281 Final Project: Intel Classification

Amanda Murray, John Van, Kang Wang, and Charles Lucas

Abstract

In an era marked by rapid technological advancements, Intel classification serves as a monumental milestone for bolstering security measures, growing environmental monitoring and shaping computer vision algorithms. This project focuses on developing an image classification model to categorize natural scenes into six distinct classes using the Intel Image dataset. Our approach involves a comprehensive set of features, including luminance histograms, object count, histogram of oriented gradients, Gray-Level Co-Occurrence Matrix (GLCM), bag of visual words, and the ResNET50 convolutional neural network. We capture these features through two models - a multi-class logistic regression and a random forest. Through extensive testing, we find that both models yield optimistic results and are ready to be enhanced to classify a larger subset of scenic labels. In a future where technology becomes more integrated into our daily lives, this project serves as a foundational step in the direction towards more advanced computer vision technologies, and stands as a preview of just how powerful computer vision can be.

1. Introduction

Our project aims to design an image classification model capable of categorizing natural scenes into six distinct classes. This model is trained on the Intel Image dataset,[1] which contains natural scenes separated into 6 classes: buildings, sea, glaciers, forests, mountains and streets. The dataset contains a total of 14,034 images with a fairly even class distribution. Each image in the dataset is 150x150 pixels and the dataset contains a mix of color and grayscale images. The table below contains sample images from each class, as well as the total number of images in the classes:

Buildings *(2191 training images)*		**Sea** *(2274 training images)*		**Mountains** *(2512 training images)*	
					
Glaciers *(2404 training images)*		**Forest** *(2271 training images)*		**Street** *(2382 training images)*	
					

Figure 1: Sample Images

Given the approximately 2000 training images per class, our model will use specific image features to accurately distinguish one class from the other. These image features described below include luminance histograms, histogram of oriented gradients, and complex neural network features. Through this comprehensive approach of combining multiple

[1] https://www.kaggle.com/datasets/puneet6060/intel-image-classification

feature values in a clear and concise model, our project aims to classify natural scenes with a high degree of accuracy and efficiency.

2. Features

2.1 Image Pre-Processing

While the majority of the images were 150x150 pixels, any that were not were re-sized by cropping to the center and rescaling. Additionally, because over 500 of the images were grayscale, all images were converted to grayscale for classification. Finally, for the majority of the features, with the exception of luminance histograms, histogram equalization was used to normalize the contrast of the images across all classes and make it easier to pick up certain features like key points or edges.

The images were also split into a train, validation, and test set. The training set consisted of 11,230 images, the validation set consisted of 2,804 images, and the test set consisted of 3,000 images. All of the visualizations below were created using the training set.

2.2 Luminance Histograms

Luminance histograms can be used to pick up differences in the luminance of the image. This feature was chosen because we hypothesized that some of our classes may have similar shapes but different luminance values. In particular, a lot of the glacier images look very similar to mountains, but are covered in snow. The luminance histogram may pick up the addition of white pixels, allowing for distinction of these classes. To generate the feature vector, values of the greyscale image were scaled 0 to 1, and separated into 25 bins, and the frequency in the bins were used as the feature vector. The plot below shows examples of these histograms for six images, one from each class.

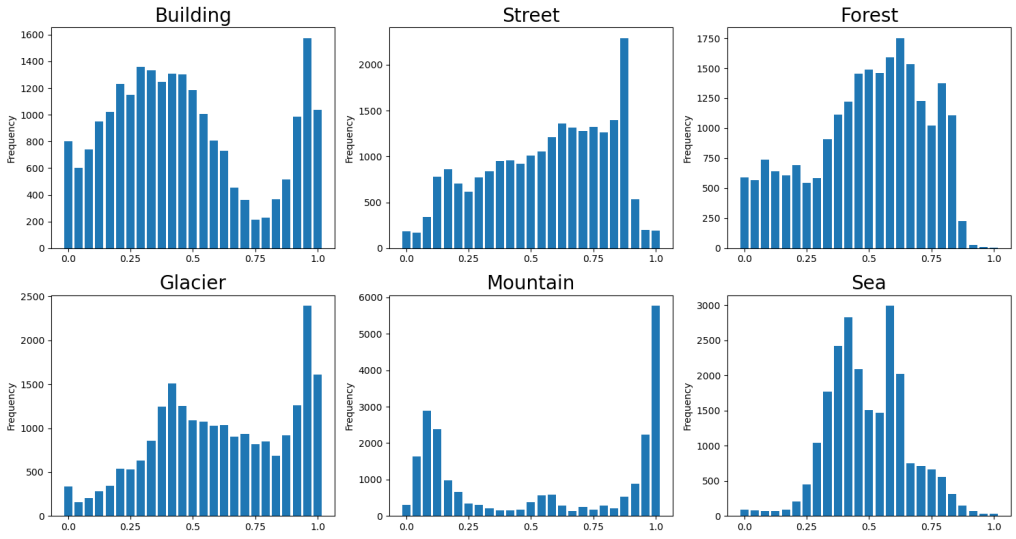


Figure 2: Sample Luminance Histograms by Class. Each histogram was separated into 25 bins.

The histogram and t-Distributed Stochastic Neighbor Embedding (t-SNE) plot below show how well this feature alone may classify images.[2] t-SNE is a non-linear method of dimensionality reduction that can be used to visualize how well a feature is distinguishing between classes. If clustering is present in the t-SNE plot, this means the feature is adequately distinguishing between classes for the classification task. However, a lack of clustering does not necessarily mean the

[2] https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

feature will not be effective in distinguishing between classes, since the dimensionality reduction can reduce some of the variance beneficial for the classification task.

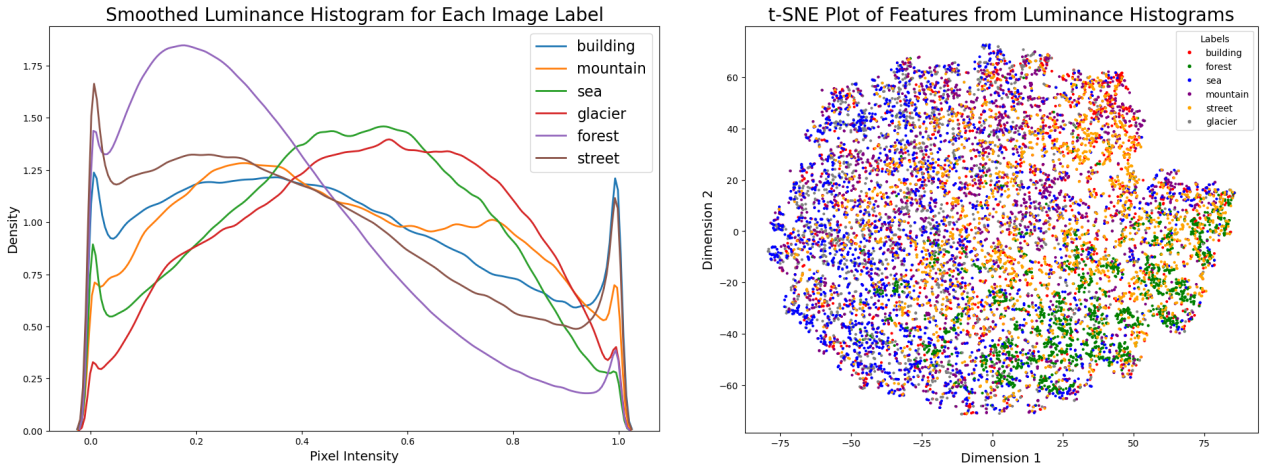


Figure 3: Histograms and t-SNE Plots for Luminance Histograms

As hypothesized, we can see some distinction between the mountain and glacier classes, with glaciers having a higher pixel intensity, potentially from the snow. The t-SNE plot does not show clear clustering between classes.

2.3 Object Count

Object count is a feature that detects the number of unique objects in an image using edge detection. We hypothesized that this feature may help detect differences between street and building images, which are very similar in terms of what they are depicting, but the street images may have more unique objects due to people, cars, or stones on the road. To create this feature, images were processed using gaussian blur to reduce noise then and Canny Edge Detection was applied using cv2.[3] Finally, the "Find Contours" function was used to count the number of objects in the image.[4] The histograms below show the distribution of object count across classes.

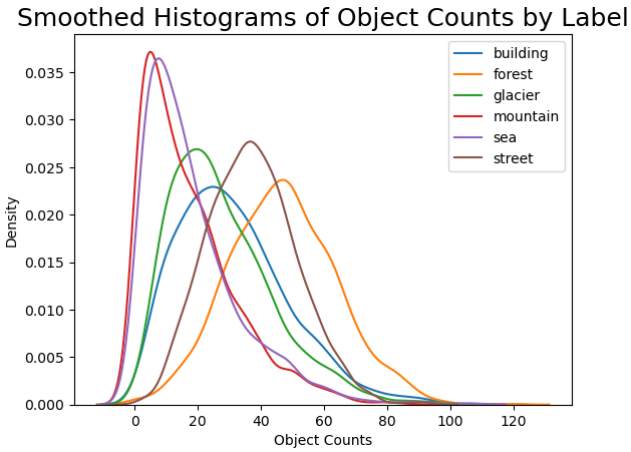


Figure 4: Histograms of Object Count by Label

As hypothesized, object count is picking up some differences between buildings and mountains, as well as differentiating between other classes.

[3] https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html
[4] https://docs.opencv.org/3.4/df/d0d/tutorial_find_contours.html

2.3 Histogram of Oriented Gradients (HOG) Feature Vector

Histogram of Oriented Gradients (HOG) is a powerful feature that can be used for many tasks, including object recognition.[5] HOG feature vectors show the magnitude and orientation of gradients at different locations in the image. We chose this feature for our image set because our image classes appear to have distinct gradients. For example, buildings or trees in the forest may be mostly vertical, whereas mountains or glaciers may have diagonal gradients leading up to the peaks.

The images below display sample HOG feature vectors for example images from each class. Parameters of (20, 20) pixels per cell and (1, 1) cells per block were chosen in order to maximize the patterns HOG displayed, while minimizing the length of the feature vector for efficiency and generalizability. This generates a feature vector of length 2,025. The HOG feature vectors are picking up some of the differences in orientation of gradients across the images.

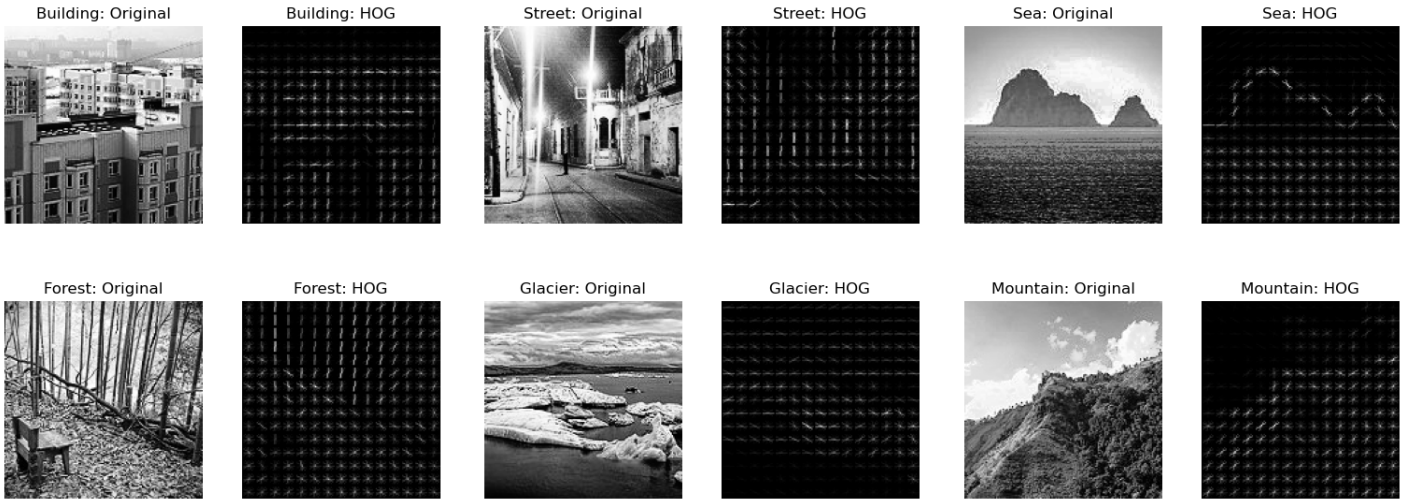


Figure 5: Sample Histogram of Oriented Gradient Feature Vectors for Six Images

Finally, a t-SNE plot of the feature vectors was created to check how well these feature vectors were working with differentiation. As shown below, the t-SNE plot is showing some differentiation of the classes.

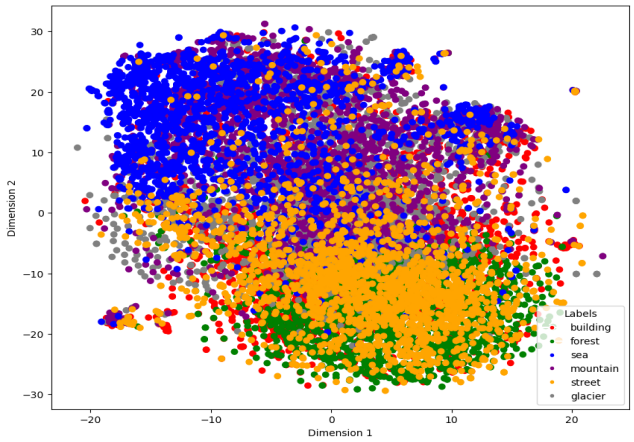


Figure 6: t-SNE Plot of HOG Feature Vectors

[5] https://ieeexplore.ieee.org/document/1467360

2.4 Gray Level Co-Occurrence Matrix

Due to the differences in texture across the scenes, we wanted to include a method for differentiating between these textures. To do this, we utilized GLCM, or Gray-Level Co-Occurrence Matrix, which measures the co-occurrence between pixels at a certain distance and angle. To determine which distance between pixels may be most effective for detecting differences in texture, we conducted some experiments with different sizes. The figure below shows an example of some of these experiments, plotting smoothed histograms for different sizes and angles using the contrast and correlation values of the GLCM matrix for the image.

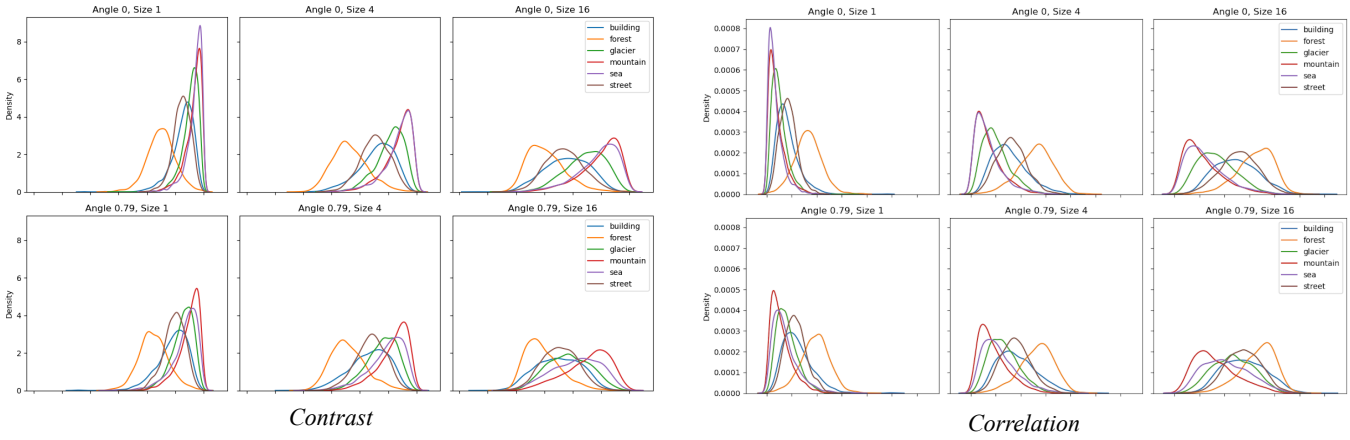


Figure 7: Experiments for Selecting Pixel Distance for GLCM Matrix

After running these experiments, we selected a pixel of 4 (the middle row of plots shown above), since this distance was differentiating between the classes better than a distance of 1, while maintaining a more limited variance within the classes than the distance of 16. We chose the angles 0, $\pi/4$, $\pi/2$, and $3\pi/4$ radians. The images below show examples of the GLCM matrices for 6 images with an angle of 0.

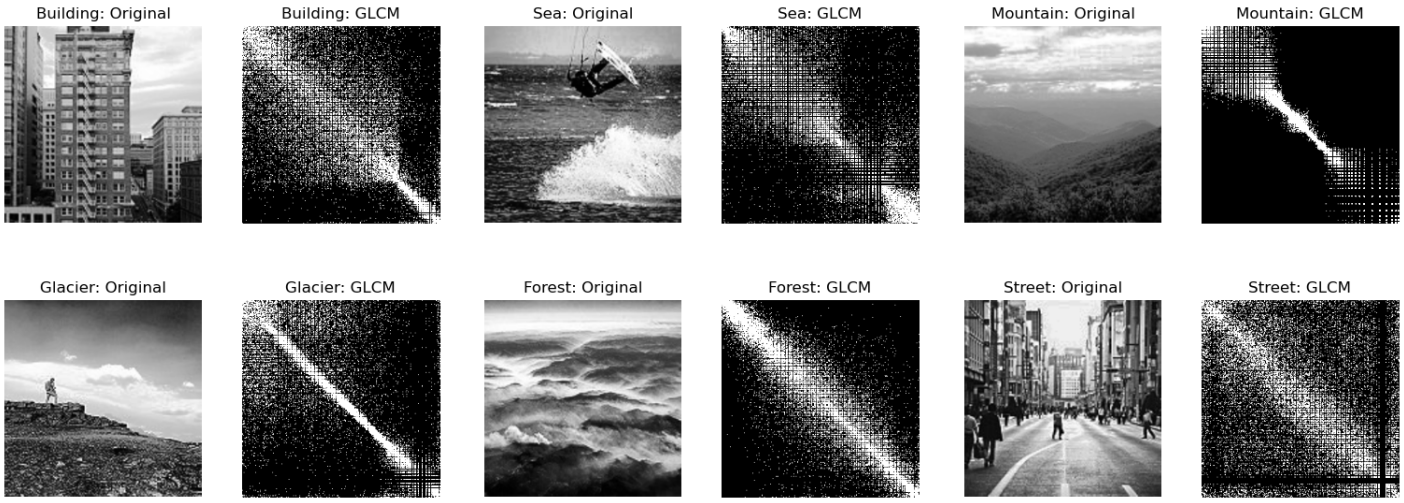


Figure 8: Examples of Raw Gray-Level Co-Occurrence Matrix for Each Class

Next, we calculated standard metrics for the GLCM matrix of the images, including contrast, correlation, dissimilarity, energy and homogeneity for all 4 angles. The plots below show histograms of these metrics for the angle 0.

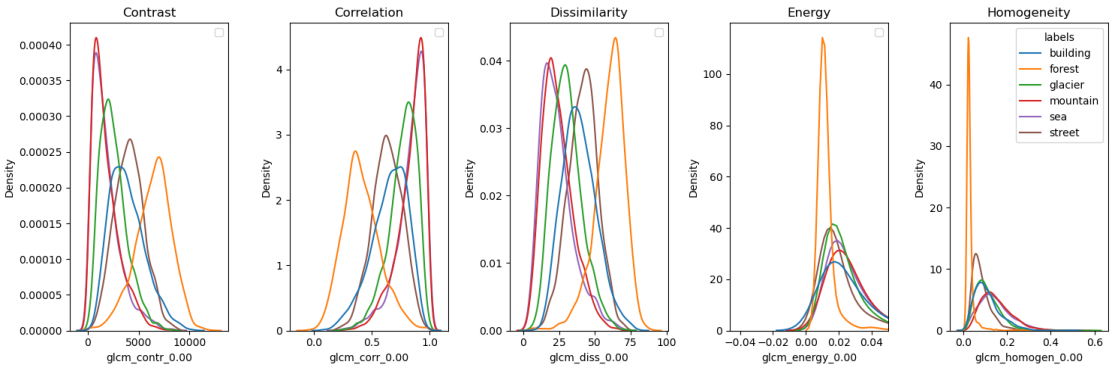


Figure 9: Smooth Histograms of GLCM Metrics for Angle=0

As shown in the histograms above, the contrast, correlation, and dissimilarity metrics seem to be differentiating between objects more effectively than homogeneity or energy.

Finally, to check how well these feature vectors were differentiating between classes when combined, we created a t-SNE plot using all GLCM features and some differentiation between classes, as shown in Figure 10.

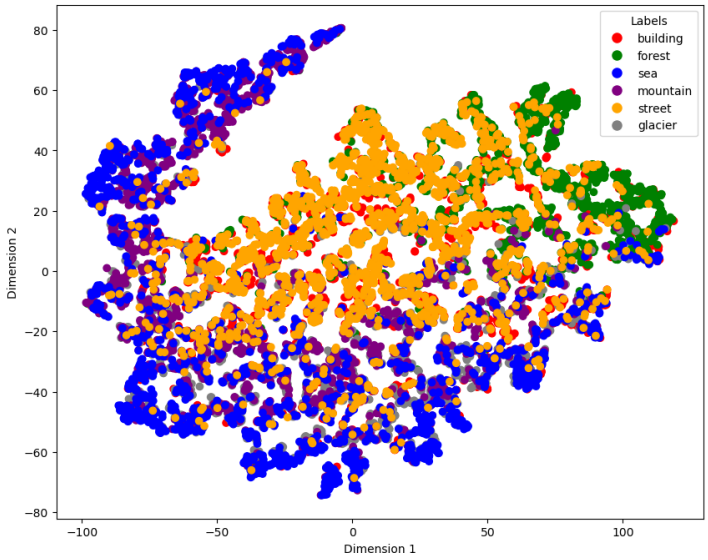


Figure 10: t-SNE Plot of GLCM Feature Vectors

2.5 Bag of Visual Words/Brisk

Bag of Visual Words is a multi-step feature that is used to identify differences between images in the areas surrounding keypoints.[6] To calculate this feature vector, we first used BRISK to identify and describe keypoints in the image.[7] Below are examples of keypoints for 6 images.

[6] https://towardsdatascience.com/bag-of-visual-words-in-a-nutshell-9ceea97ce0fb
[7] https://docs.opencv.org/3.4/de/dbf/classcv_1_1BRISK.html

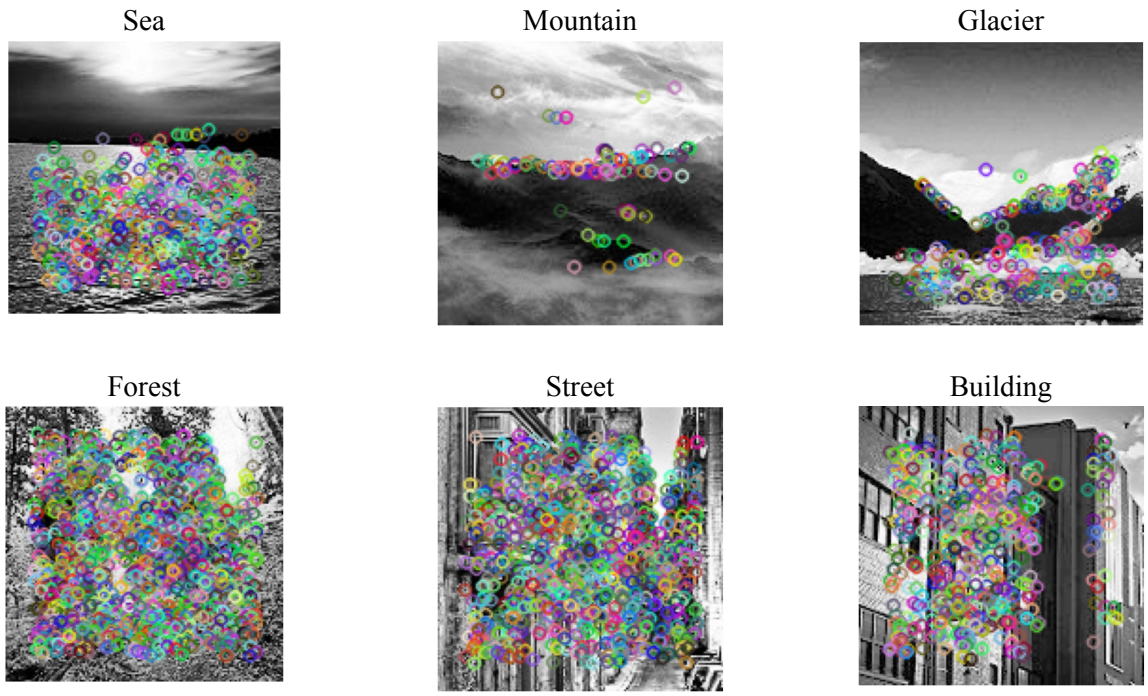


Figure 11: Key Points for 6 Sample Images

We then used k-means classification to sort these descriptors into *k* groups. We experimented with different values of *k* (100, 150, 200), visualizing the results using t-SNE, and decided on k=200. We then sort the descriptors into these 200 'visual words', and count the number of times each word appears in each image. Next, we use TF-IDF vectorization, a common vectorization approach for language-based bag of words methods, which scores the 'uniqueness' of the visual words by taking into account the total number of words in that image, and how many times that word appears across all images.[8] The two histograms below show distributions of TF-IDF scores for the first two "words" for all images in the 6 classes.

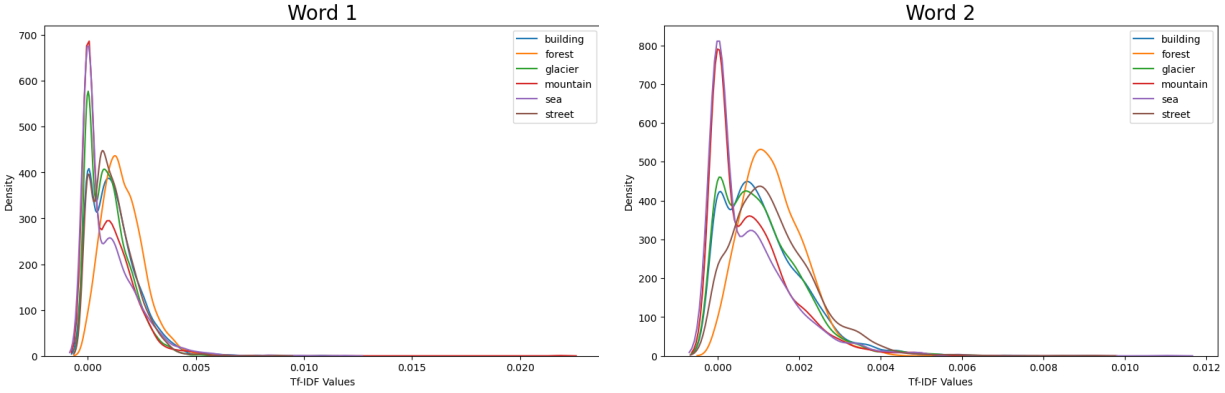


Figure 12: Histograms of TF-IDF Values for First Two "Words"

Additionally, because the number of keypoints in the image seemed to be different across classes, we appended this as an additional feature. The histogram below shows differentiation of the total "words", or keypoints, in each image across classes.

[8] https://towardsdatascience.com/tf-idf-simplified-aba19d5f5530

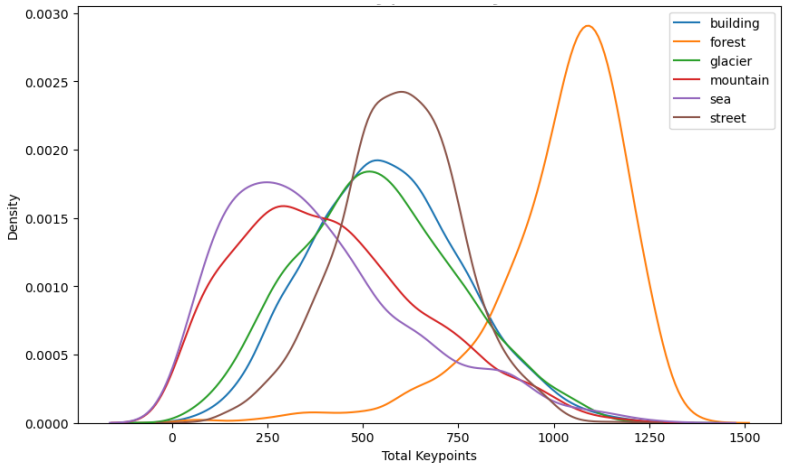


Figure 13: Total Number of "Visual Words" per Image by Class

To check how well the bag of visual words features were differentiating between classes, we plotted them in a t-SNE plot shown in figure 14, but did not find significant differentiation.

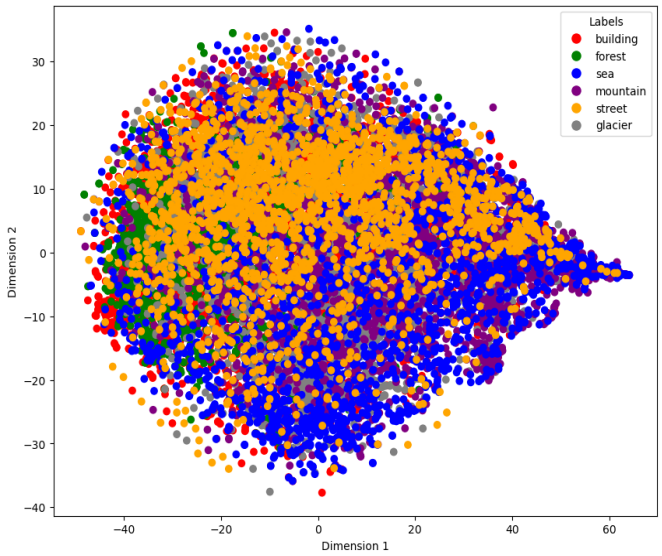


Figure 14: t-SNE Plot of BOVW Feature Vectors

2.6 ResNET50

ResNET50 is a pre-trained convolutional neural network, trained on more than a million images from the ImageNet database.[9] The network was trained on image classification tasks, and therefore, can be powerful for generating feature vectors for classification, even without additional fine-tuning. ResNET50 is 50 layers deep and outputs a feature vector of 1,000 values. To input our images, we had to resize them to 224 by 224 pixels and repeat the gray scale image channel 3 times to replicate an RGB image. The t-SNE plot in figure 15 was created using the 1000-length feature vector output from ResNET50.

[9] https://www.mathworks.com/help/deeplearning/ref/resnet50.html

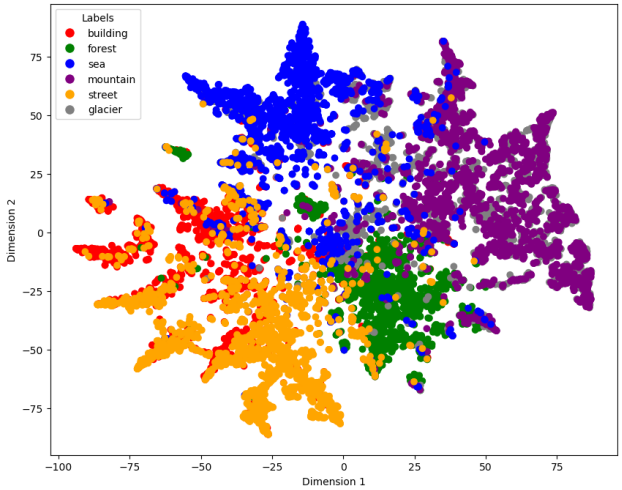


Figure 15: t-SNE Plot of ResNET Feature Vectors

2.7 Principal Component Analysis

PCA is used here to reduce the dimensionality of the data while retaining as much of the variance in the dataset as possible. Figure 16 explains the variance captured by PCA for luminance histograms, histograms of oriented gradients (HOG), gray-level co-occurrence matrices (GLCM), and bag of words (BOW) features, as well as deep learning features from a pre-trained network like ResNet. Each graph indicates how many principal components are needed to represent the data's variability effectively.

Luminance Histogram PCA shows a saturation effect as the number of features increases, with explained variance approaching a high value quickly. This indicates that a small number of principal components are sufficient to capture most of the variance in the luminance histogram data. The curve of HOG PCA steeply increases at first and then gradually plateaus, indicating that a substantial amount of variance is captured by the initial components, but additional features continue to add information. The plot suggests that more components are needed to capture a high percentage of variance in the HOG features compared to the luminance histogram. The explained variance in GLCM PCA rapidly reaches a plateau, which suggests that only a few principal components are required to capture most of the variance within the GLCM features. The plateau begins early, showing that additional features beyond a certain point do not contribute significantly to increasing the explained variance. BOVW PCA also features a steep initial slope, followed by a gradual approach to the plateau, indicating that while a significant amount of variance is captured by the first few components, additional components still contribute to capturing more variance. This linear increase from BOVW is because the process for creating the descriptors and words is designed so that each word should add a new piece of information or feature, so including all of them is necessary to maintain full variance. The ResNet PCA plot starts with a steep curve, showing a large amount of variance captured by the initial components. The curve then slowly levels off, suggesting that while the first few components are quite informative, additional components add value up to a certain point before the gain in explained variance diminishes.

The overall trend in these plots indicates that while some features have their variance captured by a few principal components (as seen in the Luminance Histogram and GLCM plots), others, like HOG, BOW, and ResNet, may require more components to capture a similar level of variance, reflecting the complexity and dimensionality of the data represented by those features.

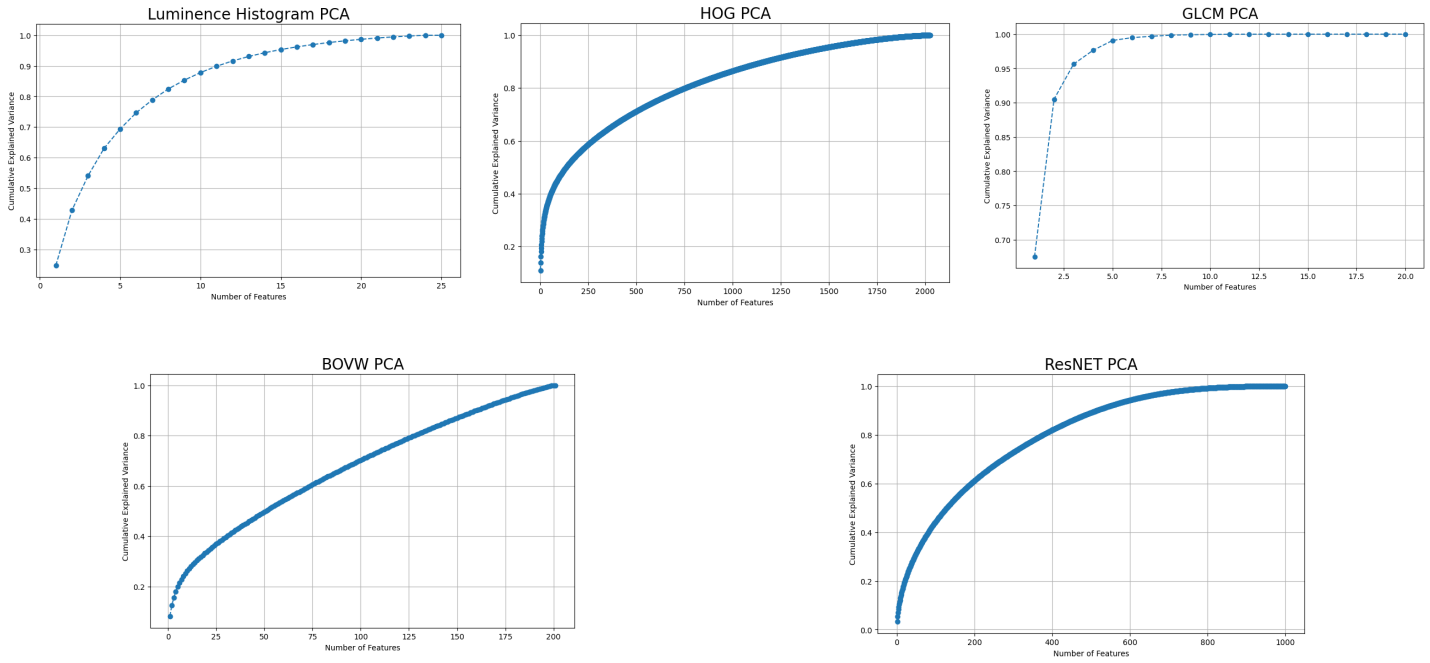


Figure 15: PCA Plots of number of features vs. Cumulative Explained Variance for unique features.

Figure 17 combines all the features to show the overall variance explained, the plot suggests that a relatively small number of principal components may be sufficient to capture a large proportion of the variance in a dataset that includes a diverse set of features. Deciding how many components to retain would involve a trade-off between the explained variance (information retention) and the desire to reduce dimensionality (complexity and computational efficiency). It would be critical for justifying the number of features retained after dimensionality reduction for further machine learning or pattern recognition tasks.

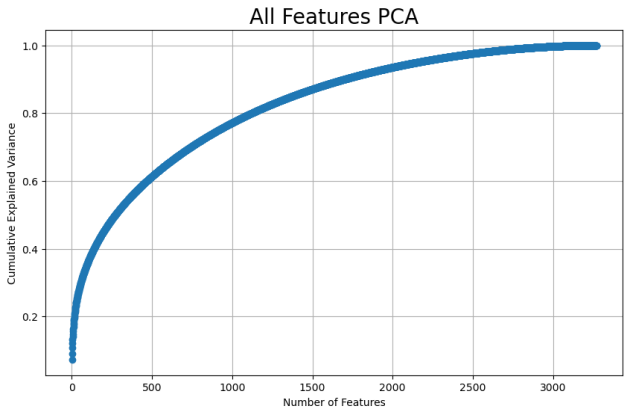


Figure 15: PCA Plots of number of features vs. Cumulative Explained Variance for all features combined.

3. Classification

In this segment, we detail the construction and examination of two distinct classifiers. The initial approach adopted was a multinomial logistic regression, apt for categorization into one of six classes. Our alternative strategy employed a random

forest algorithm, recognized for its robust default performance with non-numeric data types. We utilized feature extraction methods detailed in the previous section, leading to feature vectors of 3274 dimensions.

Our dataset division allocated 11,230 images for training, 2,804 for validation, and 3,000 for the test phase. We calibrated critical hyperparameters using a grid search strategy, optimizing for accuracy as shown in Table 1. The optimal hyperparameters were then applied to train the models, with subsequent performance assessments conducted on the validation and test subsets (Table 2) .

Table 1. Optimization of Model Parameters via Grid Explorations

Model	Parameter	Search Values	Best Value	Best Accuracy	Search Time
Logistic Regression (one vs rest)	max_iteration	100, 500, 1000, 2000	2000	0.894	714.63 seconds
Random Forest	n_estimators	10, 50, 100, 200	100	0.916	198.53 seconds
	max_depth	2, 4, 6, 8, 10	10		

Table 2. Classifier efficacy Post-Parameter Optimization

Model	Best Hyperparameters	Accuracy		
		Training Set	Test Set	Validation Set
Logistic Regression (one vs rest)	Max_iterations = 2000	0.9126	0.894	0.9058
Random Forest	N_estimators = 100	0.979	0.910	0.904
	Max_depth = 10			

3.1 Model Results

3.1.1 Logistic Regression Results

A logistic regression's strengths lie in its ability to solve binary classification problems, where it uses probability to predict whether a set of data belongs in a specific class. Although it specializes in binary classification, logistic regression is still very powerful for multi-class classification with the help of the 'One versus Rest' strategy.

In a multi-class classification problem, the 'One versus Rest' strategy simplifies a multiclass problem into a series of binary class problems for each class in the question. It does this by training a binary logistic regression classifier for each class, where the positive class consists of one class while the negative class is a grouping of all the other possible classes. We proceed to pit one class against all the rest until there is a binary classification for every class. Finally, the class with the highest probability from all the binary regressions is chosen as the final prediction. We also employed a hyperparameter search which resulted in setting our hyperparameter, max_iterations, to 2000.

As seen in the confusion matrix below, the 'One vs Rest' strategy was successful in identifying relationships within the data by yielding a test accuracy of 89.4%. Unsurprisingly, the model struggles the most at capturing the key differences

between mountains and glaciers. The model also struggles to differentiate between mountain and sea, which makes sense since some sea images can contain mountains in the background.

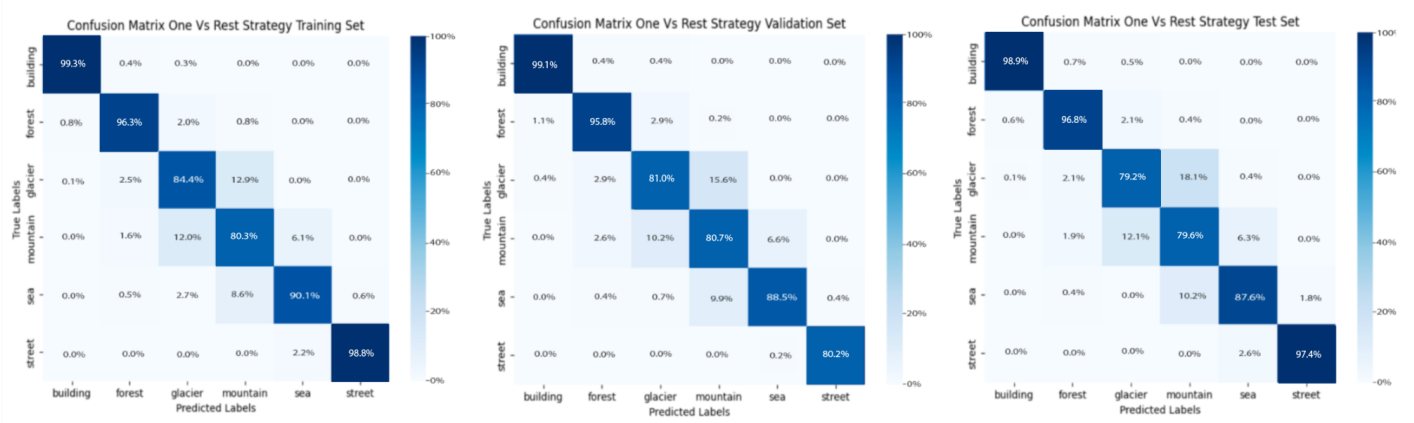


Figure 18. Comparative Confusion Matrices for Logistic Regression: Building, Forest, Glacier, Mountain, Sea, and Street Classifications Across Training, Validation, and Test Data Sets (Class sizes are not equal, but are very similar in size)

3.1.2 Random Forest Results

A random forest classifier works by generating a large number of decision trees and aggregating the results for a prediction. The individual decision trees take in new data and "vote" on which classification they think the new data belongs to. The winner of this "election" is the predicted class.

The random forest model attained an accuracy of 91.8%. A confusion matrix of the predictions shows what the model understands well and what confuses the model. Similar to logistic regression, the random forest model is easily confused about things that a human would also be confused about. The model confuses buildings and streets as well as mountains and glaciers. We can also see the model understands what a forest is very well. Sometimes the model confuses mountains and glaciers with the sea, which again may be due to some sea images in the dataset containing mountains in the background.

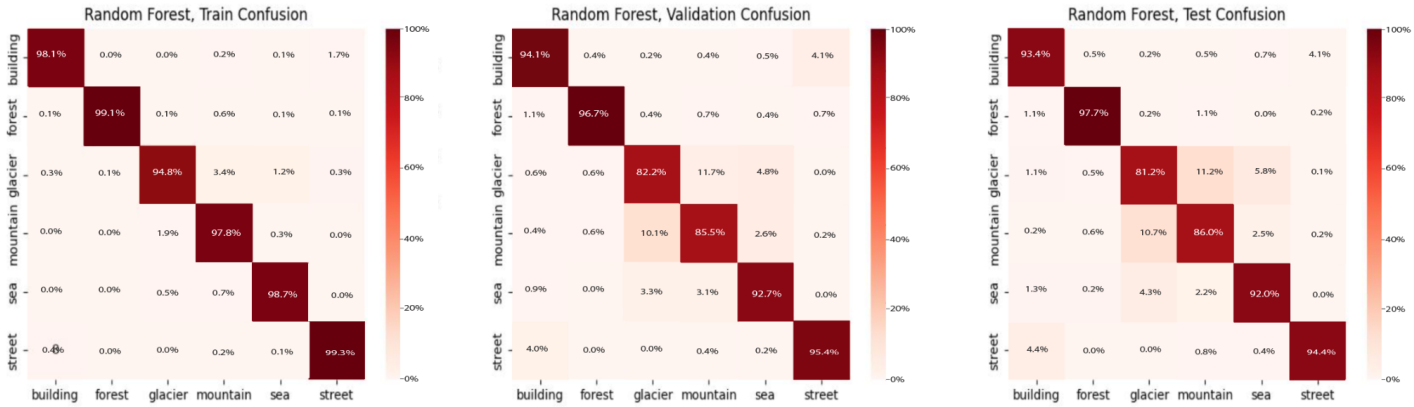
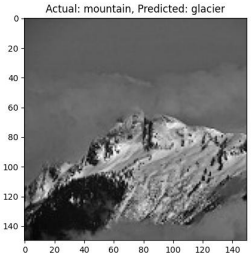


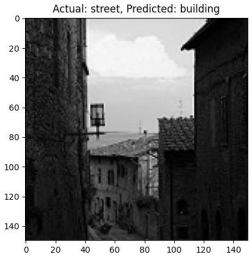
Figure 19. Comparative Confusion Matrices for Random Forest: Building, Forest, Glacier, Mountain, Sea, and Street Classifications Across Training, Validation, and Test Data Sets (Class sizes are not equal, but are very similar in size)

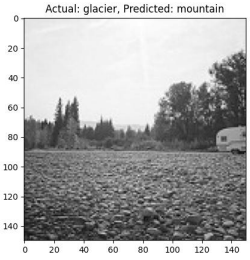
3.1.3 Model Performance

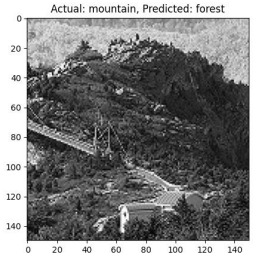
The confusion matrices reveal that both models demonstrate commendable classification accuracy across the six categories (building, forest, glacier, mountain, sea, and street) in the training, testing, and validation sets. The logistic regression model, while slightly less accurate, is competitive, indicating robust generalization from training to unseen data. The random forest model, on the other hand, shows higher accuracy, especially on the training set, but this could indicate a tendency to overfit, which is suggested by the slight drop in accuracy on the validation set.

Here we listed misclassified images samples, which allow for a deeper understanding of the model's limitations.The first group of images represents instances where the confusion is justifiable, as these images could potentially mislead even human observers due to their ambiguous characteristics or unusual representations of the categories.

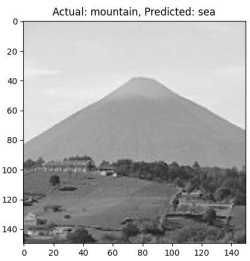


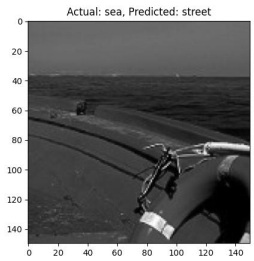




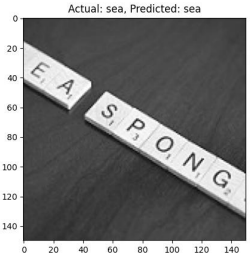


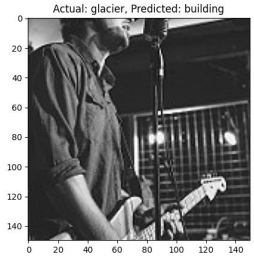
The second group highlights errors that wouldn't typically confuse a person, suggesting areas where the model may benefit from further refinement or more nuanced feature extraction.





Finally, the presence of some nonsensical images in the dataset—likely outliers or mislabeled instances—illustrates the challenges of 'big data' environments where noise is inevitable. Fortunately, such images are infrequent enough not to significantly degrade overall model performance.





3.2 Generalizability

We believe that this approach–using computer vision to produce features and training a logistic regression or a random forest on the generated features is very generalizable. The real work behind this model was on the frontend producing the features, and the kinds of features required for a good model probably depend on the domain. That said, a diverse set of features, like we have used here, will likely work well for any broad classification problem, as long as those features are tailored to the needs of the task. This approach may fail when trying to differentiate between very similar image classes

(like images of potentially cancerous moles), but will likely succeed on diverse image classes (like vegetables, clothing, landscapes, etc).

3.3 Efficiency

In our assessment of classifier efficiency, we have documented training and prediction times for our two models: Logistic Regression and Random Forest by using the full feature set and a reduced set via PCA with 30 components (applied to all features). We applied PCA to the full feature vector. The computational environment was standardized on a 3.6GHz Intel i7 8-core processor desktop with 64GB memory to ensure consistent benchmarking.

For the complete feature set (Table 3), Logistic Regression demonstrated longer training times, attributed to the iterative nature of optimization for a large number of features. Random Forest, with its ensemble approach, completed training more quickly due to parallel tree construction.

Table 3. Time Efficiency Metrics for Logistic Regression and Random Forest Models on the Full 3274-Feature Dataset Without Hyperparameter Optimization

Model	Model Training Time	Prediction Time		
		Training Set	Validation Set	Test Set
Logistic Regression (one vs rest)	429.365198s	0.273021s	0.147114s	0.149617s
Random Forest	43.345133s	0.379617s	0.181529s	0.186687s

The application of PCA resulted in a significant reduction in feature dimensions, leading to faster training times for both models (Table 4). Logistic Regression, in particular, benefited from dimensionality reduction, showing a marked decrease in training time. Random Forest training time also decreased, but to a lesser extent, given its inherent efficiency with high-dimensional data.

Table 4. Time Performance Metrics for Logistic Regression and Random Forest Models with 30 Principal Components Analysis Reduction

Model	Hyperparameter Tuning Time	Model Training Time	Prediction Time		
			Training Set	Validation Set	Test Set
Logistic Regression (one vs rest)	31.160633s	0.152606s	0.000998s	0.000000s	0.000997s
Random Forest	227.602848s	9.436282s	0.211646s	0.068425s	0.071809s

In optimizing for efficiency, we considered the trade-off between the computational speed and the accuracy of the classifiers. In this regard, we tested PCA at 2500,1500, 750, 300 and 30 components, which corresponds to cumulative explained variance around 95%, 85%, 70%, 50% and 10% respectively (Figure 17). As illustrated in Table 5, Logistic Regression maintains relatively stable accuracy levels with PCA applied even at 300 components. However, there is a noticeable drop in accuracy as we further reduce the variance threshold, highlighting a diminishing return on efficiency gains. Random Forest, on the other hand, shows a significant reduction in accuracy with PCA applied, suggesting that the reduction in feature space compromises the model's performance more drastically than with Logistic Regression.

Table 5. Comparison of Logistic Regression and Random Forest Model Performances at Different PCA Dimensionalities

Model		No PCA	PCA: 2500	PCA: 1500	PCA: 750	PCA: 300	PCA: 30
Logistic Regression (one vs rest)	Accuracy	0.886	0.867	0.853	0.849	0.853	0.715
	Run Time	441.65s	40.50s	27.53s	13.24s	2.81s	0.18s
Random Forest	Accuracy	0.907	0.678	0.701	0.732	0.740	0.714
	Run Time	41.79s	47.35s	35.57s	25.43s	15.53s	4.69s

Given these observations, Logistic Regression paired with PCA at around 300 components emerges as the more balanced choice. It strikes a strategic compromise by achieving a substantial reduction in training time while sustaining high accuracy levels. This optimized approach underscores the importance of feature selection in model performance, especially when operating within computational or time constraints.

In summary, the relative trade-offs for the solution involve balancing accuracy against computational efficiency. Logistic Regression demonstrates a minor decrease in accuracy with significant computational gains, particularly in training time, making it an appealing choice for scenarios with computational constraints. Random Forest exhibits a more substantial trade-off between accuracy and efficiency. The model loses more accuracy as the number of PCA components decreases, suggesting that Random Forest requires a richer feature set to maintain high performance.

4. Conclusion

Our project focuses on the development of an image classification model with the primary goal of categorizing natural scenes into six distinct classes using the Intel Image dataset. With over 14,000 images spanning a diverse range of landscapes, our dataset provides a robust foundation for training a sophisticated classification model.

The features used to shape our classification model include luminance histograms, object count, histogram of oriented gradients, GLCMs, bag of visual words, and the convolutional neural network, ResNET50. Our multi-faceted feature approach was designed to capture key features in each image and highlight specific areas that uniquely defined each class.

Both our implementations of a multi-class logistic regression and a random forest were able to successfully incorporate our designated features into models and accurately classify the images into the 6 desired labels. Our logistic regression utilized the ‘One vs Rest’ strategy which trained multiple binary classifiers in parallel and identified the highest probability class. Our random forest ran an ensemble of decision trees that built an overall classifier for our problem. Both models performed exceptionally well at classifying the building, forest, sea, and street classes; however, it struggles to differentiate between the mountain and glacier classes.

Given more time and resources, we would optimize our feature selections. With over 3000 feature values going into our machine learning models, we would employ feature selection algorithms and dimensionality reduction methods to try to maximize our model accuracies while minimizing as many model components as possible. We would also love to expand our model’s capabilities by introducing more classes for it to predict. Nature scenes aren’t limited to just the 6 listed in our dataset, and augmenting our model to incorporate a broader set of labels would prove to be invaluable in real world applications.

As advancements in technology continue to grow, we remain enthusiastic about the importance of computer vision technology in daily applications. Perfecting intel classification is a minor, but fundamental step in the direction towards a future filled with technological breakthroughs.