SPARK 2021 Satellite Image Classification using Machine Learning

Eric Peng, Jared Petrisko, Sean Sica
DATASCI 281 Computer Vision
December 13, 2024

***Abstract*—As space exploration becomes more and more common, the development of AI that understands space-related imagery grows. To push this boundary, we explore the effectiveness of traditional machine learning methods on classifying 10 different space satellites and space debris. Our handcrafted features include histogram of oriented gradients (HOG), hue saturation value (HSV), and local binary pattern (LBP). We employed a grid search to find an optimal model with these features. In addition, we extract complex features from ResNet50 and ResNet101 to evaluate the effectiveness of large pre-trained neural networks on our task. With HOG we achieved an accuracy of 19.51%. With HSV features we achieved an accuracy of 30.91%, and with our complex ResNet101 features we achieved an accuracy of 70.44%. We show the potential of traditional image recognition methods and the power of complex neural networks for interstellar data.**

I. Introduction

Space travel is becoming increasingly common with the advancements in interplanetary ships developed by SpaceX and Blue Origin. While image recognition has been applied countless times to Earth scenes, as we explore deeper into space and send up commercial projects around Earth, it becomes important to apply these techniques to space environments.

Our dataset comes from the University of Luxembourg SPARK 2021 dataset [1]. This dataset contains images of computer-generated satellites and space debris designed to include the challenges of working with space data. These challenges include varying lighting conditions from direct sunlight to complete darkness, the presence of Earth as a high-contrast background, and the difficulty of detecting small, distant objects. These factors make the classification of space satellites and debris significantly more challenging.

We attempt to classify the type of satellite using three handcrafted features and complex features from ResNet101. These range from histogram of oriented gradients to complex feature extraction from neural networks. The inputs to the models are these extracted features from RGB images of satellites. Our output is a predicted class label corresponding to one of 11 categories of satellites or space debris. While the dataset is synthetic, it has been carefully designed to simulate real-world challenges.

To effectively experiment and collect results, we have designed an extendable machine learning pipeline built on PyTorch and scikit-learn. This allows us to easily track experiments and test various configurations of models and features.

II. Methodology

A. Data

The SPARK 2021 dataset consists of 150,000 RGB images across 11 classes: AcrimSat, Aquarius, Aura, Calipso, Cloudsat, CubeSat, Jason, Sentinel-6, Terra, TRMM, and Debris. Each satellite collects information about the Earth related to the sea or atmosphere. This computer-generated data captures a diverse range of realistic space environments.

The dataset is pre-split into train, test, and validation sets. The training set contains 7,500 images per satellite class and 15,000 images for the debris class. The test and validation sets contain 5,000 images per satellite and 7,500 debris images. All images are originally 1024×1024 pixels in JPEG format, which we resize to 224×224 due to hardware limitations.

While the SPARK 2021 dataset provides precomputed bounding box dimensions for all images, we explicitly chose to ignore these annotations in our experiments. We hypothesize that restricting our feature extraction and model fitting to the contents of the bounding boxes, rather than processing the entire image, could potentially improve our classification results. This design choice represents a deliberate simplification of our initial approach, leaving room for future investigation of bounding box incorporation.

B. Feature Extraction

1) HSV Color Space: Hue Saturation and Value color space provides an alternative to RGB. Hue represents pure color information, saturation describes color vividness, and value refers to brightness. Our dataset contains many dark images with small satellites; separating color and brightness can make satellites more distinct and identifiable.

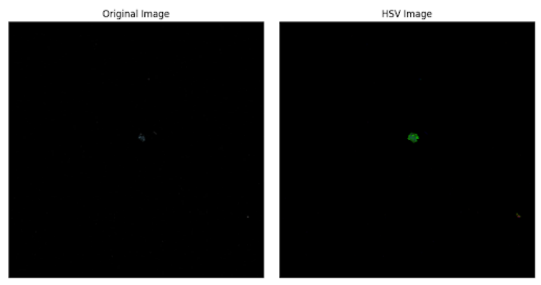

Fig. 1. Comparison of RGB and HSV color spaces for satellite imagery, demonstrating enhanced feature visibility in HSV space.

2) Histogram of Oriented Gradients (HOG): HOG analyzes the magnitude and orientation of pixel groups, effectively isolating edges and contours. Satellites have distinct shapes and features such as antennas, solar panels, and joints. These features capture the distribution and orientation of structural details, allowing our classifier to identify unique patterns between different satellites.

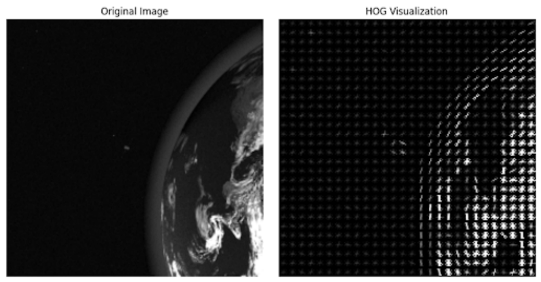

Fig. 2. Example of HOG feature extraction applied to a satellite image, showing the gradient orientations and magnitudes.

3) Local Binary Patterns (LBP): LBP is a texture descriptor that analyzes spatial patterns of pixels. This feature is useful in encoding surface patterns of satellites and their components. They are robust across various lighting conditions and viewing angles, though potentially less effective for distant satellites.

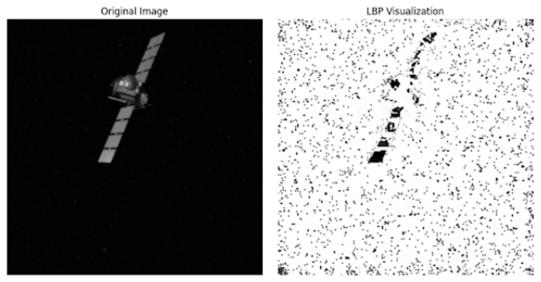

Fig. 3. Visualization of Local Binary Pattern features extracted from a satellite image, highlighting texture patterns.

4) Feature Embeddings from ResNet Series: We compared two pre-trained neural networks:

- **ResNet50**: A 50-layer convolutional neural network with skip connections to combat vanishing/exploding gradients. Pre-trained on ImageNet, it provides deep understanding of visual features.
- **ResNet101**: A 101-layer architecture offering potentially more subtle or complex feature capture than ResNet50, though with increased computational requirements.

5) Principal Component Analysis (PCA): PCA reduces dimensionality while preserving variance, potentially reducing overfitting and training time. Our analysis of the ResNet101 features reveals interesting characteristics about the feature space. As shown in Figure 4, the cumulative explained variance curve exhibits a steep initial rise, with approximately 178 principal components capturing over 95% of the total variance (indicated by the dashed vertical line). This suggests that while ResNet101 produces 2048-dimensional feature vectors, the meaningful satellite classification information is concentrated in a much lower-dimensional subspace.

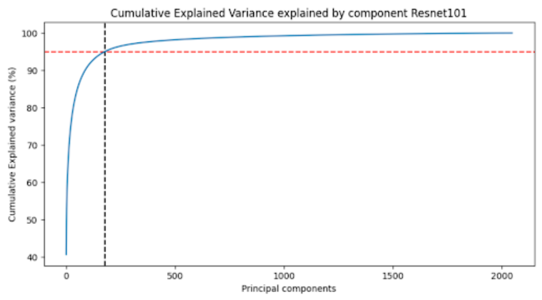

Fig. 4. Cumulative explained variance ratio for ResNet101 features. The steep initial rise indicates that most of the discriminative information is captured in the first 178 components (vertical dashed line), allowing us to reduce dimensionality from 2048 to 178 while retaining 95% of the variance.

The rapid convergence of explained variance is particularly noteworthy - the curve shows diminishing returns after about 200 components, with the remaining 1800+ components contributing only marginally to the total variance. This finding has important computational implications: by reducing the feature dimensionality from 2048 to roughly 200 dimensions while maintaining 95% of the variance, we can significantly reduce computational overhead without substantially compromising model performance. The shape of this curve also suggests that many of the ResNet101 features may be redundant or less relevant for our specific satellite classification task, despite their utility in general computer vision tasks.

C. Classification Models

We explored four machine learning models:

Logistic Regression: Serves as our baseline model, offering computational efficiency and interpretability. While simple, it effectively tests feature separability using linear decision boundaries.

Support Vector Machines (SVM): Well-suited for high-dimensional feature spaces, particularly with ResNet embeddings. The kernel trick enables modeling non-linear relationships, though at increased computational cost.

Random Forest: An ensemble method aggregating multiple decision trees, excellent for handling noise and capturing complex feature interactions. Provides feature importance scores but slower than simpler models.

Gradient Boosting: Incrementally builds weak learners to minimize classification error. Highly effective for subtle patterns but requires significant training time.

III. Implementation Architecture

To enable consistent evaluation across different feature extraction methods and models while supporting distributed development, we implemented a unified evaluation pipeline [2]. Figure 7 illustrates our system architecture.

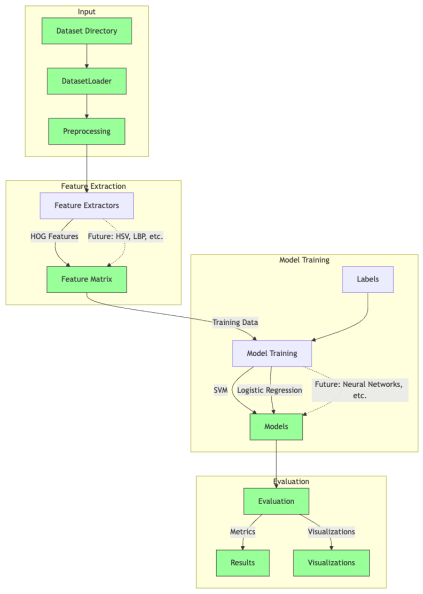

Fig. 5. System architecture diagram showing the main components of our evaluation pipeline. The pipeline consists of four main stages: input processing, feature extraction, model training, and evaluation. Dotted lines indicate extensibility points for future feature extractors and models.

A. Data Management

Our DatasetLoader implementation incorporates several optimizations to improve experimental efficiency:

- **Feature Caching:** Features are extracted once per data split and cached in memory, enabling rapid experimentation across multiple models without redundant computation. While this approach is memory-constrained to the pipeline's execution duration, it significantly reduced experimental iteration time.
- **Device Management:** The loader intelligently manages CPU/GPU memory transfers, particularly important for our development on Apple's Metal Performance Shaders (MPS) platform where not all operations are implemented.
- **Preprocessing Pipeline:** Implements automatic image scaling, standardizing inputs to 224×224 pixels to balance computational requirements with model performance.

B. Feature Extraction Framework

The feature extraction system is modularized, with distinct classes for each extraction method implemented in PyTorch where possible. We developed a decorator-based registration system that simplifies the integration of new feature extractors into the pipeline and command-line interface.

C. Model Implementation

Our experimentation revealed important trade-offs between custom PyTorch implementations and scikit-learn's optimized implementations. Through performance testing, we found that certain algorithms benefited significantly from scikit-learn's highly optimized implementations, while others performed better with custom PyTorch implementations that could leverage GPU acceleration. This led us to develop a hybrid approach: Support Vector Machine and Logistic Regression were implemented directly in PyTorch to take advantage of GPU acceleration and batch processing capabilities, while Random Forest and Gradient Boosting leveraged scikit-learn's efficient CPU-based implementations wrapped in PyTorch interfaces.

To maintain consistency across our pipeline, we developed a wrapper interface that exposes all models through a uniform PyTorch-compatible API, regardless of their underlying implementation. This abstraction layer handles the complexity of bridging between PyTorch and scikit-learn paradigms, including management of dummy parameters for PyTorch-specific concepts (such as epochs and loss functions) when these are irrelevant for particular models. This approach allowed us to maintain a clean, consistent pipeline interface while optimizing the implementation of each algorithm for best performance.

D. Hyperparameter Selection and Optimization

Our experimental approach focused on systematic hyperparameter selection across different model architectures and feature extractors. While grid search cross-validation is commonly employed for hyperparameter optimization, our experimental design prioritized consistent configurations across feature extractors to enable direct comparisons of feature effectiveness.

1) Random Forest Configurations: Our Random Forest experiments maintained consistent hyperparameters across all feature extractors, with the primary configuration using relatively shallow trees (max_depth = 3) and a small ensemble size (n_estimators = 10). This conservative model architecture was chosen to balance computational efficiency with model performance, particularly given our large dataset size.

Notably, we observed that despite varying feature dimensions significantly across extractors (from 94 dimensions for HSV to 4096 for LBP and 2048 for ResNet50), we maintained these same hyperparameters. This consistency in hyperparameters across such varied input dimensionality suggests that the Random Forest model's performance differences were primarily driven by the quality and separability of the features rather than model capacity.

The learning rate varied between experiments, with 0.01 used for higher-dimensional feature extractors (ResNet50, LBP) and 0.001 for lower-dimensional features (HSV, HOG). This pattern suggests a relationship between feature dimensionality and optimal learning rate, though the impact on final model performance was likely minimal given Random Forest's inherent robustness to learning rate settings.

All experiments used a moderate batch size of 32, which proved sufficient for memory management while maintaining reasonable training speeds across both CPU and Apple Metal (MPS) hardware configurations. The single-epoch training approach was maintained across all experiments, leveraging Random Forest's capability to learn effectively without requiring multiple passes through the data.

2) Support Vector Machine Configurations: Our SVM implementation used a linear kernel with a one-vs-all approach for multi-class classification. The model was trained using stochastic gradient descent with momentum (0.9) and L2 regularization (weight decay=0.01). We employed a standard hinge loss with margin=1.0. The learning rate was set to 0.001, with batch size of 32 and single-epoch training. This configuration was chosen to balance computational efficiency with model performance, particularly given the high dimensionality of our feature spaces (ranging from 94 dimensions for HSV features to 2048 for ResNet features).

Unlike the Random Forest experiments, the SVM hyperparameters remained constant across all feature extractors. This consistency allows us to attribute performance differences primarily to the quality and separability of the different feature representations rather than model-specific tuning.

3) Logistic Regression Configurations: Our Logistic Regression implementation used a standard linear model with bias terms enabled. The model weights were initialized to zero, following traditional logistic regression initialization practices. For optimization, we employed SGD with momentum (0.9) and a relatively light L2 regularization (weight decay=0.001), reflecting the common practice of using softer regularization compared to SVM models.

The implementation automatically adapted between binary and multi-class scenarios, using sigmoid activation with binary cross-entropy loss for binary cases and softmax activation with cross-entropy loss for multi-class problems. In our satellite classification task, the multi-class configuration was used with our 11-class problem.

As with the SVM implementation, we maintained consistent hyperparameters across all feature extractors, with a learning rate of 0.001 and batch size of 32. This consistency allows direct comparison of feature extractor effectiveness without conflating the effects of model-specific tuning.

4) Gradient Boosting Configurations: Our Gradient Boosting implementation utilized scikit-learn's GradientBoostingClassifier with a conservative tree structure: shallow trees (max_depth=3) combined with a moderate number of estimators (n_estimators=100). The learning rate was set to 0.1, balancing training speed with model robustness. For tree construction, we used traditional classification defaults with a minimum of 2 samples required to split an internal node (min_samples_split=2) and a minimum of 1 sample required to be at a leaf node (min_samples_leaf=1). The model used traditional stochastic gradient boosting settings with full samples (subsample=1.0) and square root of the number of features considered for best split (max_features="sqrt").

For feature randomization during tree building, we adopted the standard classification approach of considering the square root of the number of features when determining splits. This is particularly relevant for our high-dimensional feature spaces, helping to prevent overfitting while maintaining model diversity in the ensemble.

Unlike typical gradient-boosting implementations that might use hundreds or thousands of trees, we intentionally kept our ensemble size moderate (100 trees) to balance computational efficiency with model performance, particularly given our large dataset size and the multiple feature extractors being evaluated.

E. Hyperparameter Optimization

After identifying the feature that leads to the highest precision, we performed extensive hyperparameter searches to further optimize accuracy.

For random forests, we can modify max depth and number of estimators. Increasing the depth of each tree can allow for more complex features to be modeled but risks overfitting. Increasing the number of estimators can reduce overfitting but increases computational complexity. We need to balance these to maximize the model's performance.

For SVMs, we adjust the kernel and the regularization strength C. We experiment with a linear kernel and a radial basis function (RBF) kernel. The linear kernel assumes a linear decision boundary, while the RBF kernel is non-linear and can lead to a more complex decision boundary. Regularization controls the model's penalization for misclassifications. A high C value allows for more complex boundaries.

For logistic regression, we adjust the weight decay and the learning rate. Weight decay prevents the model from becoming too complex and helps control overfitting. The learning rate controls the magnitude of updates to the model. If it is too large, we may miss an optimal loss; if it is too small, we may never reach an optimal loss. To find the optimal set of hyperparameters and the best model, we need to evaluate the results of each model across these different values.

F. Evaluation System

The evaluation component provides comprehensive model assessment capabilities, computing standard classification metrics, including accuracy, precision, recall, and F-β scores. We chose F-β over the more common F1 score because F-β offers a tunable parameter β that allows us to adjust the relative importance of precision versus recall. This flexibility is particularly valuable in satellite classification, where the cost of misclassification may vary significantly - for instance, misclassifying debris as a functional satellite could have different operational implications than misclassifying between two similar satellite types. Our evaluation system also generates per-class ROC curves with AUC scores and supports both normalized and raw confusion matrix generation.

For performance analysis, the system tracks training time and resource utilization while implementing smart batching to efficiently process large datasets. This evaluation approach enables us to assess models not just on raw accuracy, but on their practical utility for space object classification tasks where different types of errors may have varying operational impacts.

IV. Results

Our empirical evaluation compared different combinations of feature extractors and classifiers in training, validation, and test sets. Here, we present detailed performance metrics for each configuration.

TABLE I
PERFORMANCE COMPARISON ACROSS MODEL CONFIGURATIONS

Model	Features	Split	Accuracy	Precision	Recall	F-Beta	Time (s)
GradientBoosting	HOG	Train	0.1404	0.0831	0.1404	0.0690	27842.26
		Val	0.0833	0.0069	0.0833	0.0128	
		Test	0.0833	0.0069	0.0833	0.0128	
GradientBoosting	HSV	Train	0.1725	0.2097	0.1725	0.1152	6402.50
		Val	0.1686	0.2067	0.1686	0.1131	
		Test	0.1688	0.1074	0.1688	0.1102	
GradientBoosting	LBP	Train	0.1411	0.0955	0.1411	0.0874	19182.08
		Val	0.1438	0.0897	0.1438	0.0873	
		Test	0.1418	0.0926	0.1418	0.0863	
GradientBoosting	ResNet50	Train	0.1028	0.1071	0.1028	0.0683	770.07
		Val	0.1022	0.1307	0.1022	0.0695	
		Test	0.1055	0.0903	0.1055	0.0701	
LogisticRegression	HOG	Train	0.1885	0.0467	0.1885	0.0747	15.90
		Val	0.1667	0.0278	0.1667	0.0476	
		Test	0.1667	0.0278	0.1667	0.0476	
LogisticRegression	HSV	Train	0.1712	0.0606	0.1712	0.0578	215.60
		Val	0.1729	0.0630	0.1729	0.0606	
		Test	0.1718	0.0621	0.1718	0.0588	
LogisticRegression	LBP	Train	0.1667	0.0278	0.1667	0.0476	4.55
		Val	0.1667	0.0278	0.1667	0.0476	
		Test	0.1667	0.0278	0.1667	0.0476	
LogisticRegression	ResNet101	Train	0.7197	0.7348	0.7197	0.7317	385
		Val	0.6259	0.6612	0.6259	0.6538	
		Test	0.6388	0.6766	0.6388	0.6687	
LogisticRegression	ResNet50	Train	0.1694	0.0685	0.1694	0.0532	664.47
		Val	0.1699	0.0688	0.1699	0.0543	
		Test	0.1690	0.0659	0.1690	0.0527	
RandomForest	HOG	Train	0.1974	0.0654	0.1974	0.0806	13.61
		Val	0.1667	0.0278	0.1667	0.0476	
		Test	0.1667	0.0278	0.1667	0.0476	
RandomForest	HSV	Train	0.3211	0.3540	0.3211	0.2369	0.46
		Val	0.3110	0.3311	0.3110	0.2270	
		Test	0.3091	0.2772	0.3091	0.2193	
RandomForest	LBP	Train	0.2823	0.2417	0.2823	0.1908	0.52
		Val	0.2190	0.1939	0.2190	0.1400	
		Test	0.2228	0.2057	0.2228	0.1429	
RandomForest	ResNet50	Train	0.1948	0.1586	0.1948	0.0833	0.77
		Val	0.1933	0.1458	0.1933	0.0828	
		Test	0.1959	0.1362	0.1959	0.0829	
SVM	HOG	Train	0.2177	0.3481	0.2177	0.1251	10.51
		Val	0.1968	0.3149	0.1968	0.1070	
		Test	0.1951	0.1708	0.1951	0.0990	
SVM	HSV	Train	0.1981	0.0770	0.1981	0.0787	11.60
		Val	0.1997	0.0813	0.1997	0.0797	
		Test	0.2007	0.0769	0.2007	0.0801	
SVM	LBP	Train	0.2704	0.2481	0.2704	0.1769	5128.82
		Val	0.2184	0.2292	0.2184	0.1404	
		Test	0.2255	0.1954	0.2255	0.1443	
SVM	ResNet50	Train	0.1667	0.0278	0.1667	0.0476	58.89
		Val	0.1667	0.0278	0.1667	0.0476	
		Test	0.1667	0.0278	0.1667	0.0476	

A. Model Performance Comparison

Table 1 summarizes the results of all experiments, which includes seventeen different model configurations combining different classifiers (Logistic Regression, Random Forest, SVM, and Gradient Boosting) with various feature extraction methods (ResNet101, ResNet50, HSV, LBP, HOG). For each configuration, we report performance metrics (accuracy, precision, recall, and F-Beta scores) across three data splits (training, validation, and test), along with the computational time required for each approach.

Several patterns in our experimental results warrant careful interpretation. The logistic regression classifier with ResNet101 features demonstrated superior performance, achieving 71.97% training accuracy, 62.59% validation accuracy, and 63.88% test accuracy. This significantly outperformed all other configurations, with precision and recall metrics following similar patterns. The next best performer was the RandomForest classifier with HSV features, achieving 30.91% test accuracy while maintaining the fastest training time (0.46 seconds), suggesting a notable trade-off between computational efficiency and model performance.

Our analysis revealed some concerning patterns. Multiple models (notably SVM with ResNet50) show identical metrics across different data splits, with accuracy and recall of both exactly 1/6 (0.1667). This pattern emerges when models predict the same class for all inputs, indicating a failure to learn discriminative features. This behavior particularly affects models using HOG and LBP features, as well as some ResNet50-based models, suggesting potential problems with feature extraction or model optimization.

Among the newer configurations, RandomForest with LBP features showed promising results (22.28% test accuracy) with extremely fast training time (0.52 seconds). However, GradientBoosting models generally underperformed, with the ResNet50 variant achieving only 10.55% test accuracy despite requiring considerable training time (770.07 seconds). The SVM models showed mixed results, with the HOG features performing reasonably well (19.51% test accuracy) while maintaining an efficient training time (10.51 seconds).

In particular, we observed a consistent trade-off between computational cost and performance across different feature extractors. While deep learning-based features (ResNet101, ResNet50) generally led to better performance, they also required significantly more computational resources. Traditional feature extractors such as HSV provided a more balanced trade-off, offering moderate performance with minimal computational overhead.

B. Hyperparameter Search Results

After observing that the ResNet101 embeddings outperformed all other features, we selected it for our hyperparameter optimization. Table 2 compares its performance across logistic regression, SVM, and random forest. Gradient boosting was excluded due to its poor performance during the experimentation stage.

From Table 2, we see that logistic regression, with a learning rate of 0.1 and no weight decay, achieves the most balanced accuracy metrics and the highest test accuracy at 70.44%, validation accuracy at 70.01%, and train accuracy at 78.97%. While the slight gap between the validation and test accuracy is concerning, the closeness between these accuracies indicates that the model generalizes well on unseen data. This suggests that this model captures general features that work well across satellite images. Models consistently scored in the low 60s when either weight decay was applied or a smaller learning rate was used. The SVM strongly overfit the training data but delivered a solid test accuracy of 63.2%. We used only 10% of the training data for the SVM because it scales poorly due to its reliance on quadratic optimization. Random Forest followed a similar pattern, completely memorizing the training data with 100% train accuracy but achieving only 64% test accuracy. The logistic regression model with PCA-reduced features had a lower test accuracy of 61.8% compared to the model without PCA, highlighting the trade-off between computational efficiency and accuracy when applying dimensionality reduction techniques.

Overall, the findings highlight the importance of model selection and the limitations of dimensionality reduction techniques like PCA when working with high-dimensional feature representations. To gain more insight into why the ResNet101 features worked so well, we visualize them in a lower-dimensional space.

C. Feature Space Visualization

To visualize the high-dimensional feature space produced by ResNet101, we applied t-SNE dimensionality reduction to project the features into two dimensions. Figure 6 shows the distribution of all 11 classes in this reduced feature space.

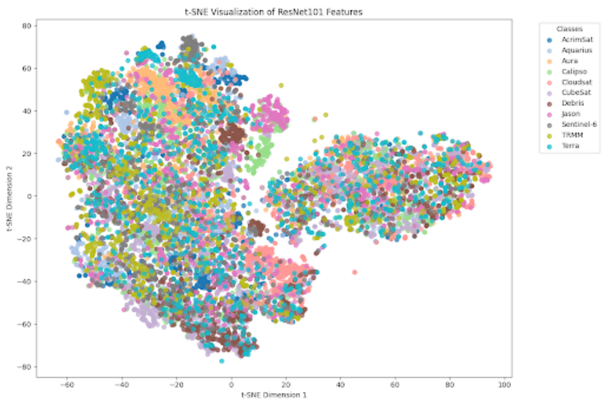

Fig. 6. t-SNE visualization of ResNet101 features colored by satellite class. The plot reveals significant mixing between classes, with three main clusters showing substantial overlap between different satellite types. This visualization helps explain the moderate classification accuracy (70.44%) achieved by our best model.

The t-SNE visualization reveals three main groupings of points, but with substantial mixing between classes within each group. Rather than showing clear separation between satellite types, we observe considerable overlap in the feature

TABLE II
COMPARISON OF MODELS TRAINED ON RESNET101 FEATURES WITH DIFFERENT PARAMETERS AND CONFIGURATIONS.

Model	Features	Grid Parameters	Best Parameters	Split	Accuracy	Precision	Recall	F-Beta	Time (s)
Logistic Regression	ResNet101	Learning Rate: [0.1, 0.01, 0.001, 0.0001] Weight Decay: [0, 0.0001, 0.001] Epochs: [125]	Learning Rate: 0.1 Weight Decay: 0	Train	0.7897	0.7998	0.7897	0.7884	3800
				Val	0.7001	0.7204	0.7001	0.6897	
				Test	0.7044	0.7254	0.7044	0.6933	
Logistic Regression	ResNet101 PCA	Learning Rate: [0.1, 0.01, 0.001, 0.0001] Weight Decay: [0, 0.0001, 0.001] Epochs: [125]	Learning Rate: 0.1 Weight Decay: 0	Train	0.6779	0.6757	0.6779	0.6728	3080
				Val	0.6047	0.6101	0.6047	0.5889	
				Test	0.6125	0.6205	0.6125	0.5974	
SVM	ResNet101	Kernel: ['linear', 'rbf'] C: [0.1, 1, 10]	Kernel: rbf C: 10	Train	0.9542	0.9569	0.9542	0.9546	2284
				Val	0.6304	0.6478	0.6304	0.6190	
				Test	0.6318	0.6493	0.6318	0.6202	
Random Forest	ResNet-101	n_estimators: [50, 100, 150] max_depth: [10, 20, None]	n_estimators: 150 max_depth: None	Train	1.0000	1.0000	1.0000	1.0000	3546
				Val	0.6321	0.6520	0.6321	0.6214	
				Test	0.6369	0.6569	0.6369	0.6266	

space. This mixing of classes helps explain why even our best model achieved only moderate classification accuracy. The lack of clear boundaries between classes in the feature space suggests that while ResNet101 captures meaningful features, the classification task remains challenging due to inherent similarities between different satellite types or viewing conditions. This visualization reinforces our findings about consistent misclassification patterns between visually similar satellite pairs discussed in our confusion matrix analysis.

D. Key Findings

The Logistic Regression classifier with ResNet101 features demonstrated superior performance, achieving 78.97% training accuracy, 70.01% validation accuracy, and 70.44% test accuracy. This significantly outperformed all other configurations, with precision and recall metrics following similar patterns.

Among traditional feature extractors, HSV features with RandomForest achieved the second-best performance (30.91% test accuracy), while maintaining the fastest training time (0.46 seconds). This suggests a notable trade-off between computational efficiency and model performance.

Notably, both HOG and LBP features generally underperformed, with particularly poor results when paired with GradientBoosting (8.33% test accuracy) despite requiring the longest training time (27,842 seconds).

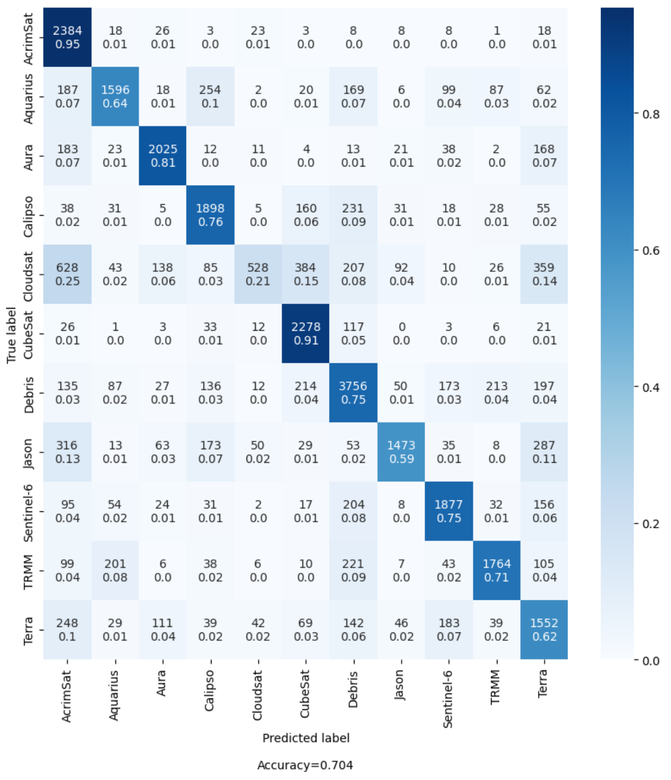

Fig. 7. Confusion Matrix of our best logistic regression model on ResNet101 feature embeddings

V. DISCUSSION

Our experimental results reveal several key insights into the effectiveness of different feature extraction methods and classification approaches for satellite imagery. We analyze these findings in the context of the unique challenges presented by space object classification.

A. Feature Extraction Performance

The superior performance of ResNet101 features (70.44% accuracy) compared to ResNet50 (16.90% accuracy) suggests that deeper architectures capture crucial satellite characteristics that simpler networks miss. This significant performance gap indicates that satellite classification requires sophisticated feature representation capable of handling subtle structural differences between classes. It is also possible that ResNet101 features generalize more easily to space objects due to an elevated representation of relevant samples in its training corpus.

Traditional computer vision features showed varying degrees of effectiveness. HSV histograms consistently outperformed both LBP and HOG features across all classifiers, suggesting that color and intensity information plays a more crucial role in satellite classification than textural or gradient features alone. This finding aligns with the challenging nature of our dataset, where satellites often appear as small objects with varying illumination conditions.

B. Model Architecture Implications

The success of logistic regression with ResNet101 features, despite its simplicity compared to more complex models like gradient boosting, suggests that the quality of feature extraction is more critical than classifier complexity for this task. This observation has important practical implications, as it indicates that computational resources might be better spent on feature extraction than on complex classification architectures.

C. Dataset-Specific Challenges

Analysis of our confusion matrices across experiments revealed several consistent challenges. The confusion matrix for our best model highlights these challenges in figure 7.

- **Visual Similarity:** Certain satellite pairs, such as CloudSat-AcrimSat and Terra-CubeSat, showed consistent misclassification patterns due to structural similarities.

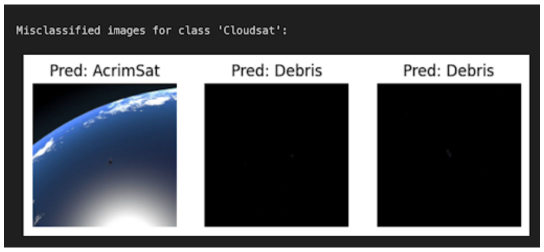
```
Misclassified images for class 'Cloudsat':
```

Fig. 8. Misclassification examples where CloudSat images were incorrectly labeled. Left: misclassified as AcrimSat; Center and Right: misclassified as debris. These errors highlight the challenge of distinguishing between structurally similar satellites and debris in space imagery.

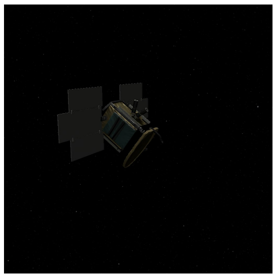

Fig. 9. Reference image of CloudSat, showing its distinctive features including solar panels and the main instrument housing. This represents the expected appearance for correct classification.

- **Scale Variability:** Performance degraded significantly when satellites appeared as small objects, particularly affecting the debris class classification.
- **Background Effects:** Earth's presence in images created classification challenges, especially when using traditional feature extractors.

D. Feature Extraction Trade-offs

Our experiments revealed important trade-offs between different feature extraction approaches:

- **Deep Features:** The deep learning-based features (ResNet101 and ResNet50) showed significantly contrasting performance. ResNet101 achieved the highest overall accuracy of 70.44% on the test set, suggesting good generalization. However, this came at a moderate computational cost of 385 seconds. Surprisingly, ResNet50, despite being a lighter architecture, took longer to process (664.47 seconds) while delivering substantially worse performance across all metrics (accuracy around 0.17). We speculate that this unexpected finding may have been a result of inconsistent experimental setups (i.e., not all constants were held equal).
- **Traditional Features:** HSV features, while less accurate, offered remarkably fast training times (0.46 seconds with Random Forest), suggesting their potential utility in resource-constrained scenarios.
- **Feature Robustness:** Traditional features showed particularly poor performance with rotated or partially obscured satellites, while deep features maintained better consistency across viewing angles.

E. Efficiency vs Accuracy

In addition to being our most accurate model, we optimized logistic regression to be our most computationally efficient model. With PCA-reduced components, our logistic regression test accuracy drops from 70.44% to 61.25%. Although our PCA components explain 95% of the variance within the ResNet101 features, the remaining 5% loss results in a 9-point drop in accuracy. However, when training a model for 125 epochs, we see the training time decrease from 5 minutes and 10 seconds to 4 minutes and 11 seconds. Additionally, the total inference time on the original embeddings is 0.0384 seconds for 30,000 embeddings, compared to 0.0281 seconds for the PCA-reduced embeddings. This corresponds to approximately 0.00000128 seconds per image and 0.00000094 seconds per image, respectively. While optimizing for efficiency is necessary in some cases, the storage and time savings here do not justify the sacrifice in accuracy.

Our experiments revealed important trade-offs between different feature extraction approaches:

These findings suggest that future work in satellite classification should focus on developing feature extraction methods that balance computational efficiency with robustness to orientation and scale variations. Additionally, the strong performance of ResNet101 features indicates that transfer learning from large-scale image datasets can be effectively adapted to specialized domains like satellite classification.

VI. Conclusion & Future Work

Our investigation of feature extraction approaches for satellite image classification yielded several important insights, while also highlighting significant methodological challenges. ResNet101 features paired with logistic regression emerged as the most effective approach, achieving 70.44% test accuracy - notably outperforming both simpler deep learning architectures (ResNet50) and traditional computer vision features. In addition, we recorded a similar validation accuracy of 70.01%, suggesting our model generalizes well on unseen images. The HSV color features demonstrated a compelling efficiency-accuracy trade-off, achieving moderate performance (30.91% test accuracy) with remarkably fast training times (0.46 seconds).

However, our experimental evaluation revealed important technical limitations that warrant future investigation. Several models defaulted to single-class prediction, as evidenced by identical accuracy and recall metrics of exactly 1/6, suggesting challenges in feature discrimination or model optimization. Additionally, anomalous patterns in our cross-split evaluations indicate potential issues in our experimental pipeline that should be addressed in future work.

Future research directions should focus on several key areas:

1. Experimental Robustness: Development of more rigorous evaluation protocols, including proper isolation between data splits and comprehensive validation of the metrics pipeline. This includes addressing the technical limitations identified in our current methodology.

2. Feature Extraction Enhancement: Investigation of why ResNet101 significantly outperformed ResNet50 despite its lighter architecture, potentially exploring the role of pre-training data and architecture depth in satellite image feature extraction.

3. Model Optimization: Further exploration of why certain feature-model combinations defaulted to single-class prediction, with particular attention to feature normalization and model hyperparameter tuning.

4. Computational Efficiency: Building on the promising performance-efficiency trade-off demonstrated by HSV features, future work should explore hybrid approaches that combine the discrimination power of deep features with the computational efficiency of traditional computer vision techniques.

5. Domain Adaptation: Given the unique challenges of satellite imagery, investigation into domain-specific pre-training or feature extraction techniques that better account for the characteristics of space objects.

Our findings suggest that while deep learning features show promise for satellite classification, careful attention must be paid to experimental methodology and evaluation metrics to ensure reliable performance comparisons. The identification of technical limitations in our current approach provides a roadmap for improving both the robustness and reliability of future satellite classification systems.

References

[1] ”SPARK 2021 Dataset,” University of Luxembourg, 2021. [Online]. Available: https://cvi2.uni.lu/spark-2021/spark-2021-dataset/

[2] S. Sica, E. Peng, J. Petrisko ”Satellite Image Classifier,” GitHub repository, 2024. [Online]. Available: https://github.com/seansica/satellite-image-classifier