Detect Aircraft in Satellite Imagery

Trent Dell'Oro
March 2019

GitHub.com/tadelloro/Detect_Aircraft_in_Satellite_Imagery

Abstract

The purpose of this project is to detect the location of airplanes in satellite images collected over airports and other spaces. The automation of satellite imagery analysis may provide unique insights into various markets such as agriculture, defense, intelligence, energy, and/or finance. The current project is structured as a binary classification problem aimed at labeling each satellite image as containing a plane or not containing a plane. Five Machine Learning models (i.e., Logistic Regression, K-Nearest Neighbors, Random Forest, AdaBoost, and Gradient Boosting), and two Convolutional Neural Networks are optimized, trained, and tested to detect airplanes in a dataset of 32,000 satellite image chips collected over California. Each image chip has three bands, and is 20x20 pixels in size. The project achieved a maximum precision of 98%, and recall of 97% on the target class using a 2D convolutional network.

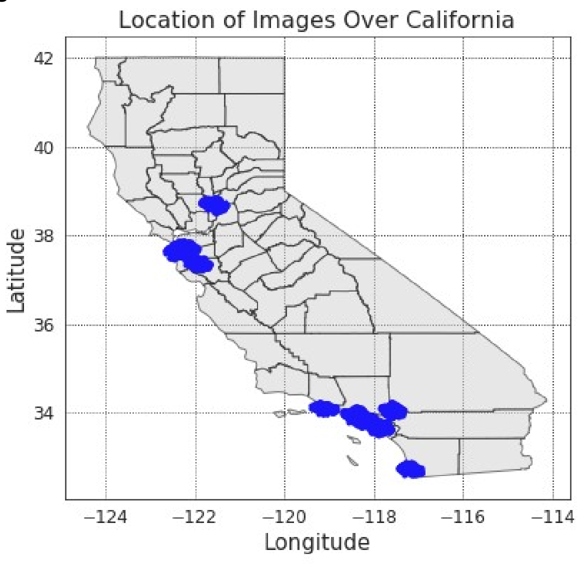
Table of Contents

1.0 Introduction

New commercial satellite imagery providers are using constellations of small satellites to exponentially increase the amount of images of the Earth captured every day. The purpose of this project is to detect the location of planes in satellite images collected over airports and other spaces. The project is structured as a binary classification problem to label each satellite image as containing a plane (i.e., 'plane') or not containing a plane (i.e., 'not-plane').

The business case for the project revolves around the idea of providing new insights into various markets, such as agriculture, defense, intelligence, energy, and finance. Motivating questions are: Can satellite imagery be used to effectively identify aircraft on the ground? How well (in terms of accuracy) can a learning model be tuned to identify planes? What learning algorithms are best suited for the problem?

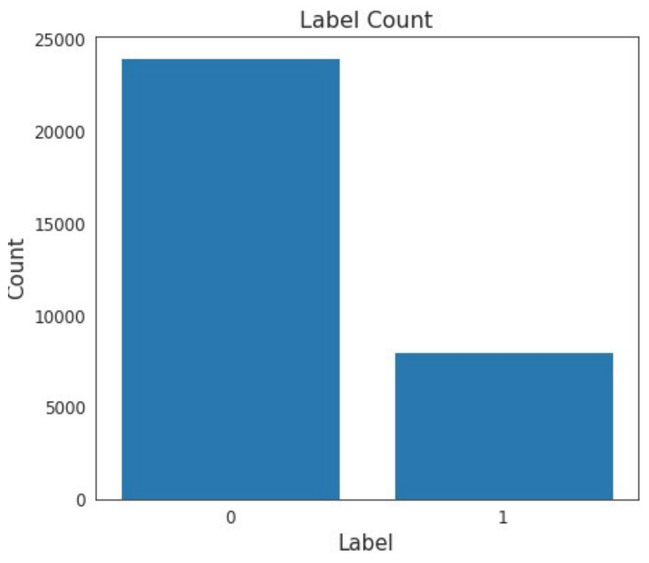
Figure 1



2.0 Data Import and Exploratory Data Analysis (EDA)

The PlanesNet satellite image chips used in this project are included in the Kaggle dataset "Planes in Satellite Imagery", provided by Planet Labs Inc. The satellite imagery used to build the PlanesNet image chip dataset is made available through Planet Labs' Open California dataset, which is openly licensed. A total of 32,000 images collected over California are provided, the images are 20x20 pixels in size, each with three bands, delivered in a JSON formatted file (190.6 Mb in size) containing image data, labels, scene id's, latitudes, and longitudes.

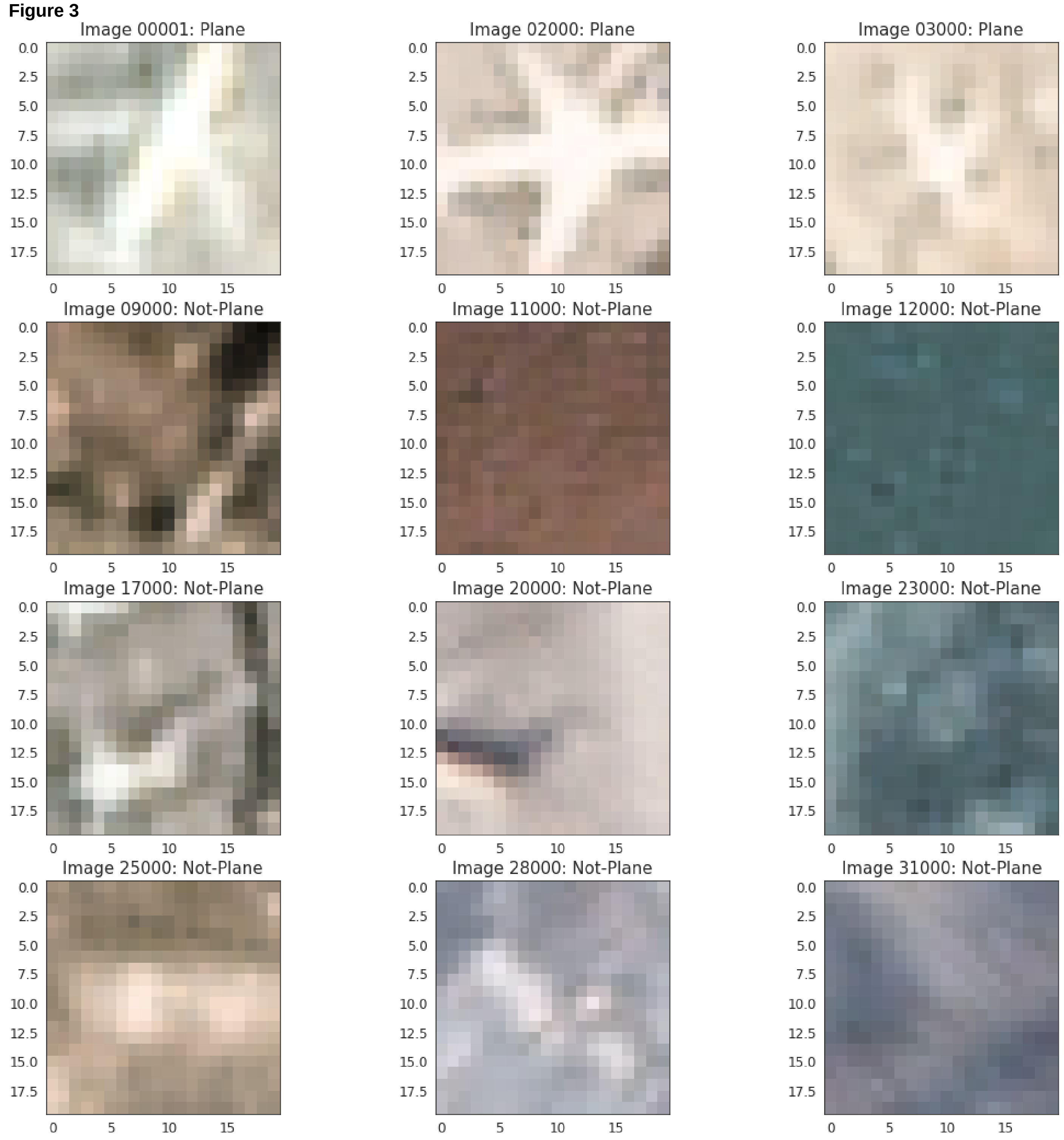
Figure 2



A map is displayed to show the locations of the satellite images taken over California, the map includes all 'plane' and all 'not-plane' image locations (Figure 1). The majority of images come from the Los Angeles area, the San Francisco area, and the San Diego area.

The 32,000 images are split into the 'plane' class (i.e., 8,000 images) and 'not-plane' class (i.e., 24,000 images) (Figure 2). The first set of 8,000 in the 32,000 images are of the 'plane' class, these are near-centered on the body of a single plane, with the majority of the plane's wings, tail, and nose also visible (Figure 3). The second set of 8,000 images are from the ‘not-plane’ class, and are a random sampling of different land-cover features — water, vegetation, bare earth, buildings, etc. — that do not include any portion of a plane. The next set of 8,000 images are partial planes that contain only a portion of a plane. The final set of 8,000 images are confusers – image chips with bright objects or strong linear features that resemble a plane.

Figure 3



RMS pixel intensity is calculated for each satellite image and displayed for visual inspection — the calculation includes all three bands (Figure 4). RMS displays can be used to visually identify pixel intensity spikes and outliers within the dataset, images that contain relatively high or low pixel intensity can be spotted easily. From the RMS display, it appears that the land-cover images in general, contain both the highest, and lowest RMS values within the dataset. This observation could be interpreted as land-cover areas tending to have higher absolute reflectivity and absorption properties (e.g., a body of water absorbing light). This interpretation can be visually cross-checked by inspecting Figure 3 again.

Red, green, and blue (RGB) bands or channels represent the primary colors of an image. These three bands are separated in order to visually inspect the pixel intensities of a few select satellite images (Figure 5). Based on visual inspection of RGB intensities from individual images, it is difficult to determine whether the bands are well balanced, or if there are an excessively dominant or weak bands per image. However, it appears that there may be some significant variation in band intensity between images from the 'plane' and 'not-plane' classes.

Gray-scale is a useful color map to use when inspecting images. When displayed in gray-scale, the value of each pixel is a single sample representing only the amount of light, that is, it carries only intensity information. Select images are re-displayed in gray-scale, along with pixel intensity distributions of the RGB bands (Figure 6 and 7). There appears to be significant variation between the

Figure 4

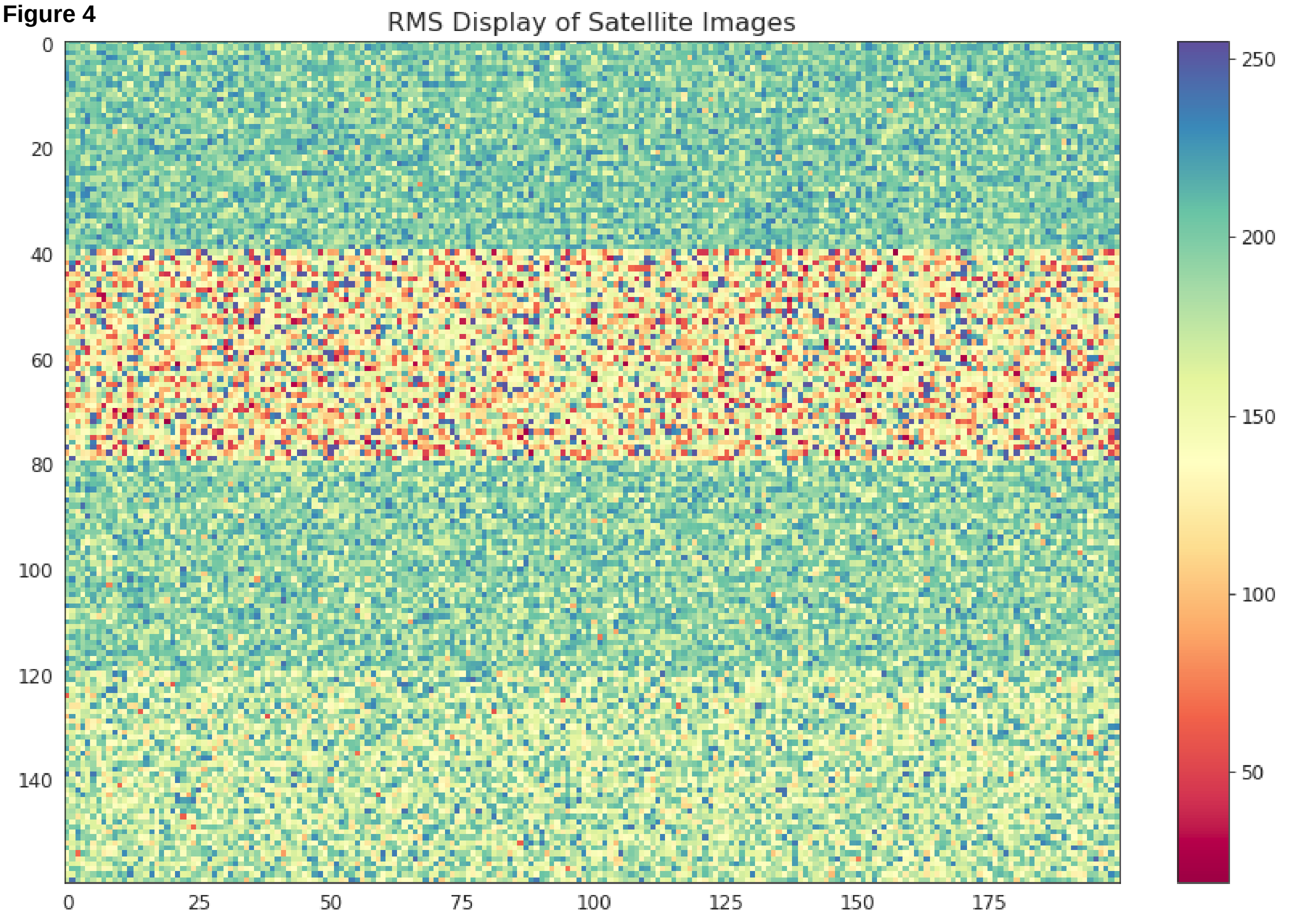


Figure 5

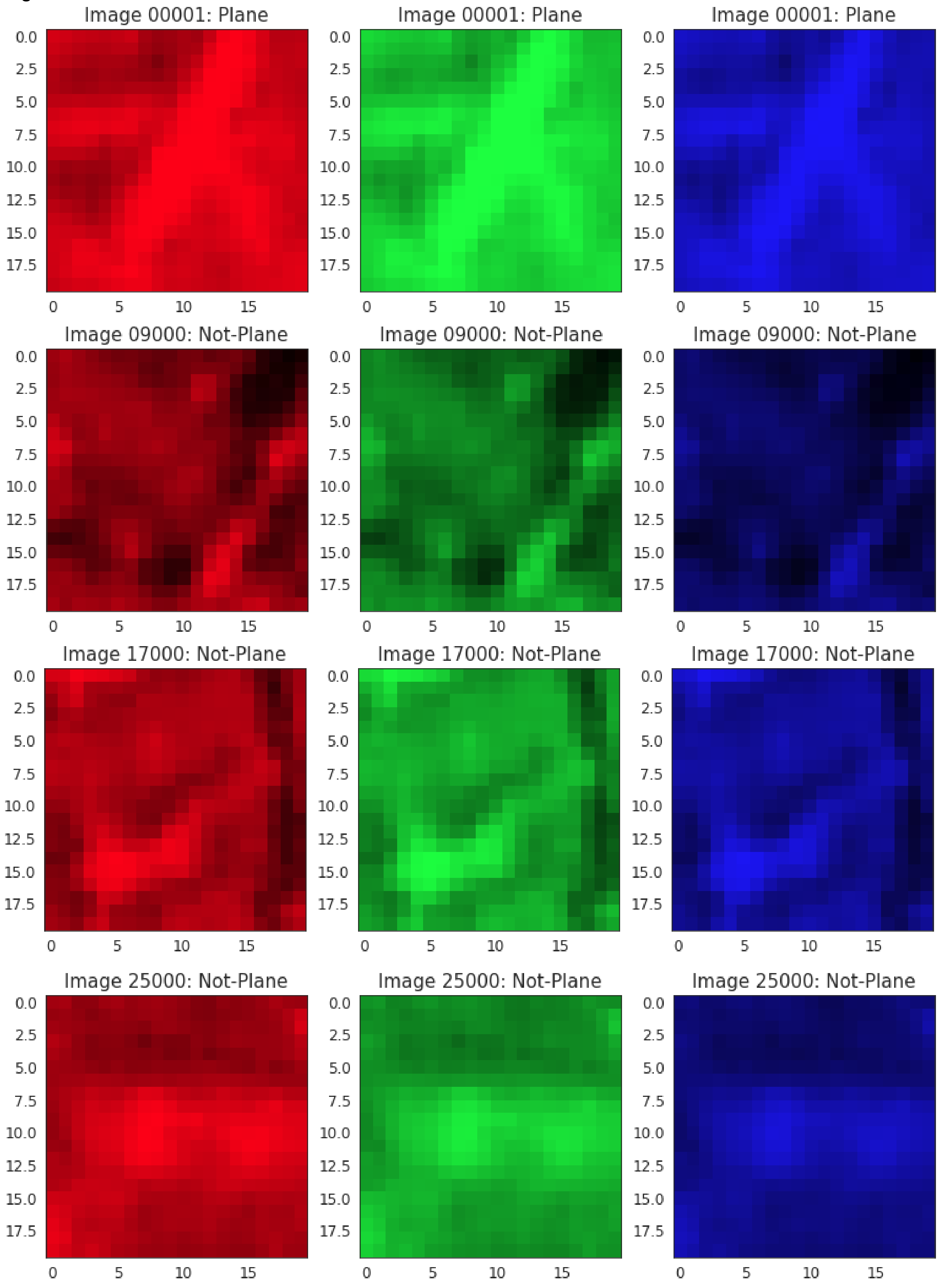


Image 00001: Plane
Image 00001: Plane
Image 00001: Plane
Image 09000: Not-Plane
Image 09000: Not-Plane
Image 09000: Not-Plane
Image 17000: Not-Plane
Image 17000: Not-Plane
Image 17000: Not-Plane
Image 25000: Not-Plane
Image 25000: Not-Plane
Image 25000: Not-Plane
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
5
10
15

Figure 6

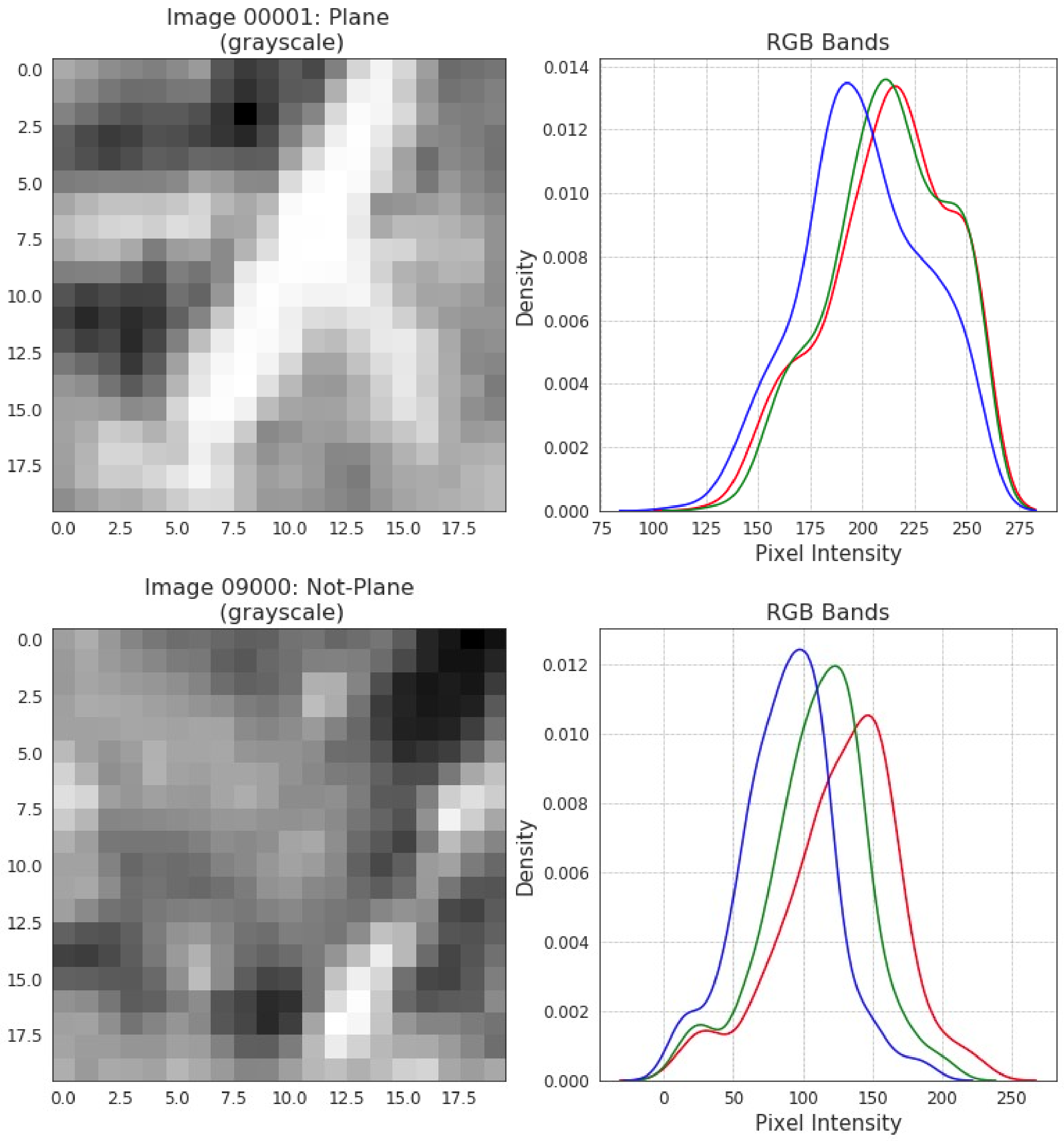


Image 00001: Plane (grayscale)
RGB Bands
Density
Pixel Intensity
Image 09000: Not-Plane (grayscale)
RGB Bands
Density
Pixel Intensity

Figure 7

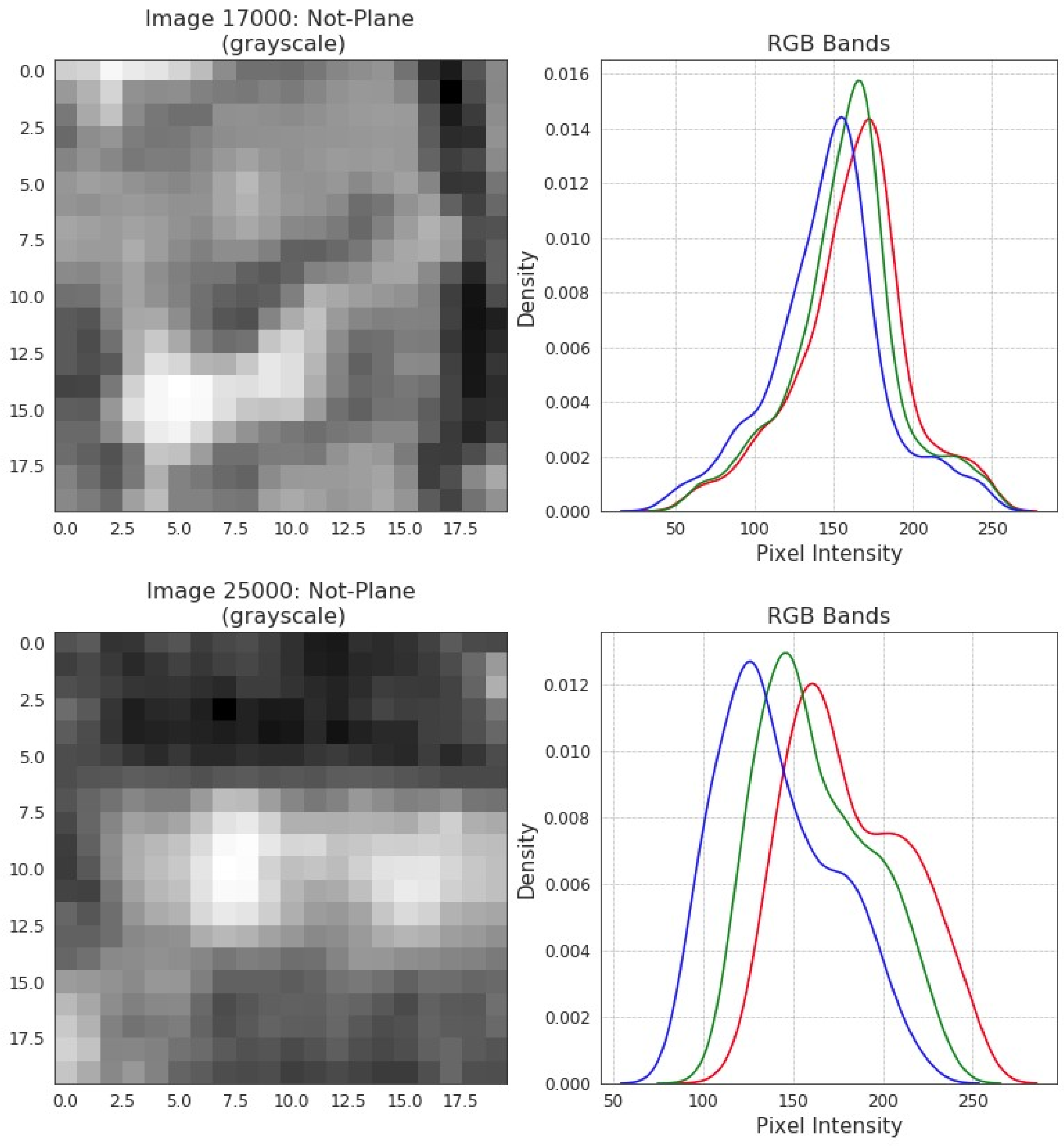


Image 17000: Not-Plane
(grayscale)
RGB Bands
Density
Pixel Intensity
Image 25000: Not-Plane
(grayscale)
RGB Bands
Density
Pixel Intensity

pixel intensity distributions of the RGB bands from 'plane' and 'not-plane' classes. From visual inspection, it appears the 'plane' distributions tend to show flatter peaks and lower mean intensities, while the 'not-plane' distributions have more defined peaks and higher mean intensities overall.

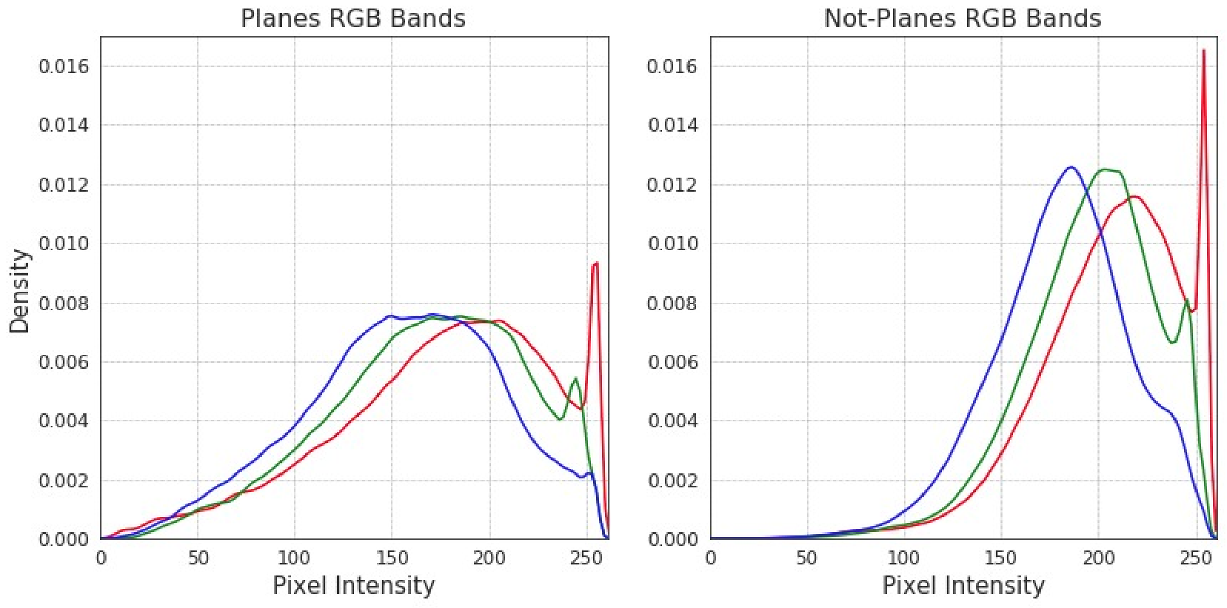
Pixel intensities from all the images are aggregated into ‘plane’ and ‘not-plane’ classes, distributions are calculated and displayed (Figure 8). It is even more apparent that the pixel intensity distributions of the RGB bands from 'plane' and 'not-plane' classes exhibit significantly different patterns.

A hypothesis test is performed to determine whether there is a statistically significant difference in the mean pixel intensity between ‘plane’ and ‘not-plane’ classes (Figure 9). The null hypothesis is that there is no significant difference in the mean pixel intensity between ‘plane’ and ‘not-plane’ classes. The alternative hypothesis is that there is a significant difference between the two groups. The hypothesis test results (i.e., p-value of 0.0) suggest that the null hypothesis be rejected, and that there is a statistically significant difference between ‘plane’ and ‘not-plane’ classes. The pixel intensities will be used as input features for classification and prediction.

3.0 Model Optimization and Testing

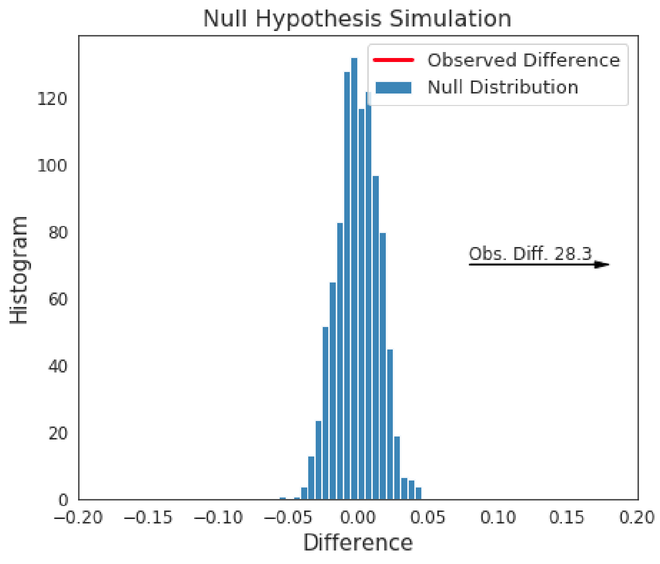
Five machine learning algorithms are optimized using GridSearchCV across a variety of parameters. The five algorithms include Logistic Regression (LRC), K-Nearest Neighbors (KNN), Random Forest (RFC), AdaBoost (ADA), and Gradient Boosting (GBC). In addition to the five machine learning algorithms, two Convolutional Neural Networks (CNNs) are designed and trained using 1D convolution layers (1D CNN), and 2D convolution layers (2D CNN). The first six models use 120 principal components (PCs) derived from the 1200 pixels as input features, while the seventh model uses the full set 1200 pixels as input features.

Figure 8



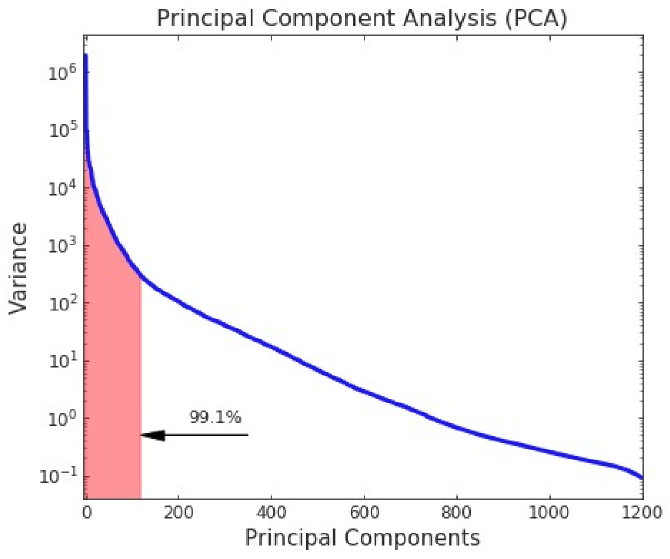
In order to optimize, fit, and test the machine learning algorithms — within the allotted time and compute resource limits — the dimensionality of the feature space (i.e., 1200 pixel intensity values per image) is reduced before processing. Principal component analysis (PCA) is performed to assess the explained variance of 1200 PCs. Prior to PCA, the data is scaled — mean centering and unit variance.

Figure 9



The explained variance of each PC is displayed, a logarithmic scale is applied to the y-axis for better visualization of the features with lower explained variance (Figure 10). We can see that the first PC accounts for the most variance in the set of 1200 PCs. From the display, it appears that the slope of the explained variance stabilizes around 120 PCs. In addition, 99.1% of the total explained variance is under the first 120 PCs. Therefore, 120 is the number of PCs chosen as input into the first five machine learning algorithms and the 1D CNN.

Figure 10



Before optimization and fitting the five machine learning algorithms, the data are split into train and test datasets — 80% for training and 20% for testing. The test dataset is set aside for model evaluation. The data are scaled and PCA is performed to reduce the number of features to 120. Algorithm parameters are optimized on the training dataset, the best training accuracy scores are printed out after each model is optimized over the chosen parameter set.

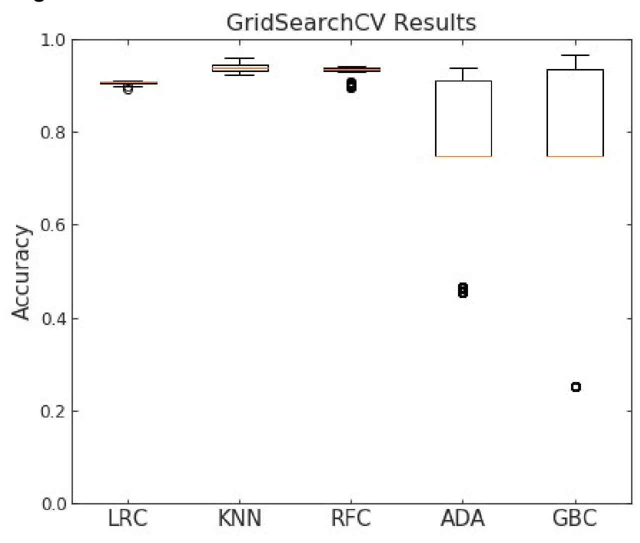
3.1 Logistic Regression (LRC)

The following parameters are tested and/or set in GridSearhCV:

- The inverse of regularization strength (i.e., C) is tested over a range of values from 0.001 to 100.
- Various solvers including 'newton-cg', 'liblinear', and 'sag' are tested.
- The penalty is set to the default of L2. The three solvers tested support L2 penalty with multi_class set to 'ovr'.

3.2 K-Nearest Neighbors (KNN)
The following parameters are tested and/or set in GridSearhCV:
• The number of neighbors (i.e., n_neighbors) is tested over a range of values from 1 to 34.
• The significance of each neighbor's vote (i.e., weights) is tested on 'uniform' and 'distance'. The former treats every neighbor's vote as equal in the prediction process, while the latter assigns a weight to each neighbor's vote based on the neighbor's distance from the query point.

Figure 11



3.3 Random Forest (RFC)
The following parameters are tested and/or set in GridSearhCV:
• The number of trees in the forest (i.e., n_estimators) is tested over a range of values from 10 to 1000.
• The minimum number of samples required to split an internal node (i.e., min_samples_split) is tested on values from 2 to 4.
• The maximum number of features to consider when looking for the best split (i.e., max_features) is set to the square root of the total number of features.

3.4 AdaBoost (ADA)
The following parameters are tested and/or set in GridSearhCV:
• The number of trees in the forest (i.e., n_estimators) is tested over a range of values from 10 to 1000.
• The learning rate or shrinkage (i.e., learning_rate) is tested over a range of values from 0.001 to 10.

3.5 Gradient Boosting (GBC)
The following parameters are tested and/or set in GridSearhCV:
• The number of trees in the forest (i.e., n_estimators) is tested over a range of values from 10 to 1000.
• The learning rate or shrinkage (i.e., learning_rate) is tested over a range of values from 0.001 to 10.
• The minimum number of samples required to split an internal node (i.e., min_samples_split) is tested on values from 2 to 4.
• The maximum number of features to consider when looking for the best split (i.e., max_features) is set to the square root of the total number of features.
• The loss function (i.e., loss) is set to 'deviance', as recommended in the sklearn documentation for classification problems.

3.6 1D Convolutional Neural Network (1D CNN)
A CNN is trained using a feature set with reduced dimensionality (i.e., 120 PCs). The 120 PCs are shaped into 120x1 arrays, and split into train, test, and validation datasets.

First, the full dataset is split into train and test datasets — 80% for training and 20% for testing. The test dataset is set aside for model evaluation. Then the train dataset is split to separate a validation dataset for training, 10% of the train dataset is used for validation.

Four 1D convolutional layers are used to build a sequential model, each with batch normalization and a relu activation function. Padding is used in each convolutional layer to maintain spatial size and preserve information as the data is processed through the network. The adadelta optimizer is used with an initial learning rate (i.e., lr) of 1.0, an exponentially weighted average decay factor (i.e., rho) set to 0.95, and a learning rate decay (i.e., decay) of 0.0. The output layer is a single node with a sigmoid activation function for binary classification.

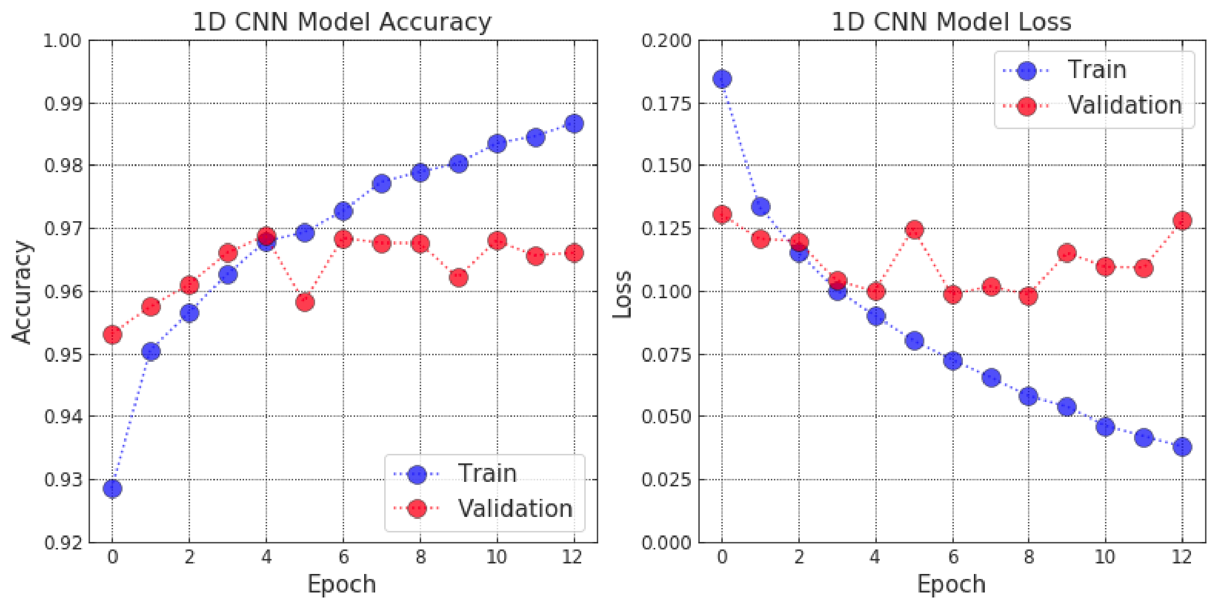
3.7 2D Convolutional Neural Network (2D CNN)

A CNN is trained using the full feature set available (i.e., 1200 pixel intensities), as opposed to using a feature set with reduced dimensionality (i.e., 120 PCs). The 1200 features are shaped into 20x20x3 arrays, and split into train, test, and validation datasets.

First, the full dataset is split into train and test datasets — 80% for training and 20% for testing. The test dataset is set aside for model evaluation. Then the train dataset is split to separate a validation dataset for training, 10% of the train dataset is used for validation.

Five 2D convolutional layers are used to build a sequential model, each with batch normalization and a relu activation function. Max pooling is added after the fourth 2D convolutional layer. Padding is used in each convolutional layer to maintain spatial size and preserve information as the data is processed through the network.

Figure 12



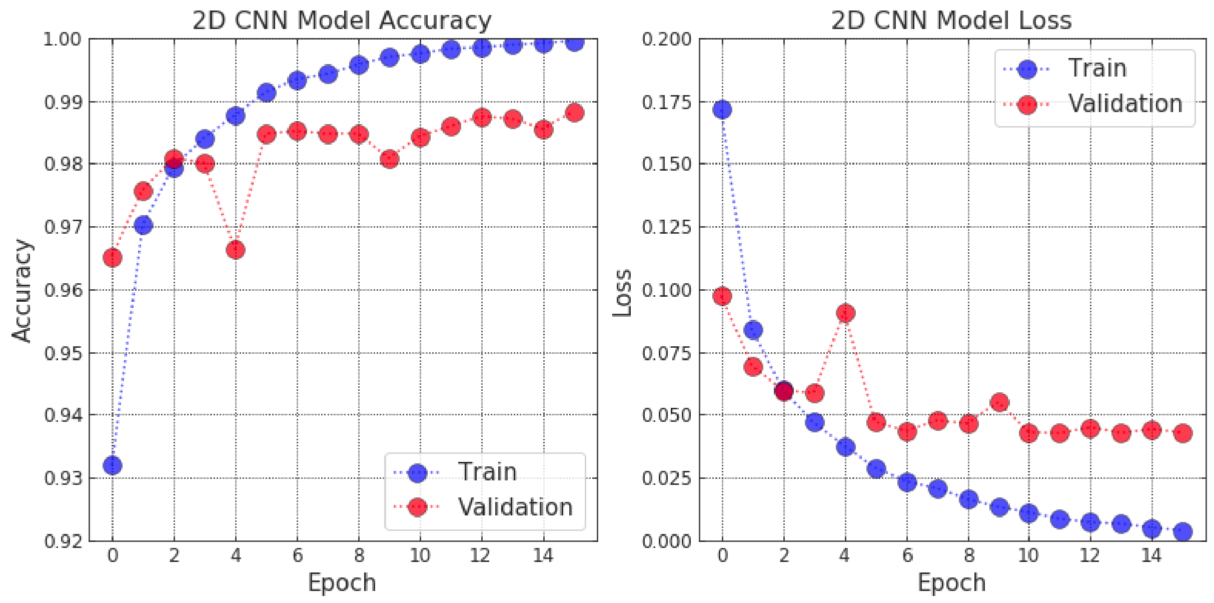
The adadelta optimizer is used with an initial learning rate (i.e., lr) of 0.1, an exponentially weighted average decay factor (i.e., rho) set to 0.95, and a learning rate decay (i.e., decay) of 0.00001. The optimizer used in the 2D CNN is parameterized differently than the optimizer in the 1D CNN, that is because the parameter adjustment offered better results for the 2D CNN, while the adjustment did not significantly improve results for the 1D CNN. The output layer of the 2D CNN is a single node with a sigmoid activation function for binary classification.

4.0 Results and Conclusion

The results of GridSearchCV performed over the five machine learning algorithms are compared in terms of model accuracy on the test split of the cross validation using the training dataset (Figure 11). LRC, KNN, and RFC result in relatively narrow accuracy ranges, however, both ADA and GBC have significantly low-accuracy outliers, with scores below 0.45 and 0.26 respectively. The low-accuracy outliers in both the ADA and GBC GridSearchCV results correspond to test runs with the learning rate (i.e., learning_rate) set to 10.0.

The main objective in a CNN learning model is to reduce (or minimize) the loss function's value by changing the weight vector values through backpropagation. Performance metrics of the 1D and 2D CNN models are recorded at each epoch during training (Figures 12 and 13). The validation accuracy of the 1D CNN begins to decouple from the train accuracy at the seventh epoch, and by the twelfth epoch, the validation accuracy shows signs of over fitting. This suggests that the model might be improved, and generalize to unseen data better, by training over less epochs (e.g., 7 or 8 epochs rather than 12). In contrast, the 2D CNN validation accuracy continues to increase with each epoch, albeit at a

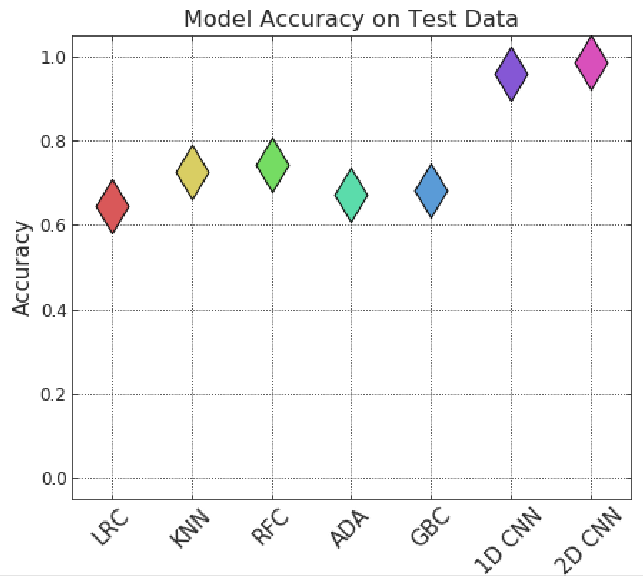
Figure 13



slower rate than the training accuracy. This observation suggests that the model continues to improve, rather than over fit with each epoch — this is reflected in the validation loss display as well.

Each of the five machine learning algorithms are fit — using the best parameters from the GridSearchCV — to the training dataset, and tested with the test dataset. The two CNN models are also fit to the train dataset and accuracy scores are calculated for all of the models and compared (Figure 14). The performance of the five machine learning models is mediocre at best. This could be due to poor feature selection, or inappropriate model selection for image classification. Better results might be obtained by using higher quality/resolution images (e.g., in terms of pixel count), or by using different image features such as Canny edges or band distribution features. The 1D CNN and 2D CNN models perform the image classification with test accuracy scores of 95.98% and 98.69% respectively. Therefore, it is recommend that the client select either one of the two CNN models. If compute resources are limited and the ability to scale is important, then the 1D CNN is recommended. If maximizing accuracy is more important, then the 2D CNN is recommended.

Figure 14


Model Accuracy on Test Data
Accuracy
LRC
KNN
RFC
ADA
GBC
1D CNN
2D CNN