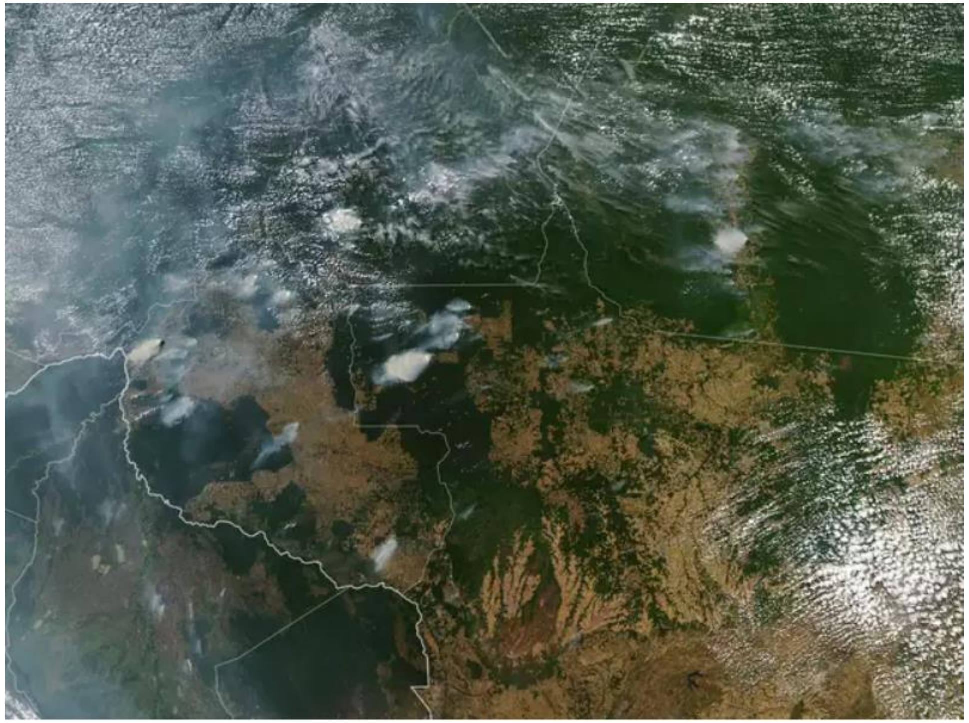
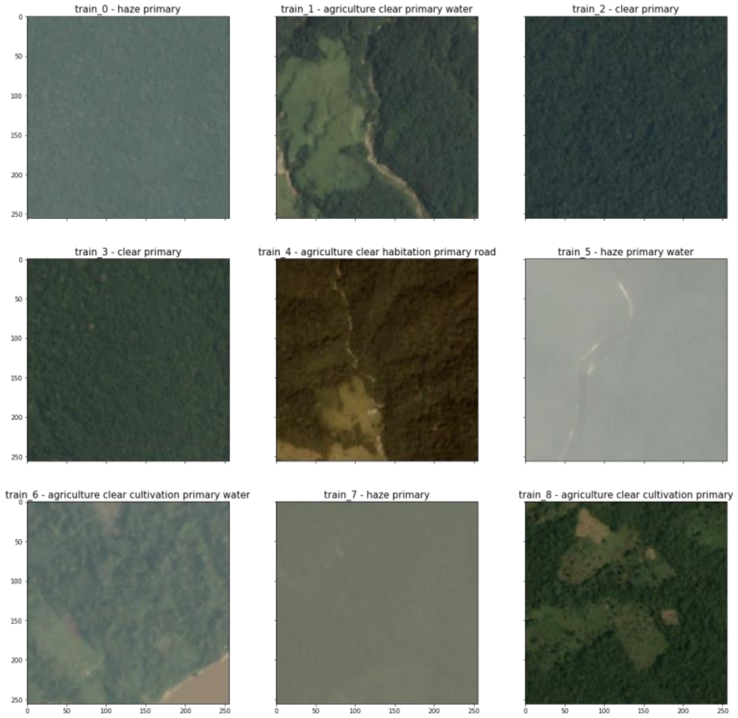
Amazon Rainforest Image Classification





CONNOR MCANUFF - NOVEMBER 2019

Content	Slide Index
Project Overview:	3-5
Data Wrangling:	6-8
Exploratory Data Analysis:	9-24
Machine Learning:	25-45
Conclusions:	46-48

Project Overview – Amazon Rainforest

- **6,000,000 km^2** across multiple countries.
- Several million species of plant, tree, insect, and animal life.
- **Contains human civilization** resulting in encroachment, exploitation, deforestation, and other forms of destruction:
  - **Reduced biodiversity;**
  - **Habitat loss;**
  - **Climate change;**
  - **Desertification;**
  - **Soil erosion.**

Project Overview – Satellite Imagery

- Can be used to monitor large areas of land.
- Previously, 30 m/pixel or 250 m/pixel imagery used – cannot identify small-scale deforestation/degradation.
- **Planet (company) plans to collect daily 3-5 m/pixel imagery of Earth's surface.**

Project Overview – Value in Classifying Satellite Images

- Scale and frequency of the Amazon Rainforest imaging means it would be **extremely cumbersome and expensive to manually classify images.**
- **Automated classification** would allow for **rapid mapping** of Amazon Rainforest to identify deforestation/degradation – action can be taken by governments or NGOs.
- Other uses such as infrastructure, aid, census, and resource planning.

Data Wrangling – Dataset Overview

- Sourced from former Kaggle competition.
- **40,479 satellite images:**
  - 256 x 256 pixels (946.2m x 947.2m on ground).
  - TIFF (4-channel) and JPEG (3-channel) format.
- **17 classifications (multilabels):**
  - Atmospheric conditions:
    - *Cloudy, partly cloudy, hazy.*
  - Common land cover/use:
    - *Primary rainforest, water, habitation, agriculture, road, cultivation, bare ground.*
  - Uncommon land cover/use:
    - *Slash and burn, conventional mining, selective logging, artisanal mining, blooming, blow down.*

Data Wrangling – Dataset Overview (2)

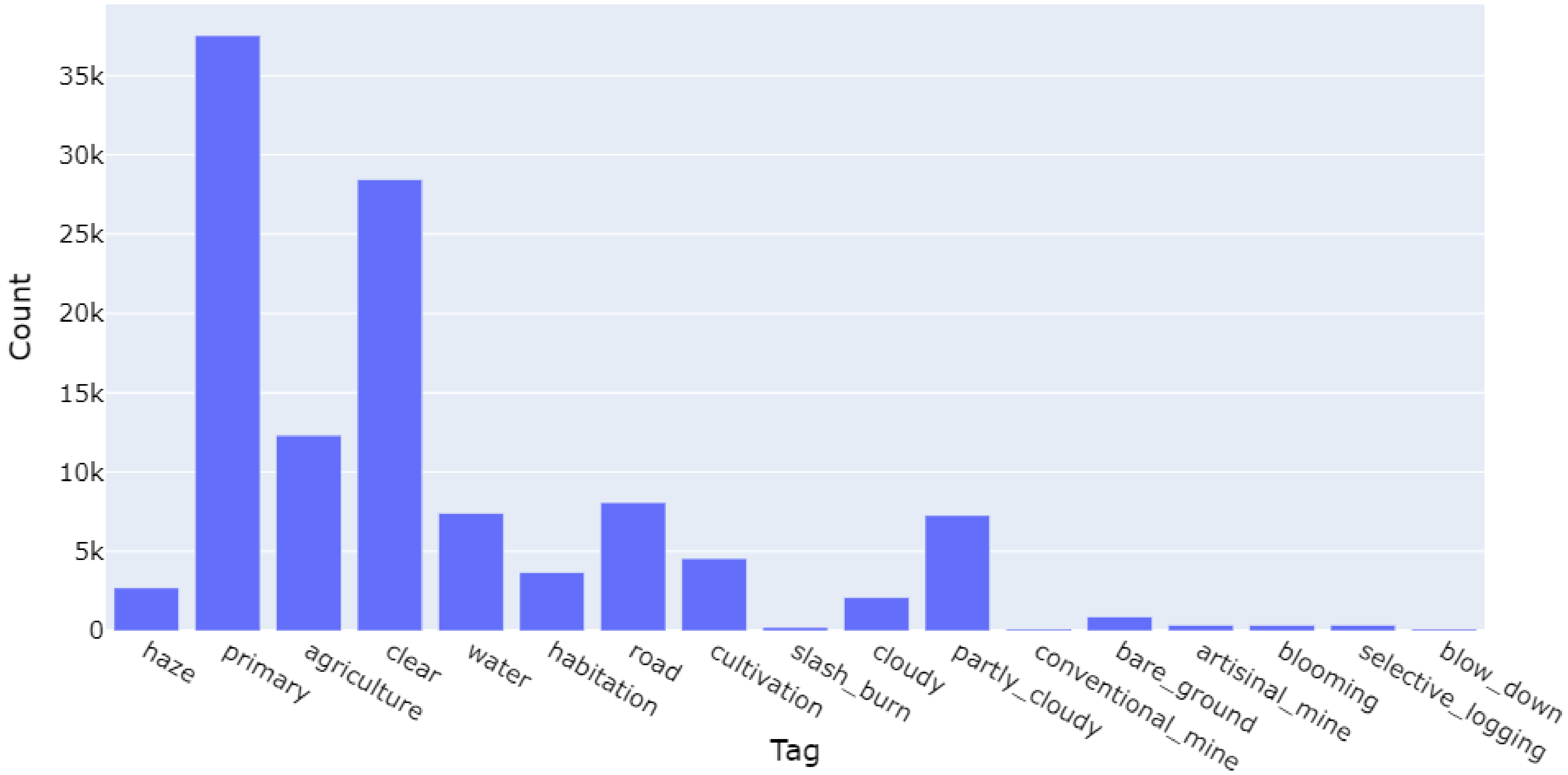
- CSV file with list of image file names and their associated labels/tags:

image_name	**tags**
train_0	haze primary
train_1	agriculture clear primary water
train_2	clear primary
train_3	clear primary
train_4	agriculture clear habitation primary road
train_5	haze primary water
train_6	agriculture clear cultivation primary water

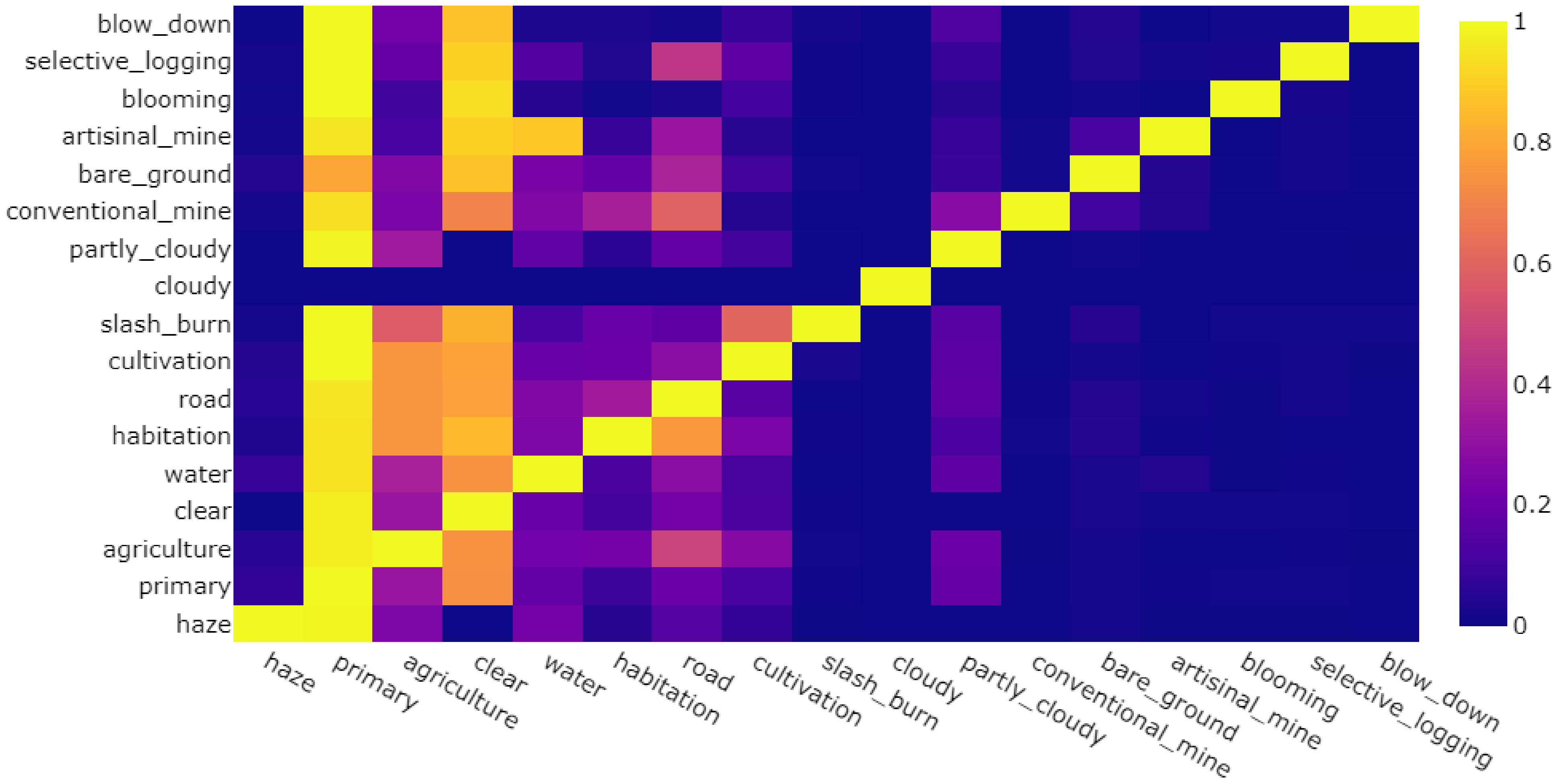
Data Wrangling – Importing and Cleaning

- CSV file read directly into a Pandas DataFrame.
- Images brought in for viewing using opencv methods:
  - imread
  - imshow
- Dataset file constructed for use in machine learning models.

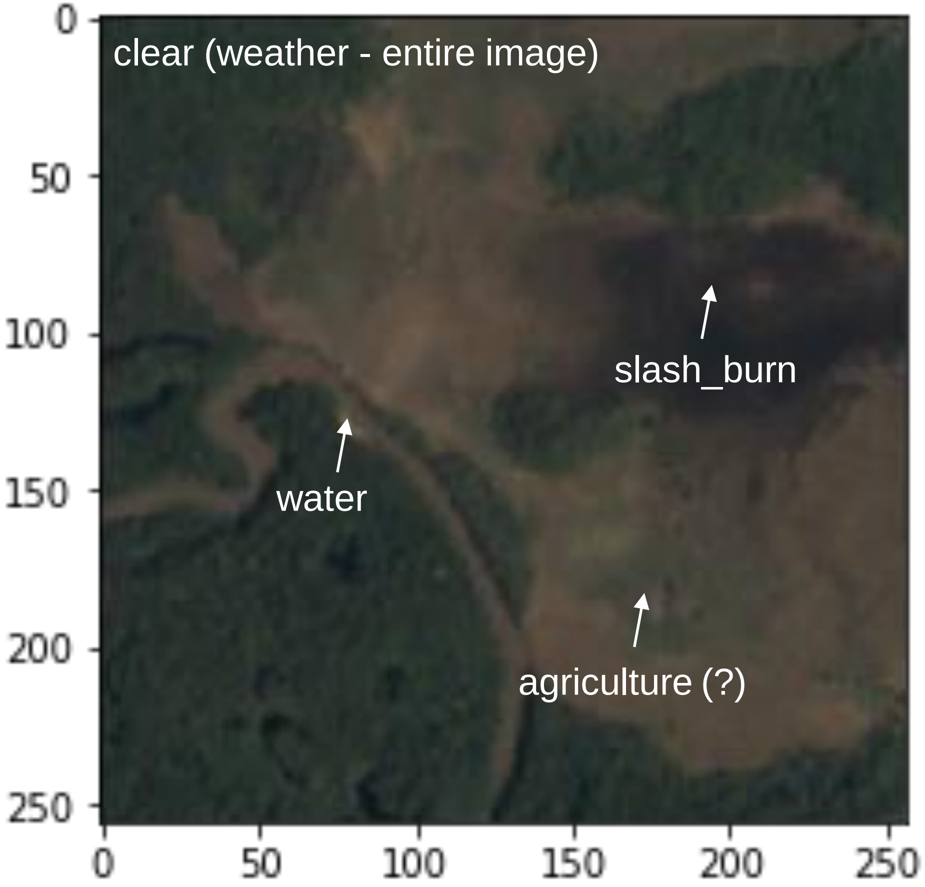
Exploratory Data Analysis – Tag Occurrences



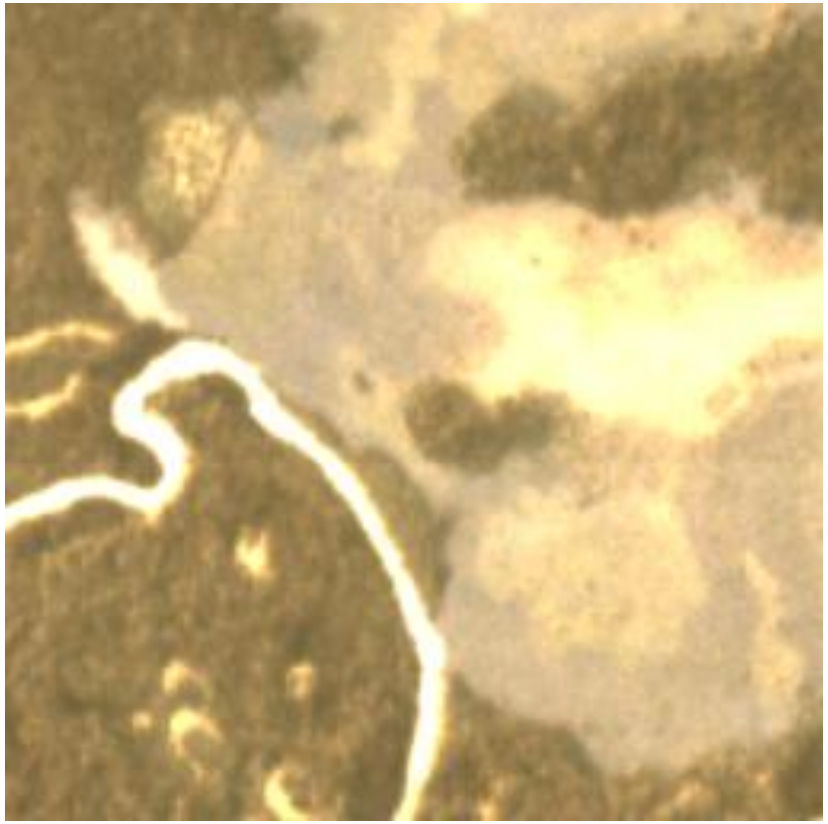
Exploratory Data Analysis – Tag Co-occurrences



Exploratory Data Analysis – Image Samples

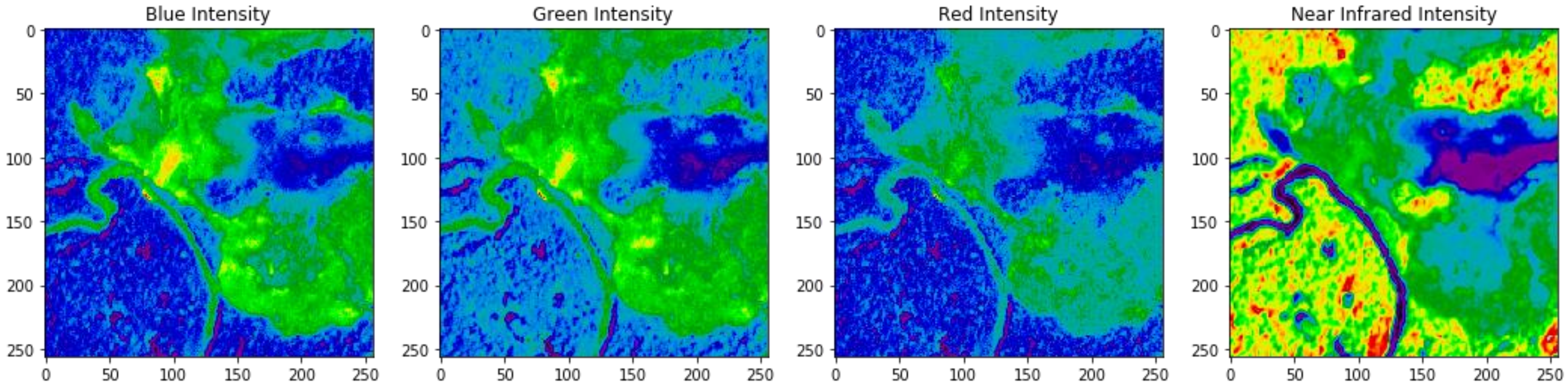


train_10 JPEG

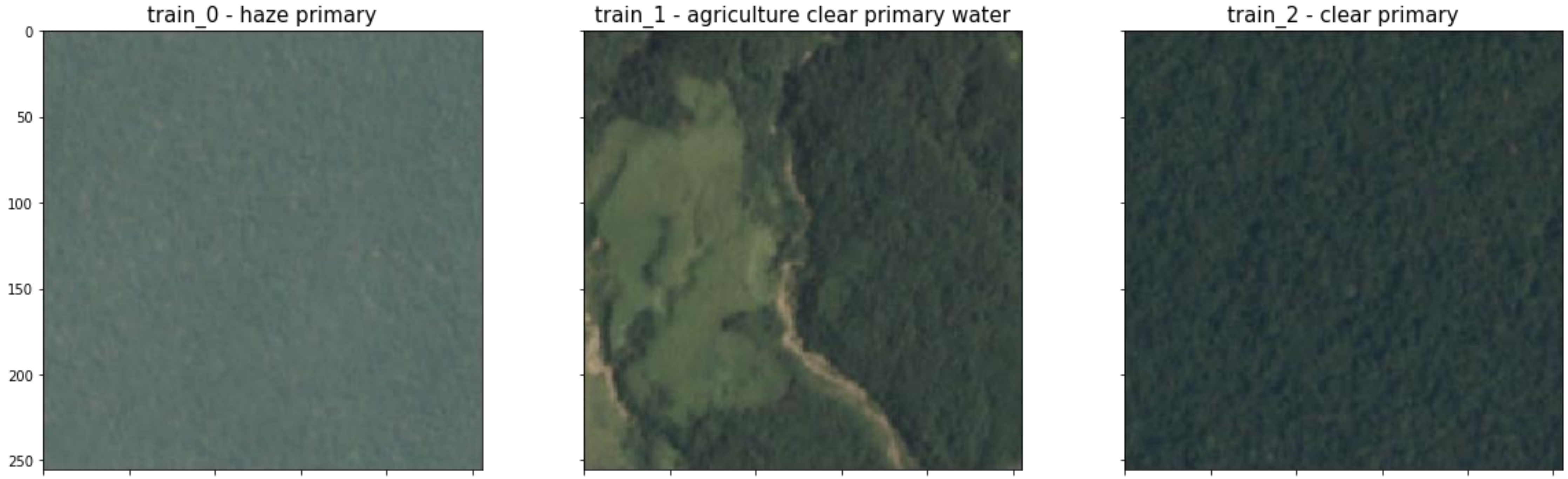


train_10 TIFF

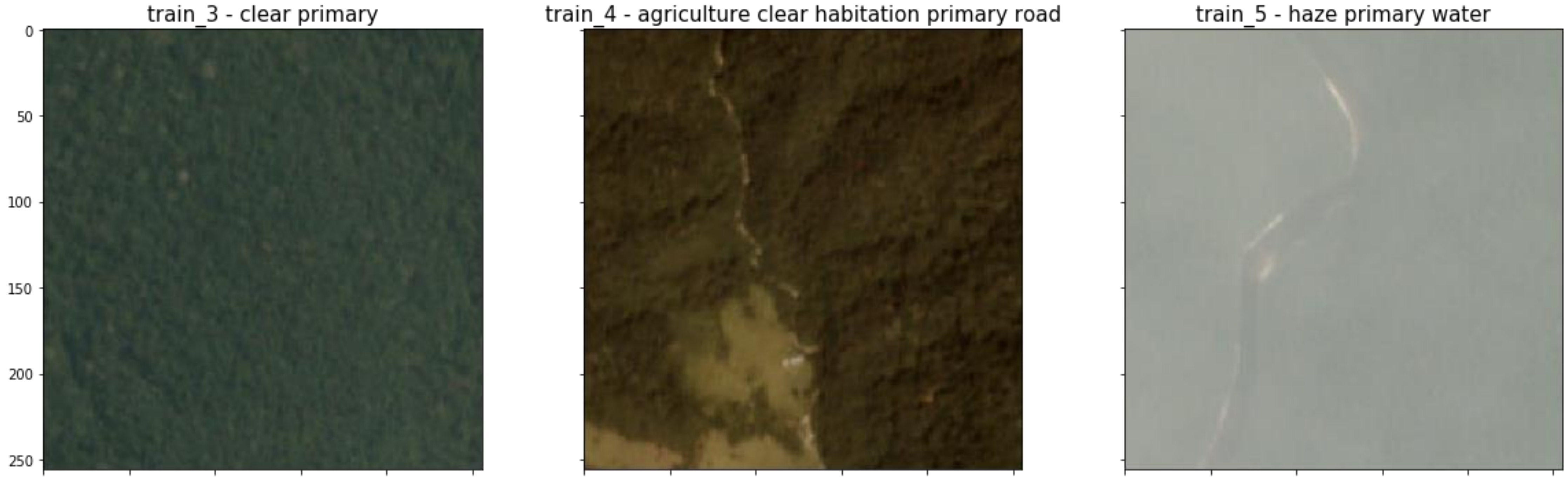
Exploratory Data Analysis – TIFF Images



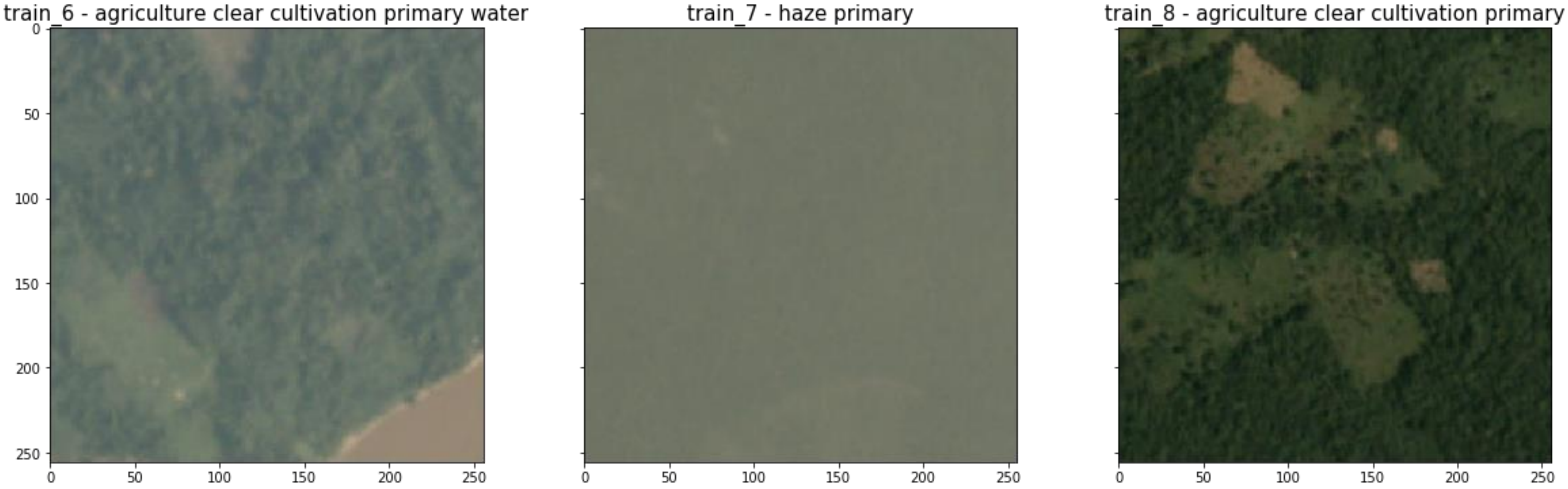
Exploratory Data Analysis – Image Samples



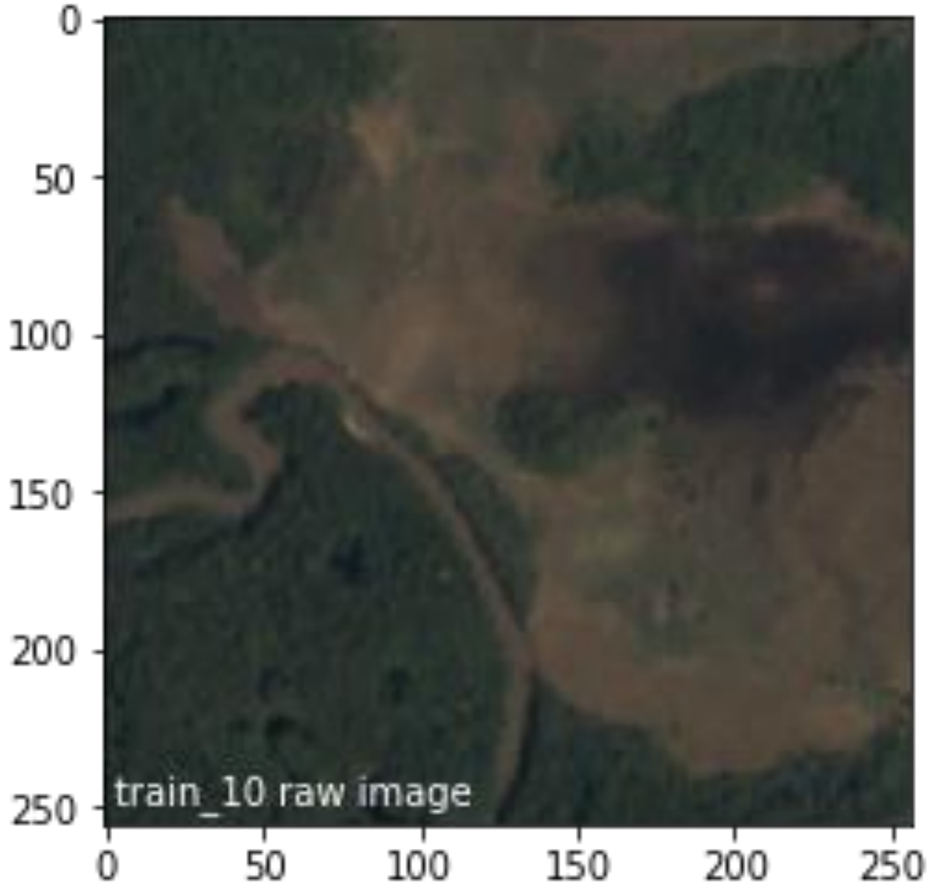
Exploratory Data Analysis – Image Samples (2)

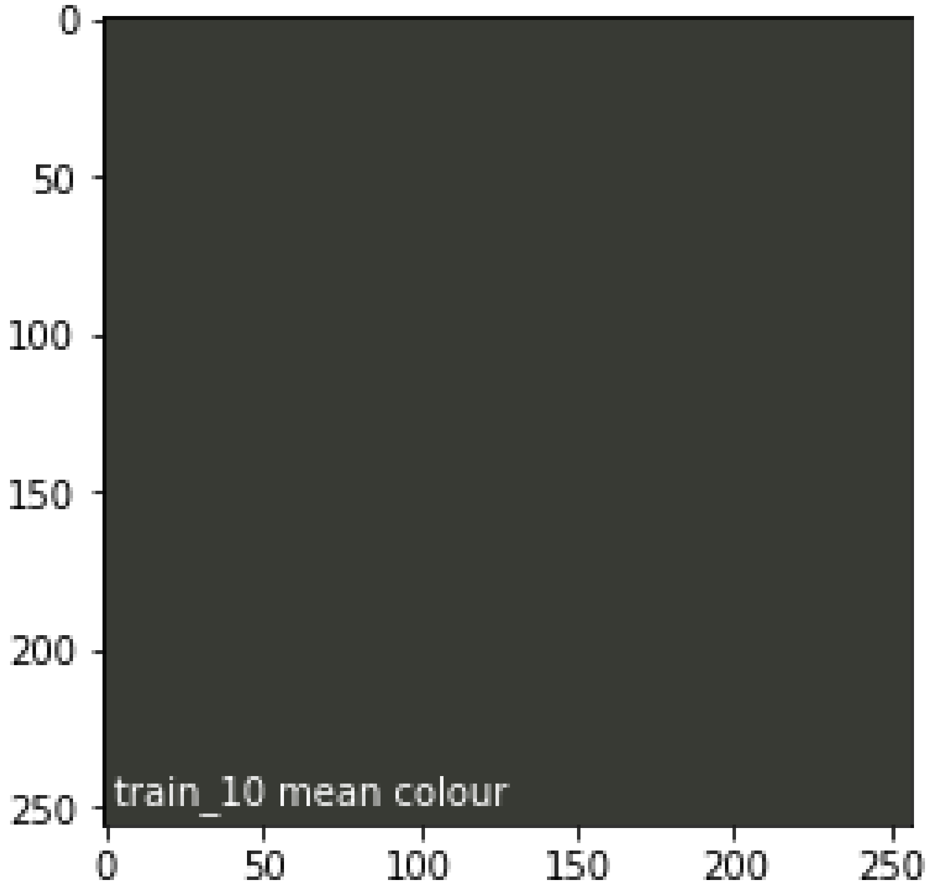


Exploratory Data Analysis – Image Samples (3)

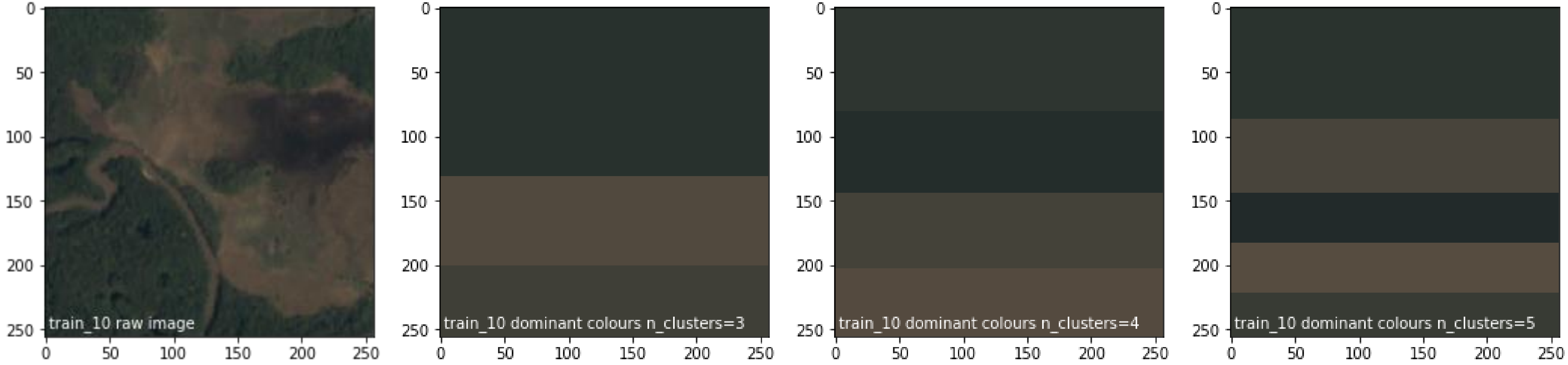


Exploratory Data Analysis – Single Image Mean Colour

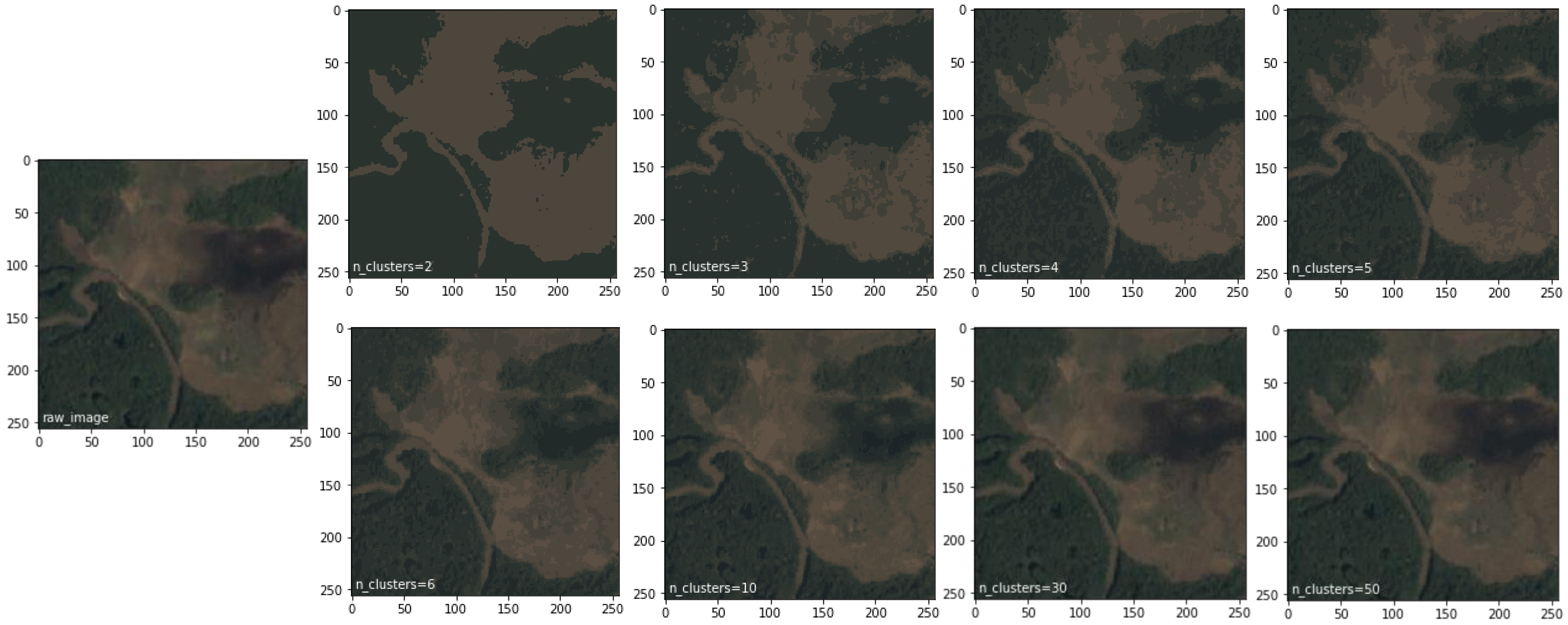




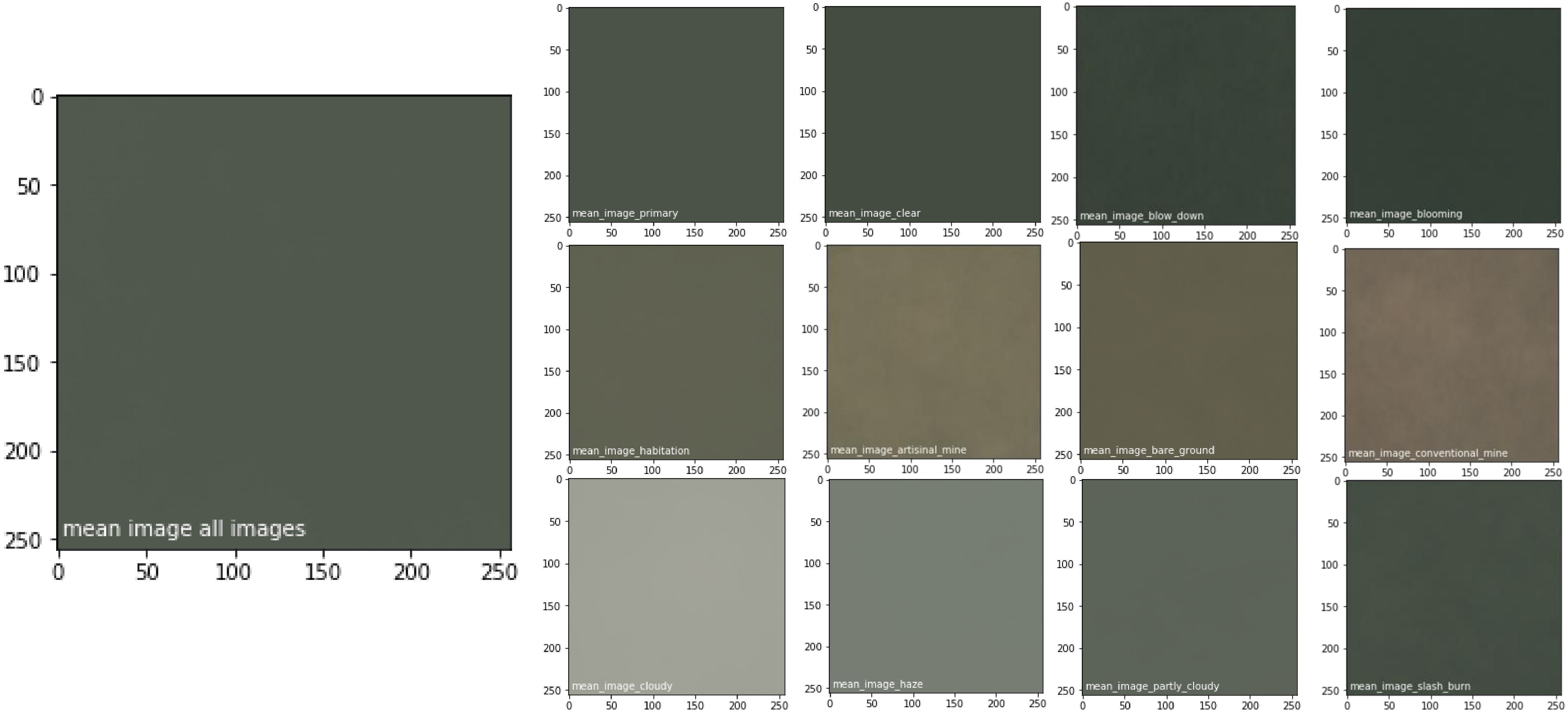
Exploratory Data Analysis – Dominant Image Colours



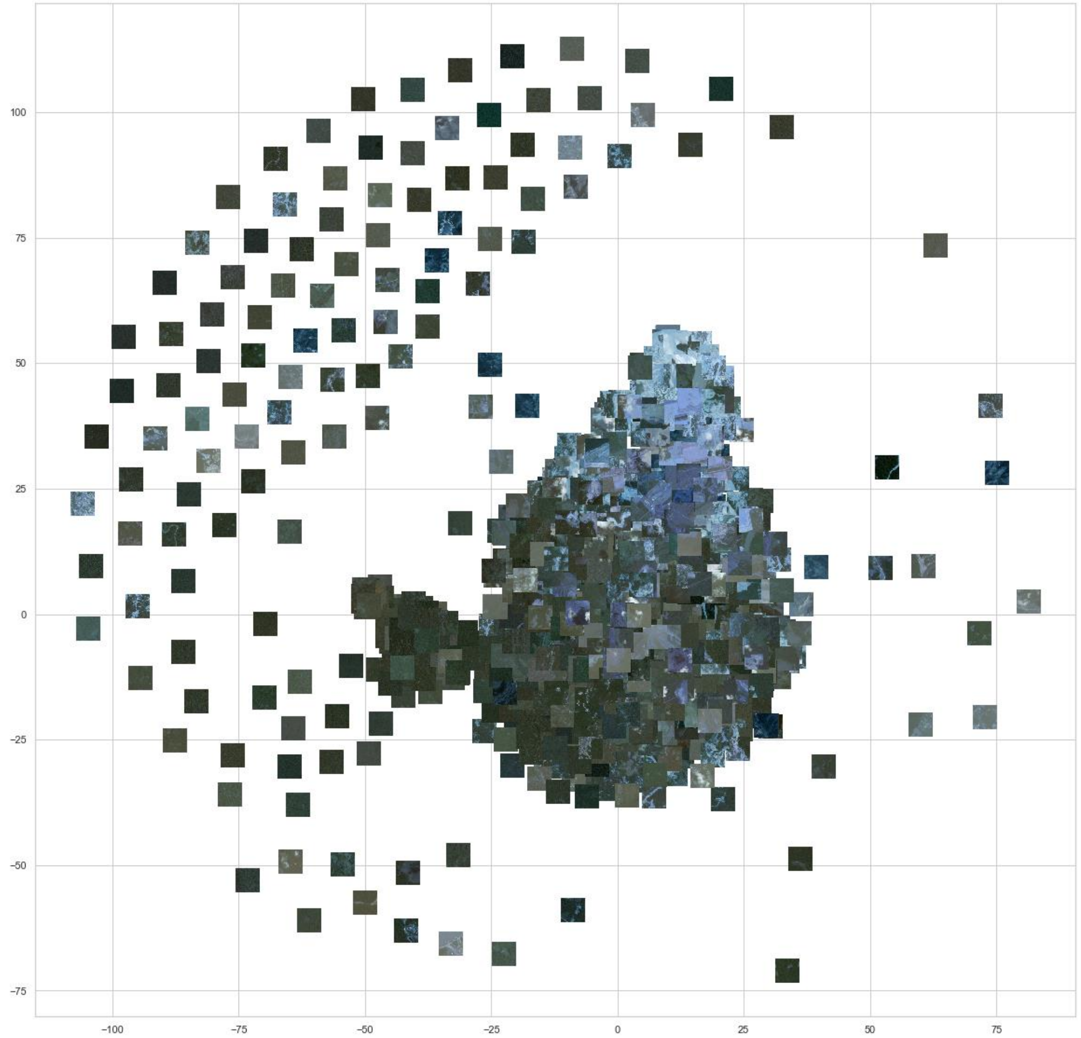
Exploratory Data Analysis – Dominant Image Colours (2)



Exploratory Data Analysis – Image Set Means

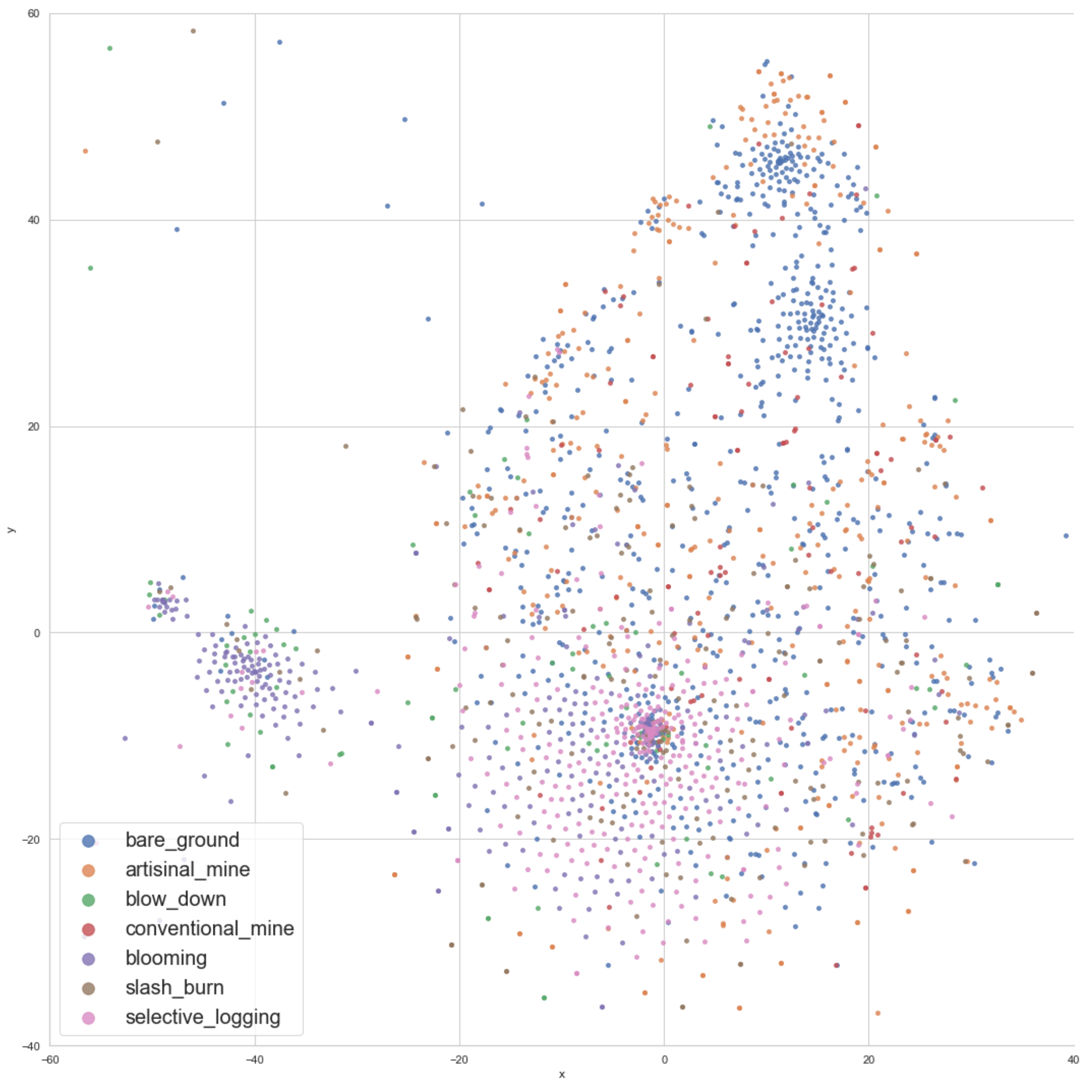


Exploratory Data Analysis – Image Clustering

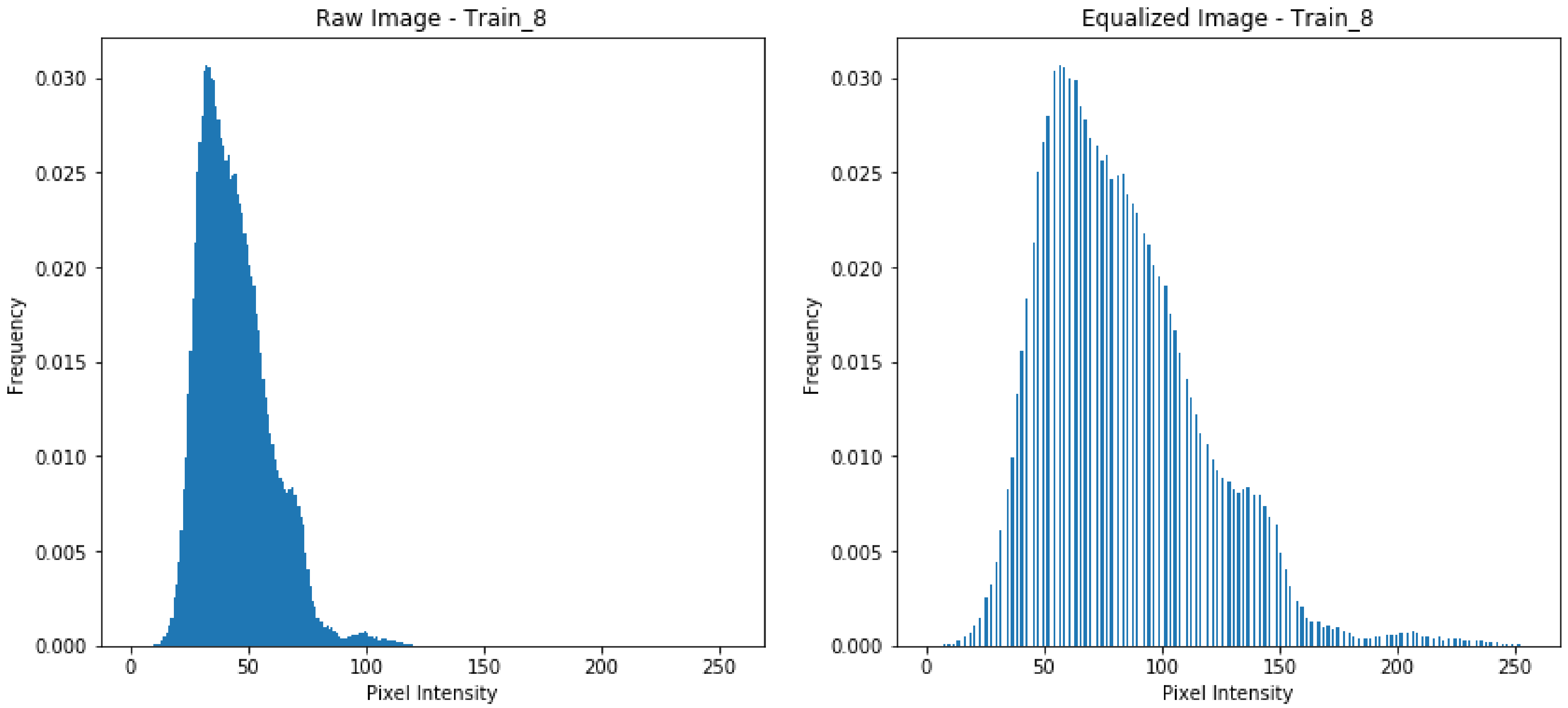


Exploratory Data Analysis – Image Clustering (2)

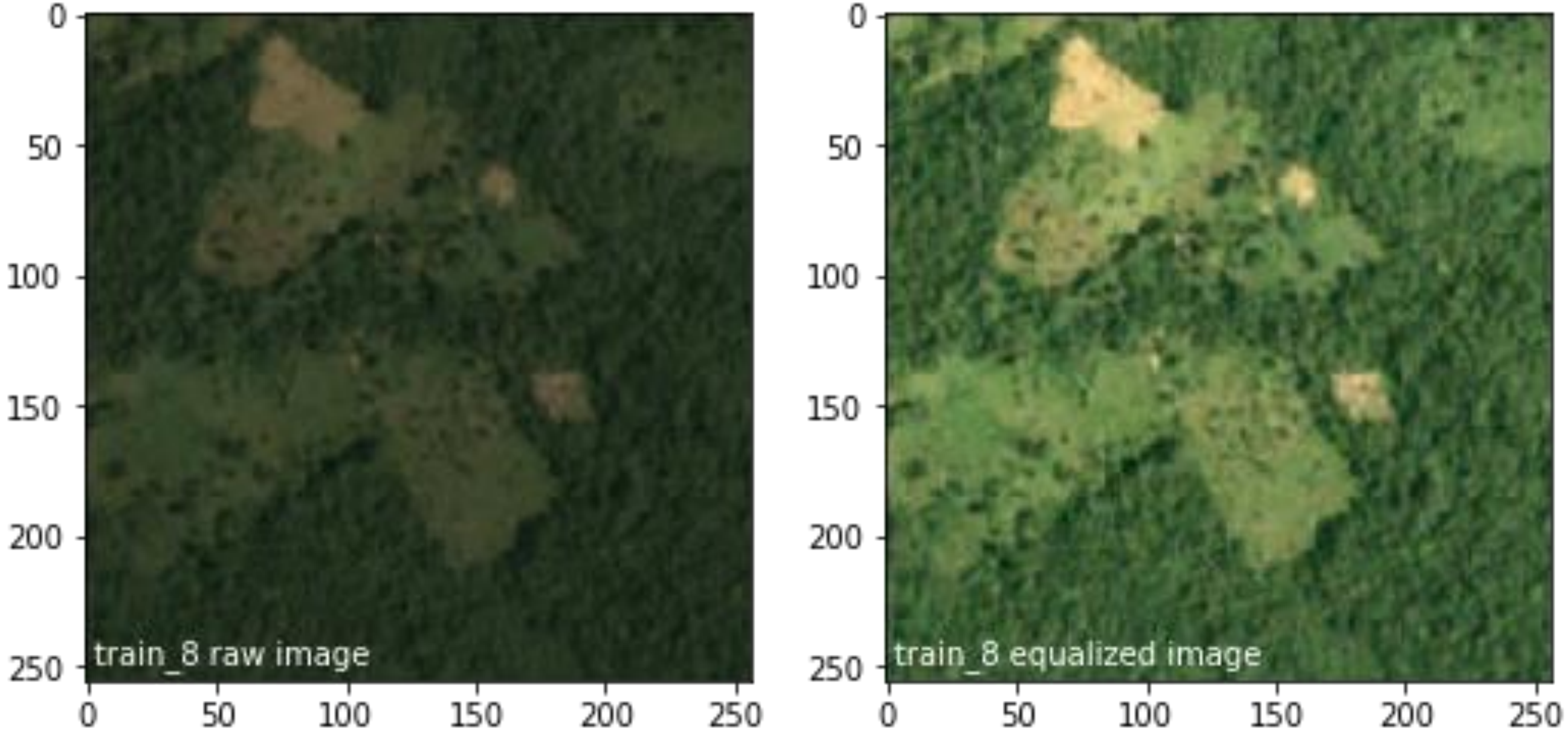
Exploratory Data Analysis – Image Clustering (3)



Exploratory Data Analysis – Histogram Equalization



Exploratory Data Analysis – Histogram Equalization (2)



Machine Learning – Overview

Purpose: Classify satellite images of the Amazon Rainforest as having one or more weather/land use tag.

- Convolutional Neural Networks (CNNs) are widely used to classify images.
- Several CNN models have been developed, trained, evaluated, and compared.
- Models have varying architectures.
- Implemented in Python using Keras and Tensorflow backend.

Machine Learning – Overview (2)

Purpose: Classify satellite images of the Amazon Rainforest as having one or more weather/land use tag.

Model #	Model Name	Optimizer	Regularization	Image Augmentation
1	base	SGD		
2	dropout	SGD	✓	
3	augmentation	SGD		✓
4	dropout_augment	SGD	✓	✓
5	adam	Adam		
6	adam_dropout	Adam	✓	
7	adam_augment	Adam		✓
8	adam_dropout_augment	Adam	✓	✓
9	transfer	SGD		

Machine Learning – Dataset Formatting

Dataset formatting is required to organize the raw images to be input into the CNN models.

1) Use dictionary of tags (keys) and integers (values) as map to create the one hot encoding sequence for each observation.
2) Load each image using keras.preprocessing.image.load_img().
3) Convert each image to numpy array.
4) Append the one hot encoding sequence and the image numpy array to the lists of targets and images.
5) Array of images becomes X (train and test) and one hot encoding sequence list becomes y (train and test).
6) Dataset is saved in compressed format for later loading.
7) Once dataset is loaded, it is split into 70% train and 30% test.

Machine Learning – Model Evaluation

$$F1 = 2 * \frac{precision * recall}{(precision + recall)}$$

F1 Score: Harmonic mean of precision and recall. Equal weight for precision and recall.

Machine Learning – Model Evaluation

$$F1 = 2 * \frac{precision * recall}{(precision + recall)}$$

F1 Score: Harmonic mean of precision and recall. Equal weight for precision and recall.

$$\Big\downarrow \text{ Introduce Beta term}$$

$$FBeta = (1 + Beta^2) * \frac{precision * recall}{(Beta^2 * precision + recall)}$$

Fbeta Score: Beta < 1, more weight on precision. Beta > 1, more weight on recall.

Machine Learning – Model Evaluation

$$F1 = 2 * \frac{precision * recall}{(precision + recall)}$$

Introduce Beta term

$$FBeta = (1 + Beta^2) * \frac{precision * recall}{(Beta^2 * precision + recall)}$$

Set Beta=2

$$FBeta\ (F2) = 5 * \frac{precision * recall}{(4 * precision + recall)}$$

Note: Precision, recall, Fbeta calculated for each observation and then averaged. Tags with fewer observations will have less affect on the Fbeta score.

F1 Score: Harmonic mean of precision and recall. Equal weight for precision and recall.

Fbeta Score: Beta < 1, more weight on precision. Beta > 1, more weight on recall.

Multi-label precision and recall:

$$precision = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap h(x_i)|}{|h(x_i)|} \qquad recall = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap h(x_i)|}{|Y_i|}$$

$where:$

$n = \#\ of\ observations$

$Y_i = true\ label\ assignments\ of\ the\ i^{th}\ observation$

$x_i = i^{th}\ observation$

$h(x_i) = predicted\ label\ assignments\ of\ the\ i^{th}\ observation$

Machine Learning – CNN Model Theory and Architecture

- CNNs work by extracting features from images.
- Each additional model layer increases the complexity of the learned features.
- Base model architecture:

1) Images input as shape (128,128,3)
2) 2x convolutional layers
3) Max pooling layer
4) 2x convolutional layers
5) Max pooling layer
6) 2x convolutional layers
7) Max pooling layer
8) Flattening layer
9) Fully connected layer
10) Output layer for prediction

Machine Learning – CNN Model Theory and Architecture

- CNNs work by extracting features from images.
- Each additional model layer increases the complexity of the learned features.
- Base model architecture:

1) Images input as shape (128,128,3)
2) 2x convolutional layers → **(32)** 3x3 filters, ReLU activation, 'He' weight initialization.
3) Max pooling layer
4) 2x convolutional layers → **(64)** 3x3 filters, ReLU activation, 'He' weight initialization.
5) Max pooling layer
6) 2x convolutional layers → **(128)** 3x3 filters, ReLU activation, 'He' weight initialization.
7) Max pooling layer
8) Flattening layer
9) Fully connected layer
10) Output layer for prediction

Machine Learning – CNN Model Theory and Architecture

- CNNs work by extracting features from images.
- Each additional model layer increases the complexity of the learned features.
- Base model architecture:

1) Images input as shape (128,128,3)
2) 2x convolutional layers
3) Max pooling layer → **2x2**
4) 2x convolutional layers
5) Max pooling layer → **2x2**
6) 2x convolutional layers
7) Max pooling layer → **2x2**
8) Flattening layer
9) Fully connected layer
10) Output layer for prediction

Machine Learning – CNN Model Theory and Architecture

- CNNs work by extracting features from images.
- Each additional model layer increases the complexity of the learned features.
- Base model architecture:

1) Images input as shape (128,128,3)
2) 2x convolutional layers
3) Max pooling layer
4) 2x convolutional layers
5) Max pooling layer
6) 2x convolutional layers
7) Max pooling layer
8) Flattening layer → **Converting to 1D vector.**
9) Fully connected layer → **Shape=128, ReLU activation, 'He' initialization.**
10) Output layer for prediction → **Shape=17 for 17 possible tags.**

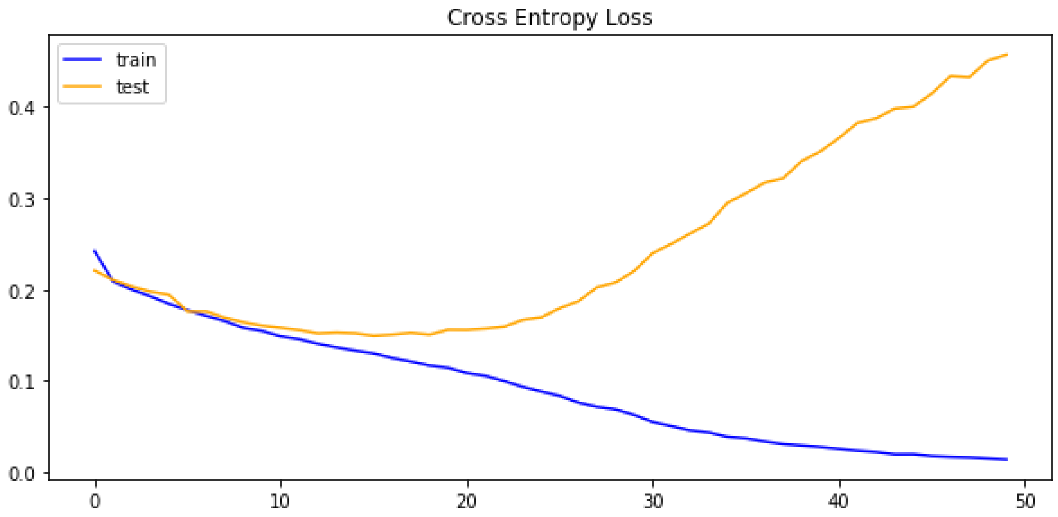
Machine Learning – Fitting and Evaluating

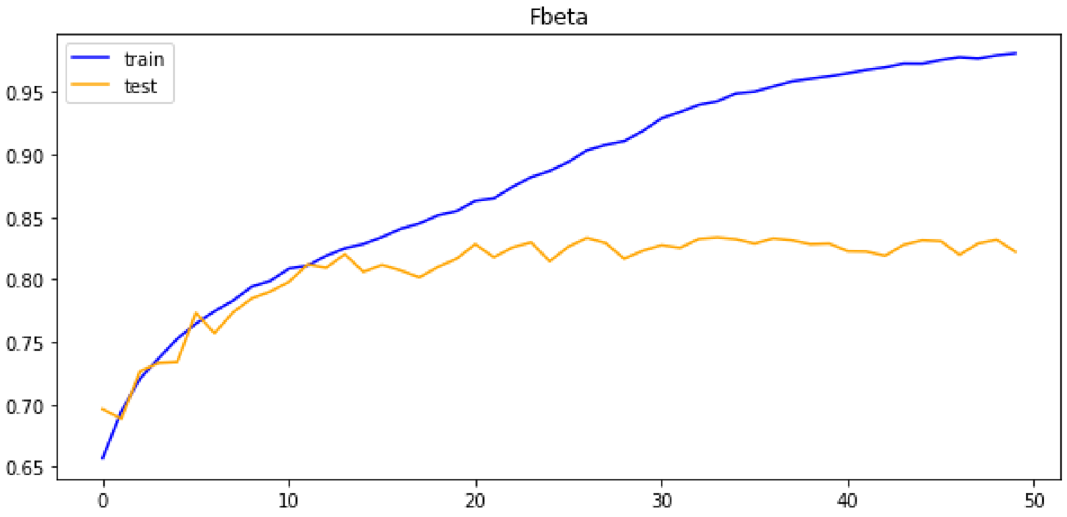
1) Create data generator using ImageDataGenerator. Images are rescaled by a factor of 1/255.
2) Use the data generator to create train and test iterators. Batch size = 128.
3) Fit the model using fit_generator for a set number of epochs.
4) Evaluate model using evaluate_generator.

Machine Learning – Base Model

- Model architecture as described previously.

Epochs: 50
Test loss: 0.534
Test Fbeta: 0.822

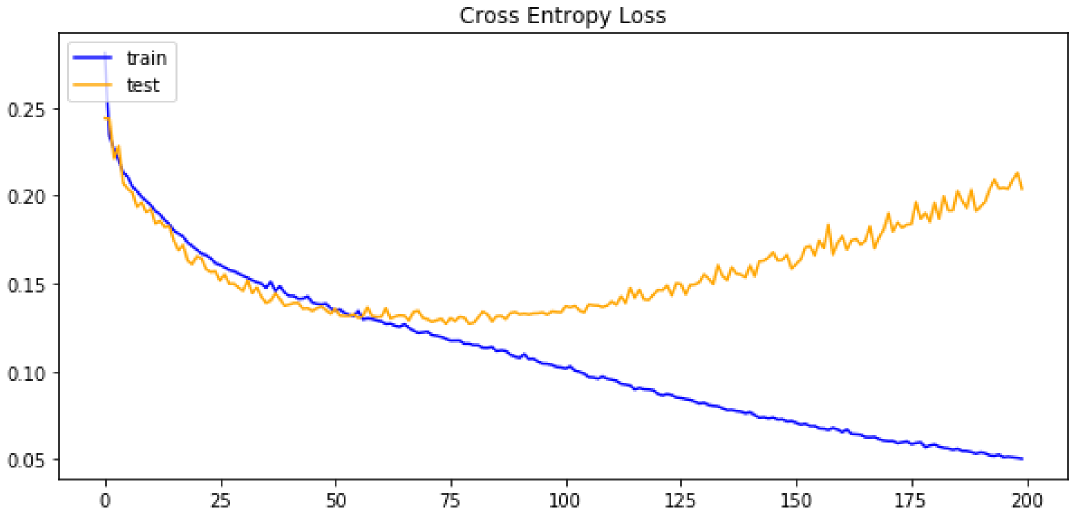


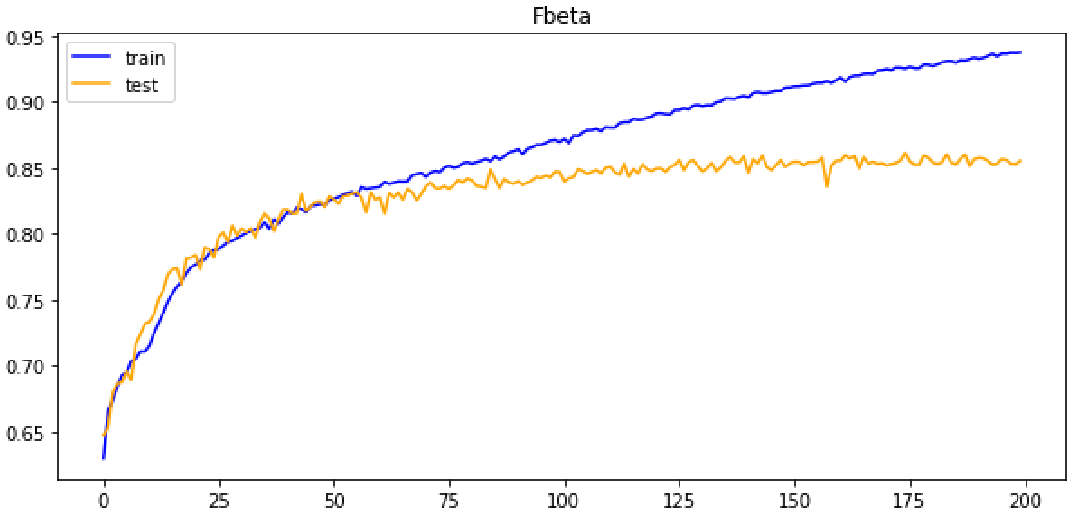


Machine Learning – Dropout Regularization

- Reduce overfitting.
- Dropout layers added after each convolutional block (20% inputs dropped) and after fully connected layer (50% inputs dropped).

Epochs: 200
Test loss: 0.235
Test Fbeta: 0.855

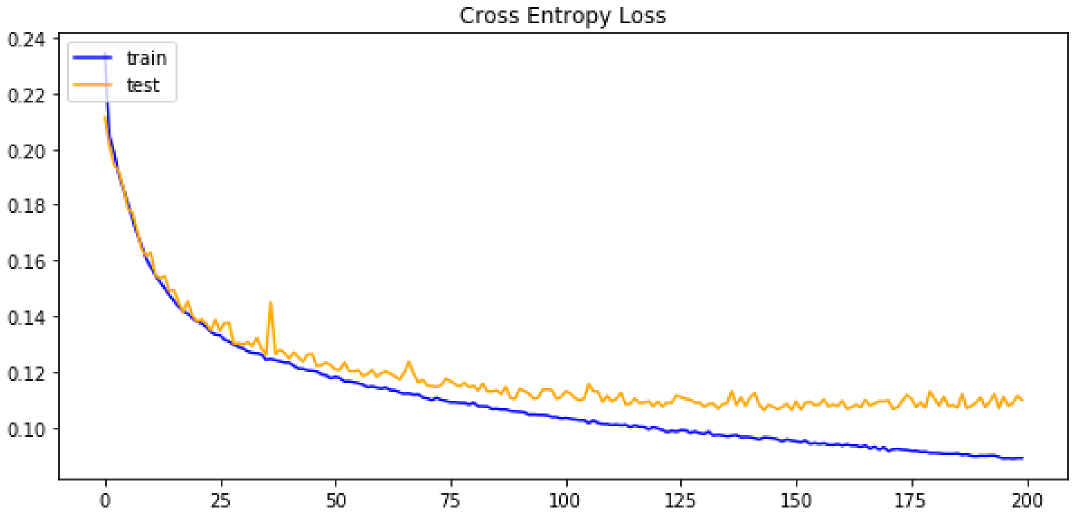


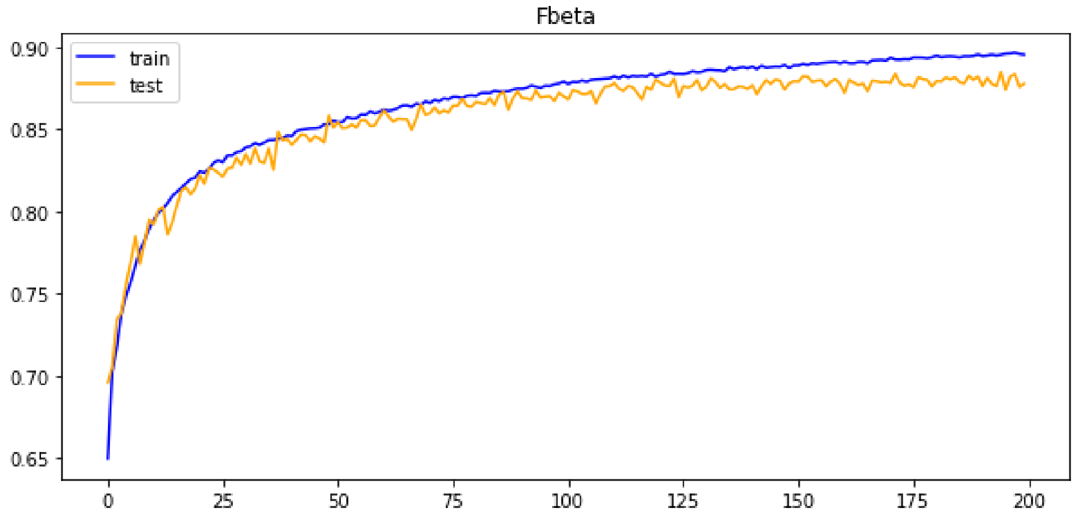


Machine Learning – Image Data Augmentation

- Artificially increase size of training dataset by creating modified versions of raw images.
- Train data generator set to flip, rotate raw images.

Epochs: 200
Test loss: 0.126
Test Fbeta: 0.878

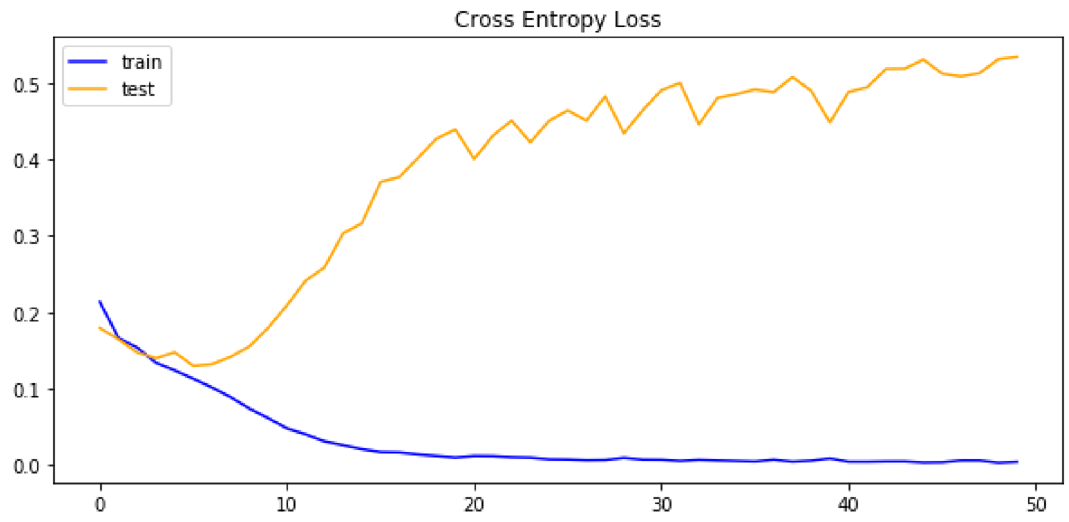


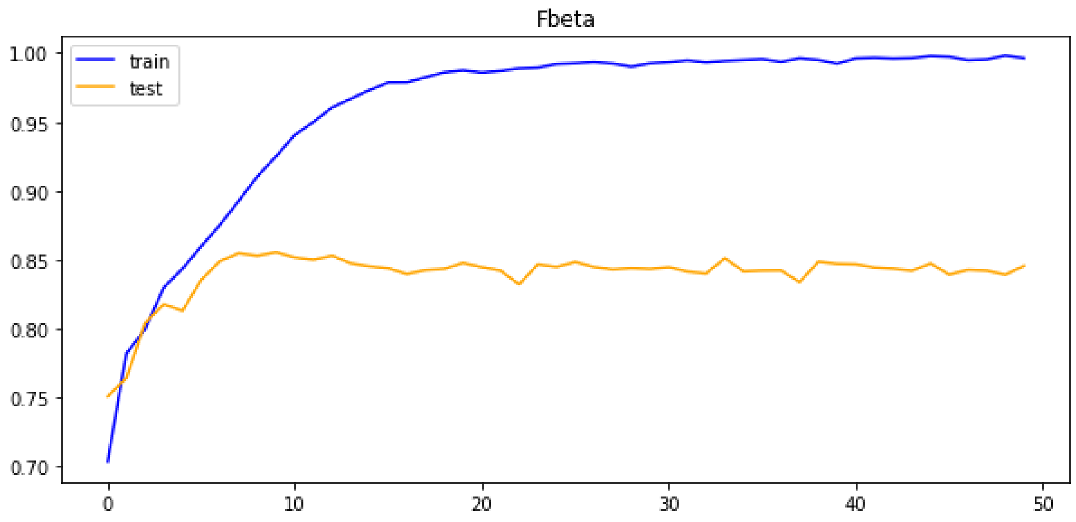


Machine Learning – Adam Optimizer

- Adaptive learning rate.
- Well suited for problems large in data and with noisy/sparse parameters.

Epochs: 50
Test loss: 0.711
Test Fbeta: 0.845

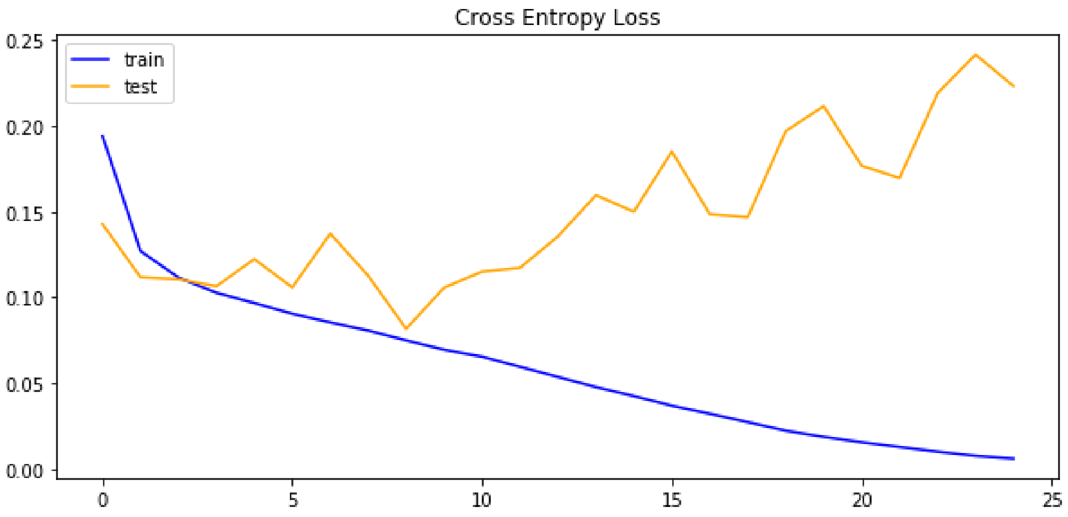


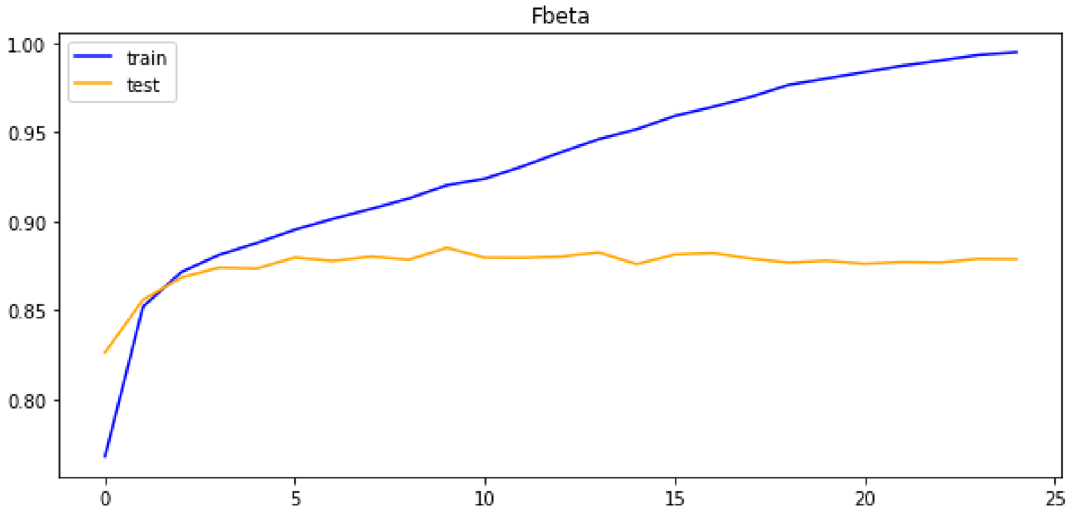


Machine Learning – Transfer Learning

- VGG-16 pre-trained model.
- Modified for Amazon Rainforest dataset input and predictions.

Epochs: 25
Test loss: 0.325
Test Fbeta: 0.879

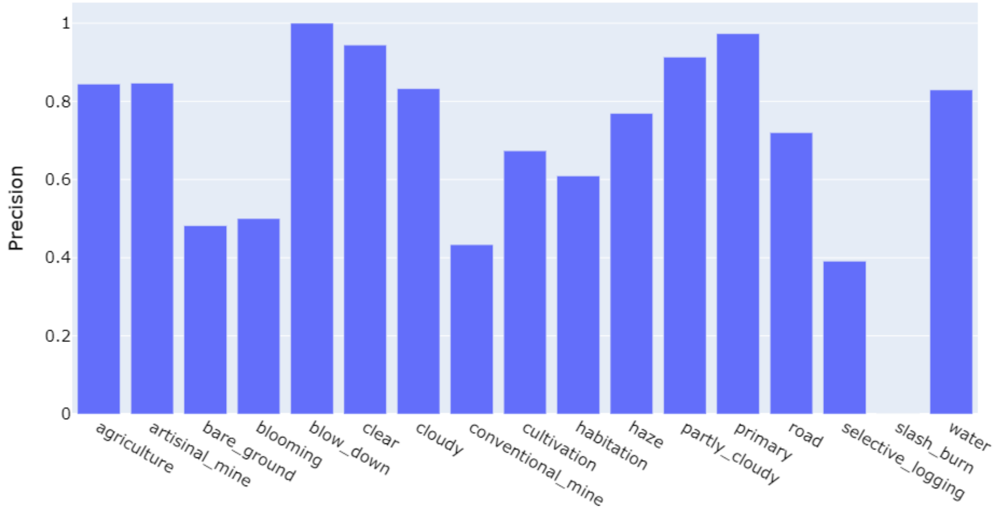


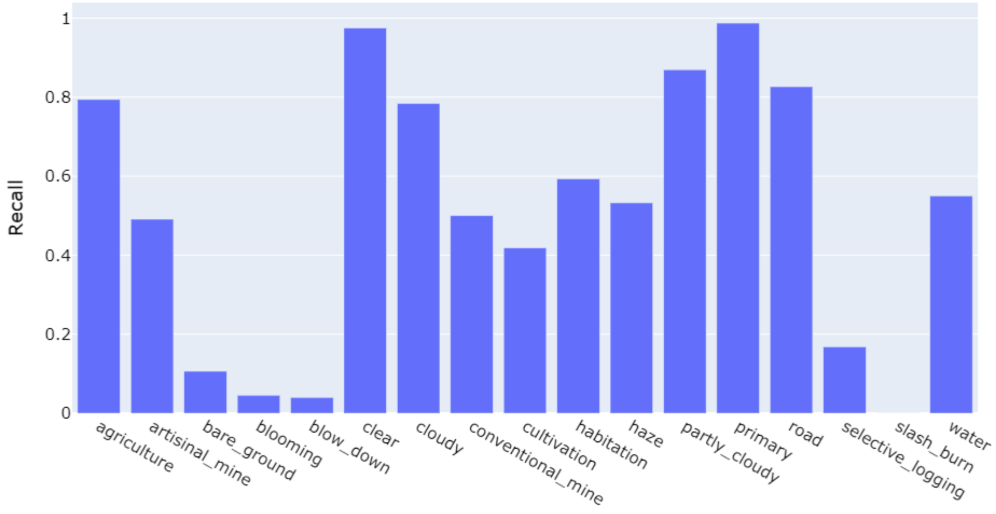


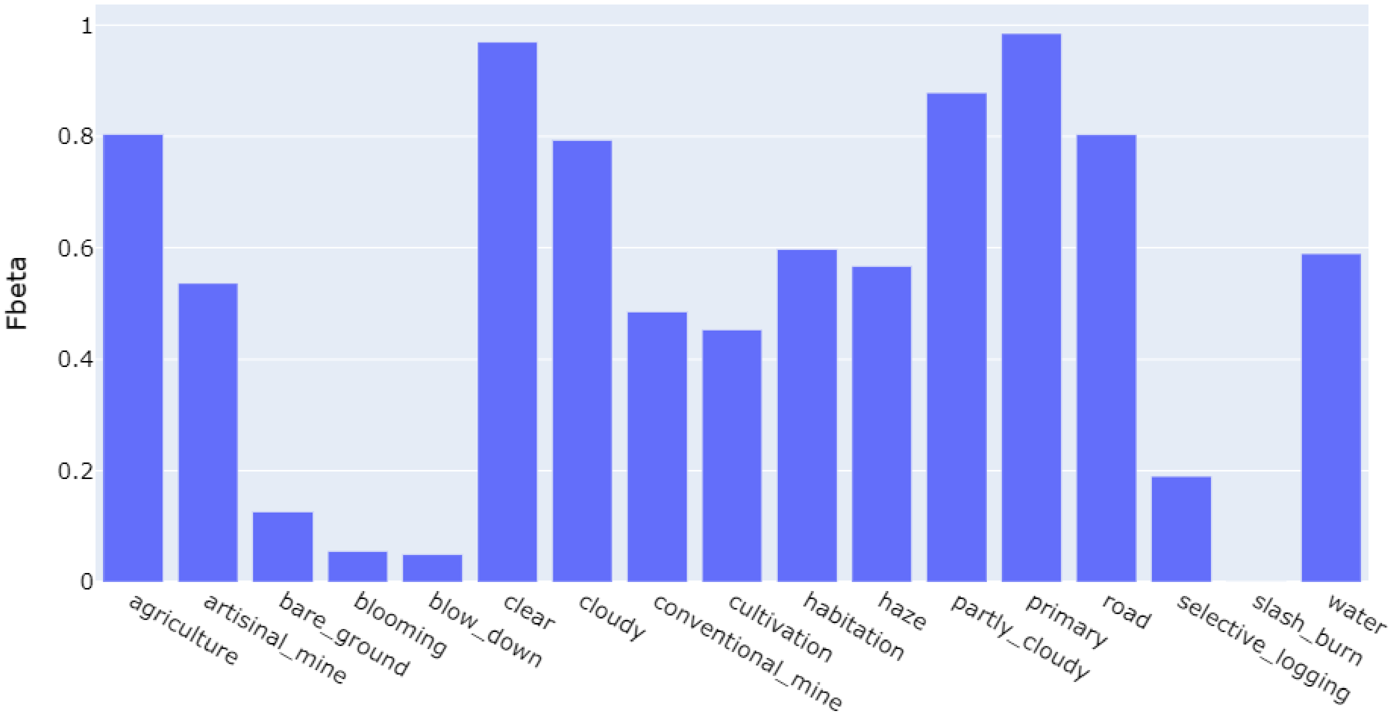
Machine Learning – Model Comparison

Model #	Model Name	Optimizer	Regulariz-ation	Image Augmenta-tion	Loss	Fbeta
1	base	SGD			0.596	0.822
2	dropout	SGD	✓		0.330	0.855
3	augmentation	SGD		✓	0.138	0.878
4	dropout_augment	SGD	✓	✓	0.121	0.840
5	adam	Adam			0.910	0.845
6	adam_dropout	Adam	✓		0.345	0.862
7	adam_augment	Adam		✓	0.104	0.878
8	adam_dropout_augment	Adam	✓	✓	0.125	0.870
9	transfer	SGD			0.205	0.879

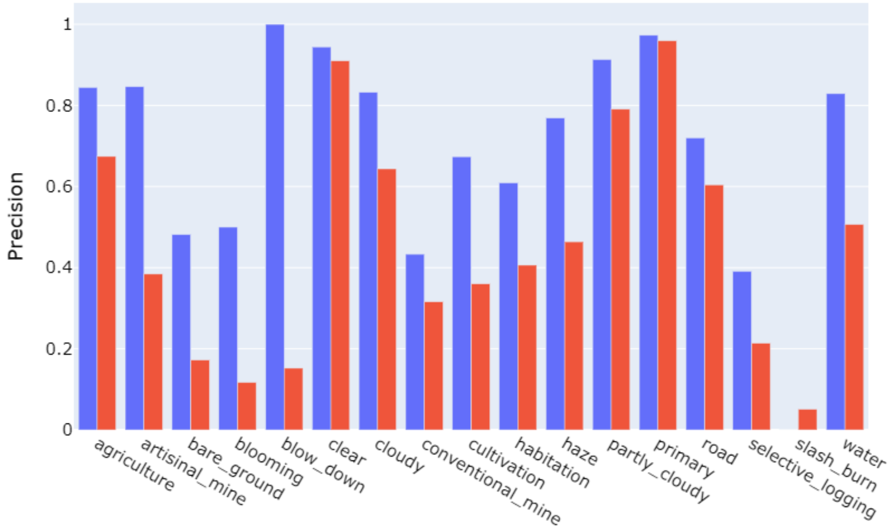
Machine Learning – Less Common Tag Prediction (1)

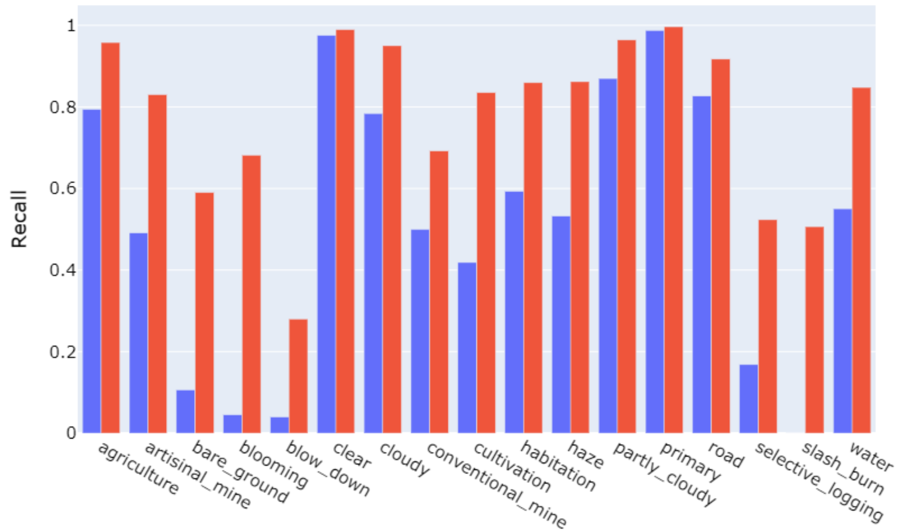




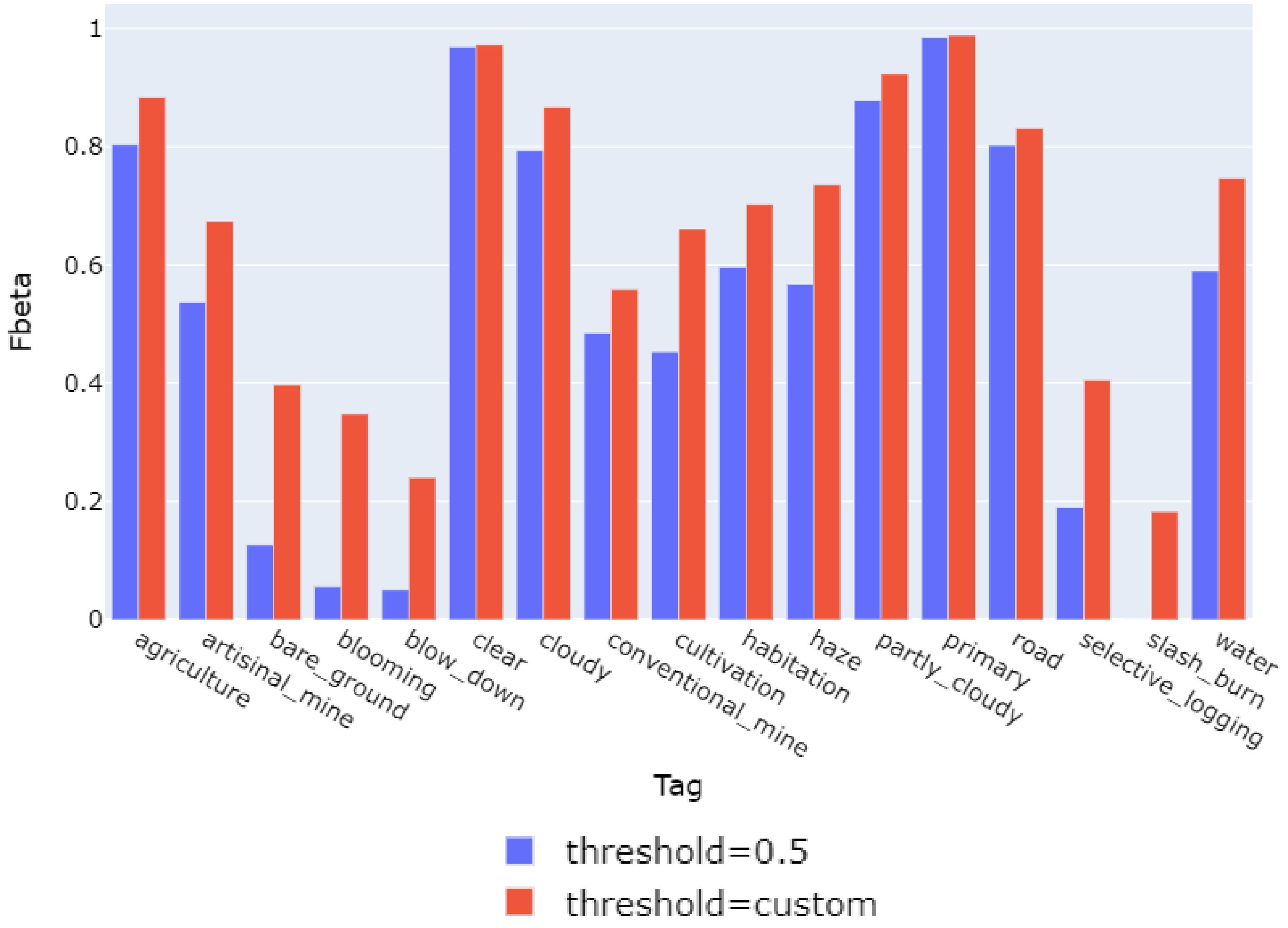


Machine Learning – Less Common Tag Prediction (2)





Custom Threshold vs Threshold=0.5 for Prediction:



Machine Learning – slash_burn model (1)

- **Original dataset:**
  - 209 images tagged with slash_burn.
  - 40,270 images not tagged with slash_burn.
- **Slash_burn model dataset:**
  - Train:
    - 584 raw images tagged with slash_burn:
      - 146 raw images
      - 146 raw images rotated 90°
      - 146 raw images rotated 180°
      - 146 raw images rotated 270°
    - 584 random raw images from original dataset not tagged with slash_burn.
  - Test:
    - 63 raw images tagged with slash_burn
    - 437 random raw images from original dataset not tagged with slash_burn.

Machine Learning – slash_burn model (2)

- **Original augment_model slash_burn predictive performance:**

Metric	**Prediction Threshold = 0.5**	**Prediction Threshold = Optimized**
Test Precision	0.000	0.050
Test Recall	0.000	0.506
Test Fbeta	0.000	0.182

- **Specialized slash_burn model:**

Metric	**Prediction Threshold = 0.5**	**Prediction Threshold = Optimized**
Test Precision	0.314	0.313
Test Recall	0.793	0.809
Test Fbeta	0.608	0.614

'Guessing' all ones achieves Test Fbeta = 0.419

Conclusions (1)

- Dropout regularization, image augmentation, and the Adam optimizer all increased model performance individually.
- The VGG-16 model was successfully used as a transfer learning model to match the performance of the best performing model built from scratch.
- The Fbeta score of all models is misleading – the metric evaluates the mean Fbeta score across all observations – the unbalanced dataset results in poorly performing tags having little effect on the overall Fbeta score.
- The Fbeta scores for less common tags are poor for all models.

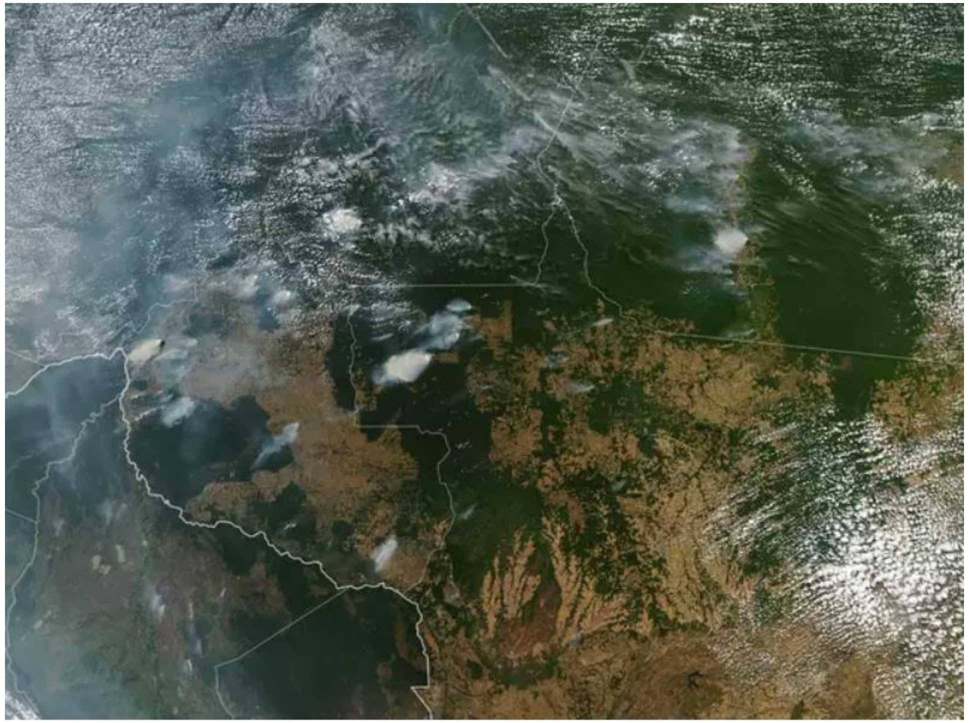
Conclusions (2)

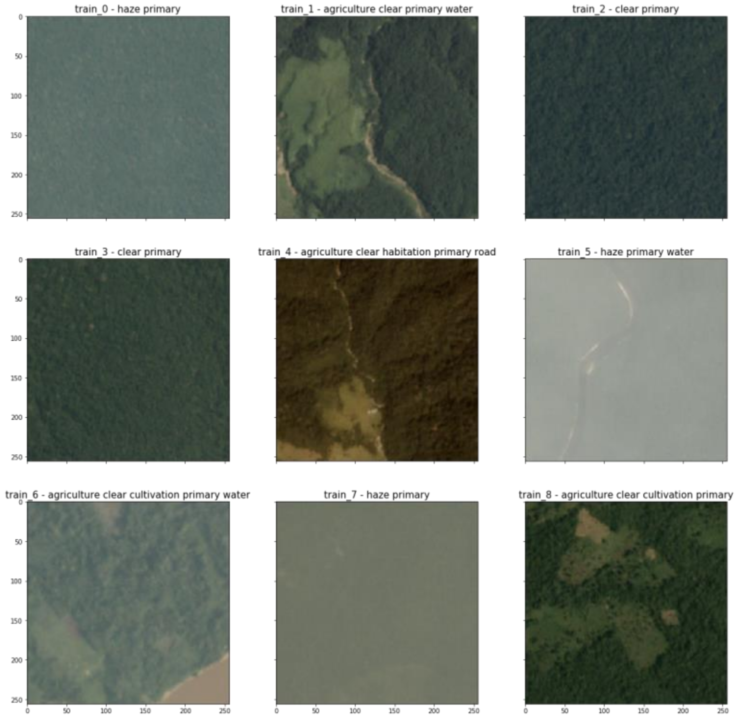
- Optimizing the prediction threshold can result in increased Fbeta scores but will result in a decrease in recall or precision. The costs and benefits should be weighed to determine if the optimized thresholds should be used.
- Using a specialized, one-vs-all model for predicting slash_burn tag, Slash_burn Fbeta was increased from 0 to 0.608 using a predictive threshold of 0.5, and from 0.182 to 0.614 when using an optimized predictive threshold.
- Image equalization did not increase the predictive performance of the specialized slash_burn model.

Further Work

- One vs all models for less common, more important tags.
- Additional data augmentation methods for expanding the datasets for less common tags.
- Test-Time augmentation.
- Ensemble model experimentation.
- Tuning learning rate.

Amazon Rainforest Image Classification





CONNOR MCANUFF - NOVEMBER 2019