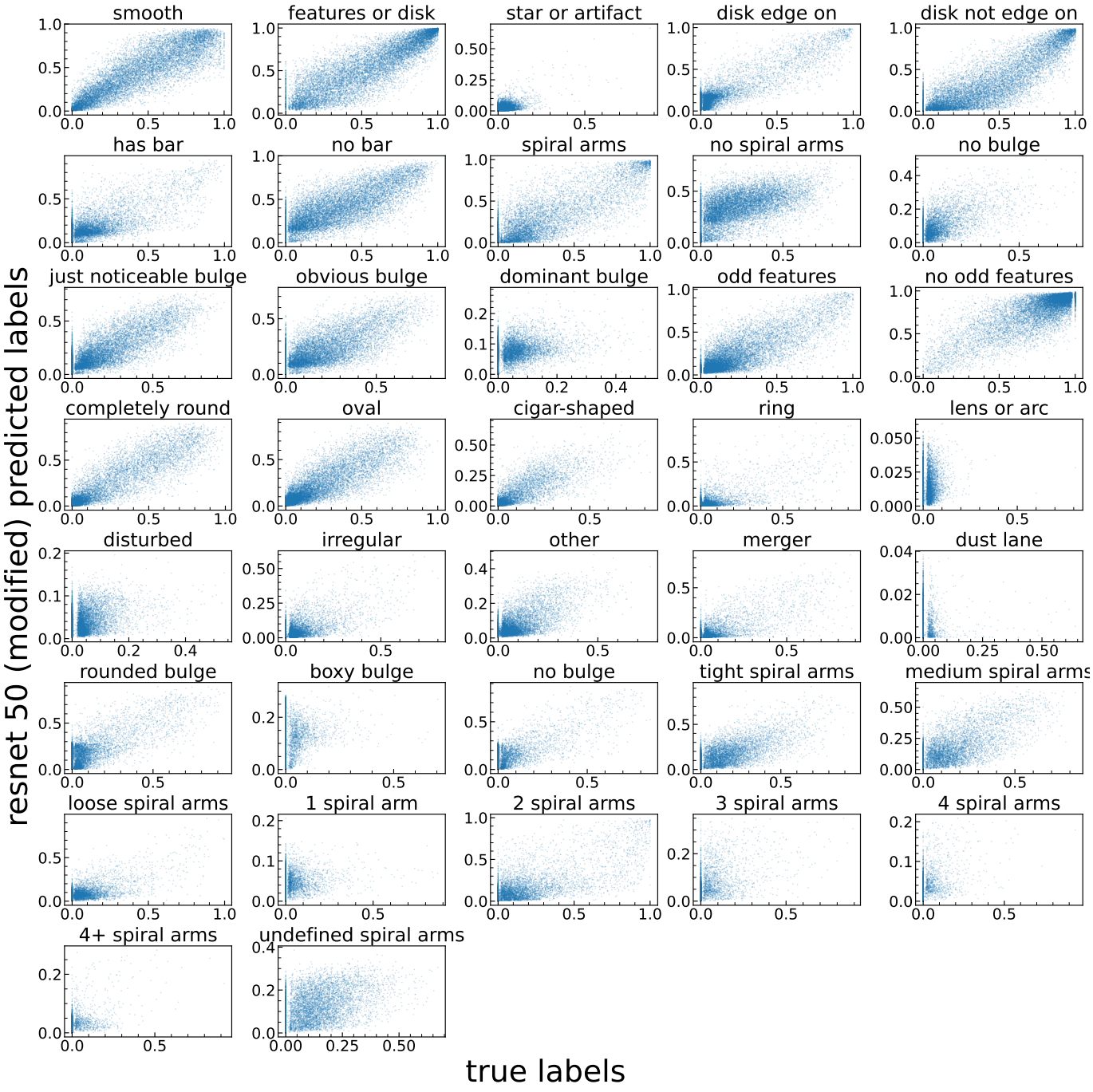

Figure 13. Scatter plots comparing the "true" and "predicted" values of all 37 labels for the validation set, with the ResNet50 model optimized with learning rate reduction, data augmentation, and label dependencies re-weighting.

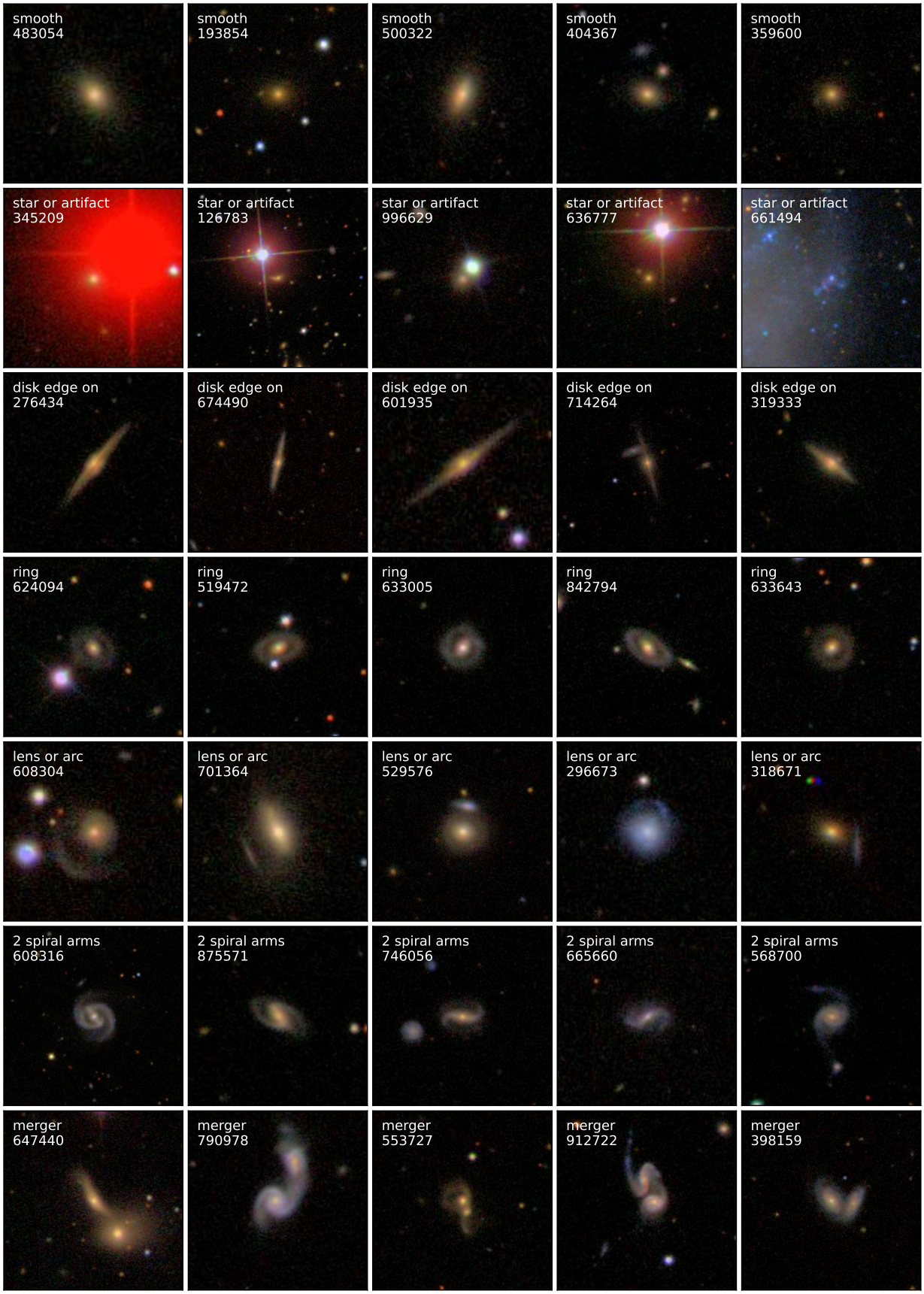

Figure 14. Images of the top 5 label prototypes for the 7 labels of interests. These images are selected from the true labels of the validation set.

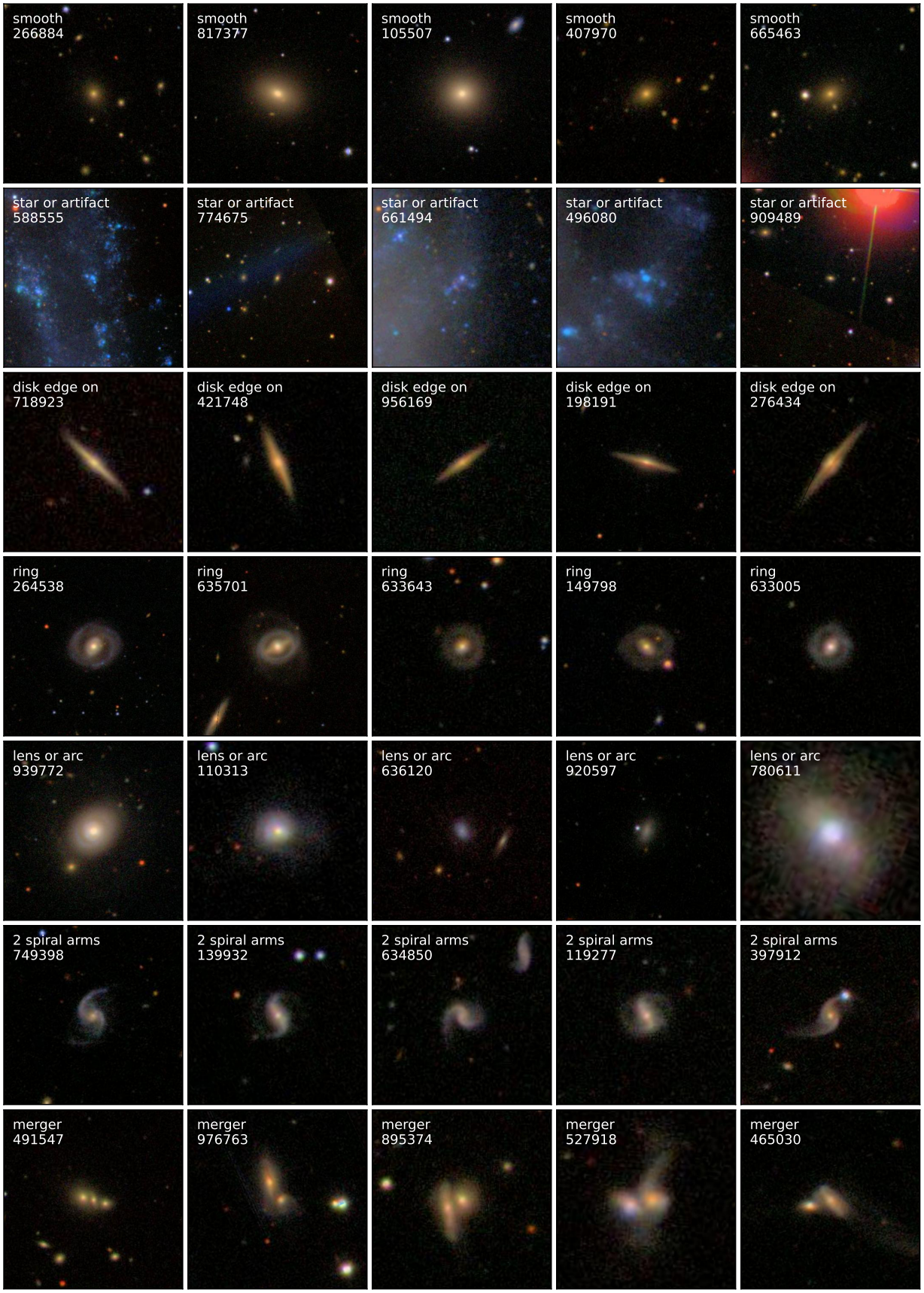

Figure 15. Images of the top 5 label prototypes for the 7 labels of interests. These images are selected from labels predicted by the custom CNN model.

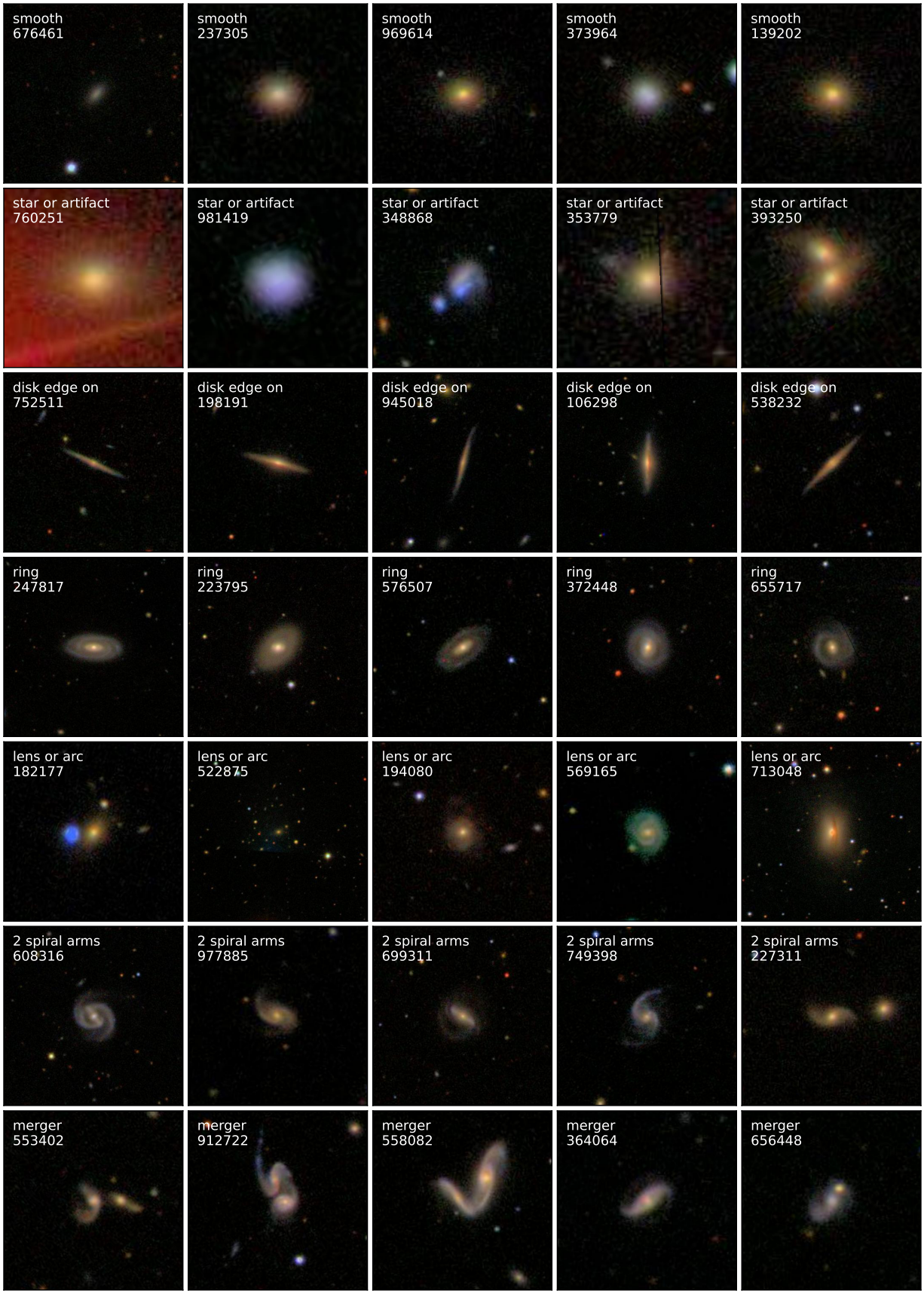

Figure 16. Images of the top 5 label prototypes for the 7 labels of interests. These images are selected from labels predicted by the default ResNet50 model.

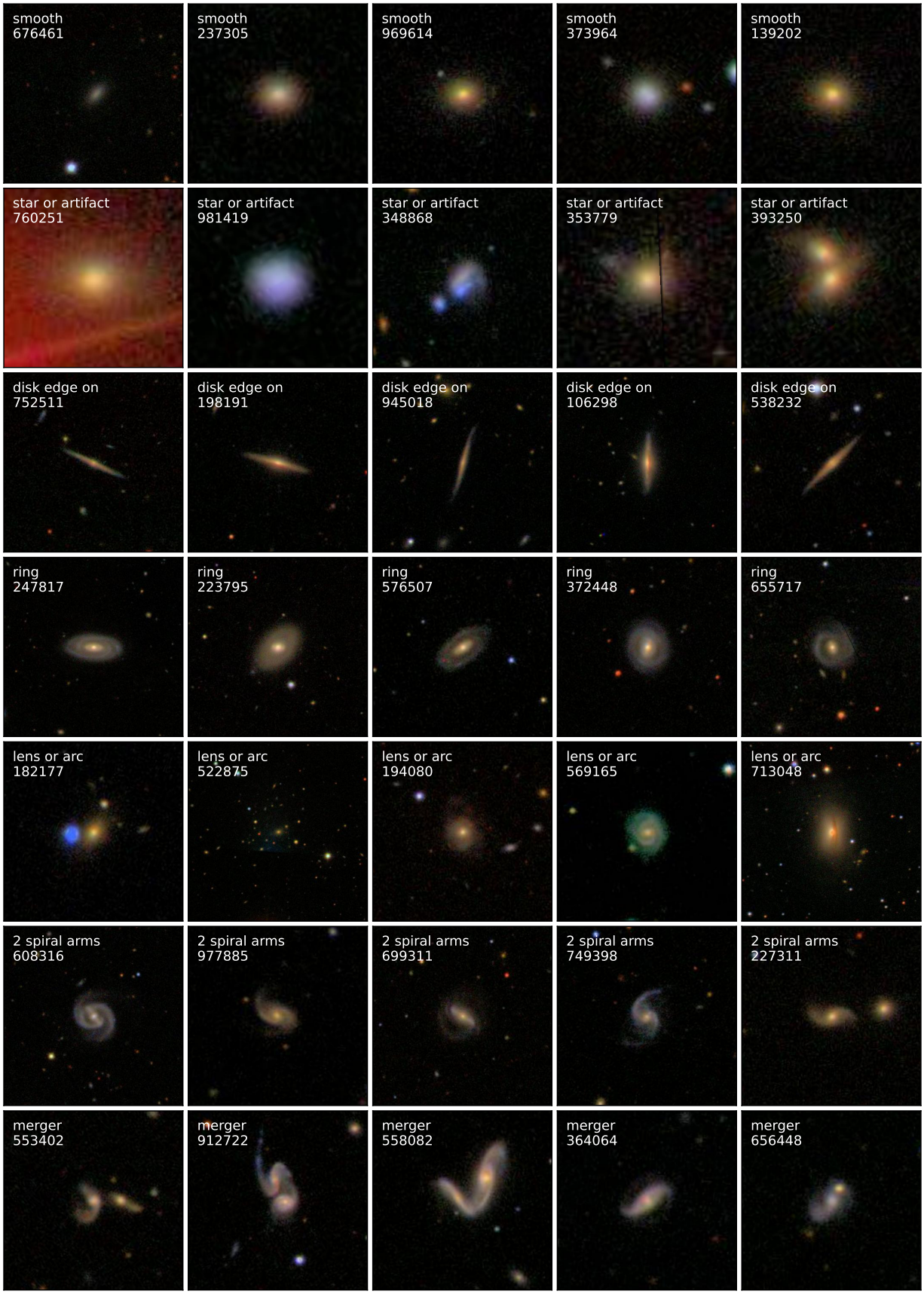

Figure 17. Images of the top 5 label prototypes for the 7 labels of interests. These images are selected from labels predicted by the ResNet50 model optimized with a learning rate scheduler, data augmentation, and label dependencies re-weight.

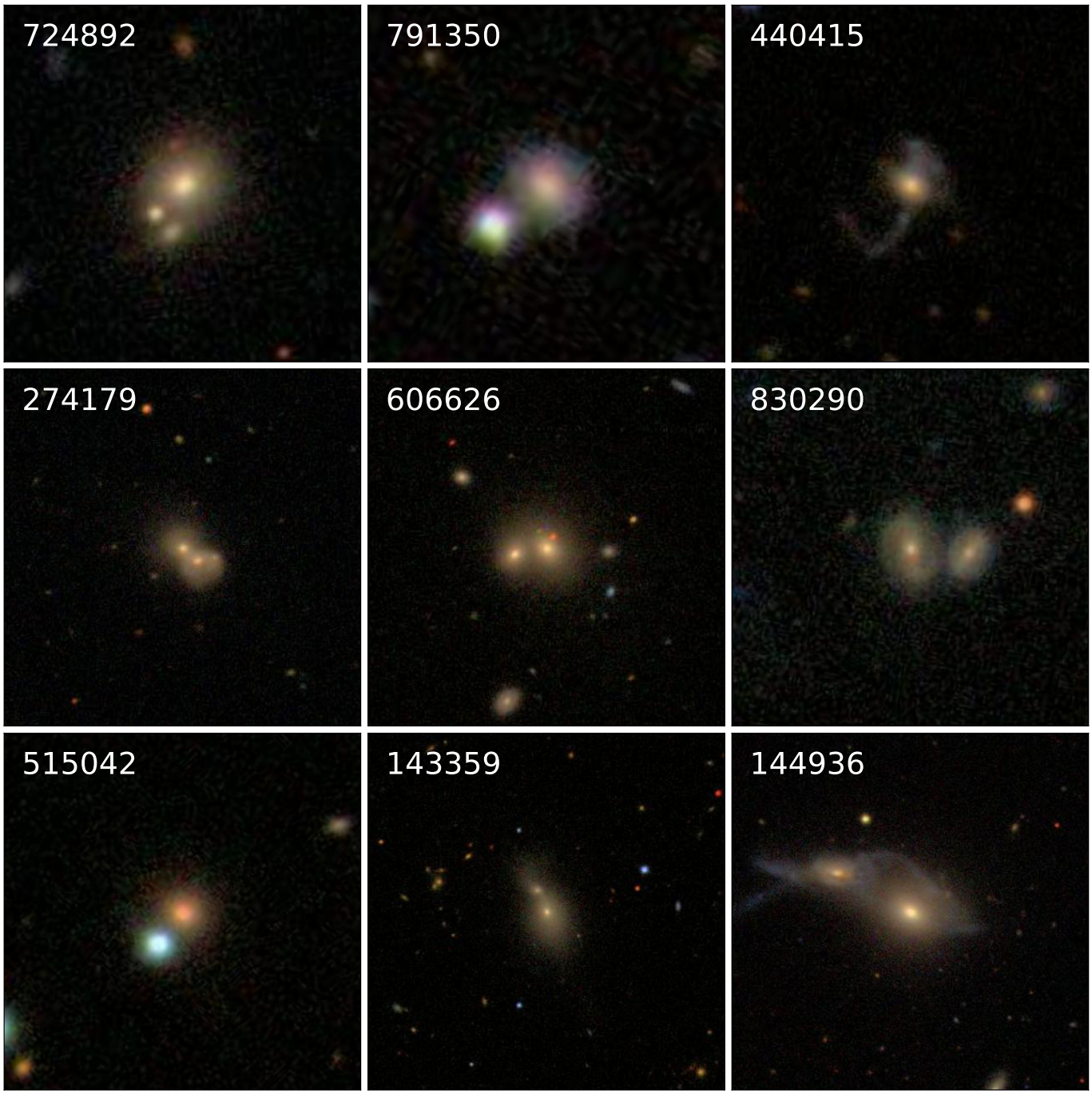

Figure 18. Nine random images of merger with probability values greather than 30%.