Mars Surface Terrain Image Classification

Michael Laucella[1], Kevin Chow[2], and Nicholas Lin[3]

[1]mikelaucella@berkeley.edu
[2]kchow2020@berkeley.edu
[3]nicholasllin@berkeley.edu

Friday 13th December, 2024

Abstract

The classification of Martian surface terrain is a critical component in advancing our understanding of the planet's geological history and planning future exploration missions. This study aims to leverage machine learning and computer vision techniques to classify Martian terrain features using a dataset from NASA's HiRISE, comprised of high-resolution images captured by the Mars Reconnaissance Orbiter. We explored multiple feature extraction methods, including Histogram of Oriented Gradients (HOG), Scale Invariant Feature Transform (SIFT), Canny edge detection, Gray-Level Co-occurrence Matrix (GLCM), and the pre-trained CNN model ResNet-50. Three classification models, Logistic Regression, Histogram-based Gradient Boosting Classification Tree, and Support Vector Machine (SVM), were employed to categorize the terrain into seven distinct classes. The three models were combined into a final voting classifier that produced a final test accuracy of 87%. The results demonstrate the effectiveness of combining traditional feature extraction methods with deep learning, particularly in handling challenges such as class imbalance and overfitting.

1 Introduction

In recent years, Mars exploration has received increasing attention from both the scientific and public communities, driving advancements in the little red planet's research. A key aspect of this research is the classification of Mars' surface terrain, which provides insights into the planet's geological history and surface activity. Understanding the surface terrain is essential not only for growing our scientific knowledge of the planet, but also for planning future missions to Mars by supporting resource identification and mission planning.

To that goal, this project aims to leverage computer vision and machine learning techniques to classify Mars surface terrain features using a dataset of images from NASA's High Resolution Imaging Science Experiment (HiRISE)[2][3], one of six instruments aboard NASA's Mars Reconnaissance Orbiter launched back in 2005. The classification is focused on seven categories of terrain features (see Figure 1).

We reprocess and rebalance the dataset before employing simple feature extraction on the images via Histogram of Oriented Gradients (HOG)[7], Scale Invariant Feature Transform (SIFT)[1], Canny edge detection[1], and Gray-Level Co-occurrence Matrix (GLCM)[7]. We also perform complex feature extraction using the pre-trained CNN model ResNet-50[4][5]. For classification, we utilize Logistic Regression[6], Histogram-based Gradient Boosting Classification Tree[6], and Support Vector Machine (SVM)[6] models. After examining individual features and different combinations of features, we utilize HOG, SIFT BOVW, and ResNet-50 features in a majority voting classifier[6] model for optimal results. In this paper, we share our approach and experimental findings in pursuit of advancing Mars terrain classification and contributing to the broader field of planetary exploration.

2 Data Preprocessing

The dataset used in this study, containing images from NASA's HiRISE, was posted to Zenodo by scientists from the NASA Jet Propulsion Lab (JPL). The images were originally sourced from 232 separate high-resolution images of Martian terrain, each containing multiple landmarks. They were then processed into smaller images, 227×227 pixels, each containing the full extent of individual landmarks. The smaller images were then augmented to generate six additional versions via:

- 90-degree clockwise rotation
- 180-degree clockwise rotation
- 270-degree clockwise rotation
- Horizontal flip
- Vertical flip
- Random brightness adjustment

Upon exploring the dataset, challenges of class imbalances, duplicate images, and image artifacts, in the form of blacked-out regions, were discovered. Duplicate images were removed and images were reprocessed by applying rotation, zooming, and cropping to remove blacked-out areas, ensuring that only the relevant terrain features remained. The final image size of 227×227 pixels was maintained and all reprocessed images were re-augmented to produce their six additional versions. Finally, we rebalanced the data into training, validation, and test sets following a 60/20/20 split per class. Training and validation sets included the original images and their augmentations, while the test set only included the un-augmented, original images. Our final dataset consisted of 8449 training images, 2821 validation images, and 403 testing images.

3 Feature Extraction

In this section, we explore various methods of feature extraction from the images, discuss

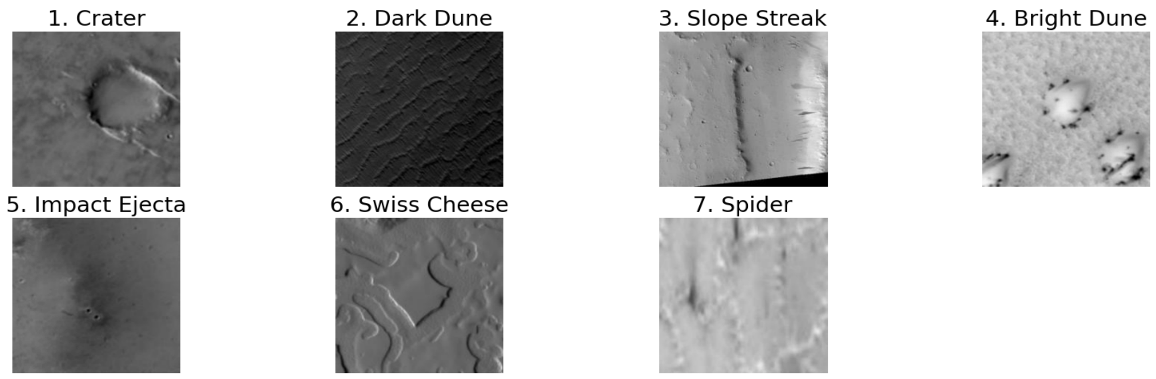


Figure 1: Example Images for Each of the Seven Classes

the findings, and explain why we chose certain features for our final classification model.

3.1 HOG

HOG extracts edges oriented in different directions by looking at the histogram of oriented gradients across the images. It is particularly useful for detecting the shape of an object, or in our case the shapes present in Martian terrain. We tuned various parameters such as orientations, pixels per cell, and cells per block to best identify significant characteristics of each class. The final parameters used were: 8 orientations, (8×8) pixels per cell, and (2×2) cells per block.

As shown in Figure 2, the extracted edges of each class are quite distinct from each other. This feature space had a dimension of $27 \times 27 \times 2 \times 2 \times 8$. Using the flattened feature vector of the raw HOG features, we noticed significant overfitting on the training data with baseline logistic regression models. For better generalization, we used two binning methods (across cells and per cell) across the dimensions to identify bins of mean, maximum, range, and interquartile range. This resulted in a final feature space of 720.

3.2 SIFT

SIFT detects key points and descriptors of an image that are invariant to scaling, rotation, and small distortions. This is useful when image features such as corners are clear from a zoomed out perspective; however, if the scale changes, the corner might not be detected. Since we're working with satellite imagery that included various perspectives and distances, we thought this would be a helpful feature to use.

As seen in the sample images in Figure 3, SIFT did a decent job on identifying key points of each class, such as the edges of a Crater, impact points of Impact Ejecta, and the edges of Dark Dunes. To further improve this feature, we converted it into a bag of visual words (BOVW) [1]. We grouped similar descriptors using k-means clustering. With 1000 components, the cluster centers formed the dataset's vocabulary. For each image, we computed the frequencies of these clusters to create a BOVW representation.

3.3 Canny Edge Detection

Canny edge detection identifies edges in an image by detecting areas with rapid changes in intensity, using Gaussian filtering for noise

[1]SIFT key points are not homogeneous every image can emit a different number of features

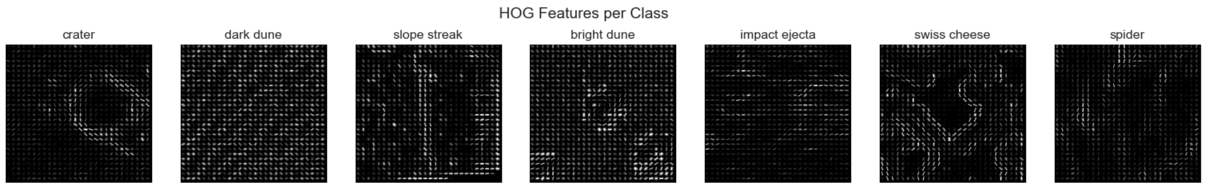


Figure 2: Example HOGs for Each of the Seven Classes

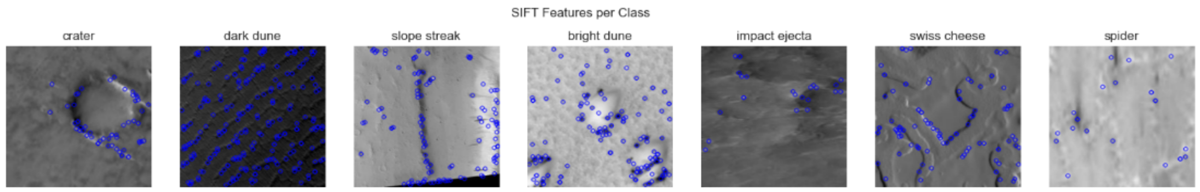


Figure 3: Example SIFT Key Points for Each of the Seven Classes

reduction and edge tracking to finalize valid edges. By highlighting sharp transitions in pixel values, the algorithm emphasizes significant features within the Martian terrain. It is useful for outlining boundaries and structures, such as edges of Craters or long lines in Slope Streaks.

Initially, we experimented with using the raw Canny edge maps and different thresholds, but found that they did not provide sufficient information as features. With our image size at 227 × 227, the raw edge features resulted in extremely large 227 × 227 arrays of binary True / False values (see Figure 4). As a result, testing this on a baseline logistic regression model saw significant overfitting. Thus, we instead converted the edges to contours and found summary statistics of these contour lengths including total length, mean length, standard deviation of the length, and number of contours. This significantly reduced the dimensionality while also improving the issue of overfitting.

3.4 GLCM

GLCM analyzes the spatial relationship between pixel intensities to extract texture features of contrast, correlation, energy, and homogeneity. We utilized this method of feature extraction to identify different textures across the landscape. For instance, in the Figure 1 sample images, the texture of the landscape in the Swiss Cheese class is much smoother than the texture in the bright dune class or spider class. It also provided valuable global feature information about the images with a relatively low feature space.

Figure 5 shows the distributions of each texture feature using a distance of 3 and an angle value of 90 degrees. There are differences in distributions among some classes for dissimilarity, energy, and correlation. To expand the feature more, we also included the following parameters into GLCM's final feature vector: distances of [1, 3, 5] and angles of [0, 45, 90] degrees. This left the final feature vector at a length of 45.

3.5 ResNet-50

ResNet-50 is a convolutional neural network (CNN) architecture that performs extremely well on image classification. Its architecture includes 50 layers divided into 4 main parts: the convolutional layers, identity block, convolutional block, and the fully connected layers. This architecture addresses the issue of

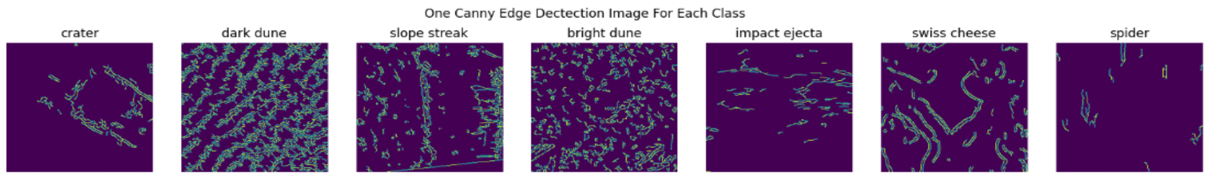


Figure 4: Example Edge Detection for Each of the Seven Classes

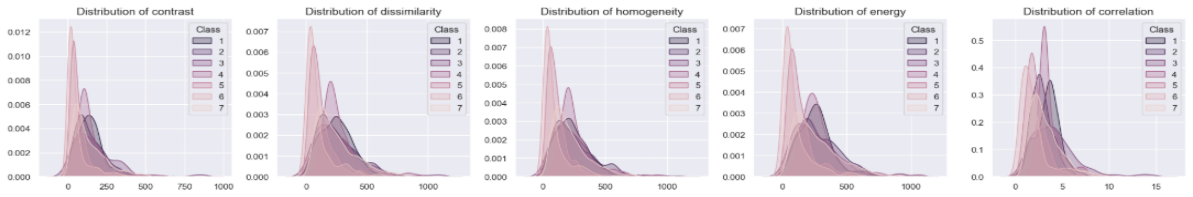


Figure 5: GLCM Distributions

vanishing gradients. Due to its effectiveness, we chose ResNet-50 to extract complex features from the Martian terrain images.

Using this pre-trained model, we employed transfer learning by using the embeddings from these networks as an input to our classifiers. These embeddings capture various features of the input image. As an example, Figure 6 shows a sample of the feature maps pulled from ResNet-50's block 1 of a "Spider" image, showing what the model sees as important attributes.

3.6 Dimension Reduction

To help increase computational efficiency, reduce overfitting, and determine the final features we wanted to use in our models, we analyzed the results from dimension reduction methods. For Principal Component Analysis, (PCA)[6], we wanted to identify the features that could be reduced while maintaining a threshold of explained variance. Initially, we tried keeping all features at 0.95 (Figure 7) explained variance. However, to address the issue of overfitting, we reduced each feature further by measuring model performance against validation and stopping when the results became unstable.

Compared to PCA, using t-SNE[6] for dimensionality reduction allows us to better visualize the data in two dimensions. In theory, this method could help identify which features were doing a better job in separating the different classes. As shown in Figure 8, the only two features that produced some separation was the feature embeddings from ResNet-50 and the SIFT BOVW feature.

Based on the analysis of dimensionality reduction and testing baseline models of individual features, we settled on HOG, GLCM, SIFT BOVW, and ResNet-50 to be examined in our final classification models.

4 Classification

When we recombined the reduced features we found that their dimensionality was still too high, which lead to overfitting. This motivated us to further compress down to 150 dimensions (see Table 1). With the reduced features, we trained and analyzed three classification models: Logistic Regression, Histogram-based Gradient Boosting Classification Tree, and Support Vector Machine (SVM). Based on this we opted to remove GLCM as it did not perform well and reduced accuracy scores

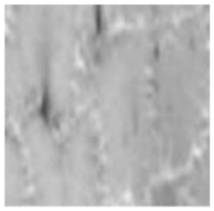
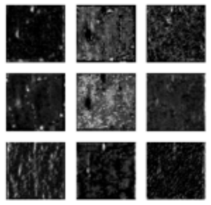
Example Spider Image	ResNet-50 Layer 1 Feature Embeddings
	

Figure 6: Resnet50 Feature Embedding Example of ”Spider” Image

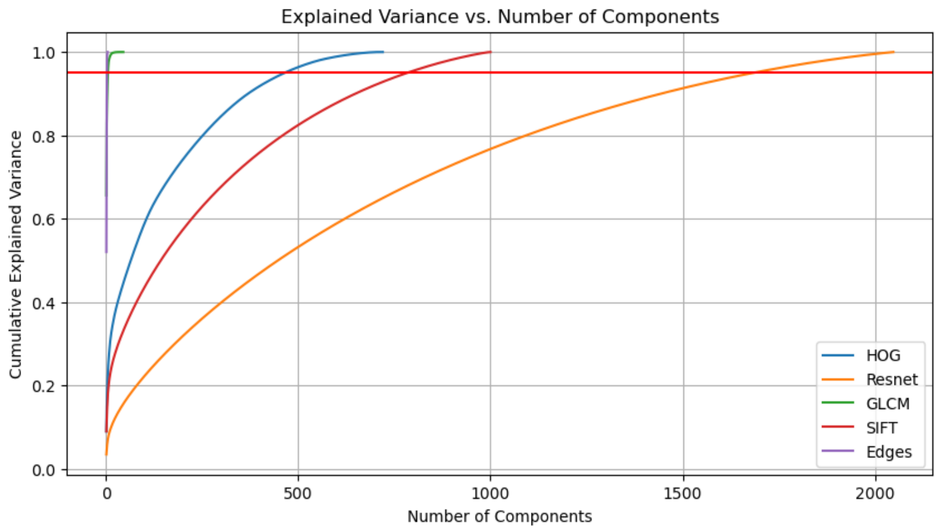


Figure 7: Dimensionality Reduction via PCA

across all the models. Final confusion matrix plots of each classifier can be found in appendix B and final model performance metrics by class can be found in Table 4.

4.1 Results

We ran our dataset against Logistic Regression, Support Vector Machine (SVM), and Histogram-based Gradient Boosting Classification Tree to compare simplicity versus performance. Starting with Logistic Regression gave us a simple and efficient baseline to improve on. With SVM we could leverage non-linear relationships. With a Gradient Boosted Classification Tree, we could address overfitting as they are complex ensemble models where every additional iteration adds a new tree that guarantees an increase in training performance.

Logistic Regression. Logistic Regression is a simple and widely-used algorithm for classification tasks. It calculates the probability of an outcome using a logistic function applied to a linear combination of features, which are then mapped to a class label. It is common for Logistic Regression to struggle with non-linearly separable data. However, after grid searching for the optimal hyperparameters, our Logistic Regression model performed the best on the validation set out of all three models (see Figure 2).

Support Vector Machine. Support Vector Machines are another supervised machine

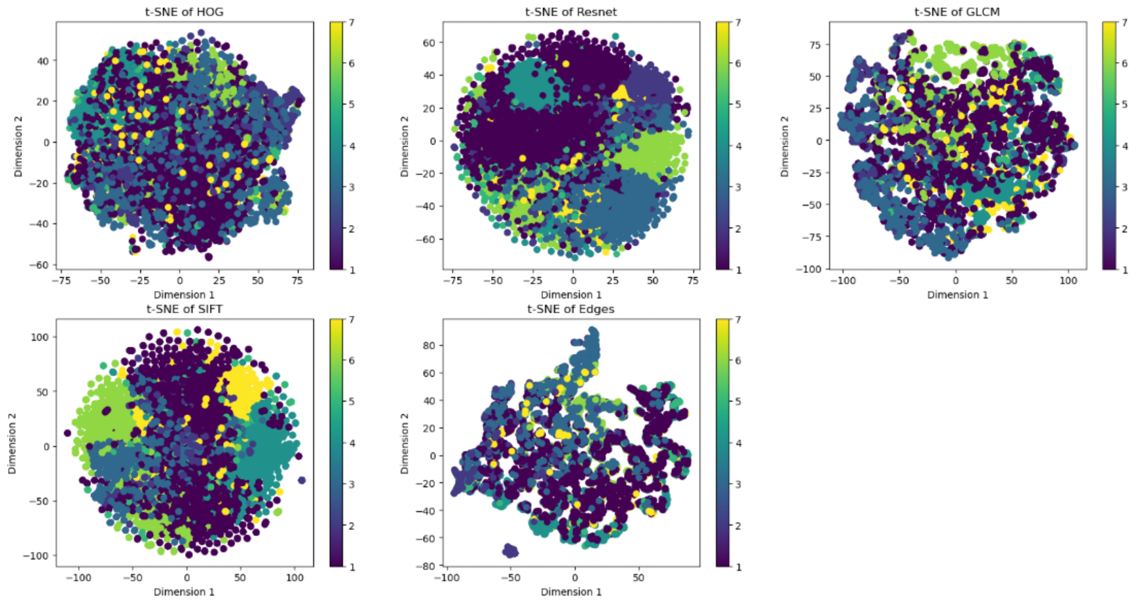


Figure 8: Dimensionality Reduction of Each Feature via t-SNE

Feature	**Original Dimension**	**New Dimension**	**Explained Variance**	**Percent Reduction**
HOG	720	5	0.2	99%
GLCM	45	15	0.6	67%
SIFT (BOVW)	1000	30	0.3	97%
ResNet-50	2048	100	0.3	95%

Table 1: Final Features Summary

learning algorithm used for classification tasks. In the context of image classification, SVMs are effective for distinguishing between classes by finding the optimal hyperplane that separates that data points. It does this by maximizing the margin between the closest points of each class. Some limitations of the model include having poor scalability for large datasets and needing tuning of hyperparameters to perform optimally. However with our reduced feature space, the SVM had a relatively quick training time (Table 5) and performed decently on both validation and test accuracy as seen in Table 2.

Histogram-based Gradient Boosting Classification Tree. Histogram-based Gradient Boosting Classification Tree builds decision trees sequentially to correct errors from previous trees. Through utilization of histograms to speed up tree building, it is more efficient and effective for large datasets. Features also do not require normalization or standardization. One drawback of Histogram-based Gradient Boosting Classification Tree models is that overfitting becomes inevitable if they are allowed to run indefinitely, as each iteration continuously learns from prior residuals. Additionally, this classification model requires careful tuning of numerous hyperparameters, making them complex, challenging to interpret, and computationally intensive. Despite these drawbacks, our grid searched model performed only 1% (Figure 2) worse on the validation set than the next best model, SVM.

For all 3 models, we performed thorough randomized grid searches scored against the validation set to determine optimal hyperpa-

Model	PCA	Parameter Tuned	Training Accuracy	Validation Accuracy	Test Accuracy
Majority Class	N	N	0.39	0.39	0.39
Logistic Regression	N	N	1.00	0.82	N/A
Logistic Regression	Y	Y	0.96	0.83	0.86
Hist Gradient	Y	Y	1.00	0.81	0.86
SVM	N	N	1.00	0.81	N/A
SVM	Y	Y	1.00	0.82	0.86
Voting Classifier	Y	Y	1.00	0.84	0.87

Table 2: Model Classification Results

rameters (see Table 3). Once found, we retrained the three models using their optimal settings and calculated the scores for train, validation, and test datasets. Across all three base classifiers, there were consistent issues with correctly identifying the Impact Ejecta and Spider classes. Rather than the individual classifiers struggling with these classes, we believe there were other problems that led to their degraded classification. The struggle to classify Impact Ejecta likely stems from its under-representation in the original images, with it making up only 3.7% of dataset. The classification of the Spider class for ResNet-50 actually gets worse when adding in HOG and SIFT BOVW features. As a result, we believe the addition of the other features hampered our classifier models' performances on the Spider class while boosting performances for other classes.

After viewing the confusion matrices for each of the three models, we noted that each had different strengths. Therefore, we fit an additional voting classifier that averaged the class probabilities across the models and selected the highest probability. This voting model achieved our highest test accuracy score of 87% (see Table 2), indicating a reasonably strong generalizability.

Our training process could be further improved through a few additional methods. Additional augmentations of our training and validation images, beyond what is already captured, could lead to better performance. Regarding the Spider class, a manual deep dive of incorrect examples could be done to further analyze its misclassification. Lastly, we could fully utilize a deep neural network to perform classifications, as ResNet-50 was our strongest performing feature.

4.2 Fairness

The dataset was highly imbalanced (see Table 4). As a result, naive models with decent overall scores (82% or more) often had low accuracies, $<\approx 2\%$, on the highly under-represented class Impact Ejecta (see Figure 9). Through dimensionality reduction and the combination of several models, we achieved an 10-fold improvement in accuracy for the Impact Ejecta Class (Figure 13).

4.3 Efficiency vs. Accuracy

With a relatively low final feature space of only 135 dimensions, there was little issue with training and inference time. Table 5 shows the full runtimes of each model along with the final test accuracy. There are some tradeoffs that could be made. To be specific, using the voting classifier required training three individual models and going through grid search to achieve the highest test accuracy. However, the tuned SVM classifier showed extremely high efficiency at the cost of model fairness and accuracy.

5 Conclusion

In conclusion, we were able to leverage various feature extraction methods to produce several classification models that were generalizable, efficient, and accurate. With a proper 60/20/20 data split for training, validation, and testing, we based our experimental decisions only by looking at validation metrics throughout the process, preventing data leakage. The final features included were HOG, SIFT BOVW, and feature embeddings from ResNet-50. Using a voting classifier that included SVM, Logistic Regression, and Histogram-based Gradient Boosting we achieved a test accuracy of 87%. Key steps that helped achieve our observed efficiency and accuracy included experimenting with feature combinations, using PCA to reduce feature dimensions, and hyperparameter tuning.

References

[1] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[2] Baris Dincer. *HIRISE Mars Reconnaissan / NASA - IMAGE/LABEL*. 2022. URL: https://www.kaggle.com/datasets/brsdincer/hirise-map-mars-nasa-image.

[3] Steven Luand Kiri Wagstaff Gary Doran Emily Dunkel. *Mars orbital image (HiRISE) labeled data set version 3.2*. 2020. zenodo: 4002935. URL: https://zenodo.org/records/4002935.

[4] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.

[5] TorchVision maintainers and contributors. *TorchVision: PyTorch's Computer Vision library*. https://github.com/pytorch/vision. 2016.

[6] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[7] Stefan Van der Walt et al. "scikit-image: image processing in Python". In: *PeerJ* 2 (2014), e453.

Appendix A Tables

Model	**Parameter**	**Search Values**	**Best Value**	**Search Time**
Logistic Regression (n_iter = 150)	Penalty	['l1', 'l2', 'elasticnet', 'None']	'l1'	∼ 4.8 Min.
	C	[0.01, 0.1, 1, 5, 10]	1	
	solver	['liblinear', 'sag', 'newton-cg', 'lbfgs', 'saga']	'liblinear'	
	max_iter	[50, 100, 200, 250]	100	
	l1_ratio	[0, 0.25, 0.5, 0.75, 1]	None	
Histogram Gradient Boosting (n_iter = 50)	learning_rate	[0.01, 0.05, 0.1, 0.2]	0.2	∼ 3.2 Min.
	max_iter	[100, 200, 300, 400]	400	
	max_leaf_nodes	[15, 31, 63, 80]	31	
	max_depth	[None, 3, 5, 7]	3	
	min_samples_leaf	[20, 50, 100]	100	
	l2_regularization	[0, 0.1, 1, 10]	10	
SVM (n_iter = 50)	C	[4, 5, 6, 7.5, 10, 15, 18]	15	∼ 2.5 Min.
	gamma	['scale', 0.00007, 0.0001, 0.0002, 0.0003, 0.0002, 0.0005]	0.0003	
	kernel	['rbf', 'poly', 'sigmoid', 'linear']	'rbf'	

Table 3: Hyperparameter Tuning (Grid Search) for our models

Class	**Precision**	**Recall**	**Accuracy**	**Count**
1	0.94	0.90	0.92	159
2	0.64	0.85	0.73	33
3	0.87	0.83	0.85	54
4	0.89	0.98	0.93	50
5	0.47	0.47	0.47	15
6	0.96	0.92	0.94	59
7	0.86	0.76	0.81	33

Table 4: Model Performance Metrics by Class (Test)

Model	**PCA**	**Parameter Tuned**	**Training Time**	**Inference Time**	**Test Accuracy**
Majority Class	N	N	N/A	N/A	0.39
Logistic Regression	N	N	0.9s	0.001s	N/A
Logistic Regression	Y	Y	4.9s	0.000s	0.86
Hist Gradient	Y	Y	5.7s	0.035s	0.86
SVM	N	N	149.3s (∼2.5m)	0.6s	N/A
SVM	Y	Y	0.8s	0.009s	0.86
Voting Classifier	Y	Y	14.8s	0.043s	0.87

Table 5: Accuracy vs. Efficiency Comparison

Appendix B Confusions

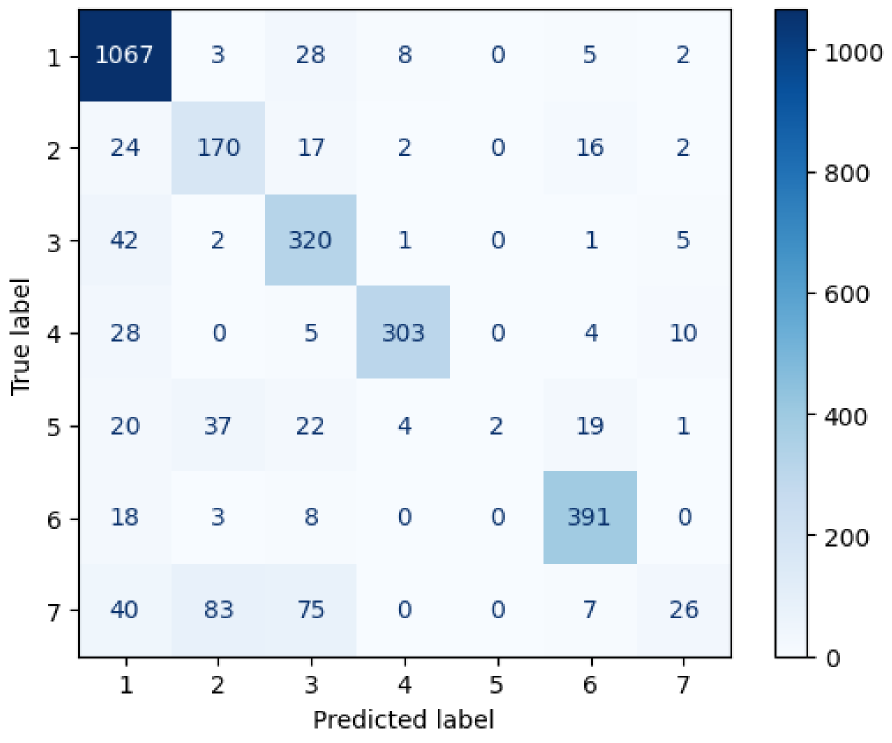


Figure 9: SVM Model Performance Confusion Matrix, No Grid Search or PCA (Val)

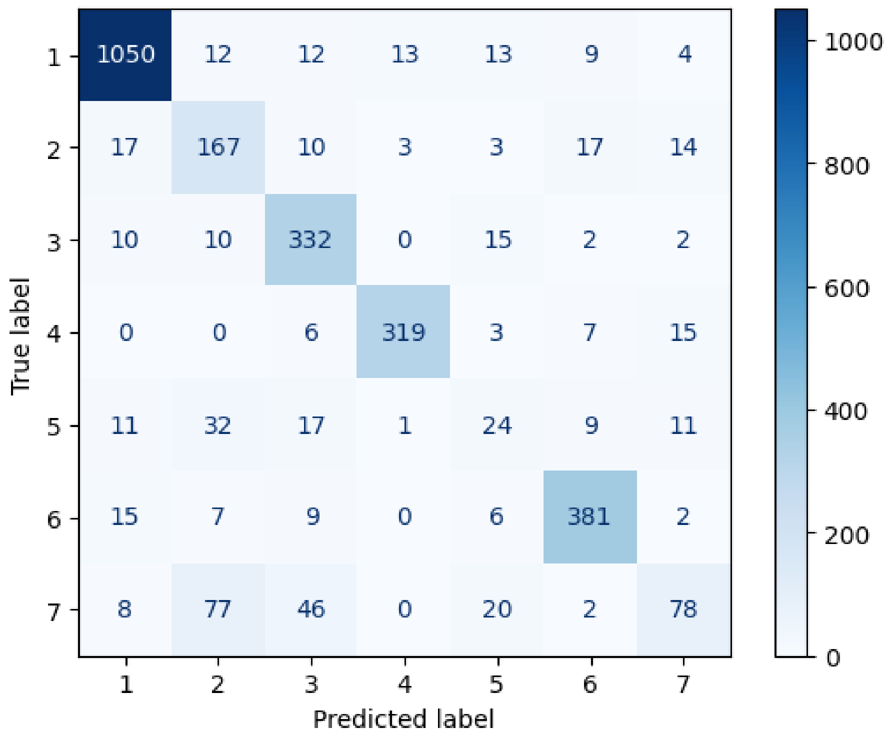
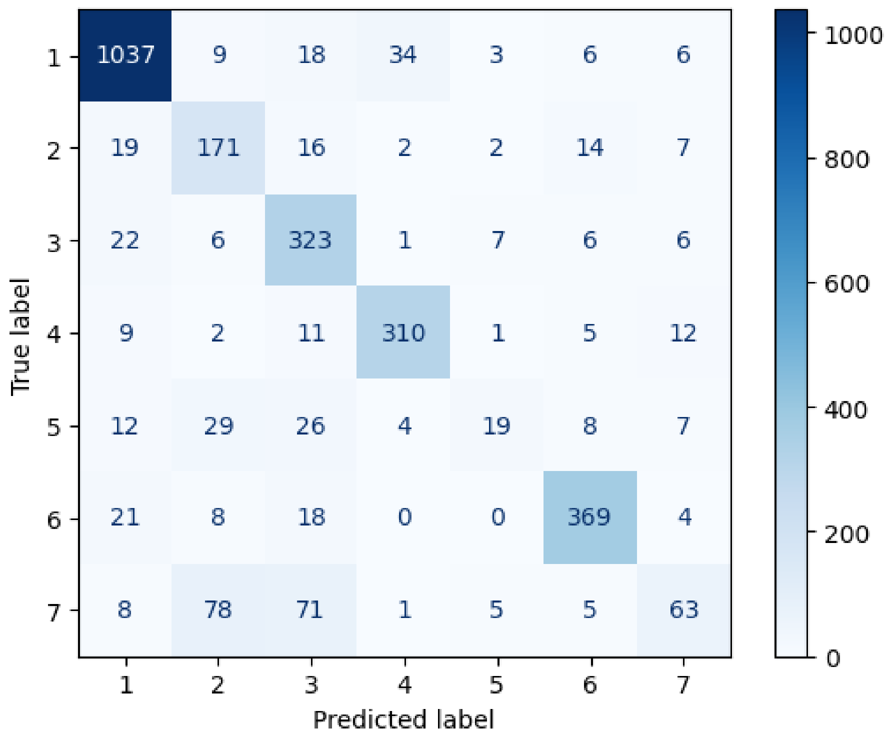
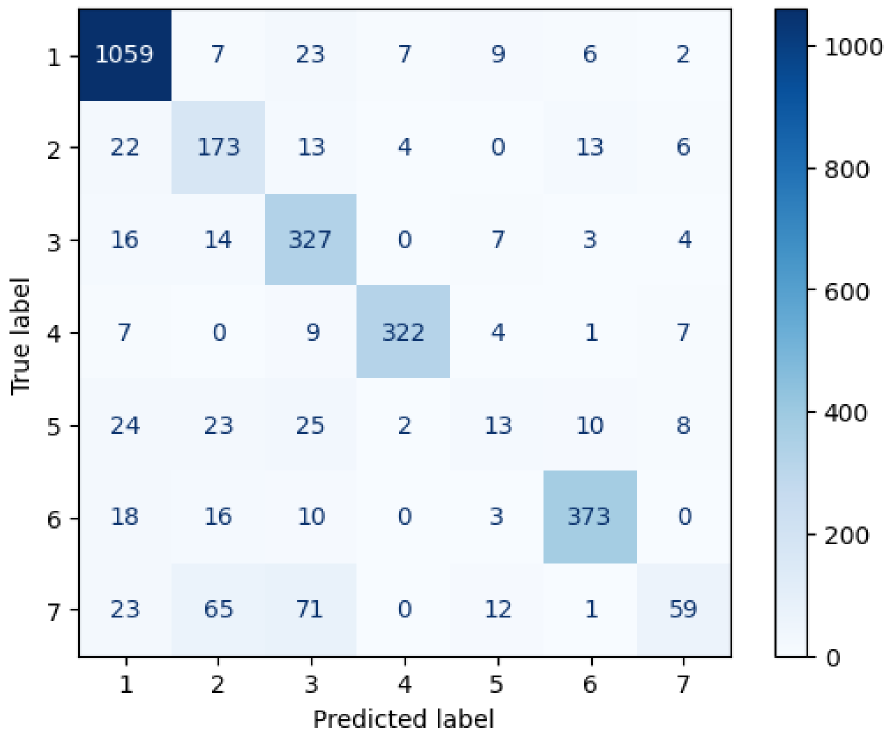


Figure 10: Logistic Regression Model Confusion Matrix (Val)



True \ Predicted	1	2	3	4	5	6	7
1	1037	9	18	34	3	6	6
2	19	171	16	2	2	14	7
3	22	6	323	1	7	6	6
4	9	2	11	310	1	5	12
5	12	29	26	4	19	8	7
6	21	8	18	0	0	369	4
7	8	78	71	1	5	5	63

Figure 11: Histogram Gradient Boosting Confusion Matrix (Val)



True \ Predicted	1	2	3	4	5	6	7
1	1059	7	23	7	9	6	2
2	22	173	13	4	0	13	6
3	16	14	327	0	7	3	4
4	7	0	9	322	4	1	7
5	24	23	25	2	13	10	8
6	18	16	10	0	3	373	0
7	23	65	71	0	12	1	59

Figure 12: SVM Confusion Matrix (Val)

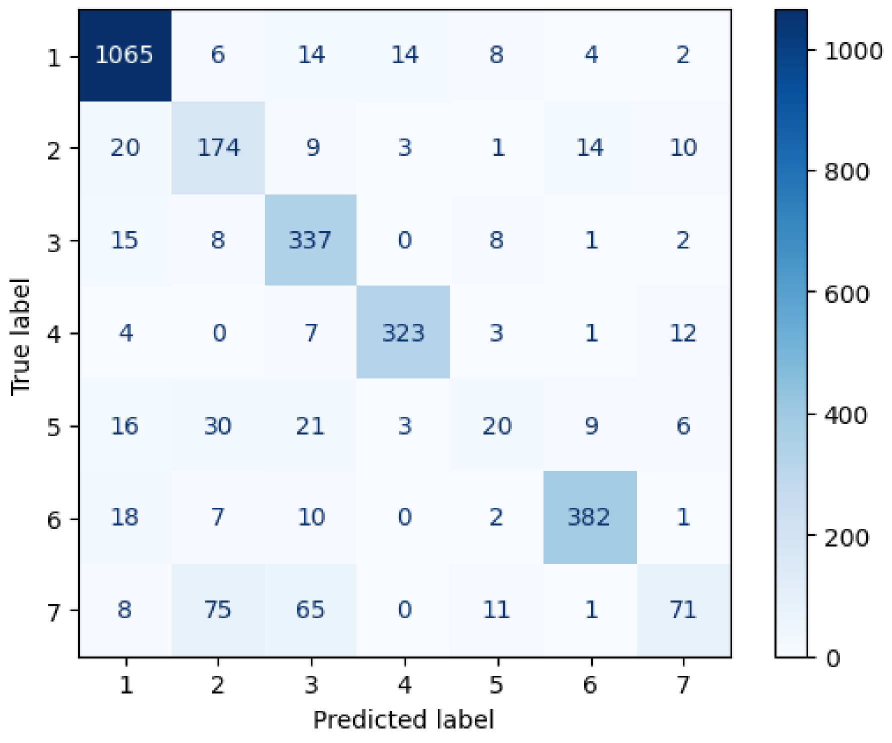


Figure 13: Final Voting Classifier Model Confusion Matrix (Val)

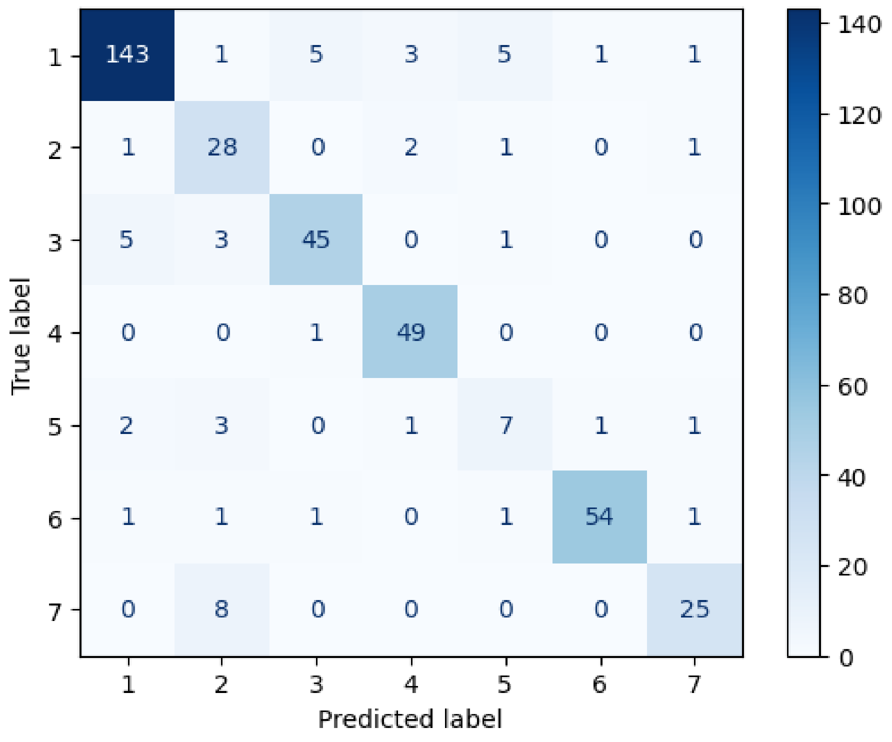


Figure 14: Final Voting Classifier Model Confusion Matrix (Test)