Galaxy Type Classification

Jake Pistotnik, Eric Danforth, Violet Li

December 2023

Abstract

This project focuses on creating a classifier designed to categorize images of galaxies into 10 broad classes. The implementation of a robust machine learning model promises to significantly accelerate the classification procedure, ensuring scalability and consistency in labeling. We utilize the Galaxy10 DECals dataset, which has been hand labeled by volunteers. We apply an innovative method of background subtraction to reduce the noise in the images, explore simple feature like color channels and HOG, advanced features such as BOVW and ResNet embeddings, apply PCA and t-SNE to reduce dimensionality, experiment with various models (weighing both accuracy and efficiency), and then analyze the results with a focus on the difficulty of this task and directions for further exploration.

1 Introduction

Our project utilizes the Galaxy10 DECals dataset [1], an advanced collection of astronomical images and their labels. This dataset originates from the Sloan Digital Sky Survey (SDSS), where the initial Galaxy 10 dataset comprised images classified by online volunteers into ten broad categories. The higher quality and resolution images from DECals (Dark Energy Legacy Survey) significantly enhance this dataset. Subsequent refinements and filtering of the classification categories and labels have produced the current set of 17,736 images.

The images are 256x256 pixels and in three-channels - green, red, and near-infrared. The galaxies are centered in the images and the dataset has been vetted to remove artifacts. To facilitate our study within the constraints of time and manageability, we have downscaled the dataset to 3,340 images. This downsizing ensures uniformity

across all ten categories, aligning with the size of the smallest class and making the dataset more practical for our analysis.

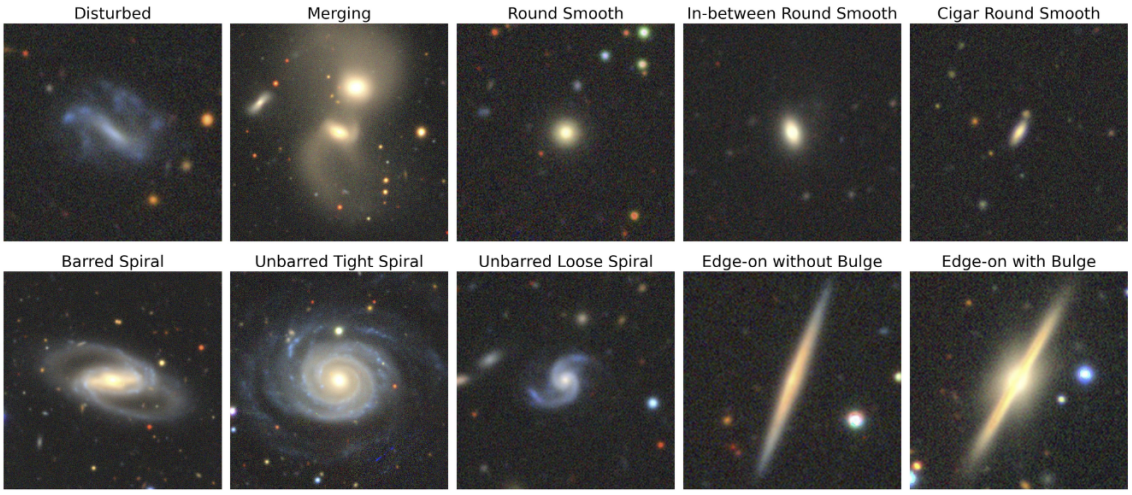


Figure 1: Examples from each of the 10 classes in the Galaxy10 DECals dataset

2 Preprocessing

2.1 Background Subtraction

A crucial aspect of our methodology is the implementation of background subtraction, which is essential for isolating galaxies from the inherently noisy astronomical images. Achieving clarity in our images is vital for the effectiveness of our modeling and analysis. Our initial attempts involved experimenting with various standard techniques such as Mixture of Gaussian and K-nearest neighbors, as well as attempting to utilize a specialized Python package, galmask[2]. However, these methods did not meet our specific requirements.

Consequently, we devised a two-part process: segmentation followed by inpainting. For segmentation, we explored various options, including scikit-learn's blob detection and Meta's advanced SAM (Segment Anything Model) [3]. Surprisingly, a more straightforward approach that combined blurring, contrast adjustment, and thresholding yielded the most effective results. In the inpainting phase, we tested several methods, including OpenCV's functions, Stable Diffusion [4], and various filling techniques using patches from unmasked areas. These approaches, however, fell short of

our expectations. Ultimately, we found success in a novel technique where we randomly selected pixels from unmasked regions outside the central area of the images. This method allowed us to recreate the subtracted backgrounds with a tailored level of noise for each image, thereby achieving the desired clarity and accuracy in our dataset.

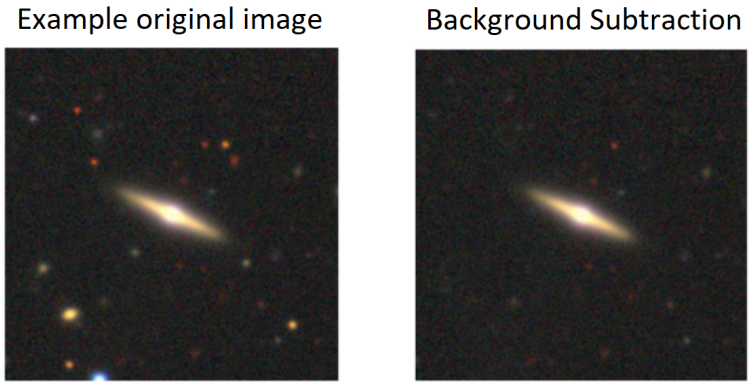


Figure 2: Background subtraction applied showing appropriate color and texture

2.2 ResNet Pre-processing

Prior to extracting ResNet features, we took the required pre-processing steps, center cropping the images to 224x224 pixels, converting them into tensors, and normalizing with specific mean and standard deviation values. This ensures optimal compatibility and performance with the ResNet architecture and results in more effective classification.

3 Feature Extraction

In this section, we discuss the main features we chose to extract in order to train our model.

3.1 Mean Color Channel and Center Color Channel

Given that each galaxy type seemed to have a different level of each color channels upon initial inspection, we thought that a good initial area of exploration would be to look at the average of each color channel for each class of images. Some galaxies, like

'cigar shaped' and 'edge-on with bulge' seemed to contain more red and green colors, whereas spirals seemed to contain a larger amount of near infrared light. As we can see there is a small amount of separation among the centers of the distributions, with our three types of spiral galaxies seeming to have the largest difference in color distribution, having a higher proportion of each color across all three color channels. One thing that was notable was how similar the distributions of mean colors were across all channels.

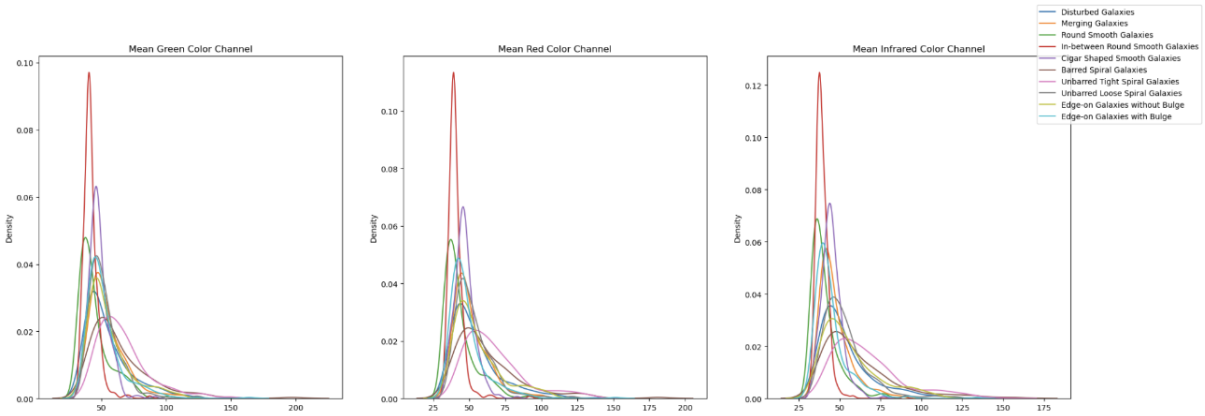


Figure 3: Density Plots of Mean Color Channels across image classes

A similar measurement was performed looking at the central color of each galaxy cluster, as research suggested [5] that galactic centers hold valuable information that cannot be found elsewhere in a galaxy. When looking into these features, we can see that the distribution of galaxy center colors is quite similar among classes for the three color channels, however some variance can be seen with the two types of edge on galaxies, so it was decided to keep the feature, and allow PCA to remove any of the redundant color channels from this feature.

3.2 Histogram of Oriented Gradients (HOG)

In order to get a better representation of the different shapes of each galaxy type, a HOG feature vector was built for each image. HOG is a method that extracts all of the different edge orientations in given segments of the image, and combines these different found edges into histograms representing the different image segments. HOG allows us toggle the number of edge orientations detected as well as the size of the image segments, in order to provide more or less granular features. We performed a HOG analysis on grayscaled and blurred image centers. HOG was helpful in

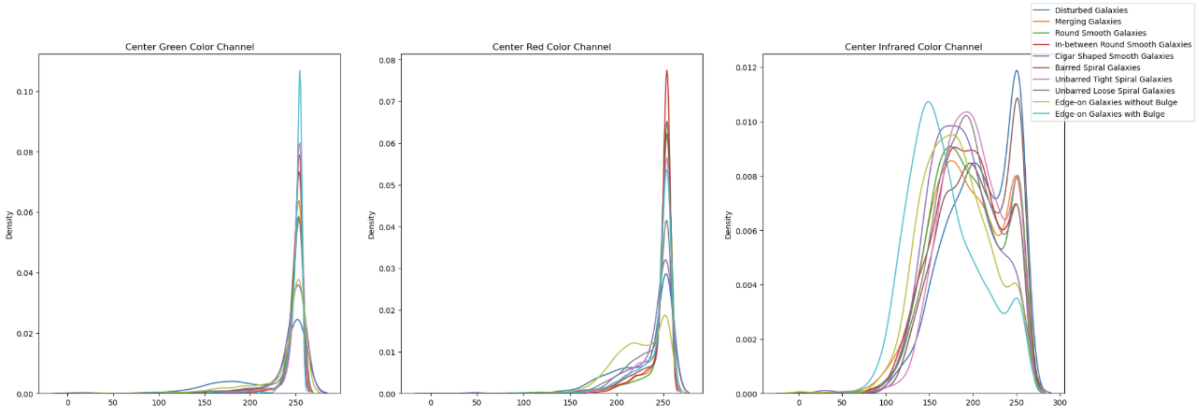


Figure 4: Density Plots of Center Color Channels across image classes

capturing the different shapes in galaxies, as well as capturing bulges in the center of galaxies, as well as capturing the more mixed orientations of merging galaxies.

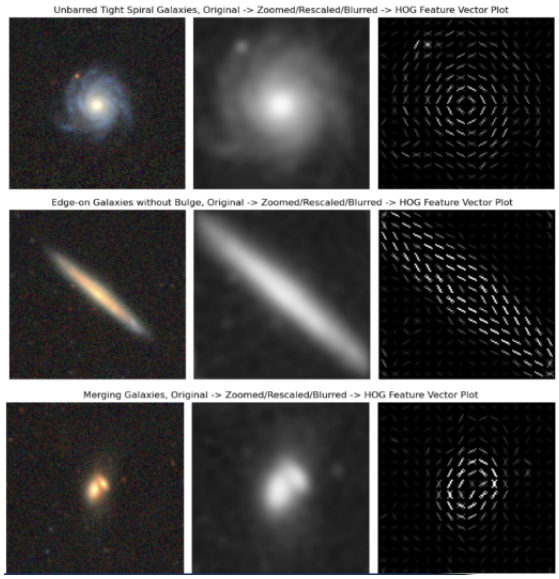


Figure 5: Histogram Of Oriented Gradients Example Images

3.3 Bag of Visual Words (BOVW)

A BOVW feature was created in order to identify key points in our galaxy images. BOVW takes its roots from the BOW approach used in NLP, where keywords are identified and stored in order to get a better understanding of the underlying text. The same approach is taken with a BOVW. We first leveraged SIFT, a feature transformer which identifies scale and rotation invariant key points in images [6]. We took 25 key points and descriptors generated by SIFT for each image and fed them into a K-means classifier with 200 clusters, to identify the most common 200 features captured by SIFT across all of our images. This created a codebook, essentially a 'vocabulary' of key points from our images. From there, we compared all of our individual SIFT descriptors for each image against this codebook and counted the frequency each key point appeared in each sift descriptor, giving us a feature vector of length 200 containing the count of each codebook feature for each of our images. In order to weigh the importance of these features more accurately, we implemented tf-idf, which reduces the weight of commonly occurring features and increases the weight of less common and unique features across the galaxy images.

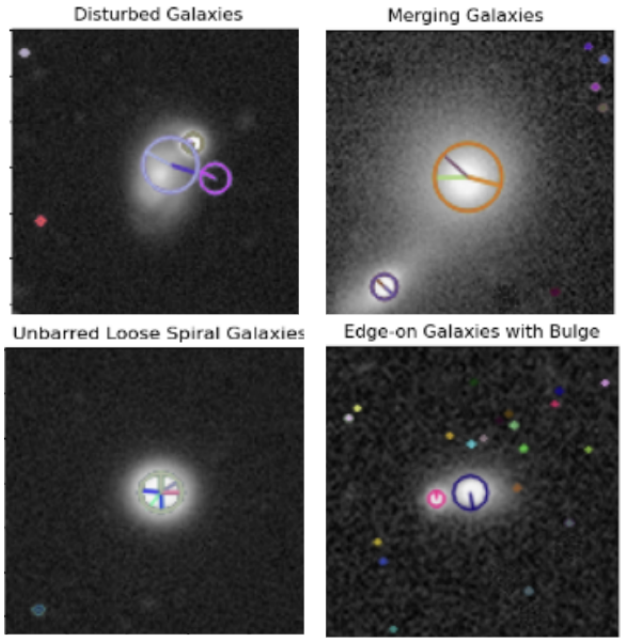


Figure 6: BOVW Keypoints

3.4 Feature Embeddings from Pre-trained Neural Network Models (ResNet)

We also leveraged ResNet101 [7], a 101 layer pretrained neural network that takes advantage of skip connections in order to address the vanishing gradients problem that models like VGG16 and AlexNet face [8]. Each layer of Resnet maps residuals from the current layer onto the previous layer in order to build a new, more robust feature representation of the image. Models like ResNet, VGG16, AlexNet, and others all employ similar processes in order to create feature embeddings of the input images. But, given more robust nature of ResNet in a variety of different classification tasks, it was decided to move forward with ResNet feature embeddings as a feature for our final model [9]. Grayscale galaxy images were preprocessed into the proper shape for Resnet (224X224) and the feature embeddings created by ResNet were extracted for training our models. As we will see in a bit, the Resnet features helped massively in improving the results of our models.

3.5 Principal Component Analysis (PCA)

A Principle Component Analysis (PCA) was then employed in order to reduce the dimensionality of the selected features, which assists in improving the efficiency of our model training, as well as reducing overfitting by removing features that explain minimal amounts of variance in the dataset. PCA was done on each of our individual feature vectors, and especially with our larger features like HOG, Visual BOW, and ResNet, we saw a huge reduction in dimensionality from over 3000 initial features to just 661 final features, improving the efficiency of our model training and somewhat reducing overfitting.

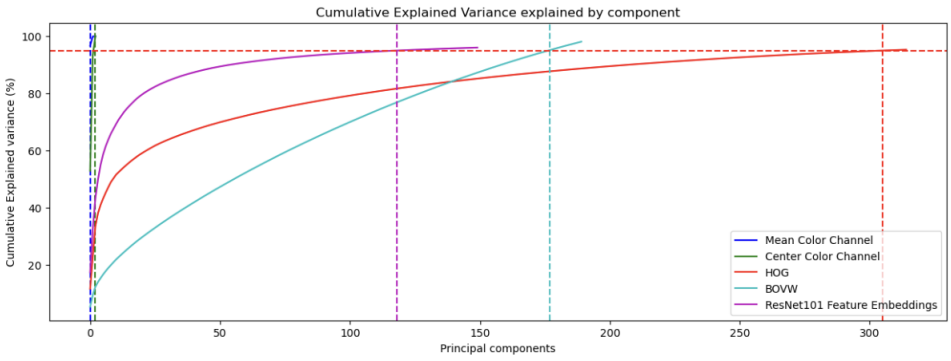


Figure 7: PCA on Individual Features (from 3,448 to 661 features)

We also ran a PCA on a concatenated version of our selected features, which reduced dimensionality even further, with 28 features that explained 95% of the features variance. That noted, this also exposed that a small number of features from the PCA output (the first four features) explained around 90% of the variance, so we decided to move forward the PCA done on each individual feature vector. Figure 8

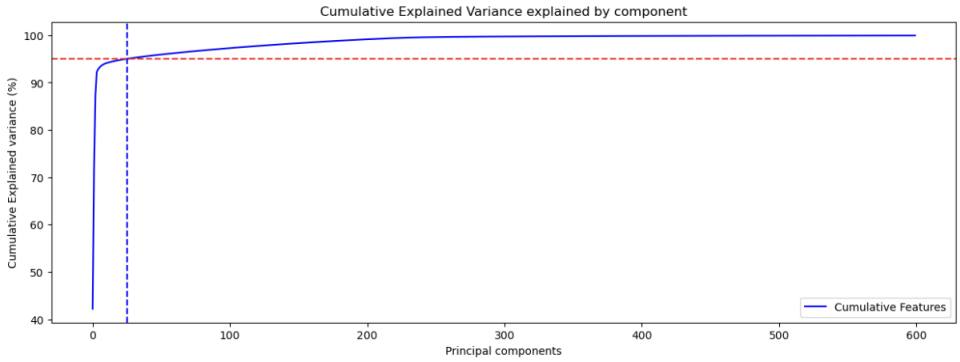


Figure 8: PCA on Combined Features (from 3,448 to 28 features)

3.6 T-distributed Stochastic Neighbor Embedding (t-SNE)

T-SNE is another dimensionality reduction method that is used to visualize high dimensional data in low dimensional spaces, and helps provide insight into how well specific features perform in terms of clustering the underlying data, and thus which features are best to use.

As we can see from the t-SNE plots for our individual feature vectors in Figure 9, none do an exceptional job in terms of clustering our data across the 10 different classes. HOG and our ResNet feature embeddings seem to do a decent job of clustering across the y-axis of the t-SNE visualization, however it is clear that there is still some overlap between the classes. This is also clear when we look at the t-SNE visualization for our combined feature vector in Figure 10.

Although t-SNE does not specifically verify that our features do a good job of clustering our data in a low dimension space, it does not indicate that our features are poor to use for classification.

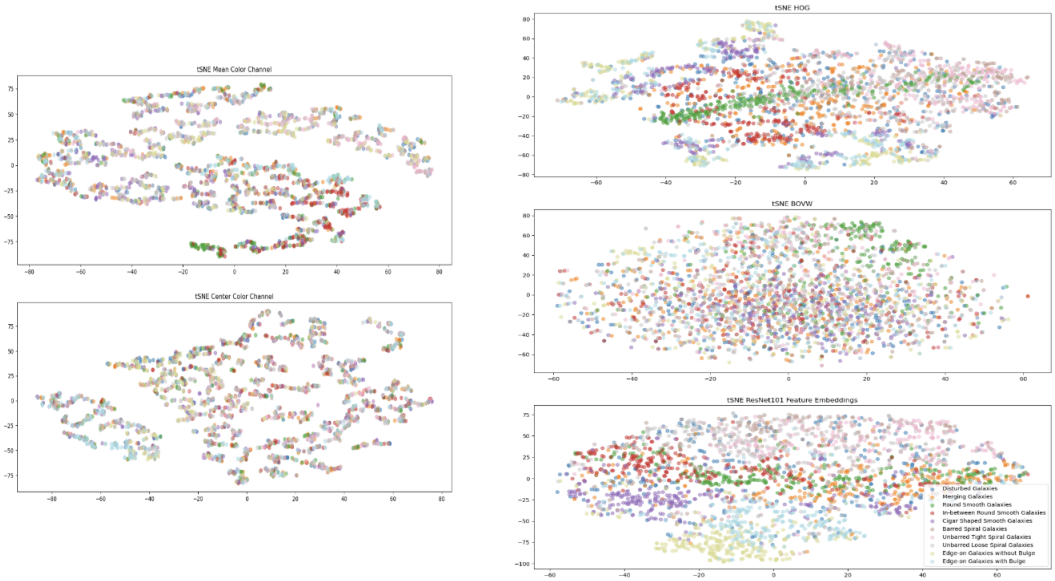


Figure 9: t-SNE Visualizations for Each Feature

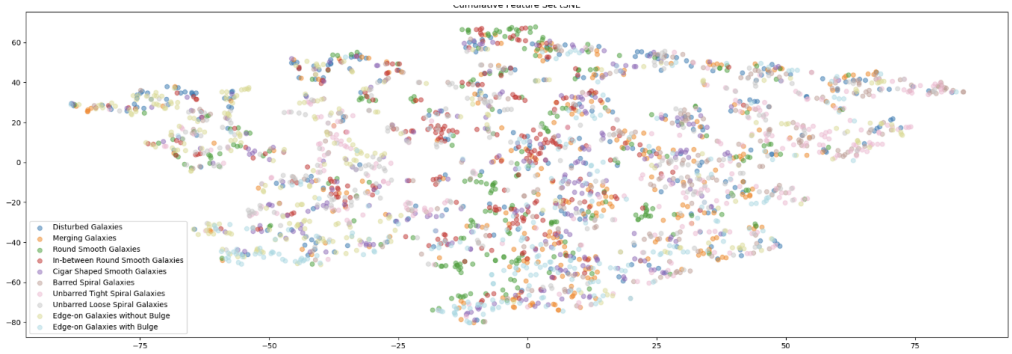


Figure 10: t-SNE Visualizations for Combined Feature Vector

4 Classification

In this section, we will talk about our classifiers, hyper-parameter tuning, and the final results.

4.1 Set-Up

Due to the large sizes of the image data, as noted in the Introduction, we faced limitations in RAM and computational resources even after downsampling. For instance, a grid search with a logistic regression model, comprising eight combinations and ten fits, requires over two hours to complete.

To speed up the run, we transitioned from traditional sklearn classifiers to GPU-accelerated classifiers. We implemented a KerasClassifier with only one Dense layer as the Logistic Regression Classifier. Furthermore, we used package cuML[10] for RandomForestClassifier and package ThunderSVM[11] for Support Vector Machine classifier.

We integrated grid search functions with result reporting capabilities. The grid search function is designed to autonomously handle model selection and hyperparameter tuning, requiring only the training and validation sets and a `model_type` keyword (one of SVM, LogReg, and RF) as input.

GPU-accelerated functions can finish a grid search of Logistic Regression in 50 minutes. A pass on SVM and RF would take around ten minutes.

4.2 Feature Selection

To optimize the selection of feature combinations and hyperparameters, we performed a series of test runs using SVM and Random Forest classifiers, excluding Logistic Regression due to its extended runtime. The outcomes of these tests are detailed in Table 1.

All models are clearly overfitting. With this in mind, based on the insights from the t-SNE analysis while considering the efficiency of training, we identified that a combination of (1) mean color channels, (2) center color channels, and (3) ResNet101 model embeddings yielded the highest accuracy. There was an discussion regarding the potential use of HOG features in place of center color channels, given their comparable accuracy levels. However, after careful consideration, we decided against HOG due to two primary factors: a higher risk of overfitting and notably slower processing speeds (HOG has 395 components, center color channels only have three). Consequently, this carefully selected feature set was adopted for subsequent use in the study.

Finally, Our final feature vectors, after transforming each to PCA separately, include 153 principal components.

Feature Combination	Model	Best Parameters	Training Accuracy	Validation Accuracy
Mean Color Channel, Center Color Channel, ResNet	LogReg	SGD, L2, 100 epochs	0.8390	0.6546
	SVM	C=1, rbf max_depth: 8	0.9364	0.6060
	RF	n_estimators: 300 split_criterion: 0	0.9749	0.5636
Mean Color Channel, Center Color Channel, BOVW	SVM	C=1, linear max_depth: 6	0.8968	0.1559
	RF	n_estimators: 300 split_criterion: 0	0.8188	0.1970
Mean Color Channel, HOG, BOVW	SVM	C=1, rbf max_depth: 8	0.9989	0.2830
	RF	n_estimators: 300 split_criterion: 0	0.9834	0.4626
Mean Color Channel, HOG, ResNet	SVM	C=5, rbf max_depth: 10	0.9995	0.5374
	RF	n_estimators: 300 split_criterion: 0	0.9995	0.5960
Mean Color Channel, HOG, BOVW, ResNet	SVM	C=5, rbf max_depth: 8	1.00	0.5175
	RF	n_estimators: 300 split_criterion: 0	0.9995	0.5798
Mean Color Channel, Center Color Channel, BOVW, ResNet	SVM	C=5, rbf max_depth: 10	1.00	0.5187
	RF	n_estimators: 300 split_criterion: 0	0.9995	0.5212
All 5	SVM	C=10, rbf max_depth: 8	1.00	0.5200
	RF	n_estimators: 300 split_criterion: 0	0.9749	0.5636

Table 1: Features Selection

4.3 Results

Using the previously identified feature sets, we conducted an extensive grid search to fine-tune the hyper-parameters across all three model types. This process ensures that the models are not only adapted to the chosen features but are also operating under the most effective parameter settings. Results are shown in Table 2.

Logistic Regression Since Logistic Regression focuses on linear relationships, it has the least impact of overfitting, though it's still prone to overfit to some degree. The combination of SGD and L2 loss emerges as the combination with highest validation accuracy.

Model	Parameter	Search Values	Best Value	Training Accuracy	Validation Accuracy	Test Accuracy	Search Time
Logistic Regression	optimizer	Adam, SGD	SGD	0.8150	0.6459	0.6257	3140.86s
	regularizer	L1, L2	L2				
	epochs	100, 200	200				
SVM	C	0.1, 1, 5, 10, 100	1	0.9374	0.6072	0.6153	703.85s
	kernel	linear, polynomial, rbf	rbf				
	gamma	scale, auto	auto				
Random Forest	max_depth	2, 4, 6, 8, 10	8	0.9759	0.5723	0.5793	774.63s
	n_estimators	10, 50, 100, 200, 300	300				
	split_criterion	0 (Gini), 1 (entropy)	Gini				

Table 2: Grid Search Results among Models and Hyper-Parameters

L2 regularization, also known as Ridge Regression, adds a penalty term proportional to the square of the model's parameters. This penalty term discourages the model from fitting overly complex or wavy boundaries, hence may help in reducing overfitting. Furthermore, recent studies show that "Adam often leads to worse generalization performance than SGD for training deep neural networks on image classification tasks" (Zhou et al., 2021)[12]. As we are working on an image dataset, SGD is thus a more suitable choice.

Support Vector Machine (SVM) SVM suffers significantly from overfitting in our model training. When we utilize all five features with 741 PCA components, the training accuracy always reaches 1.00, but the validation accuracy dramatically drops to 0.51. The HOG feature, which comprises 395 components, is identified as the primary contributor to this overfitting issue.

Another reason that SVM may overfit is the high dimensionality of the data (153 components), This may lead to a model that is too complex and thus captures noise as well as signal. We then experimented with combined PCA components, where we only have 20 components. The result did not yield satisfactory results either, which we would discuss in the Efficiency section.

Random Forest Random Forest is the model in our study that suffers most significantly from overfitting, primarily due to its completely non-linear nature and its capacity to learn complex data patterns. The optimal parameter combination we identified for Random Forest includes a maximum depth of 8, 300 estimators, and the Gini criterion.

The reasons for overfitting can be multifaceted. First, having a large number of estimates in the forest can lead to excessive complexity, where the model becomes

overly tailored to the training data. We are also surprised that the grid search says 300 estimators give the best result. Additionally, a higher max-depth increases the risk of learning noise present in the training data, as deeper trees tend to capture more intricate patterns that may not be representative of the overall data distribution. Another contributing factor is the lack of sufficient regularization, which is important in controlling the growth of the trees.

4.3.1 Model Performance

The performance of our final models are described in Table 3. The best model overall, is the Logistic Regression model.

The confusion matrix (Figure 11) reveals a notable pattern: three types of spiral galaxies cannot be accurately distinguished. We see this pattern in other two models, too. Examples of spiral galaxies are shown in Figure 13. This challenge is somewhat expected, as these spiral galaxy variants present subtle differences that are not easily discernible, even to the human eye. In real-world astronomical classification, these borderline cases often require a consensus approach, where volunteer votes play a crucial role in making the final determination. Implementation of a sub-classifier may be able to resolve this issue.

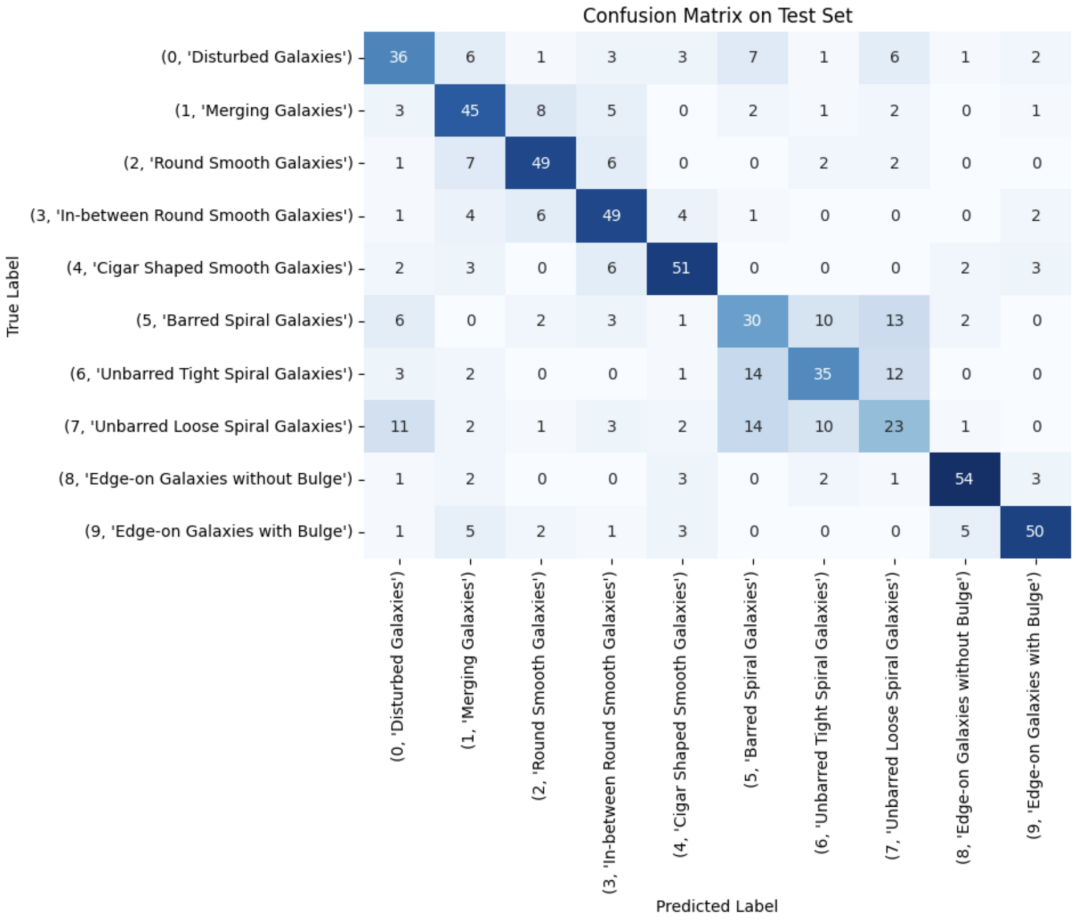
The model also struggles in correctly identifying 'disturbed galaxies' (Figure 14). This is because disturbed galaxies are galaxies that has been altered by an interaction with another galaxy. This leads to a wide range of unique and irregular shapes. Such variability directly impacts the models' ability to classify disturbed galaxies accurately.

Model	Accuracy			Training Time	Prediction Time		
	Training Set	Validation Set	Test Set		Training Set	Validation Set	Test Set
Logistic Regression	0.8139	0.6439	0.6317	29.58s	0.25304s	0.12572s	0.12005s
SVM	0.9706	0.5761	0.5689	4.44s	0.10813s	0.05681s	0.04973s
Random Forest	0.9353	0.6035	0.6138	4.26s	0.20439s	0.14798s	0.14055s

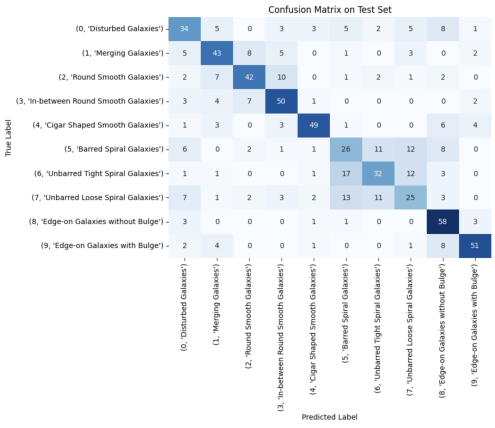
Table 3: Final Model Results Comparison

4.4 Generalizability

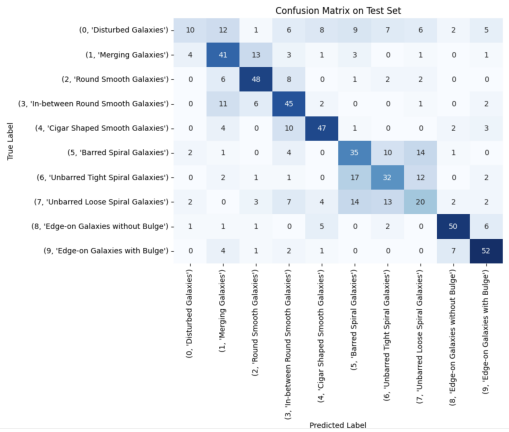
Following common practices, we hold out 20% of the data as the test set. Then we did a 70-30 train/validation split. The test set is remained out of the training process



((a)) Logistic Regression



((b)) SVM



((c)) Random Forest

Figure 11: Confusion Matrix for Each Classifier

	precision	recall	f1-score	support
0.0	0.55	0.55	0.55	66
1.0	0.59	0.67	0.63	67
2.0	0.71	0.73	0.72	67
3.0	0.64	0.73	0.69	67
4.0	0.75	0.76	0.76	67
5.0	0.44	0.45	0.44	67
6.0	0.57	0.52	0.55	67
7.0	0.39	0.34	0.37	67
8.0	0.83	0.82	0.82	66
9.0	0.82	0.75	0.78	67
accuracy			0.63	668
macro avg	0.63	0.63	0.63	668
weighted avg	0.63	0.63	0.63	668

Figure 12: Classification Report

and grid search process at all times. We only used test set accuracy to evaluate the final classifier choice.

Our training set accuracy is much larger than the test set accuracy, which tends to indicate limited generalizability. However, considering the complexity of classifying among 10 categories, the achieved accuracy of 0.63 is still reasonable. Furthermore, when the spiral galaxy classes are grouped together, the model's accuracy improves to 0.71, suggesting a degree of generalizability, albeit within a more constrained scope.

We made some attempts to enhance the model's generalizability. First, we refined the preprocessing steps, particularly improving the background subtraction results. Secondly, a significant improvement was observed upon removing HOG features, likely due to the reduction of the high-dimensional feature space. Thirdly, we tried reducing the number of PCA components, though this did not yield as substantial an impact as the previous adjustments. All together, these three attempts elevated our test accuracy from 0.5 to 0.63, a notable improvement in our model's performance.

5 Efficiency

In an attempt to reduce number of features, we opted to consolidate all individual features to a single, combined PCA. We recorded the model training time as well as

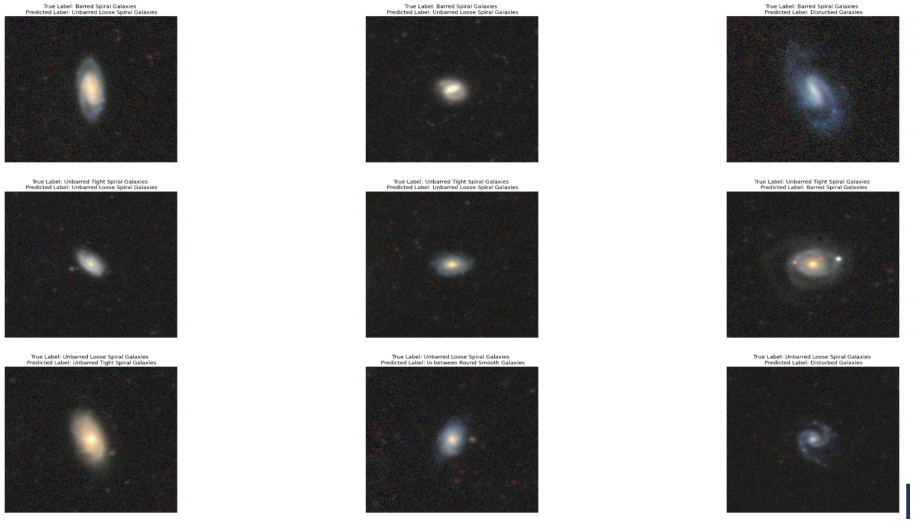


Figure 13: Misclassified Examples: 3 Types of Spiral Galaxies

Figure 14: Misclassified Examples: Disturbed Galaxies

prediction time of all runs. The result is shown in Table 4.

This brings 156 PCA components down to 28 components, significantly improving computational efficiency. Comparing the runtime in Table 4 to Table 3, we noticed that the training and prediction time for the Random Forest model was halved as before. The training time for SVM and Logistic Regression also decreased.

However, this simplification came at a cost to model performance: the average accuracy dropped from 0.6 to 0.45. Given this trade-off between efficiency and accuracy, we decided to use the model with 156 components as our final model.

Model	Validation Accuracy	Test Accuracy	Model Training Time	Prediction Time		
				Training Set	Validation Set	Test Set
Logistic Regression	0.4676	0.4671	27.51s	0.29454s	0.14783s	0.13616s
SVM	0.4838	0.4895	3.69s	0.42775s	0.097944s	0.17690s
Random Forest	0.4414	0.4237	2.01s	0.09360s	0.04681s	0.04280s

Table 4: Model Run Speed on 28 PCA Components

6 Conclusion

To summarize, our Models did an adequate job at classifying the different galaxy types, however even after extensive experimentation with Logistic Regression, Support Vector Machine (SVM), and Random Forest models, and despite careful hyperparameter tuning, our results revealed a consistent issue of overfitting across all models.

The persistent overfitting could be attributed to several factors. Firstly, the inherent complexity and variability within astronomical images make it difficult for models to generalize well from training to unseen data. The high dimensionality of the data, even after reduction through PCA, likely contributed to this challenge. Additionally, the distinct and subtle differences between galaxy types, especially disturbed and spiral type galaxies, may require more sophisticated and novel feature extraction and classification techniques than those we employed.

Moving forward we would like to explore different avenues of custom feature extraction in order to capture the nuances between the commonly misclassified galaxy types to reduce overfitting and improve our model's performance. We would also like to explore different models, including creating an ensemble classifier that adds a separate classifier to our existing classifier for classifying between Spiral type galaxies.

In conclusion, while our models demonstrated some capability in classifying galaxy images, the task's complexity and our results suggest the need for continued research and innovation in our approach. The path forward will likely involve a combination of advanced machine learning techniques, deeper astrophysical insights, and perhaps new feature extraction techniques that have yet to be explored. Overall, we still believe that employing a model such as the one proposed provides a clear path towards improving the current inefficient and subjective methodology of classifying galaxies, providing scalability and consistency that can help improve our understanding of the

fundamental building blocks of galaxies in the universe.

References

[1] Henry Leung and Jo Bovy. Galaxy10 decals dataset. `https://www.astro.utoronto.ca/~hleung/shared/Galaxy10/Galaxy10_DECals.h5`, 2021. Department of Astronomy & Astrophysics, University of Toronto. Dataset contains 17736 colored galaxy images in 10 classes, sourced from DESI Legacy Imaging Surveys and Galaxy Zoo.

[2] Yash Gondhalekar, Rafael S. de Souza, and Ana L. Chies-Santos. galmask: A python package for unsupervised galaxy masking, 2022.

[3] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023.

[4] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

[5] K. M. Hosny, M. A. Elaziz, I. M. Selim, and M. M. Darwish. Classification of galaxy color images using quaternion polar complex exponential transform and binary stochastic fractal search. *Astronomy and Computing*, 31:100383, 2020.

[6] D.G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] Yumeng Qian. Performance comparison among vgg16, inceptionv3, and resnet on galaxy morphology classification. *Journal of Physics: Conference Series*, 2580:012009, 09 2023.

[10] RAPIDS development team. cuml - rapids machine learning library. `https://pypi.org/project/cuml/`, 2023.

[11] Zeyi Wen, Jiashuai Shi, Qinbin Li, Bingsheng He, and Jian Chen. ThunderSVM: A fast SVM library on GPUs and CPUs. *Journal of Machine Learning Research*, 19:797–801, 2018.

[12] Pan Zhou, Jiashi Feng, Chao Ma, Caiming Xiong, Steven Hoi, and Weinan E. Towards theoretically understanding why sgd generalizes better than adam in deep learning. *arXiv:2010.05627 [cs.LG]*, 2020. NeurIPS 2020, arXiv:2010.05627v2.