Final Project Report

281 Spring 2024 - River Schieberl, Sophia Chen, Sanurak Natnithikarat

Problem Statement

There is a popular online game called GeoGuessr in which the player is dropped into a Google Street View somewhere in the world and the player has to click somewhere on the map to guess where they are in the world. This is a problem that humans have become fairly good at, often being able to consistently come very close to the true location. Some experts at the game have become so good at the game that they add extra challenges like making the screen black and white, blurry, or only able to see part of the screen.

Humans employ a variety of techniques that often incorporate knowledge of the “real world” such as vegetation, building architecture, and road configuration. Our project’s goal is to train models to play this same game, but we will simplify the task to match our available data.

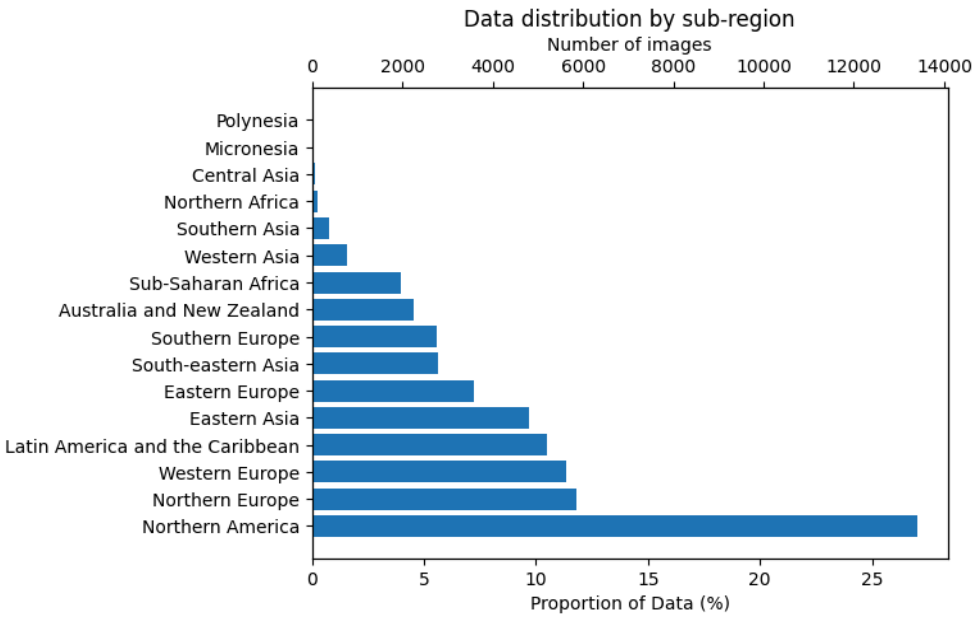
Data Source

https://www.kaggle.com/datasets/ubitquitin/geolocation-geoguessr-images-50k

Data Description

The dataset linked above contains around 50,000 images from 129 countries. We felt that there would not be substantial variation between countries to warrant having so many classes to predict, so we decided to merge countries together into subregions to cluster similar regions.

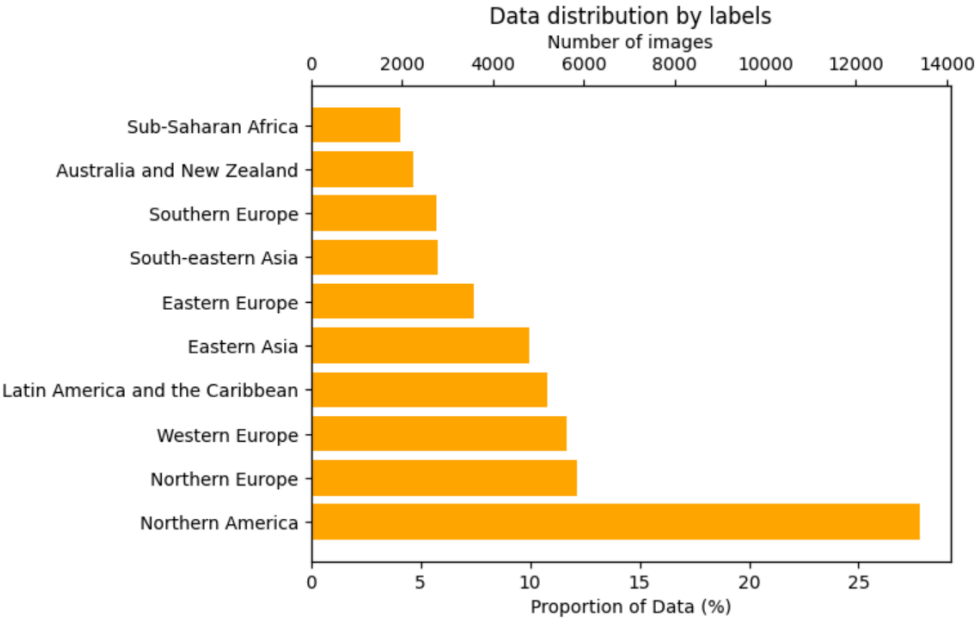
A map showing the subregions that the countries are grouped into.
This map is very similar to the actual groupings, but there are some minor differences.



A histogram showing the number of images for each subregion.

However, we have identified a strong data imbalance persisting after checking the dataset distribution based on these labels. Northern America contains nearly a quarter of the data and Polynesia contains a single digit number of images.

We propose dropping Polynesia, Micronesia, Northern Africa, Central Asia, Southern Asia, and Western Asia. The more balanced distribution of labels is shown below, with the smallest class containing 1,960 images.



A histogram showing the number of images for each subregion after modifying smaller classes.

Data Cleaning

There were several steps involved in the cleaning process. We first started by ensuring that all of the images in our dataset were of a consistent size. We noticed that there were 48,177 images that all had the same resolution (1536 x 662) but there were 54 images that had a resolution of (1030 x 703). Since these outlier images were not the same aspect ratio, we eliminated these entirely.

The next step was to remove the excess images from the larger classes to maintain an equal balance of images in all categories. Since we knew we would have the ability to double the size of all classes from data augmentation, we randomly downsampled every class to have a maximum of twice the smallest category (Sub-Saharan Africa).

Afterwards, we cropped the images to remove the extraneous user interface elements which only add unnecessary noise to the model. In order to make each image smaller for easier processing, we took two subregions from each larger image.

The red region shows approximately where the two subregions were taken from.

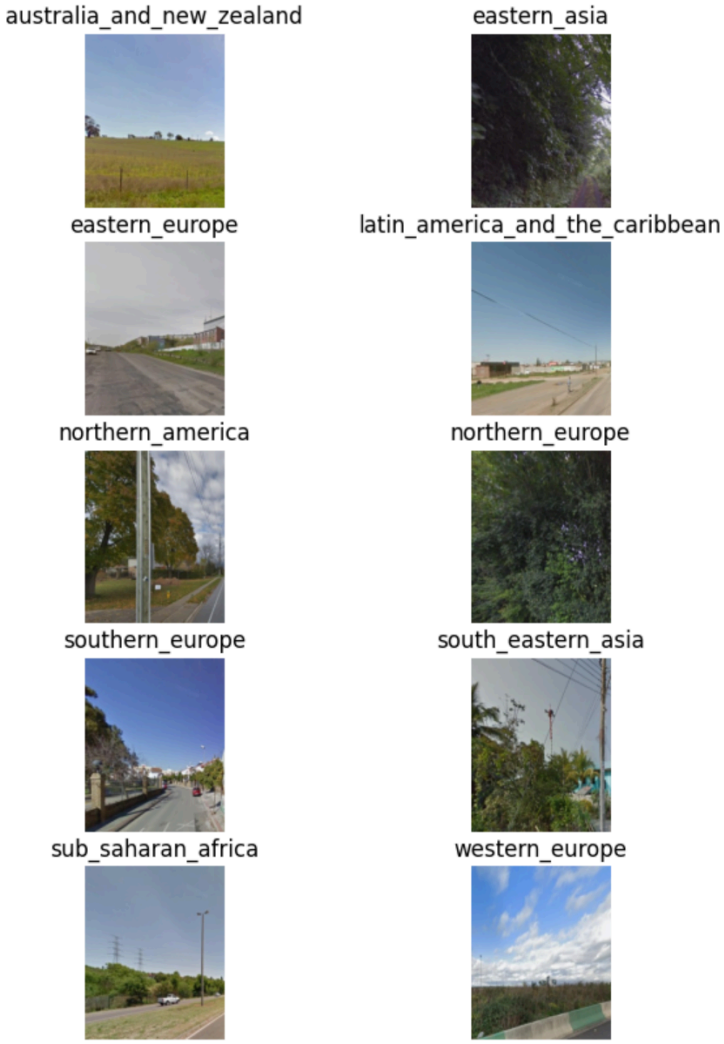
Since we did not need the finer details, we shrank every image down to a resolution of (235 x 293). At this point, we had the following number of images in every class:

- 7,828 in Northern America
- 7,828 in Northern Europe
- 7,828 in Western Europe
- 7,828 in Latin America and the Caribbean
- 7,828 in Eastern Asia
- 7,148 in Eastern Europe
- 5,552 in South-eastern Asia
- 5,506 in Southern Europe
- 4,516 in Australia and New Zealand
- 3,914 in Sub-Saharan Africa

Data Augmentation

In order to make each category have the same number of images and avoid introducing bias towards one particular class, we augmented the smaller classes by flipping the images along the vertical axis until every class had exactly 7,828 images in it. The images which were flipped were randomly selected, but data points which originated from the same base image were always kept together throughout the train-validation-test split.

We considered flipping the right subregion without flipping the left subregion during the cropping phase to take advantage of the symmetry as discussed in class, however we decided that it would introduce more bias since the road is not consistently running through the middle of the cropping divider.



An example of an image from each class.

Afterwards, all of the data was split into training, validation, and testing sets using a 70%, 20%, and 10% ratio resulting in 54,790 training images, 15,650 validation images, and 7,840 test images.

Feature Extraction - Methodology

There were three feature extraction methods our team used: Local Binary Patterns (LBP), Histogram of Oriented Gradients (HOG), and a pretrained Convolutional Neural Network (CNN) called DenseNet 121.

We were interested in exploring the local binary patterns because they have the ability to describe the texture of an image and it is intuitive that different regions around the world would have their own distinct textures from things like buildings, streets, etc. The resulting feature vector had a length of 20.

The histogram of oriented gradients is another feature that we looked forward to researching as it has potential for describing the structure and shape of an object. Since we anticipate different regions to have sufficient variability in their structures, we had hypothesized that this could lead to succinctly synthesizing the information in the images. The resulting feature vector had a length of 840.

The convolutional neural network is one of the most powerful descriptors of images as it can understand patterns at every resolution by layering many convolution and pooling layers whose weights can be modifying through gradient descent. We used the pretrained network without any additional fine-tuning and removed the last layer to get the dense vector before it gets classified. The resulting feature vector had a length of 1024.

A diagram of DenseNet121's architecture.

We ran our entire dataset through these embedders and saved the resulting vectors before attempting any classification, the times to embed all of the images are shown below.

	LBP	HOG	CNN
Training	768s	785s	430s
Validation	244s	353s	125s
Testing	123s	178s	121s
Total	**1135s**	**1316s**	**676s**

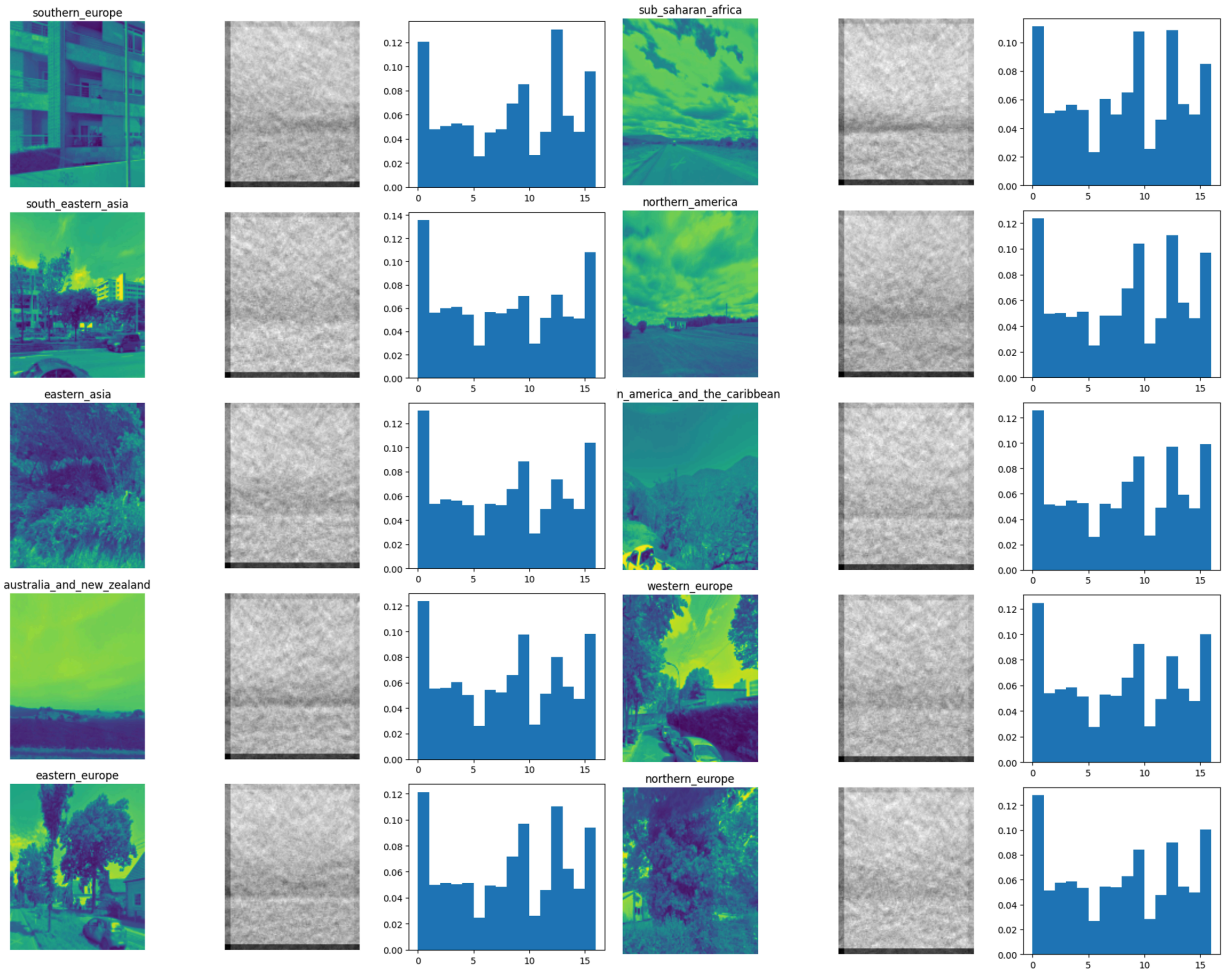
A table showing the number of seconds each dataset took to have its features extracted.

Since the dataset is too large to store all of it in memory at once, our code used data generators to dynamically read the images as needed. This means all times include the time to read from disk. The timing benchmarks were not conducted on google colab since one of our team members had adequate hardware to run the full dataset for long periods of time. The LBP and HOG only ran on the CPU while the CNN had GPU acceleration. The benchmarks were rounded to the nearest second and performed on a desktop computer with the following specifications:

- CPU - Intel i7-9700K @ 3.60GHz
- GPU - NVIDIA GeForce RTX 2080
- Disk - Samsung NVMe 980 Pro

Feature Extraction - Analysis

Looking at the histogram of the average local binary pattern for each class, they are more similar than we were hoping. There are some small differences between classes, but the high and low indices are consistent between every region.

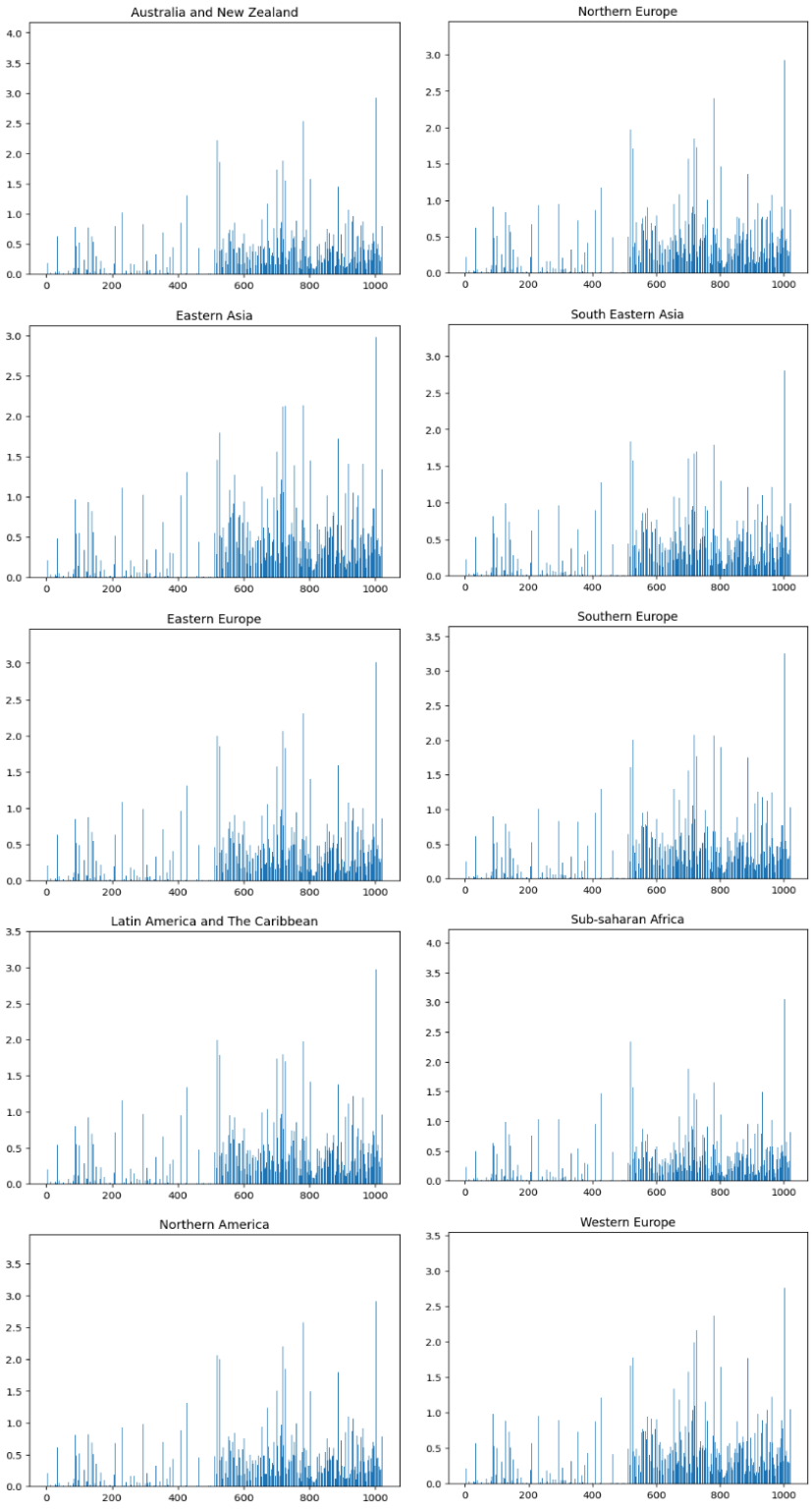


An example image, the average LBP, and the average bar chart per class.

We also looked at the visualizations of the histogram of oriented gradients for each class, although there doesn't appear to be a consistent pattern to differentiate between classes.

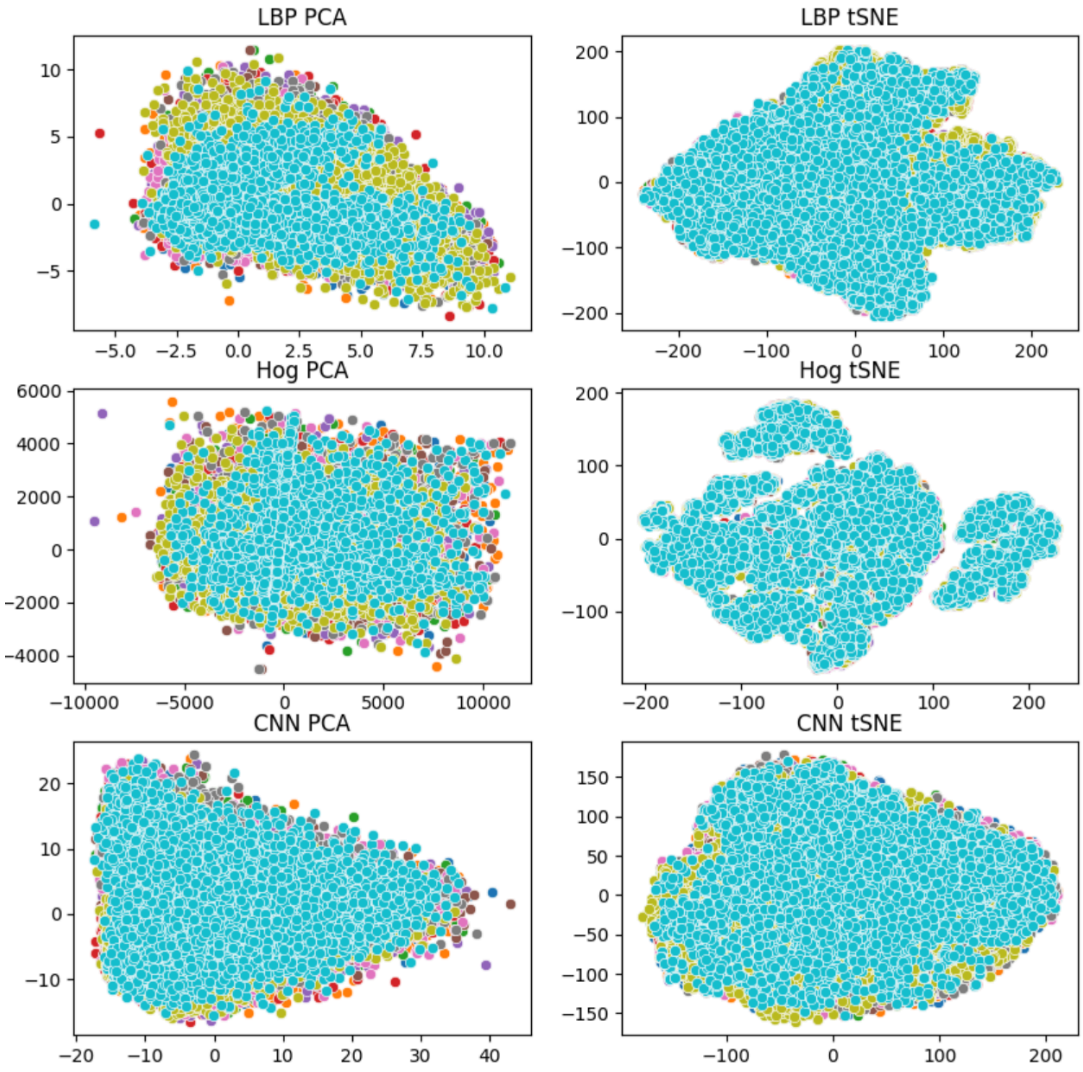
An example image and its HOG for each class.

Lastly, we also visualized the CNN features. These graphs are similar to the LBP histogram which shows the magnitude at each index of the vector. All of these appear to be rather similar, with some small differences between the average vector of each class.



The average CNN bar chart for each class.

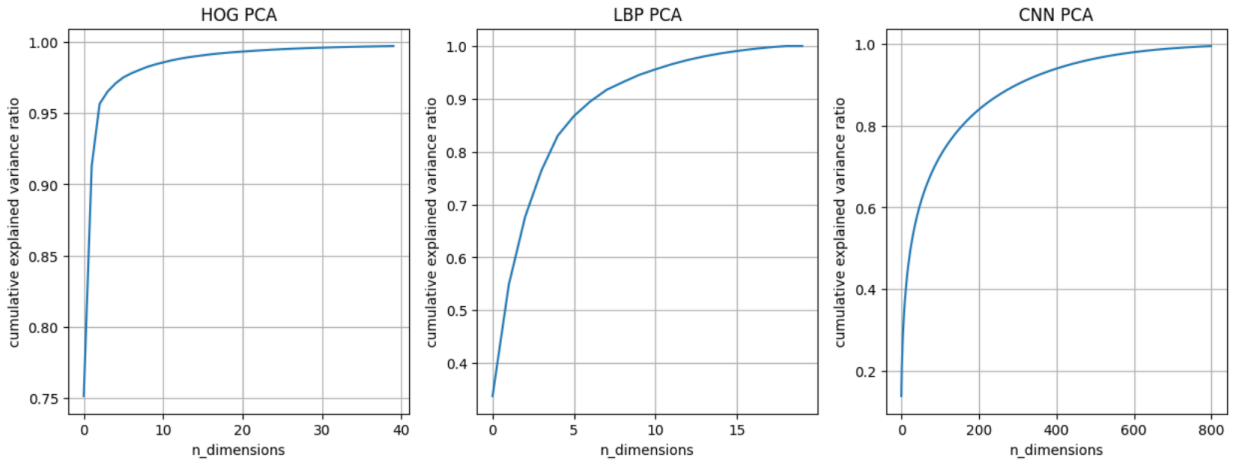
After embedding the entire dataset, we wanted to look at the data separability of the training dataset through a principal component analysis (PCA) and a t-distributed Stochastic Neighbor Embedding (t-SNE).



Dimensionality reduction graphs showing each feature's PCA & t-SNE plot.

Reviewing the PCA & t-SNE plots for each feature vector type, we were apprehensive about how well a classifier could be trained as it seems that the different classes are fairly distributed through the PCA & t-SNE 2D space. One notable plot is the HOG t-SNE as we can see clear clusters, but our classes are still fairly distributed. We hypothesize that it is finding a different pattern in the textures, perhaps the differences between nature and city scenery.

We were also curious about how quickly each feature vector type reached a satisfactory variance ratio while increasing the PCA dimension.



Graphs showing the cumulative explained variance as the number of PCA dimensions increases expressed as a percentage.

The HOG feature vectors reach 95% explained variance very quickly, with only 3 dimensions of the 840 available. The LBP grows a little slower, reaching that threshold after 10 of the 20 available dimensions. The CNN grows the slowest, hitting 95% after around 450 of the 1024 available dimensions.

Classification

We tested two different types of classifiers in this experiment, the logistic regression (LR) and histogram gradient boosted (HGB) classifiers. For logistic regression, we used the Newton-Cholesky solver with the maximum iterations hyperparameter arbitrarily set to 100. For the histogram gradient boosted classifier, we used a tolerance hyperparameter of 1e-7. Both classifiers also had a variant run on the vectors that the PCA produced with dimension 10 for HOG & LBP and 600 for the CNN. For further discussion on hyperparameters, see the section on generalizability below.

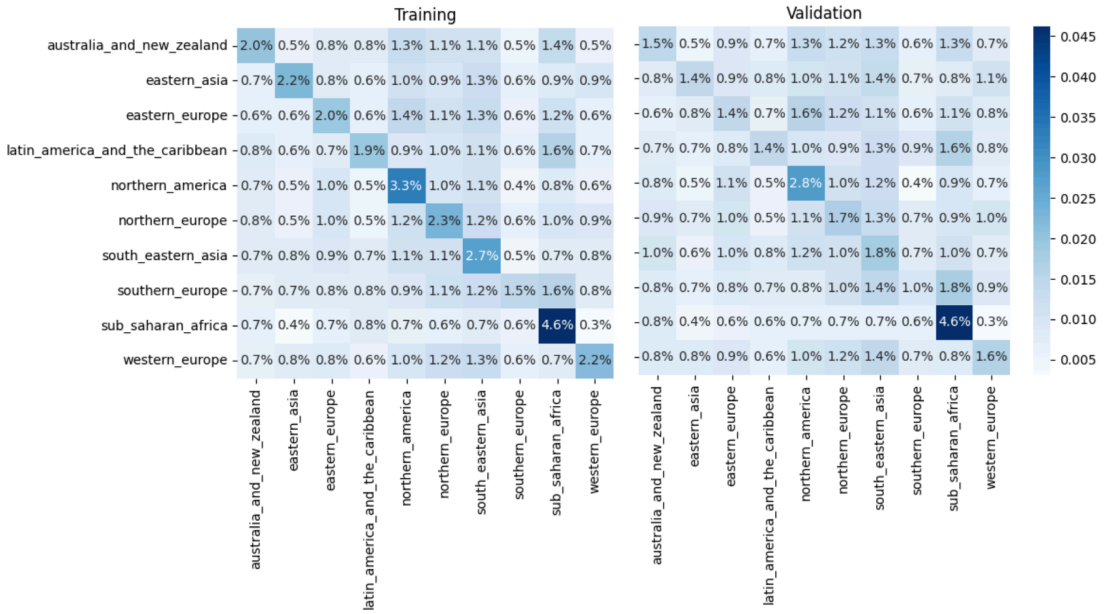
A summary of the results are provided below before diving into how specific classifiers performed. The section on this page is for all of the logistic regression and the next page has all of the histogram boosted gradient classifiers.

	Training Accuracy	Validation Accuracy	Duration
HOG x LR	24.69%	19.27%	28.76s
HOG-PCA x LR	13.91%	14.35%	4.38s
LBP x LR	19.62%	18.33%	1.02s
LBP-PCA x LR	17.96%	17.03%	1.19s
CNN x LR	52.32%	**46.91%**	22.13s
CNN-PCA x LR	47.71%	44.24%	17.69s

HOG x HGB	21.80%	15.65%	3.52s
HOG-PCA x HGB	23.82%	16.03%	3.68s
LBP x HGB	43.08%	20.87%	2.44s
LBP-PCA x HGB	39.93%	20.10%	3.11s
CNN x HGB	52.33%	**46.91%**	90.41s
CNN-PCA x HGB	82.35%	40.04%	58.33s

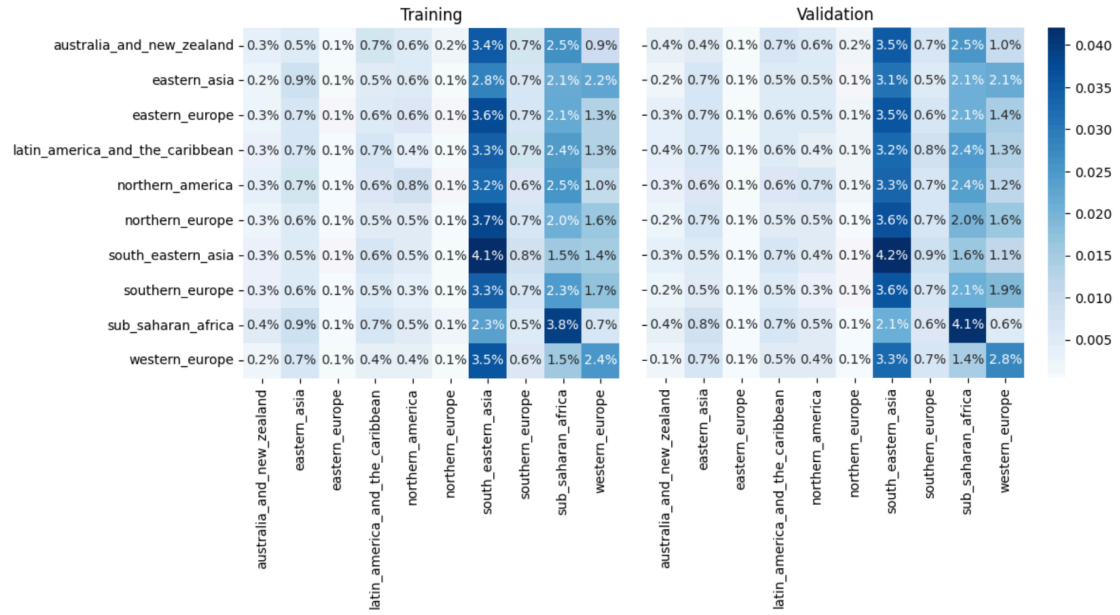
A table showing the training and validation accuracies for every classifier.

The comparison between the logistic regression and the histogram gradient boosted classifiers are different for each feature vector type. The HOG saw a drop in validation accuracy while the LBP saw a small boost and the CNN remained exactly the same. Adding the PCA step before regression on all of the pairs shows a direct trade-off losing accuracy for speed of training.



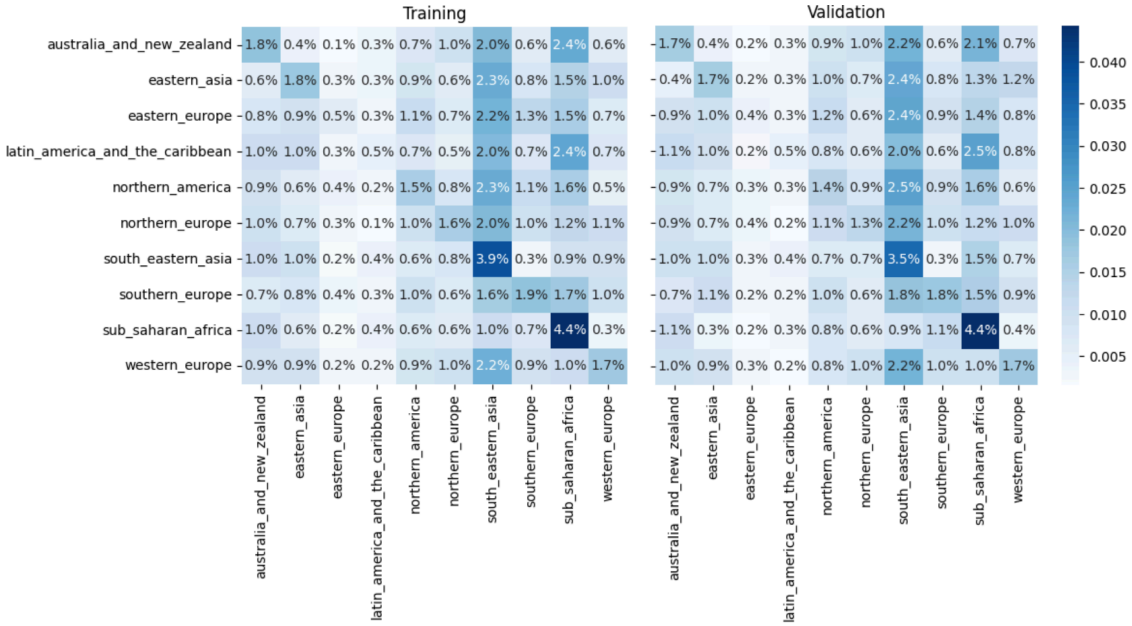
HOG Logistic Regression

It seems that the HOG was able to find a substantive pattern for Sub-Saharan Africa compared to the rest of the world, but it was not able to satisfactorily distinguish the remaining classes.



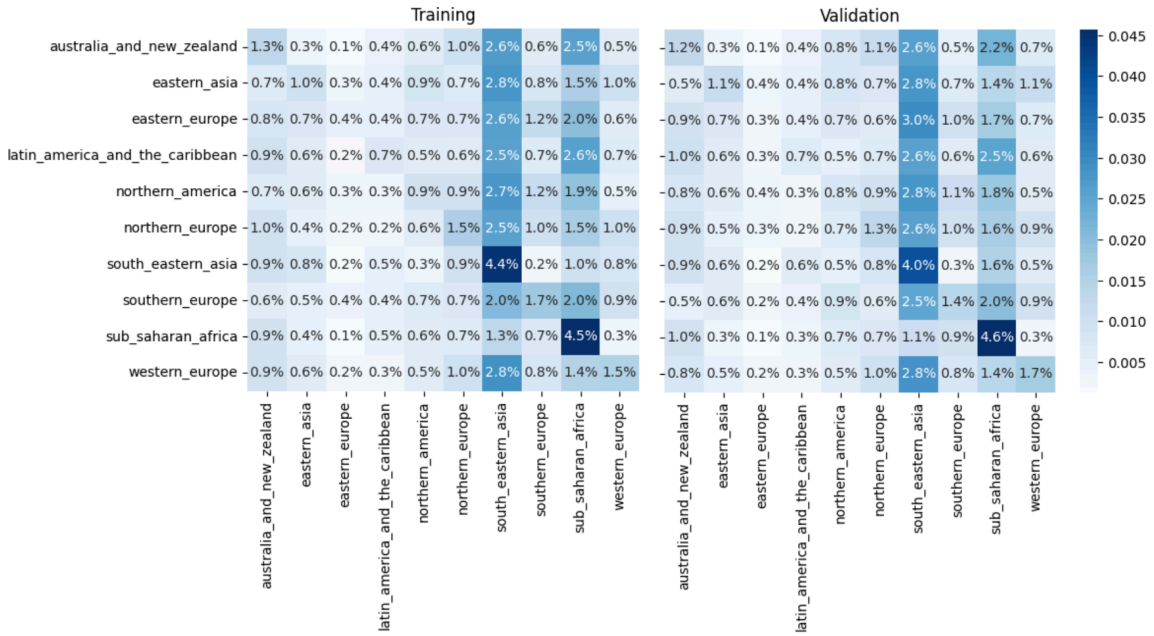
HOG-PCA Logistic Regression

The HOG features ran through PCA did not appear to reasonably converge. Even after increasing the number of maximum iterations into the tens of thousands, well beyond the starting value of 100, it did not show any improvement.



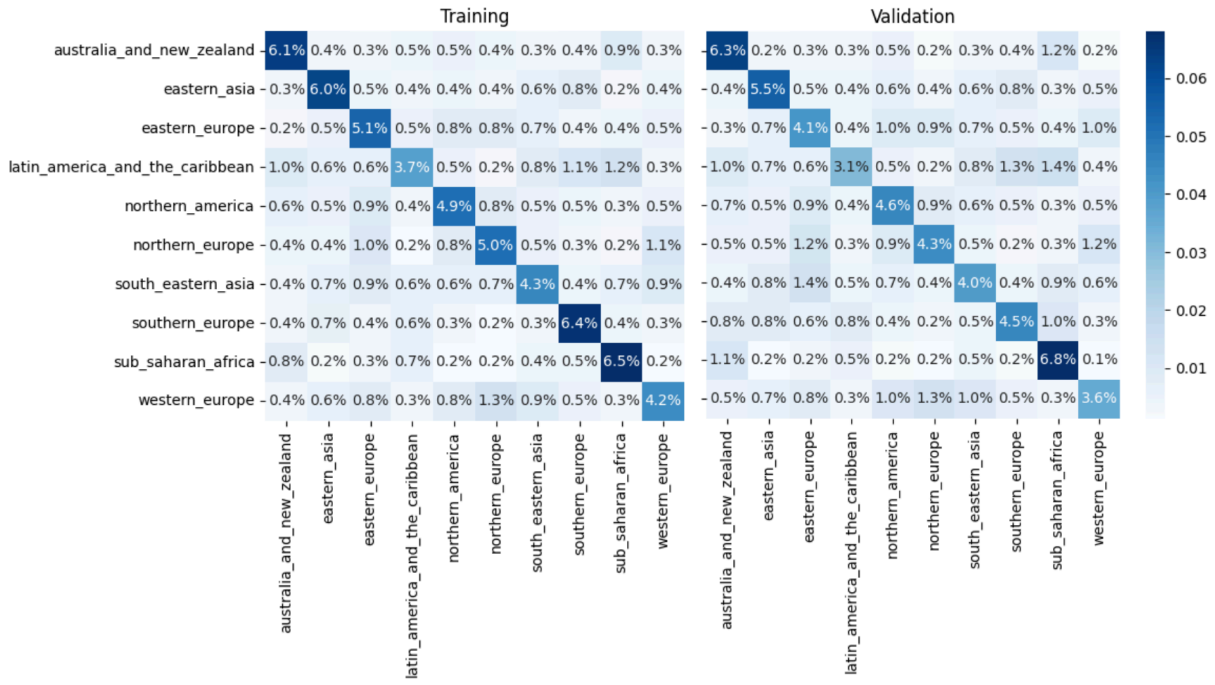
LBP Logistic Regression

The LBP was able to find the same success for Sub-Saharan Africa as the HOG did, but also appeared to struggle with over-classifying into South Eastern Asia and Sub-Saharan Africa as the HOG-PCA did, despite all of the classes being equal in the training set.



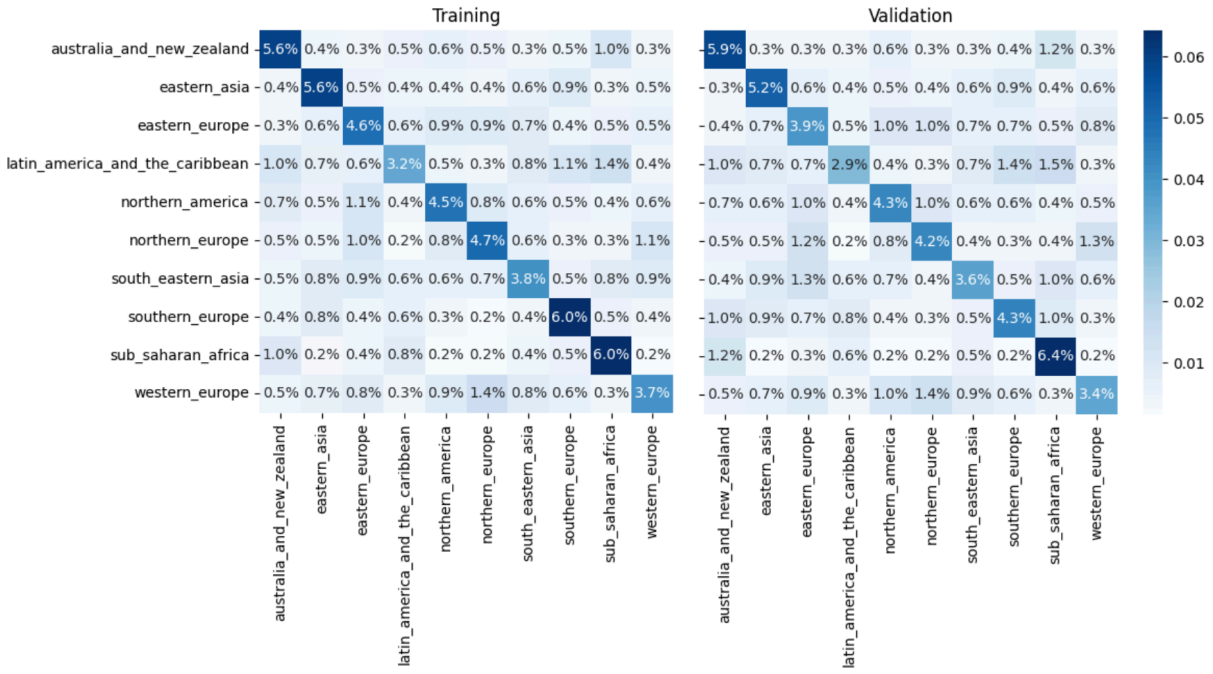
LBP-PCA Logistic Regression

Running the LBP vectors through PCA before regression did not appear to make any difference.



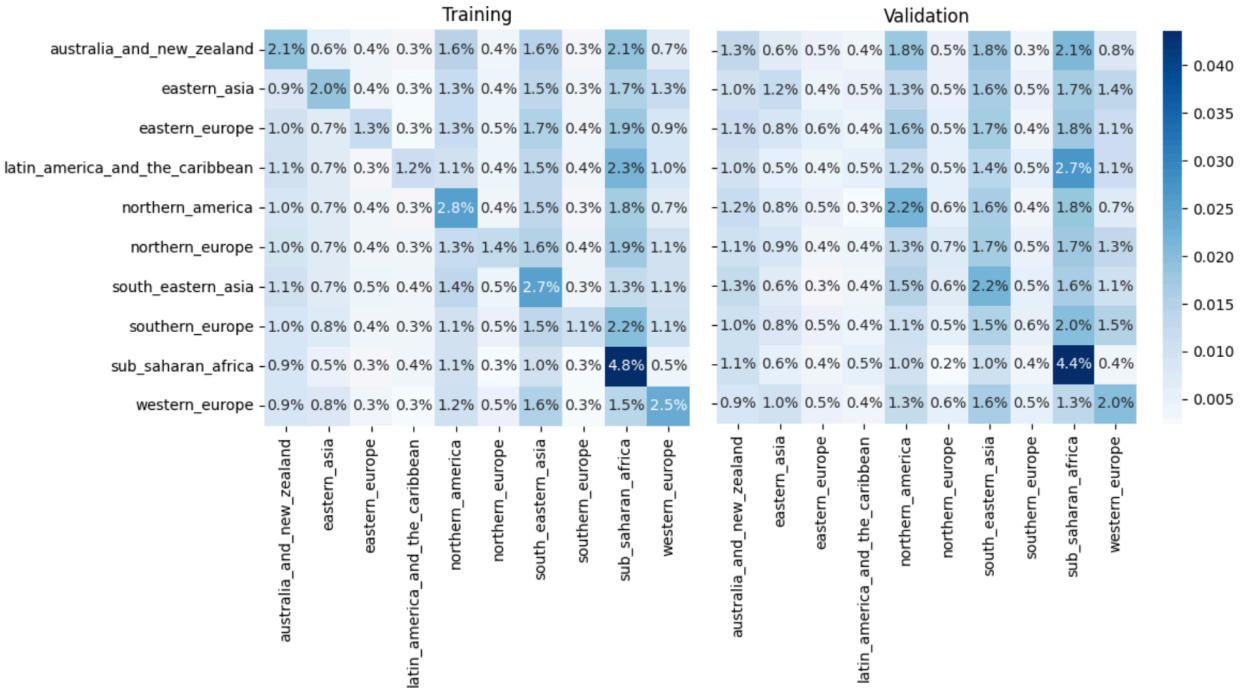
CNN Logistic Regression

The pretrained CNN had the best results of the three feature vector types by a substantial margin. There does not seem to be any bias towards one particular class.



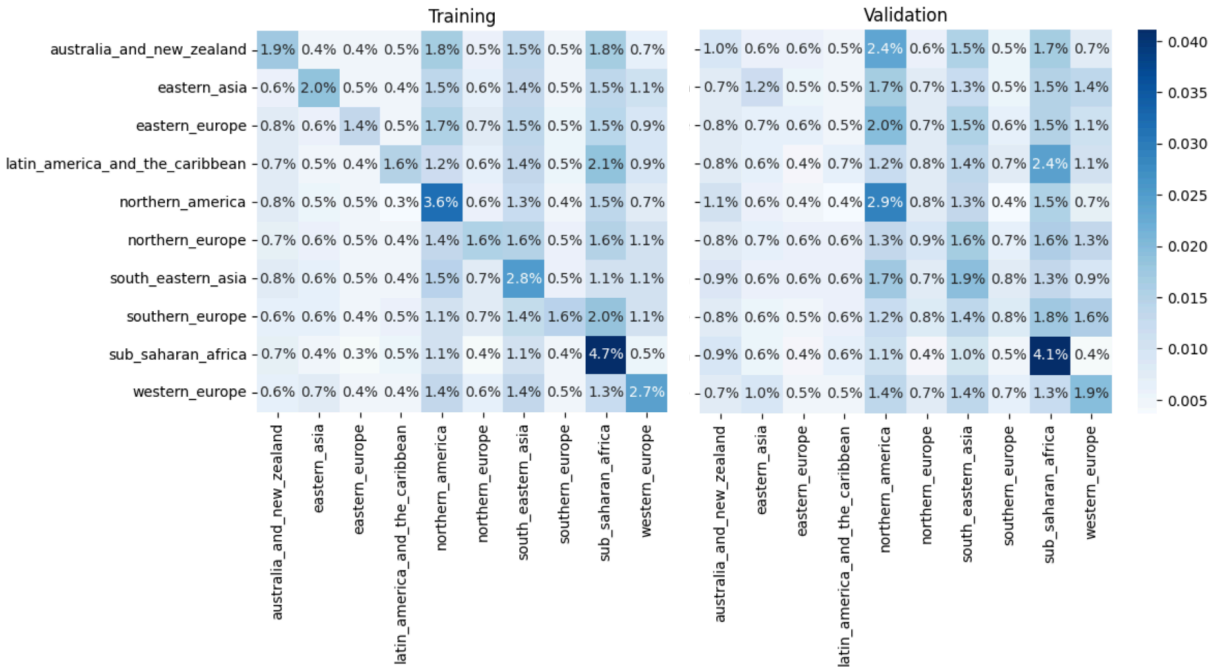
CNN-PCA Logistic Regression

Running the CNN vectors through PCA before regression did not make any visual impact.



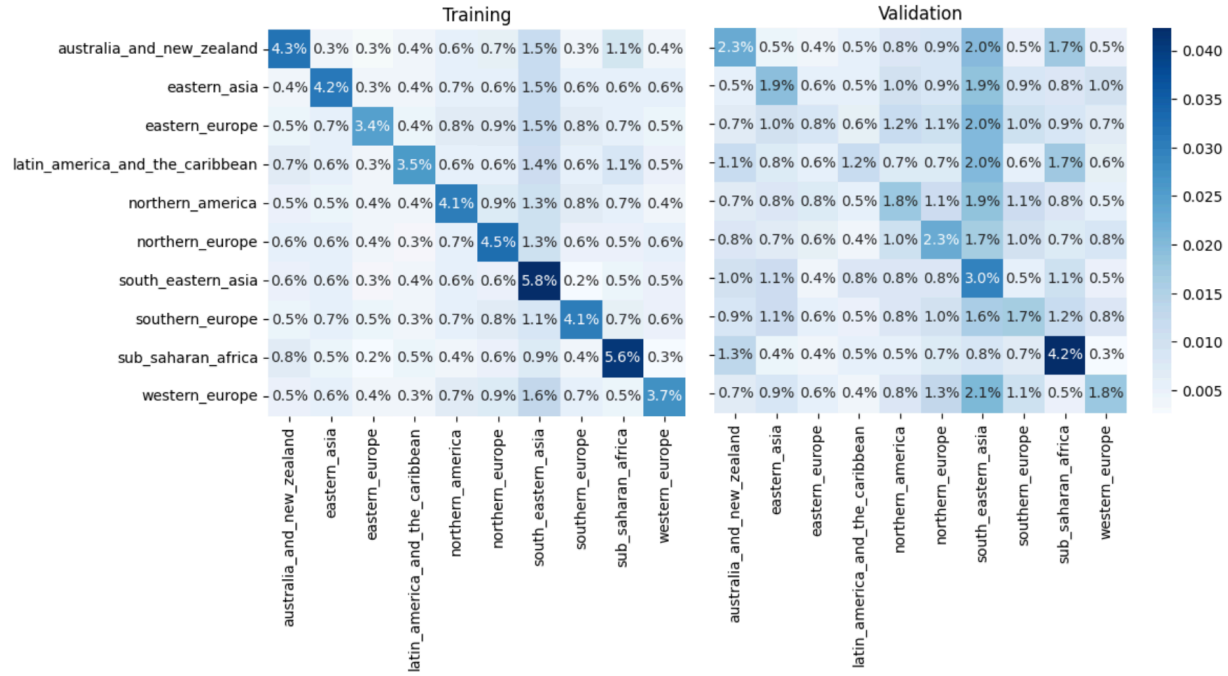
HOG Histogram Boosted Gradient

Using the histogram boosted gradient classifier instead of logistic regression seems to be a mistake for HOG, it is showing the problematic vertical stripes.



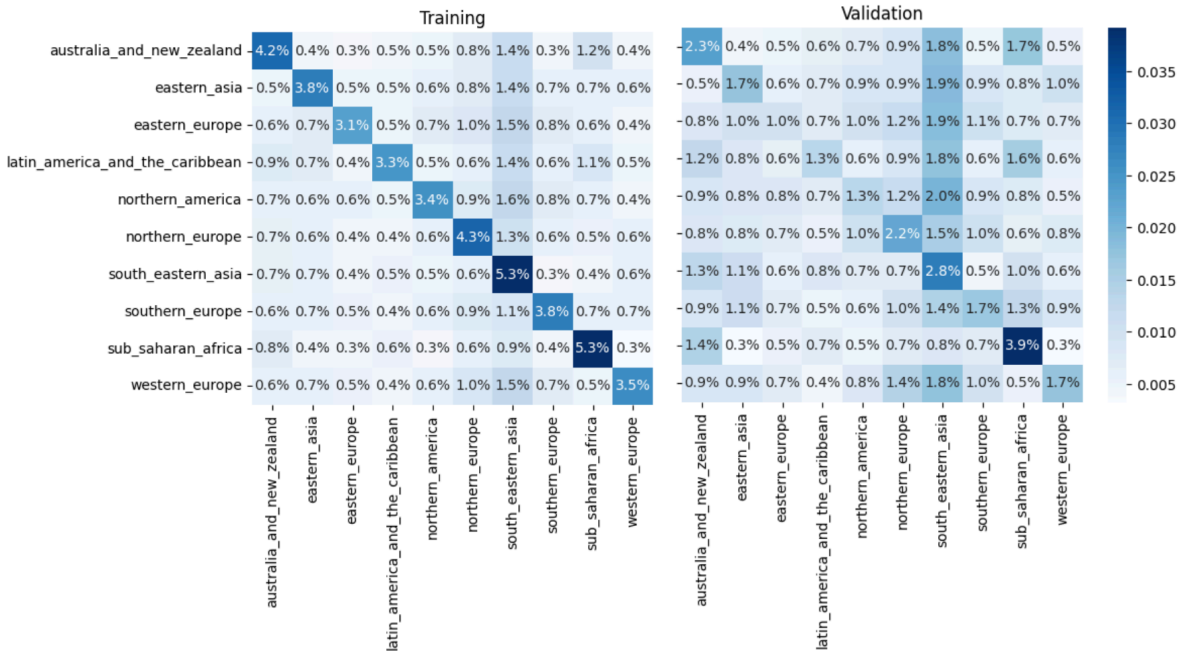
HOG-PCA Histogram Boosted Gradient

Adding PCA to the HOG doesn't offer any visual improvement.



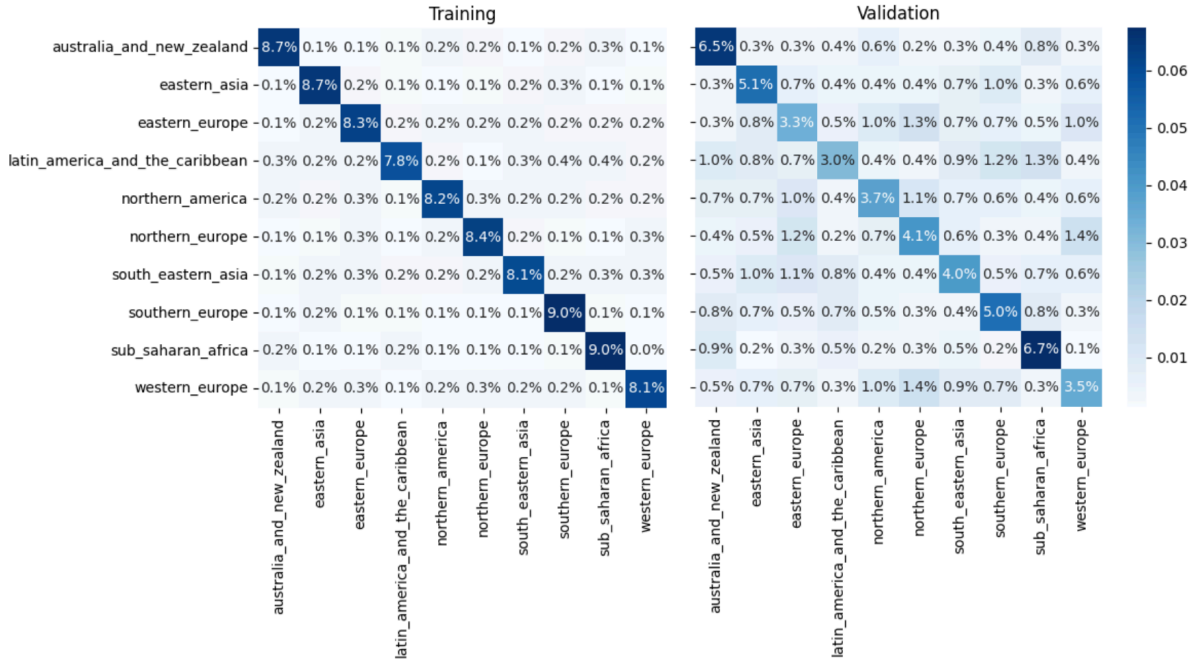
LBP Histogram Boosted Gradient

Using the histogram boosted gradient classifier for LBP has now moved the overclassification issue into solely the south eastern asia class.



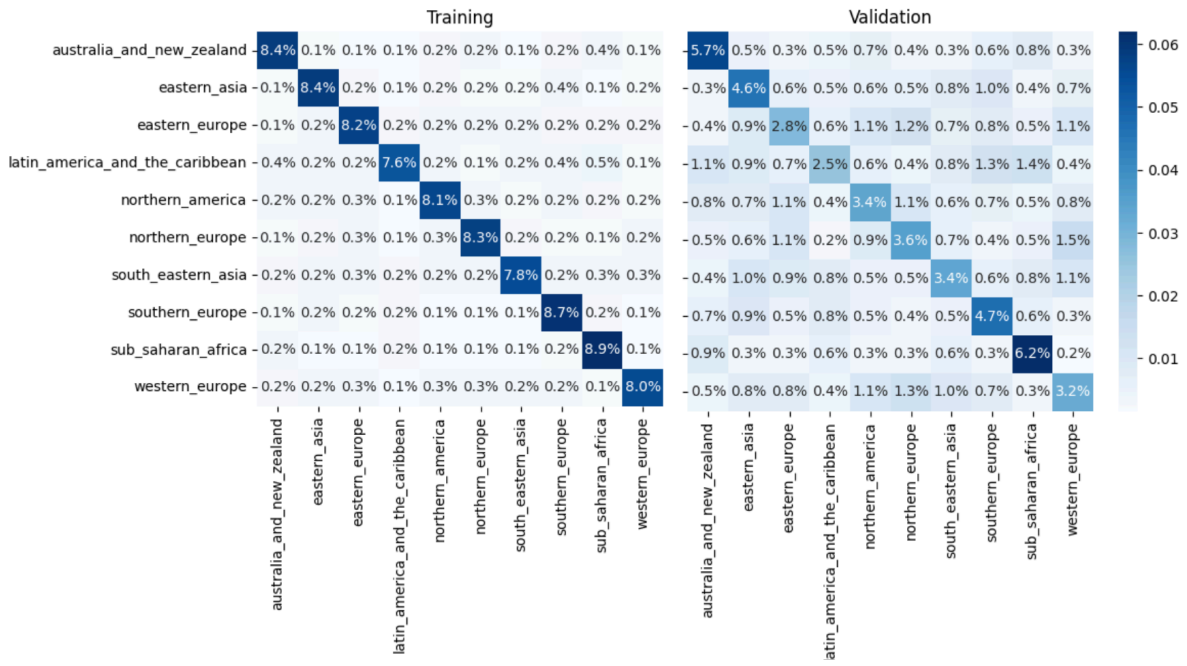
LBP-PCA Histogram Boosted Gradient

Adding PCA to the LBP doesn't offer any visual improvement.



CNN Histogram Boosted Gradient

Using the histogram boosted gradient classifier for the CNN overfits on the training set, but offers similar performance on the validation set.



CNN-PCA Histogram Boosted Gradient

Adding PCA to the CNN did not offer any visual improvement.

Generalizability

In order to find the optimal hyperparameters for the classifiers, we performed a grid search across several different possible hyperparameters. For the models that didn't rely on PCA, we tested the maximum iterations for 50, 100, 200, and 500 and the tolerance for 1e-5, 1e-4, and 1e-3. For models that do rely on PCA, we also tested the dimensions as 5, 10, 20, and 30 for HOG; 5, 10, 15, and 18 for LBP; and 100, 300, 500, and 700 for CNN.

We also decided to use a 5-fold validation strategy for performing this grid search to avoid having the optimal hyperparameters overfit to one part of the validation dataset and maximize generalizability. Below is a table of the grid search results, the logistic regression is on this page while the histogram gradient boosting classifier is on the next page.

	Combos	Time	Best Val. Acc.	Prev. Val. Acc.	Improvement
HOG x LR	60	753s	19.27%	19.27%	0.00%
HOG-PCA x LR	240	816s	15.80%	14.35%	1.45%
LBP x LR	60	10s	18.33%	18.33%	0.00%
LBP-PCA x LR	240	83s	18.19%	17.03%	1.16%
CNN x LR	60	393s	**46.91%**	46.91%	0.00%
CNN-PCA x LR	240	2208s	44.89%	44.24%	0.65%

HOG x HGB	60	249s	22.41%	15.65%	**6.76%**
HOG-PCA x HGB	240	1018s	16.08%	16.03%	0.05%
LBP x HGB	60	33s	20.87%	20.87%	0.00%
LBP-PCA x HGB	240	192s	20.17%	20.10%	0.07%
CNN x HGB	60	1104s	**46.91%**	46.91%	0.00%
CNN-PCA x HGB	240	3385s	44.63%	40.04%	4.59%

A table showing the number of fits performed, the grid search duration, the best validation accuracy recorded from all of the attempts, the previous validation accuracy chosen with arbitrary hyperparameters, and the improvement the grid search offered.

The grid search did not offer any improvement for the CNN, but it did offer small improvements for some of the other classifier types.

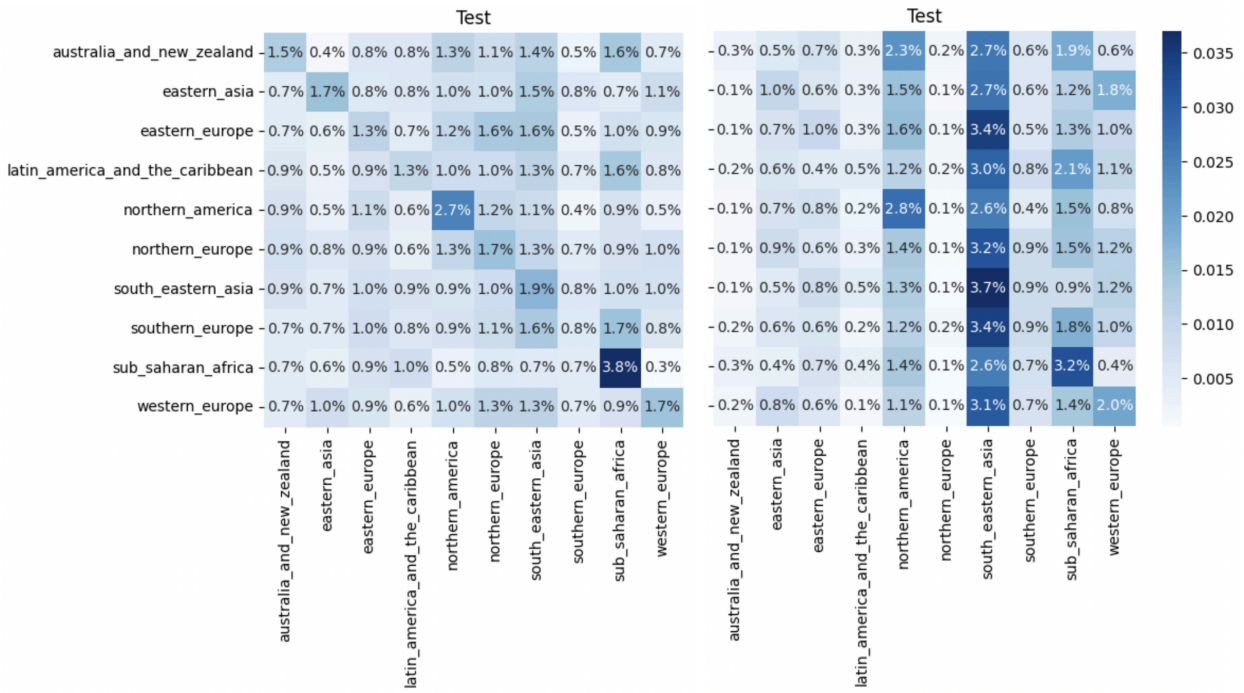
The final results ran on the test set are recorded below using the optimized hyperparameters.

	Inference Time	Test Accuracy	Validation Accuracy
HOG x LR	0.09s	18.32%	19.27%
HOG-PCA x LR	0.13s	15.38%	15.80%
LBP x LR	0.02s	18.57%	18.33%
LBP-PCA x LR	0.02s	18.50%	18.19%
CNN x LR	0.08s	**46.53%**	46.91%
CNN-PCA x LR	0.23s	44.57%	44.89%
HOG x HGB	0.91s	19.69%	22.41%
HOG-PCA x HGB	0.33s	15.73%	16.08%
LBP x HGB	0.14s	18.69%	20.87%
LBP-PCA x HGB	0.40s	18.79%	20.17%
CNN x HGB	2.46s	46.03%	46.91%
CNN-PCA x HGB	0.91s	40.92%	44.63%

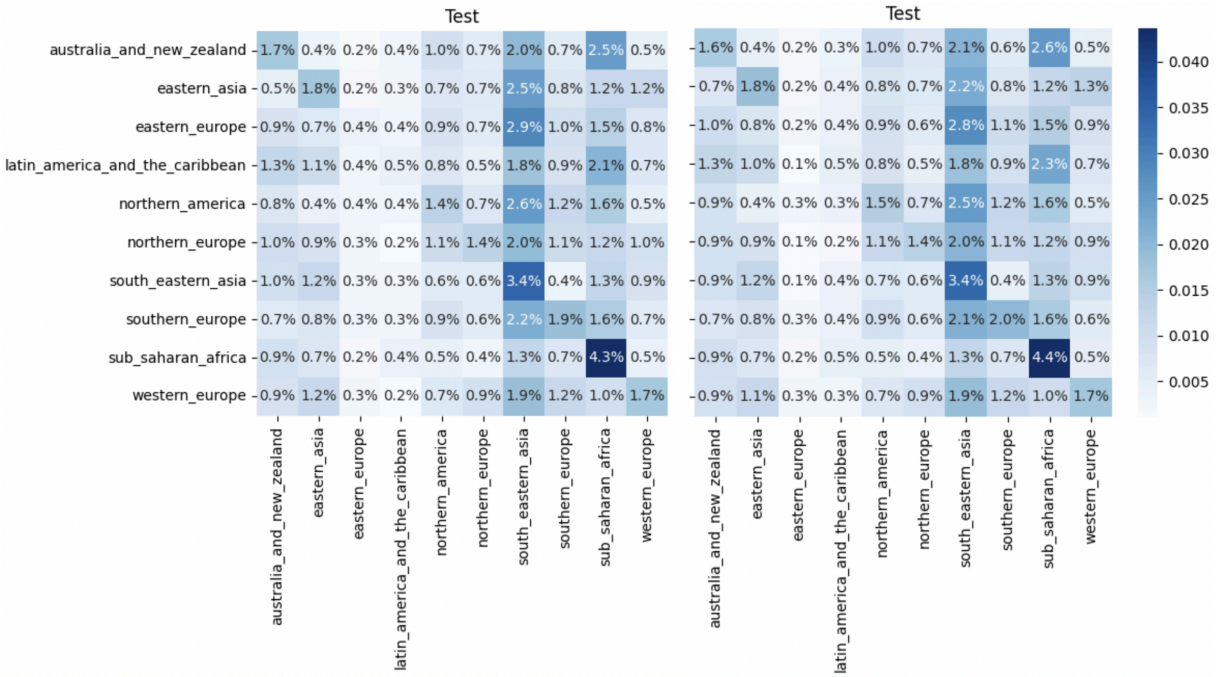
The inference time, test accuracy, and validation accuracy for every classifier.

We found that the pretrained CNN feature vector without PCA using the logistic regression classifier is the most accurate on the test set. The local binary pattern does have the best overall efficiency since it is the fastest to train, hyperparameter search, and conduct inference on, but the accuracy is far lower. The CNN is fairly close in terms of efficiency too, only barely losing in most of the categories and being the fastest to embed by far.

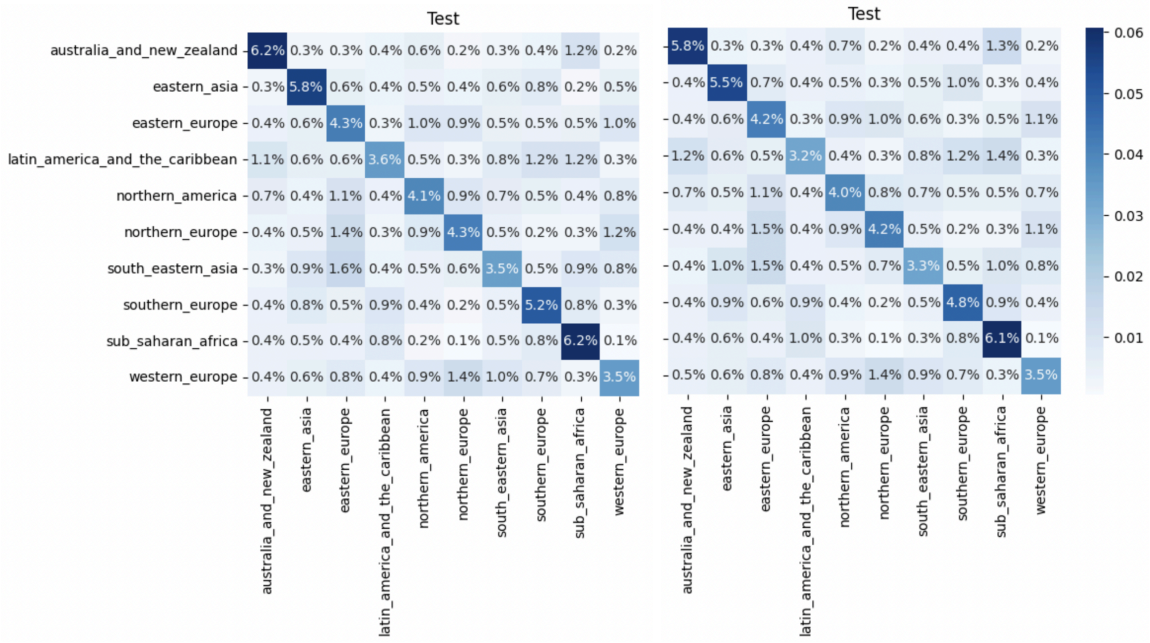
All of the confusion matrices run for the test sets display similar patterns to the validation confusion matrices, so unfortunately it appears that tuning the hyperparameters with the grid search did not eliminate these biases.



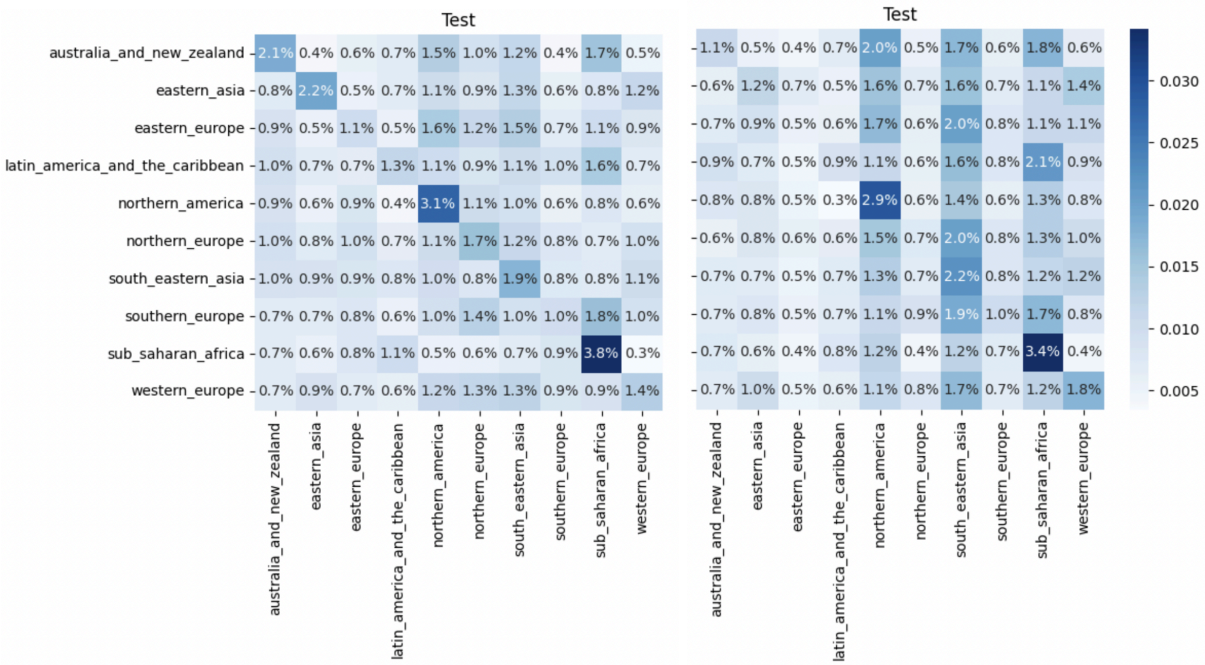
The test confusion matrices for HOG (left) and HOG-PCA (right) logistic regression classifiers.



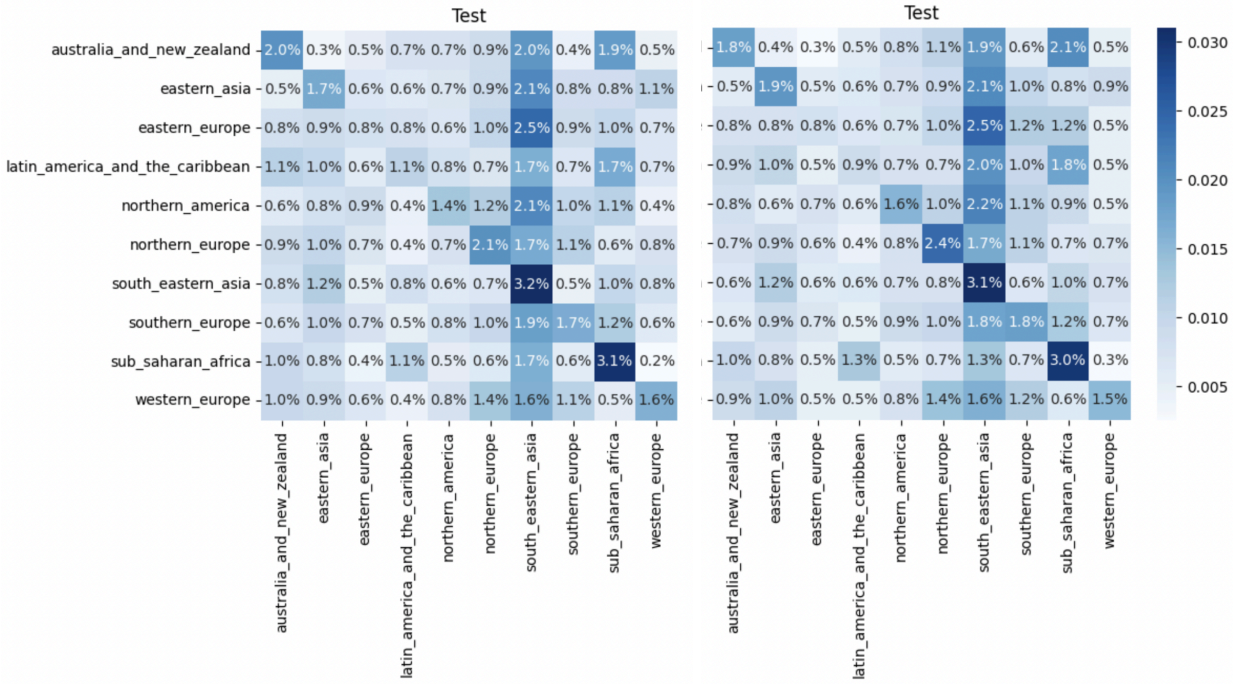
The test confusion matrices for LBP (left) and LBP-PCA (right) logistic regression classifiers.



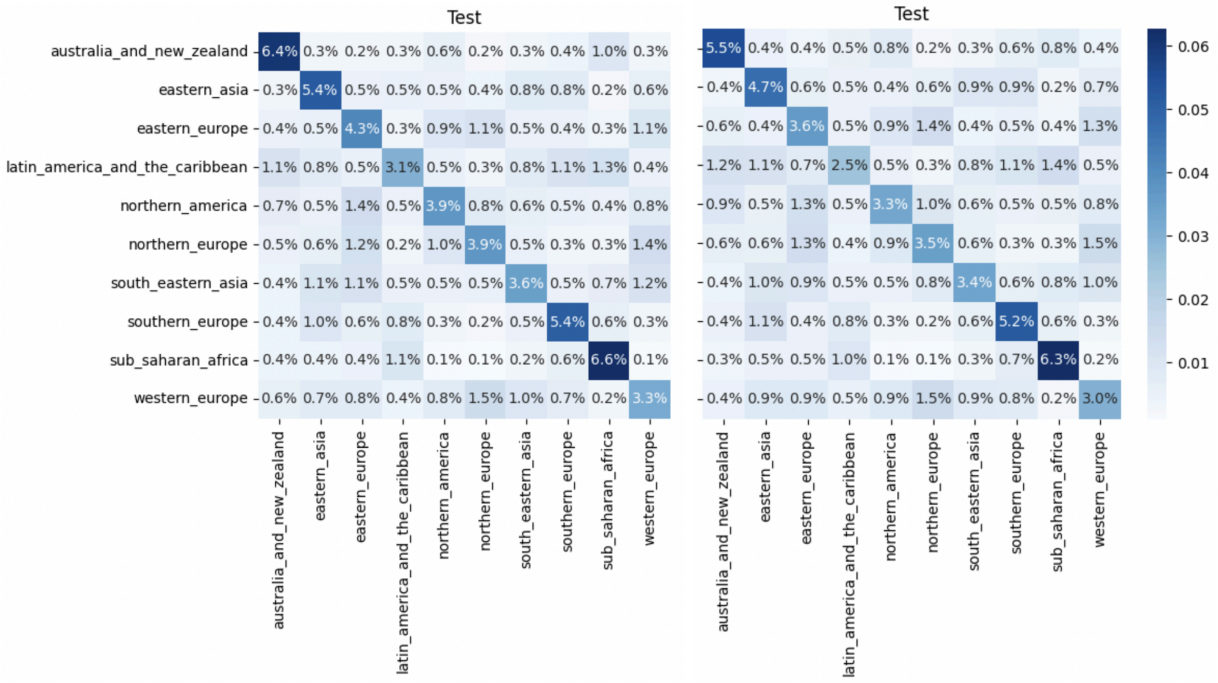
The test confusion matrices for CNN (left) and CNN-PCA (right) logistic regression classifiers.



The test confusion matrices for HOG (left) and HOG-PCA (right) histogram gradient boosted.



The test confusion matrices for LBP (left) and LBP-PCA (right) histogram gradient boosted.



The test confusion matrices for CNN (left) and CNN-PCA (right) histogram gradient boosted.

Conclusion

Overall, this data set has been quite challenging to work with since we have a lot of variations of images within each category. Several logistic regression classifiers did not converge properly because of this reason. We inferred that the South-Eastern Asia group seemed to have the most variability as we can see that the classifier had significantly more false positives for this particular category. Sub-Saharan Africa is another region that the classifiers struggled with, but the accuracy for that group was slightly better than South-Eastern Asia, and it was consistently the best out of all categories. We anticipate that to happen due the small beginning sample size of this group. This strongly indicates that the more image augmentation we had in our samples, the better the models can learn about the images. The more complex features generated from the CNN embeddings also gave better results, proving that it is able to capture the most complexity and details from the images. The Histogram Gradient Boosted Classifier, an ensemble model, proved its strong ability to learn the training set, but yet no matter how we adjusted the parameters (tolerance, max iterations, etc.), it still tends to overfit, which is what we would anticipate to see for a decision tree-based model.

After tuning the hyperparameters, we saw that the results barely improved on the logistic regression models, but had slightly better results for the histogram gradient boosted classifier with HOG and CNN features. Considering testing accuracy and efficiency, the best model we have concluded was the CNN features with logistic regression classification without PCA with a test accuracy of 46.53%.

Some potential improvements we can make is that since we have quite some variations within each category, we can try to explore those subdivisions and focus on just one subdivision to categorize at a time. For example, we put all the images with buildings together and run a classifier on the building images, versus the scenic views. Hopefully, this could address both the overfitting and the divergence in some of the models. Another improvement we did not get a chance to make was to include the same amount of data augmentation for each group and use other methods to balance the sample sizes to address the imbalance in performance among different groups. Last but not least, we can further explore more methods for feature extraction as well as other versions of classifiers.