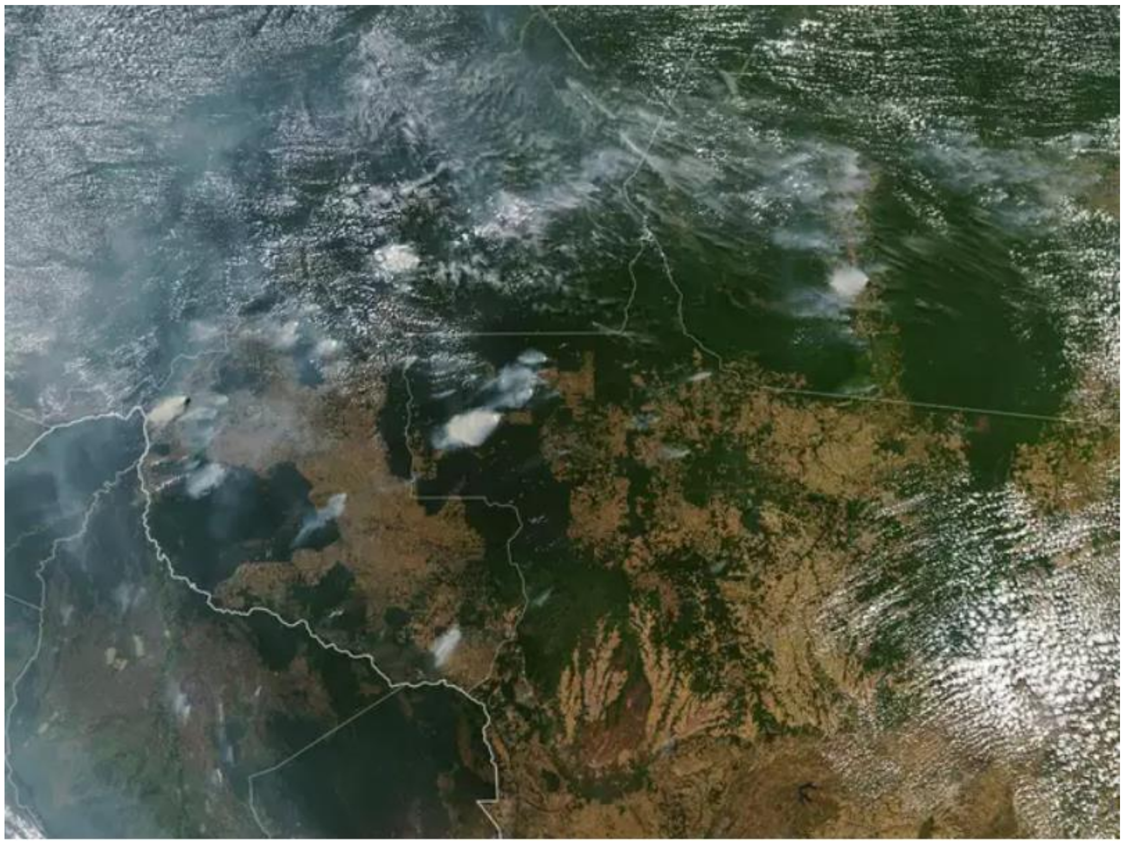
Springboard Capstone Project 2 – Final Report
Amazon Rainforest Image Classification



Connor McAnuff

November 28, 2019

1. Overview

1.1 Problem Statement

1.1.1 Amazon Rainforest

The Amazon rainforest covers an area of 6,000,000 km^2 across multiple countries and is located in the largest river basin in the world. It contains several million species of insect, animal, plant, and tree life. Many areas of the Amazon also contain human civilization.

Despite efforts beginning in the 1990s by the Brazilian government and international bodies to protect the rainforest, human encroachment, exploitation, deforestation and other forms of destruction continue to harm the health of the Amazon Rainforest, causing reduced biodiversity, habitat loss, climate change, desertification, and soil erosion, among other issues [1][2].

1.1.2 Satellite Imagery

Due to the immense area of the Amazon Rainforest, monitoring deforestation is difficult, if not impossible to do so from the ground. Previously, research on tracking changes in forests has been performed using satellite imagery with a resolution of 30 m/pixel (Landsat) or even 250 m/pixel (MODIS). These resolutions are too coarse to identify small-scale deforestation/degradation or identify whether the cause is human or natural. The company Planet plans to collect daily satellite imagery of the Earth’s surface at a resolution of 3-5 m/pixel, which would allow for manual (and perhaps machine-learning automated) classification of Amazon Rainforest satellite images, including the location and cause of small-scale deforestation/degradation [3].

1.2 Value to client

Automated classification of Amazon Rainforest satellite images would provide great value to governments and other organizations. The scale of the area and frequency of the images means that it would be extremely cumbersome and resource intensive for images to be manually classified.

Automated classification would ideally allow for the ‘human footprint’ in the Amazon to be accurately and frequently mapped to identify deforestation/degradation. The human footprint map can be used by governments or organizations to take action to prevent further harm. The data would also be valuable for other reasons – for example, it could be used for infrastructure, aid, census, and resource planning.

2. Data Wrangling

2.1 Dataset Overview

The dataset to be used in for this project has been made available for a former Kaggle competition that occurred in 2017 [4]. The images were collected using Planet satellites and are given as chips (tiles). Each chip has been manually assigned a label (or multilabel) through crowd sourcing of labour. The client has stated that some mislabels are present, however this issue is common in satellite imagery classification and the proportion of mislabels is thought to be very small.

The labels stem from three categories: atmospheric conditions, common land cover/use, and rare land cover/use. The labels are unbalanced - for example there are many more images tagged with primary forest than there are images tagged with selective logging. Images commonly have more than one label.

2.2 Dataset Format

Dataset available: https://www.kaggle.com/c/planet-understanding-the-amazon-from-space/data

- 40,479 satellite images (chips) of amazon rainforest:
    - 256 x 256 pixels (947.2m x 947.2m on ground) in GeoTiff (stripped of GeoTiff metadata) and JPEG formats
    - 21.2 GB total
    - Tiff images have four bands of data: red, green, blue, near infrared (16-bit)
    - JPEG images are in the standard format of RBG arrays
    - 17 classifications (unbalanced):
        - More common:
            - Cloudy
            - Partly cloudy
            - Hazy
            - Primary rainforest
            - Water (rivers and lakes)
            - Habitation
            - Agriculture
            - Road
            - Cultivation
            - Bare ground
        - Less common:
            - Slash and burn
            - Selective logging
            - Blooming
            - Conventional mining
            - "Artisanal" mining
            - Blow down
- List of image file names and their associated labels (train_v2.csv)
    - CSV format

2.3 Data Importing

The train_v2.csv file can be read directly into a Pandas DataFrame to create a table of image names (train_0, train_1, train_2, …) and each image's associated, manually labelled tags (i.e. labels). The tags column can have each observation split into a list of tags (strings) and then a list of all non-unique tags can be constructed to determine the frequency of each tag.

The images can be brought in for viewing using opencv methods imread and imshow.

2.4 Cleaning and Organization

There does not appear to be any missing or incorrect information in the train_v2.csv table of image names and associated tags. As all 40,000+ images cannot realistically be viewed and confirmed to be error free manually, irregularities will be searched for during exploratory data analysis.

2.5 Formatting

Dataset formatting is necessary to put the image information into the format required for neural network machine learning. This process includes the following steps:

1) Create a dictionary of tags (keys) and integers (values) that will be used to create the one hot encoding sequence.
2) Load each image one by one using keras.preprocessing.image.load_img.
3) Convert each image to a numpy array using keras.preprocessing.image.img_to_array.
4) Use the image tags and dictionary mappings to create the one hot encoder sequence for each image. The target variable for each observation is a list of 17 integers, 0 if the image does not have the tag, and 1 if the image does.
5) The numpy image array and one hot encoding sequence are appended to lists of images and targets, respectively.
6) The image and one hot encoding lists are converted to arrays X and y.
7) The dataset is saved into a compressed format (.npz) and can be loaded at any time.

3. Exploratory Data Analysis

3.1 Data Storytelling

3.1.1 Tag Occurrences

Total tag occurrences for all 40,479 images are plotted in Figure 1. The tag occurrences are unbalanced, with 37,513 and 28,431 of the images having tags primary and clear, respectively. The more important tags (those that will be used to identify deforestation/degradation) are much less common:

- Selective logging (340 images)
- Artisanal mine (339 images)
- Slash and burn (209 images)
- Blow down (101 images)

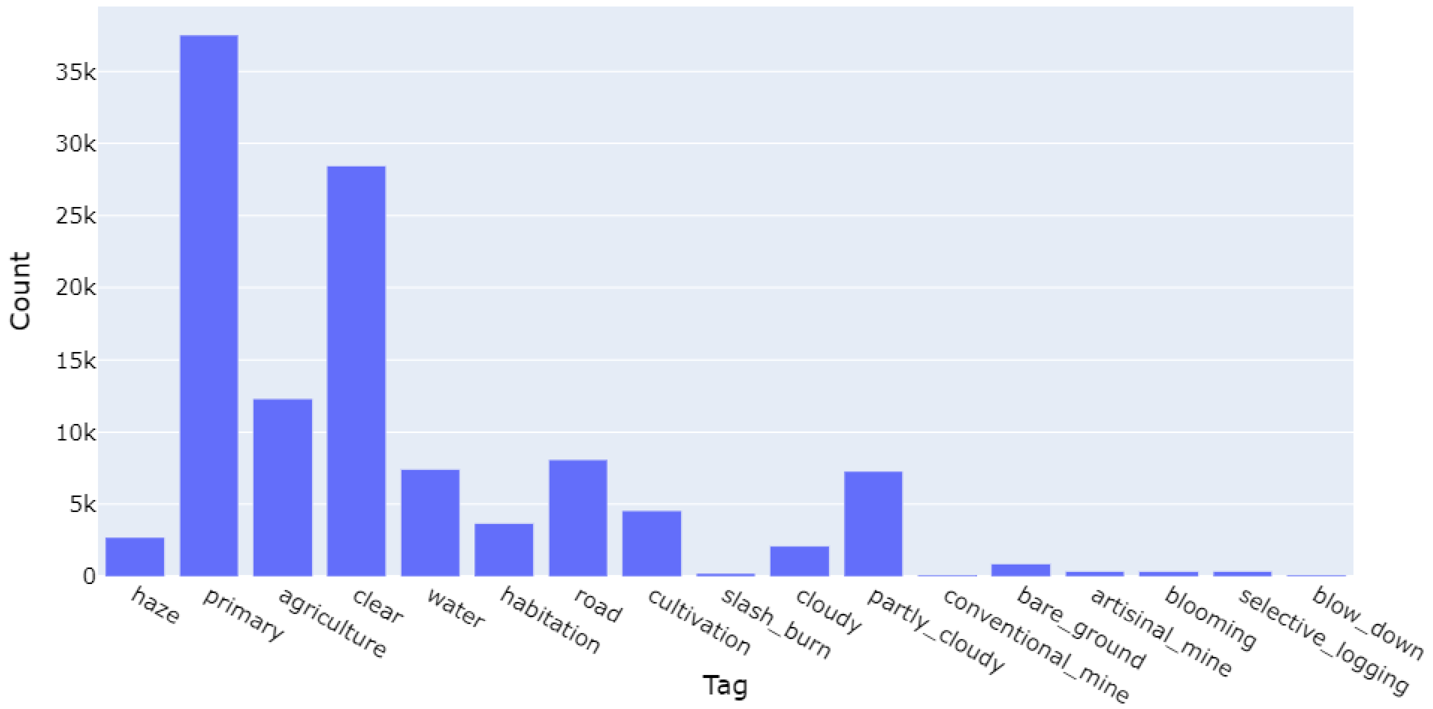


Figure 1: Total occurrences of tags for all images.

The images have an average of 2.87 tags per image. To understand which tags occur with other tags most frequently, a co-occurrence matrix is shown in Figure 2. For each tag listed on the y-axis, the co-occurrence matrix gives the proportion of images that also have each tag listed on the x-axis. For example, for all images tagged with blow_down:

- 0% of images tagged with blow_down also have tag haze.
- 100% of images tagged with blow_down also have tag primary.
- 22% of images tagged with blow_down also have tag agriculture.
- 87% of images tagged with blow_down also have tag clear.

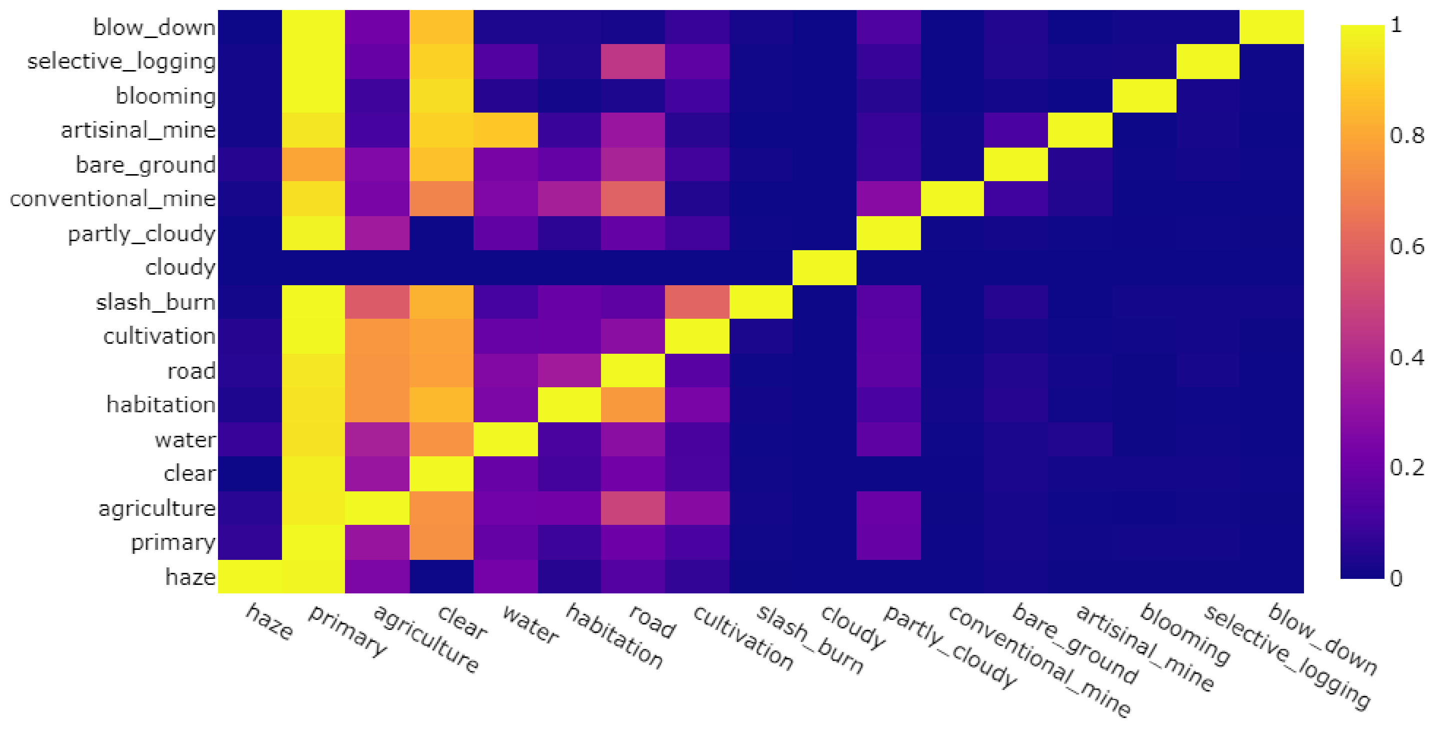


Figure 2: Co-occurrence matrix of tags. Values given are the proportion of images each tag on the y-axis that also have each tag on the x-axis.

Unsurprisingly, most tags have a high co-occurrence with primary and clear. Cloudy is the exception to this trend – images tagged with cloudy are always tagged with cloudy only. Cultivation, road, and habitation commonly occur with agriculture. Selective logging images are also tagged with road 44% of the time. Slash_burn often occurs with agriculture or cultivation.

3.1.2 Images

A sample image (train_10) is given in Figure 3. The image is 256 x 256 pixels with 3 channels for red, blue, and green intensity values. Image train_10 is tagged with primary, clear, agriculture, slash_burn, and water. Primary, water, and slash_burn are labelled in the sample image, the locations being obvious. It is not clear where the agriculture is located, however it is assumed to be the new vegetation growth nearby the slash and burn site. The tag clear applies to the entire image and is referring to the clearness of the image due to lack of clouds or haze.

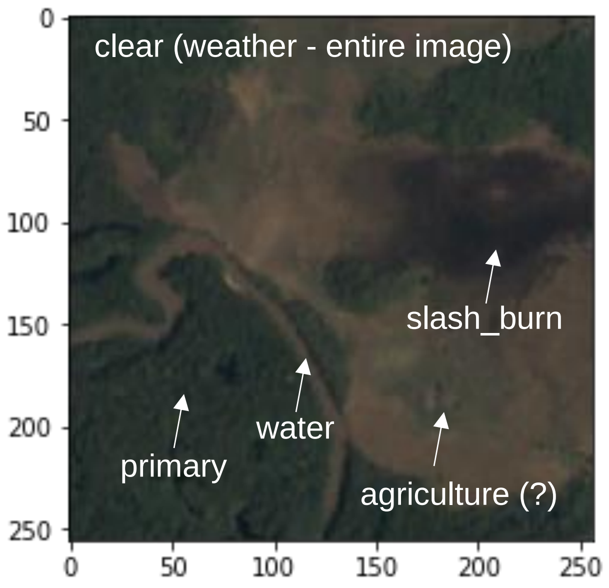


Figure 3: Sample JPEG image – train_10.

The TIFF images have 4 channels (red, blue, green, near-infrared) as opposed to the 3 channels that the JPEG images have. The additional channel, near-infrared, results in the images being 'washed-out' to the naked eye, as shown in Figure 4.

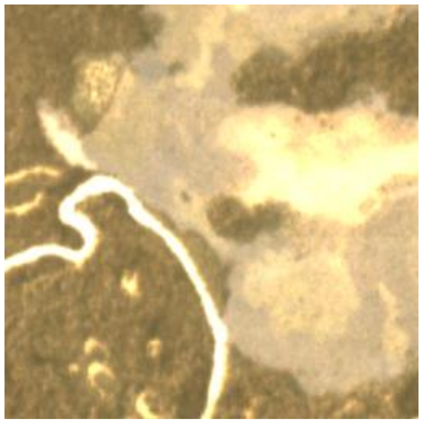


Figure 4: TIFF raw image – train_10.

The TIFF images are better viewed when each of the 4 channel intensities are shown (Figure 5). Near-infrared intensity is shown to be very low for water and dark earth, and very high for thick vegetation. For the purposes of simplicity, JPEG images will be focused on for this project.

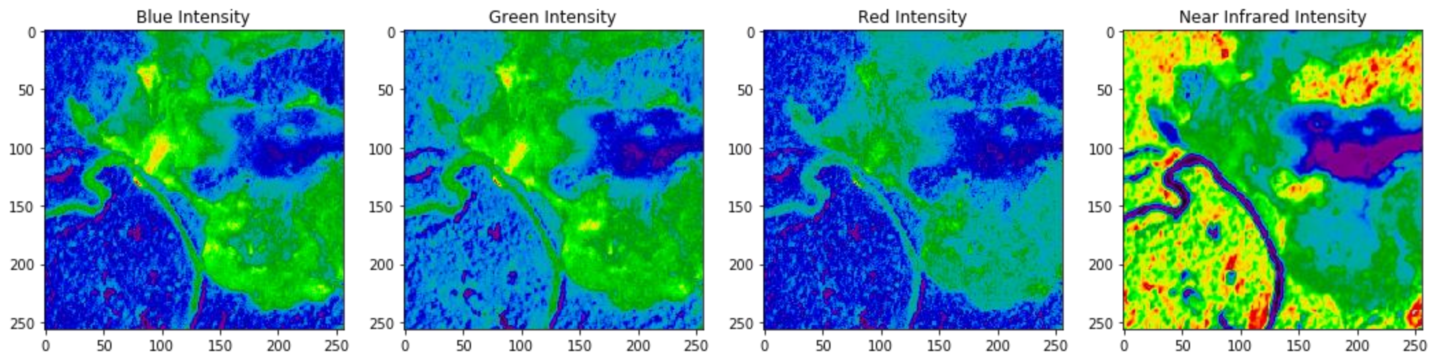


Figure 5: Sample TIFF image (4 channels) – train_10.

A further 9 image samples (in JPEG format) are shown in Figure 6. Images tagged with haze appear to have a transparent white/tan layer over the entire image. Images are tagged with water even if only a small portion of the image contains water. Images are tagged with agriculture when there are green/beige clearings in the primary forest.

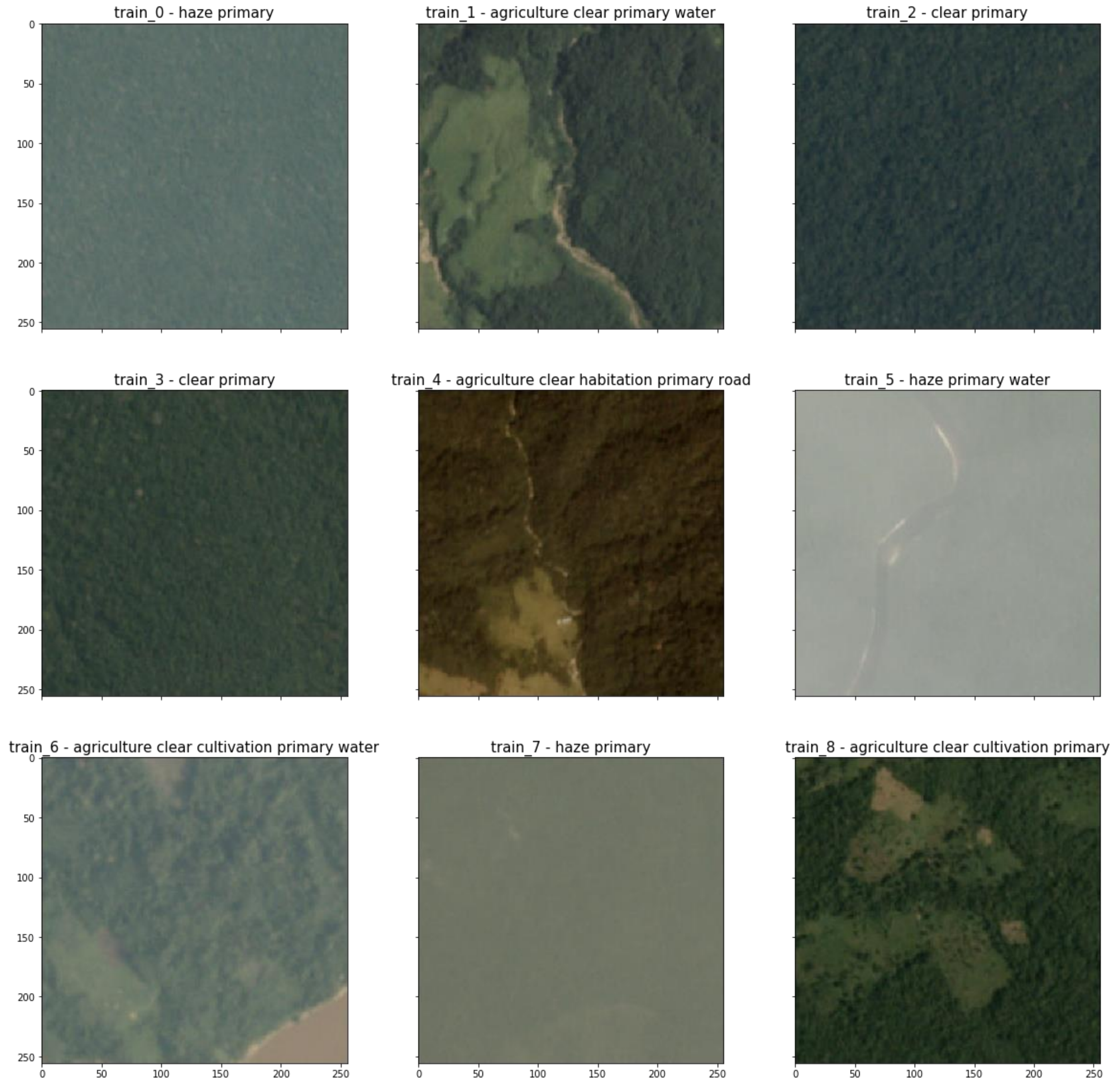

train_0 - haze primary
train_1 - agriculture clear primary water
train_2 - clear primary
train_3 - clear primary
train_4 - agriculture clear habitation primary road
train_5 - haze primary water
train_6 - agriculture clear cultivation primary water
train_7 - haze primary
train_8 - agriculture clear cultivation primary
0
50
100
150
200
250

Figure 6: 9 image samples in JPEG format and their associated tags.

3.1.3 Image Histogram Equalization

Image histogram equalization is a technique used to increase contrast in images by detecting the distribution of pixel intensities (ranging 0-255) and 'stretching' them to better cover the pixel intensity spectrum. The technique is best understood using an example – Figure 7 shows pixel intensity histograms and images for the raw and equalized image train_8. The equalized histogram has the same shape vertically as the raw histogram, however the shape has been stretched along the x-axis to force the pixel intensities to use the entire 0-255 intensity range. The equalized image is much brighter and higher in contrast overall.

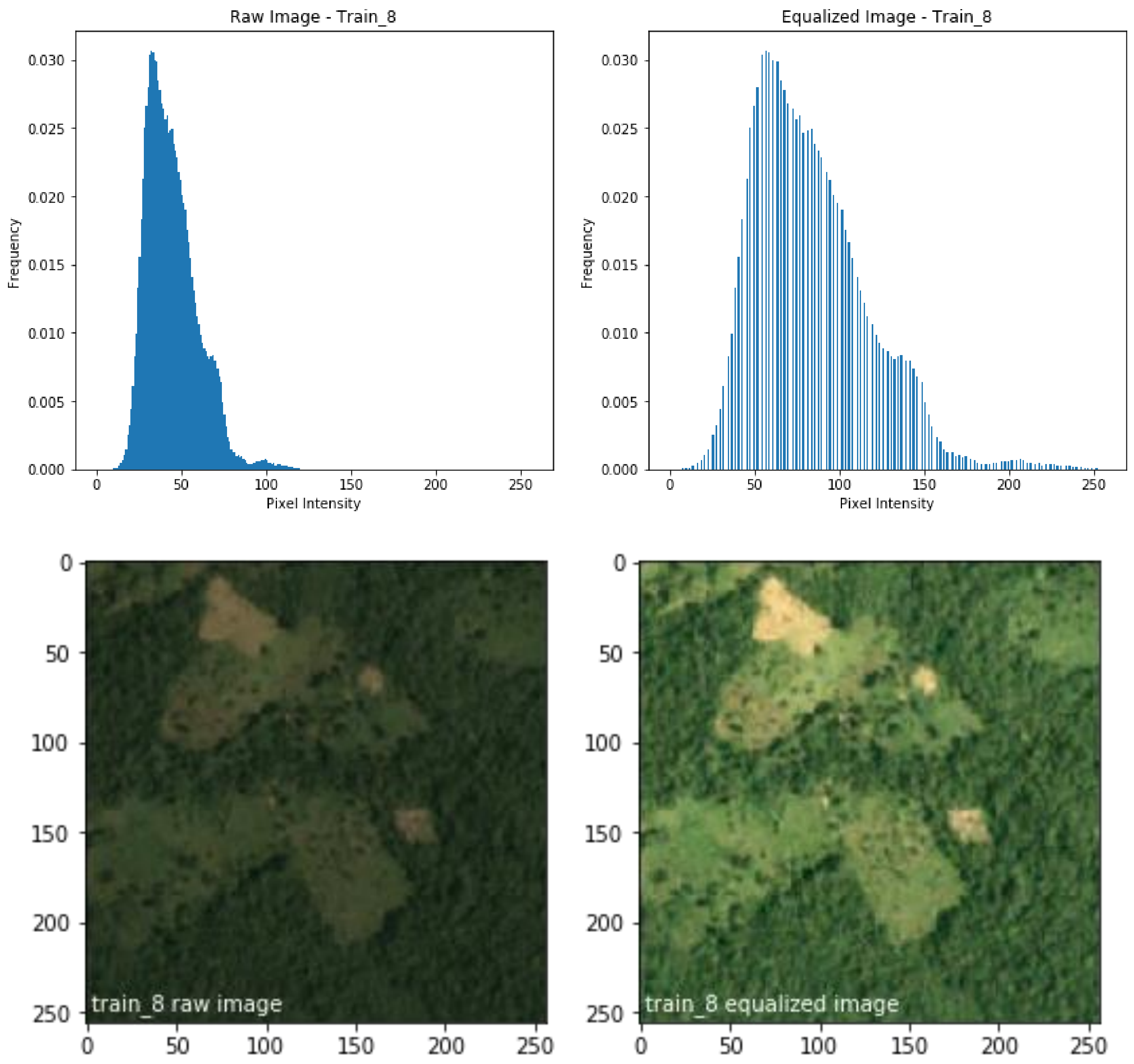


Figure 7: Raw and equalized train_8 pixel intensity histograms and images.

Image equalization can be used as a method of image augmentation when training image classification models. The modified images may allow for the machine learning model to better identify features and therefore provide more accurate classification results.

3.1.4 Single Image Means and Dominant Colours

Single images can be simplified by calculating their mean colour. The mean colour is calculated using the following process:

1) Take the mean value of the numpy array row-wise producing the mean R, G, and B values for each column.
2) Again take the mean value of the numpy array row-wise producing the mean R, G, and B values for all pixels.
3) Round the float means and convert to integers.
4) Multiply the R, G, B mean array by an array of ones in the shape of the original image. Each pixel of the final mean image has the mean pixel values.

A mean image sample is given in Figure 8. The mean image appears to be a mix of brown and green, as expected. It is not very exciting to look at and doesn't say much about the image itself, as the mean image colour does not necessarily appear frequently in the raw image. It would be more informative to explore the dominant colours of an image.

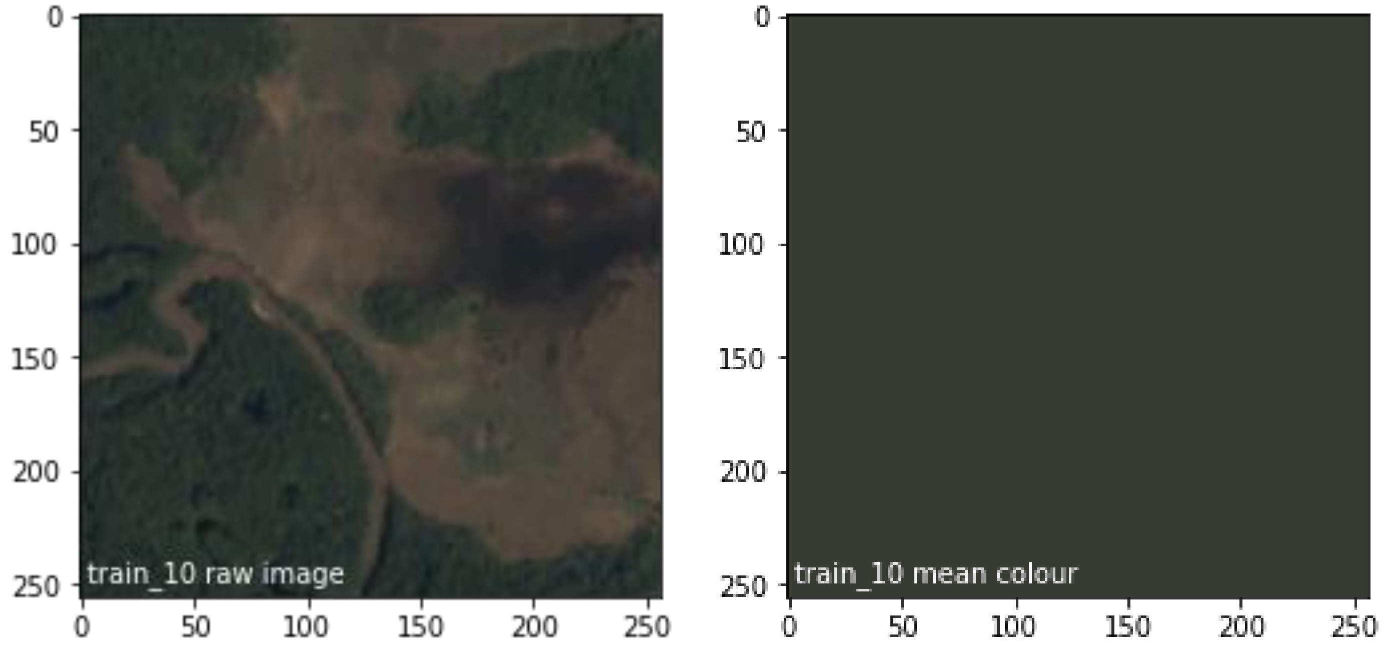


Figure 8: train_10 raw and mean colour.

Obtaining the dominant colours in an image can be performed by k-means clustering. The process is as follows:

1) Reshape the image array from (256, 256, 3) to (65536, 3).
2) Set the iteration criteria.
3) Initiate flags to be random centers.
4) Perform k-means clustering with the selected number of clusters/colours to obtain labels and centers (RGB colours).
5) Calculate the number of pixels in each label cluster.

6) Use numpy.argsort to get the sorting indexes of the number of pixels array and reverse the list.
7) Create a cumulative frequency array holding the proportion of pixels with each label.
8) Calculate the location of the separation between dominant colour rows according to the image shape.
9) Create the dominant colour image by filling the rows with the dominant colours according to their proportions.

The dominant colours of image train_10 according to k-means clustering are shown in Figure 9. Depending on whether 3, 4, or 5 clusters are implemented, the resulting dominant colour image shows various shades of green and brown, as can be seen in the raw image.

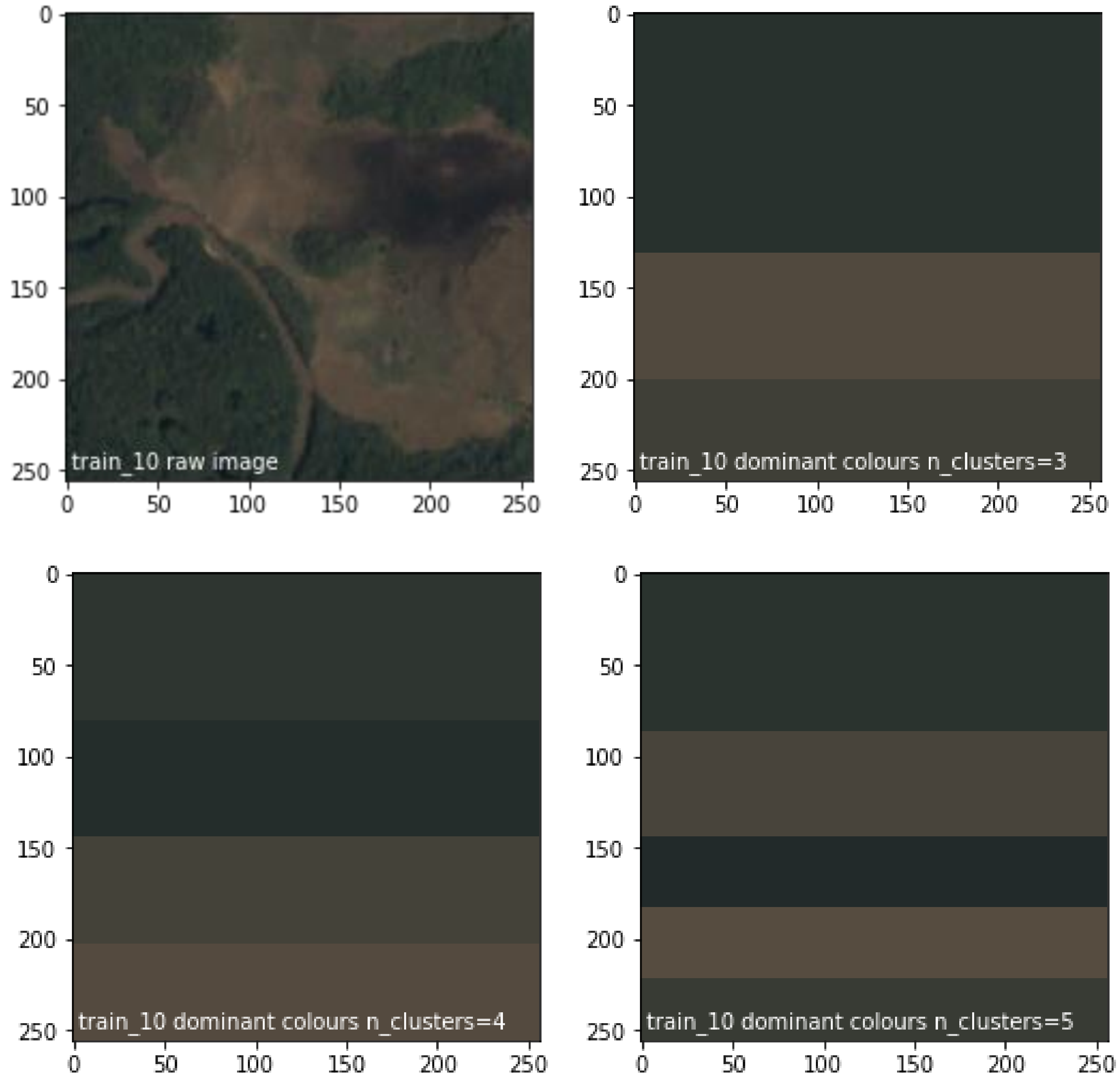

train_10 raw image
train_10 dominant colours n_clusters=3
train_10 dominant colours n_clusters=4
train_10 dominant colours n_clusters=5
0
50
100
150
200
250

Figure 9: Dominant colours of image train_10 found using k-means clustering: Raw image, dominant colours using n_clusters=3, n_clusters=4, and n_clusters=5.

Extracting the dominant colours removes information from the image in that the features that would be searched for by a deep learning algorithm are removed. The dominant colour images are a function of the colours in the image and their relative frequency. Images tagged with road will have varying amounts of ‘road pixels’ in the image. This process is unlikely to be useful for machine learning models but is an interesting exercise none-the-less.

Lastly, a tool to replace each pixel by the dominant color cluster that it belongs to has been created. Figure 10 shows images where the raw image pixels have been replaced by their associated dominant colour cluster using a varying number of clusters. As the number of clusters increases, so does the detail in the image. At 50 clusters (50 dominant colours used to re-create the image), the image is essentially indistinguishable from the raw image. This method is a method of colour quantization, wherein the number of distinct colours used in an image is reduced to compress the image file size.

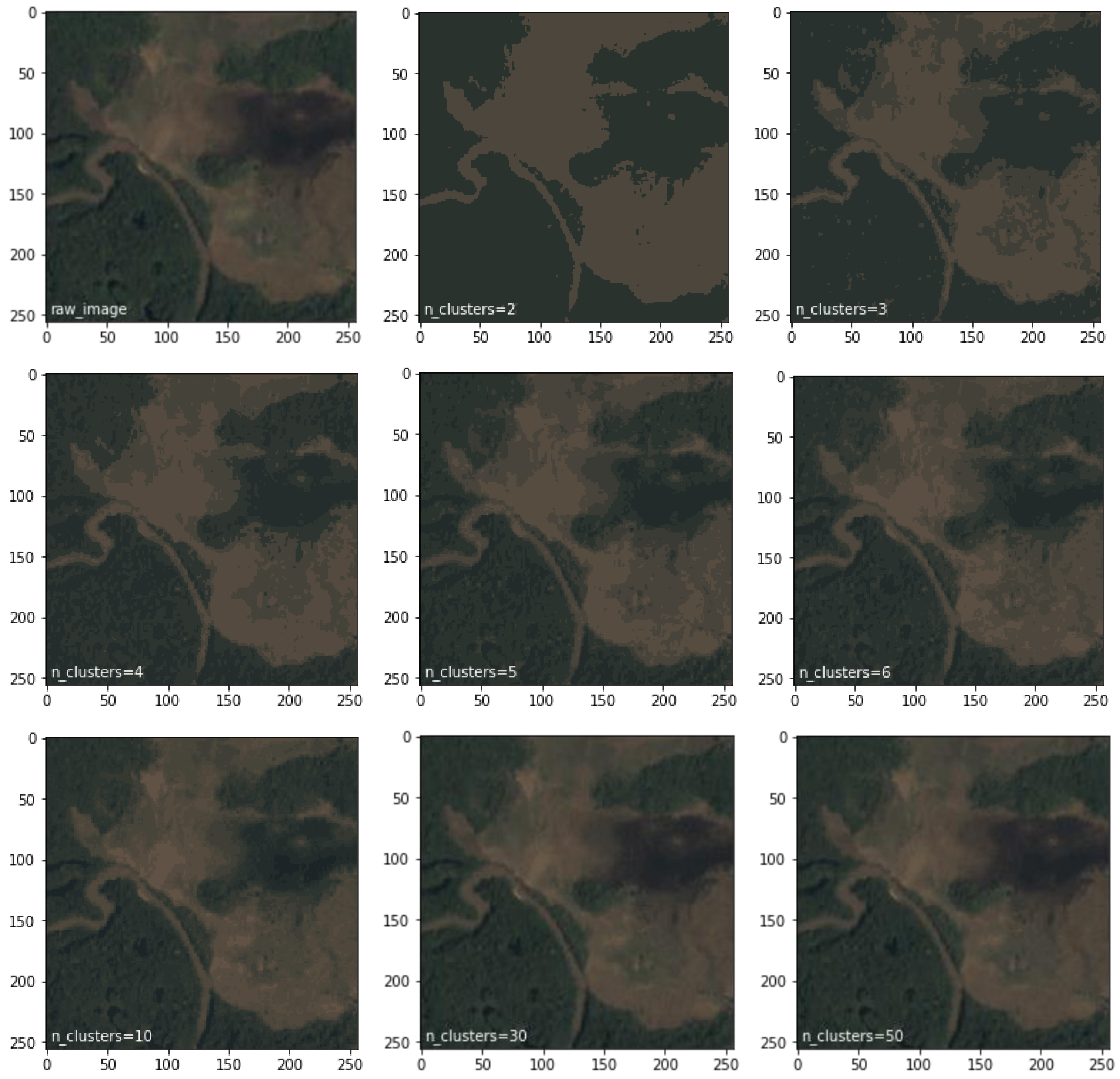


Figure 10: Image pixels represented by their associated dominant colour cluster.

3.1.5 Image Set Means

Sets of images can be combined into a single mean image by the following process:

1) Initialize the mean image as a numpy array of all zeros with the same shape as the raw images.
2) For each image in the set, add the image to the mean image.
3) Divide the mean image by the number of images in the set.
4) Round the float values and convert to integers.

The mean image for all images combined is shown in Figure 11. It is a shade of green which makes sense as most of the images are tagged with primary and thus have trees in the image.

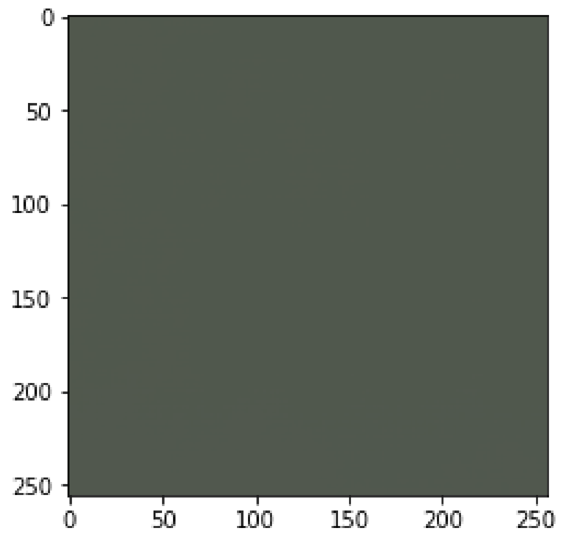


Figure 11: Mean image for all images combined.

A set of image lists can be constructed for which each list contains the list of images tagged with a given tag. The mean image for all images with that tag can then be calculated.

The mean images for each tag are given in Figure 12. The mean images by tag can give an idea of the colour scheme and prevalence of tagged information in the image sets. Of the more rare tags, selective_logging, blooming, slash_burn, and blow_down appear to be the most similar to each other and the primary mean image. Conventional_mine and artisinal_mine are very different from primary, and different than each other. Overall, there is at the least slight variability between tag mean images, indicating images with different tags do indeed have varying colours.

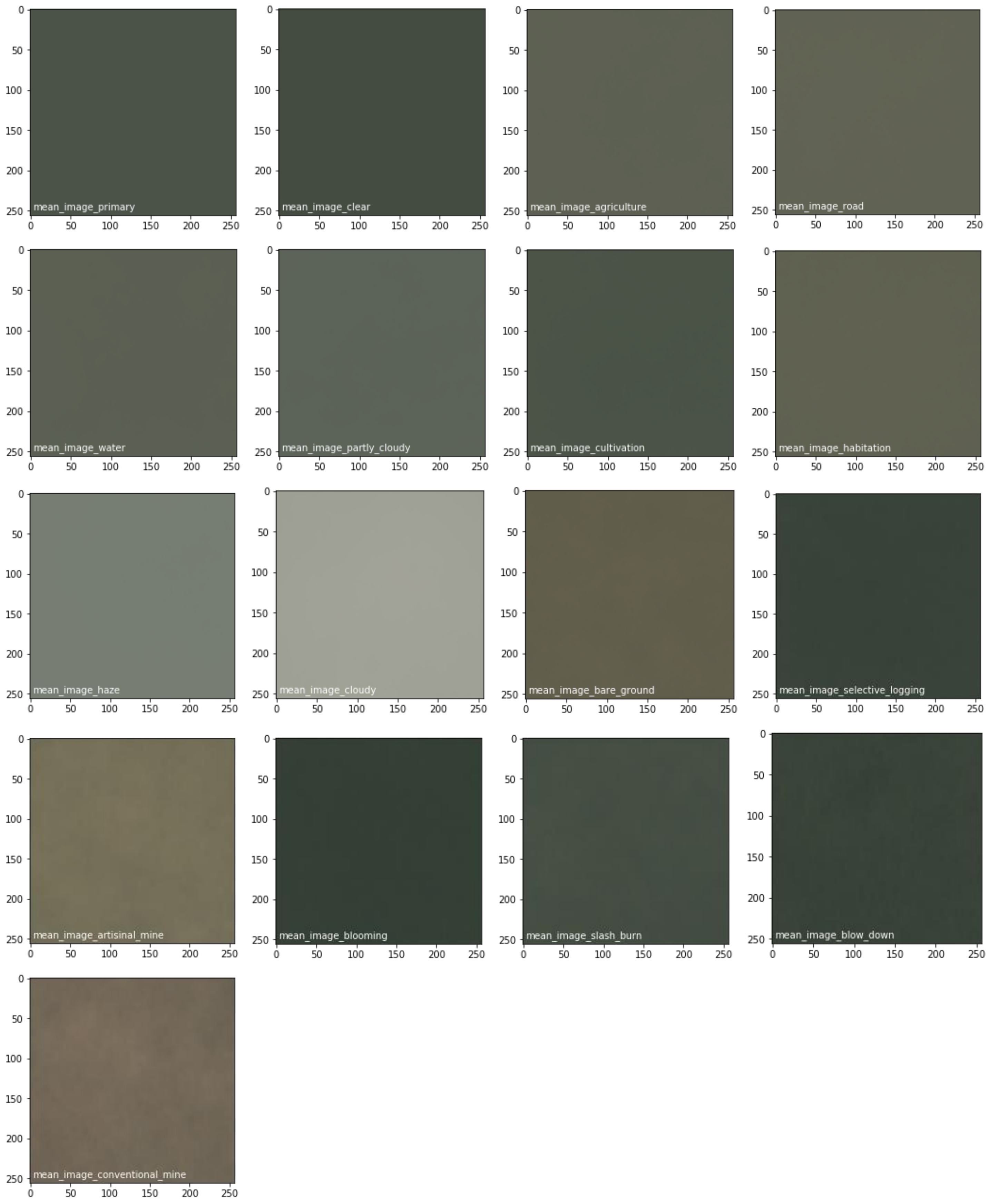


Figure 12: Mean images for sets of images with each tag.

3.1.6 Image Clustering

The images can be clustered to determine similarity using the TSNE method. For this clustering task, images tagged with only one of the rare image tags are included:

- Bare_ground
- Selective_logging
- Artisinal_mine
- Blooming
- Slash_burn
- Blow_down
- conventional_mine

The clustered images are plotted on using the two components of the TSNE clustering as the x- and y-axes (Figure 13). This plot shows that there are somewhat well-defined clusters, and several outliers. The size of the images obscures the location of many of the points – the images would be better plotted as points.

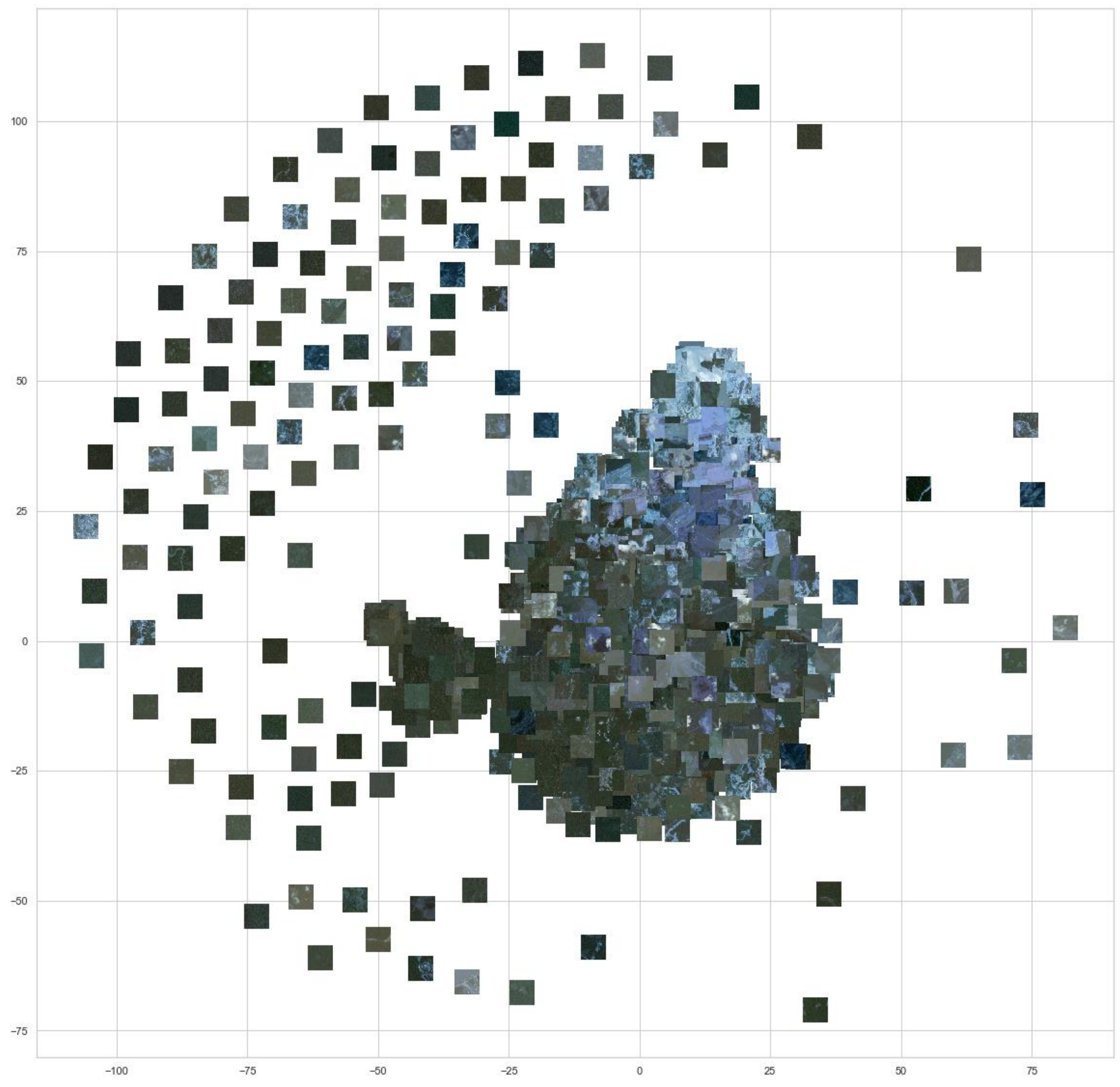


Figure 13: TSNE clustering of rare tag images.

In addition to plotting the images as points, the points can be clustered according to their rare tag to compare the results of the clustering to the image tags (Figure 14). Figure 15 shows a magnified section of the original plot.

There are multiple clusters that correspond well to rare image tags. The following observations can be made:

- Images with tag bare_ground are clustered in 3 locations and are spread around outside of all clusters.
- Images with tag artisanal_mine are partially clustered with bare_ground images. Artisanal mining would clear the land vegetation, leaving the ground bare.
- Images with tag blow_down are loosely clustered (at (-40, -5)) and are also spread around.
- Images with tag conventional_mine are not well clustered.
- Images with tag blooming are clustered in 2 locations and are also spread around.
- Images with tag slash_burn are spread around.
- Images with tag selective_logging are well but loosely clustered in the largest cluster.

Overall, there are many images that do not cluster well with others with this method. Clustering will not be able to be used to predict the labels of images accurately enough - a neural network will need to be used to identify features in the images on a smaller scale.

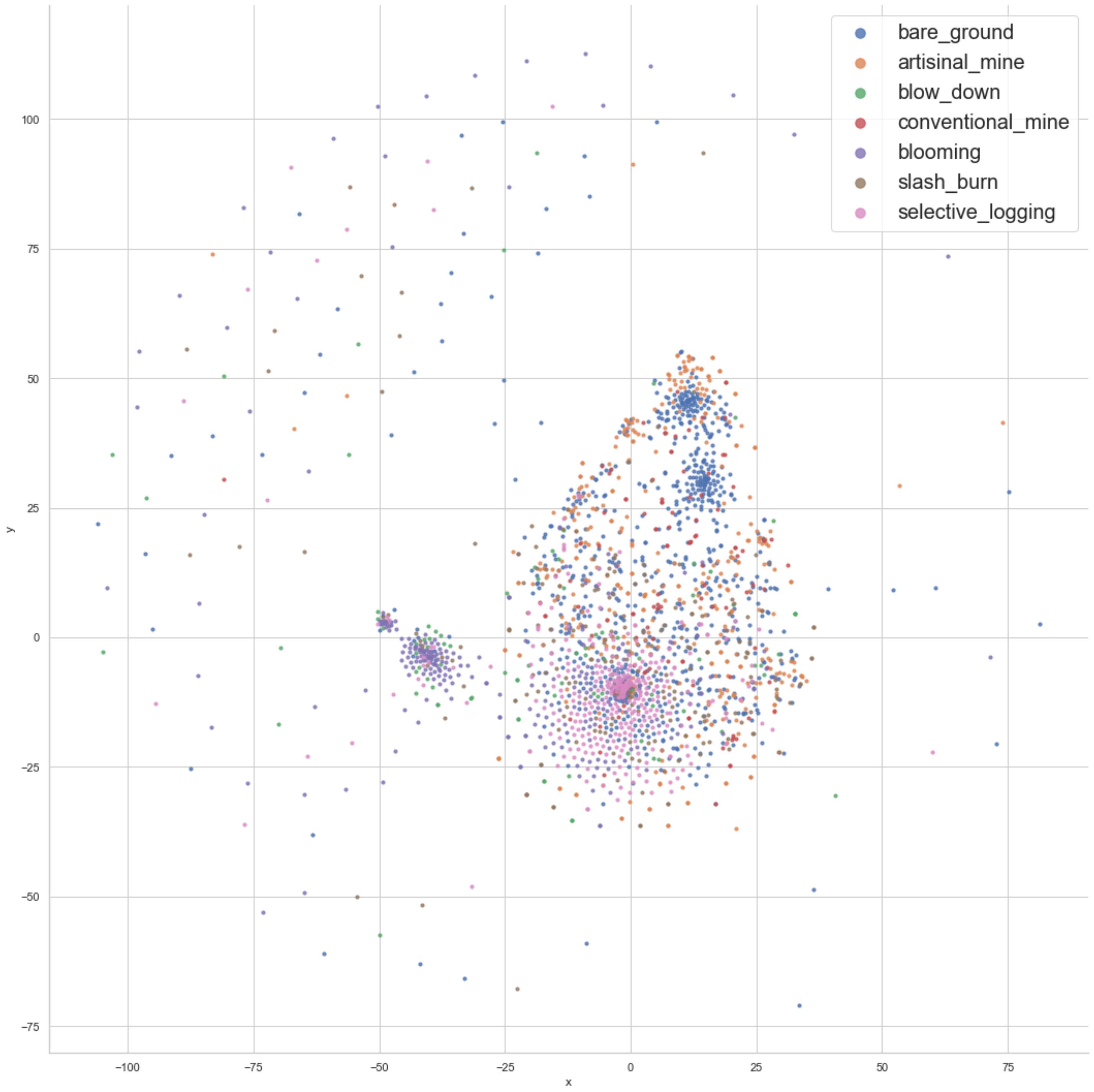


Figure 14: TSNE clusters with points coloured by rare tag.

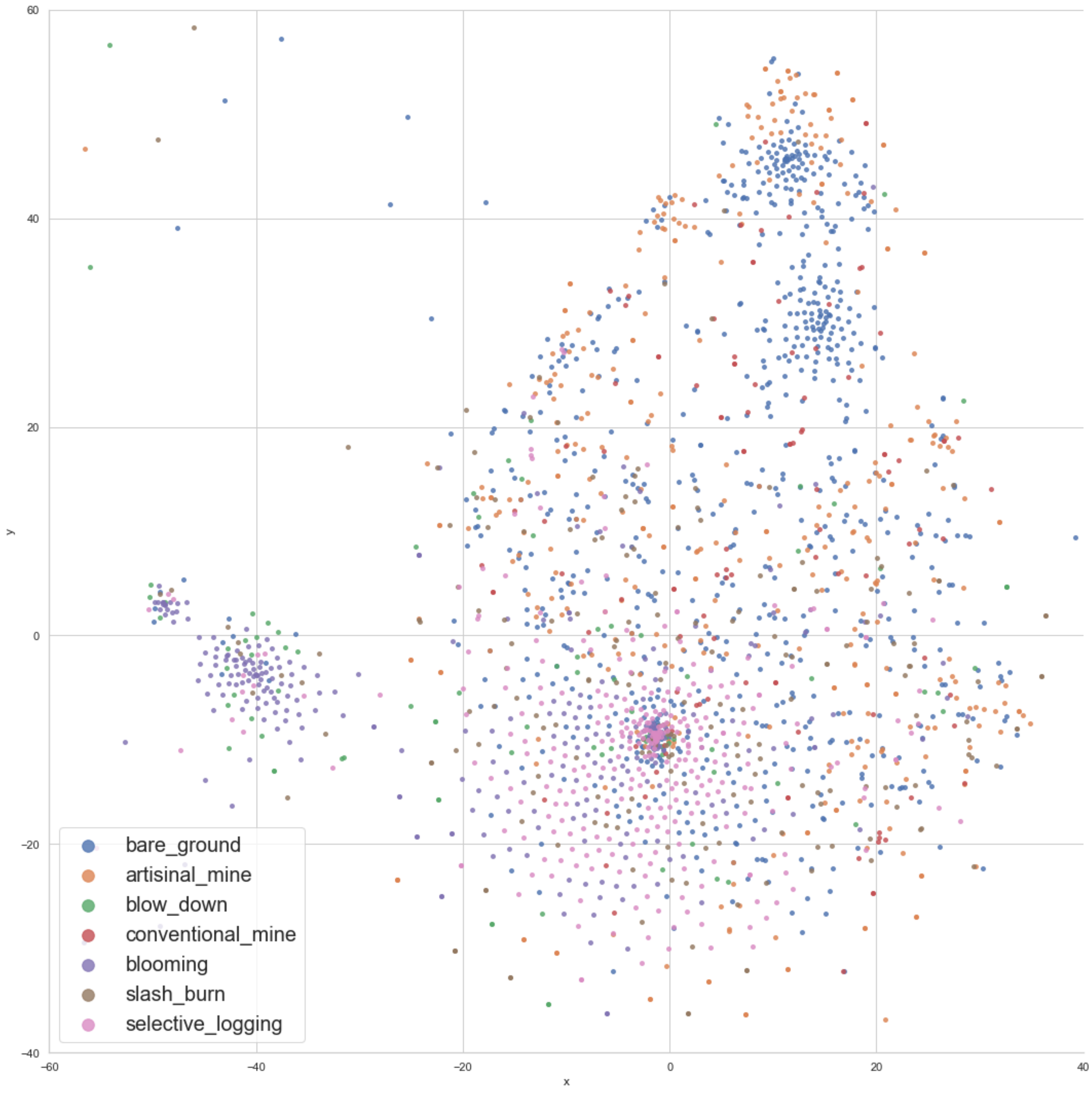


Figure 15: TSNE clusters with points coloured by rare tag (magnified).

4. Machine Learning

4.1 Overview

The purpose of the machine learning task is to classify images as having one or more of the 17 tags listed as follows:

- Clear
- Cloudy
- Partly cloudy
- Haze
- Primary rainforest
- Water
- Habitation
- Agriculture
- Road
- Cultivation
- Bare ground
- Slash and burn
- Blooming
- Selective logging
- Conventional mining
- Artisanal mining
- Blow down

State-of-the-art image classification is typically done using deep learning. Convolutional Neural Networks (CNNs) are widely used to classify images. Several CNN models have been developed, trained, evaluated, and compared in their ability to predict tags of satellite images of the Amazon Rainforest. Varying techniques and combinations of techniques have been used between models. The models are implemented using keras with tensorflow backend. An overview of the models is given in Table 1.

Table 1:CNN models overview.

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		

4.2 Model Evaluation Framework

The following two aspects of the data and prediction task must be addressed to determine an appropriate evaluation metric:

1) Classification of images is multi-label.
2) Occurrences of each label are unbalanced.

The metric that will be used for evaluation is the "F2 score". It is related to the more common "F1 score" which is an average of precision and recall. The F1 score is not an arithmetic mean of precision and recall, rather it as the harmonic mean. Equal weight is attributed to precision and recall. The equations for precision, recall, and F1 are given in Equations 1-3.

$$precision = \frac{true\ positives}{(true\ positives + false\ positives)} \quad (1)$$

$$recall = \frac{true\ positives}{(true\ positives + false\ negatives)} \quad (2)$$

$$F1 = 2 * \frac{precision * recall}{(precision + recall)} \quad (3)$$

The F-Beta metric introduces a new term (beta) – this term is a weight that determines how important recall is compared to precision. For Beta < 1, more weight is given to precision, while for Beta > 1, more weight is given to recall. The generalized F-Beta equation is shown in Equation 4.

$$FBeta = (1 + Beta^2) * \frac{precision * recall}{(Beta^2 * precision + recall)} \quad (4)$$

For the purposes of this project, Beta=2, as that is a common value for Beta and the value used in the Kaggle competition for this dataset. It is appropriate to consider recall more important than precision for this task, as it is desired to identify deforestation/degradation – the consequences of a false negative are worse than those for a false positive. Thus, the evaluation metric for the CNN models is as follows in Equation 5:

$$F - Beta\ (F2) = 5 * \frac{precision * recall}{(4 * precision + recall)} \quad (5)$$

The standard definitions of precision and recall apply to binary classification (i.e. predicting positive or negative for each observation). This task, however, involves multi-label classification (i.e. predicting positive or negative for a set of classes for each observation), and therefore precision and recall are calculated as shown in Equations 6-7.

$$precision = \frac{1}{n}\sum_{i=1}^{n} \frac{|Y_i \cap h(x_i)|}{|h(x_i)|} \quad (6)$$

$$recall = \frac{1}{n}\sum_{i=1}^{n} \frac{|Y_i \cap h(x_i)|}{|Y_i|} \quad (7)$$

$where:$
$n = \#\ of\ observations$
$Y_i = true\ label\ assignments\ of\ the\ i^{th}\ observation$
$x_i = i^{th}\ observation$
$h(x_i) = predicted\ label\ assignments\ of\ the\ i^{th}\ observation$

It is important to note that the precision, recall, and F-beta are calculated for each observation and then averaged to obtain the final F-beta score. Therefore, the unbalanced tags may skew the Fbeta metric, as tags with relatively few observations will have a lower weight when calculating the average metric. It would be prudent to also determine the recall, precision, and Fbeta scores for each tag individually to assess a model's ability to predict the more rare, more important tags such as slash_burn, blow_down, blooming, artisanal mine, and selective_logging.

4.3 Dataset

Dataset formatting is necessary to put the image information into the format required for CNN learning. This process includes the following steps:

8) Create a dictionary of tags (keys) and integers (values) that will be used to create the one hot encoding sequence.
9) Load each image one by one using keras.preprocessing.image.load_img.
10) Convert each image to a numpy array using keras.preprocessing.image.img_to_array.
11) Use the image tags and dictionary mappings to create the one hot encoder sequence for each image. The target variable for each observation is a list of 17 integers, 0 if the image does not have the tag, and 1 if the image does.
12) The numpy image array and one hot encoding sequence are appended to lists of images and targets, respectively.
13) The image and one hot encoding lists are converted to arrays X and y.
14) The dataset is saved into a compressed format (.npz) and can be loaded at any time.

4.4 Convolutional Neural Networks

4.4.1 Model Theory and Architecture

Convolutional Neural Networks (CNNs) are a class of deep learning neural networks commonly used to classify images. CNNs work by extracting features from images, eliminating the need for manual feature extraction. Each additional layer of a model increases the complexity of the learned features. CNN model architecture will vary by application and dataset - the basic architecture used for this project is as follows:

1) Images input as shape (128,128,3)
2) 2x convolutional layers
   - 32 3x3 filters
   - ReLU activation
   - ‘He’ weight initialization
3) Max pooling layer
   - Size 2x2
4) 2x convolutional layers
   - 64 3x3 filters
   - ReLU activation
   - ‘He’ weight initialization
5) Max pooling layer
   - Size 2x2
6) 2x convolutional layers
   - 128 3x3 filters

    - ReLU activation
    - ‘He’ weight initialization
7) Max pooling layer
    - Size 2x2
8) Flattening layer for converting to 1D vector
9) Fully connected layer (shape=128)
    - ReLU activation
    - ‘He’ weight initialization
10) Output layer for prediction (shape=17)

The base model is optimized using mini-batch Stochastic Gradient Descent (SGD) with learning rate of 0.01 and momentum of 0.9. Loss is set to be measured as ‘binary_crossentropy’ and the models record the Fbeta metric while training.

Additional notes:

- The model is sequential, composed of a linear stack of layers.
- It is common practice to increase the number of filters as the output spatial volume is decreasing (through the pooling layers).
- ReLU activation is commonly used as an activation function for CNNs. It has the advantages of being cheap to compute, converging quickly, and sparsely activated. Sparsity results in concise models that have better predictive power and less overfitting.
- ‘He’ weight initialization is used to initialize weights based on the size of the previous layer which helps in attaining a global minimum of the cost function faster and more efficiently.
- Pooling is used to combine output clusters (from convolutional layers). It allows objects to be detected regardless of location within the image. Additionally, it helps reduce the number of parameters required and the amount of computation. Max pooling is commonly used for CNNs.
- Output layer has shape 17 as there are 17 tags/classes/categories.

The model is fit and evaluated by:

1) Creating a data generator using ImageDataGenerator. The images are rescaled by a factor of 1/255 to force pixel values to be between 0 and 1.
2) Using the data generator to create train and test iterators. A batch size of 128 images is used to decrease computation time.
3) Fit the model using fit_generator. A set number if epochs is input.
4) The fitted model is evaluated using evaluate_generator.

The models were fit and evaluated using Kaggle Notebooks and Google Colab. The trained models were output along with model history for later reloading and analysis.

4.4.2 Base Model

The base model was run for 50 epochs. The loss and Fbeta over epochs are shown in Figure 16. The model resulted in a test data loss of 0.534 and Fbeta of 0.822. The model quickly reaches its maximum Fbeta at approximately 10 epochs.

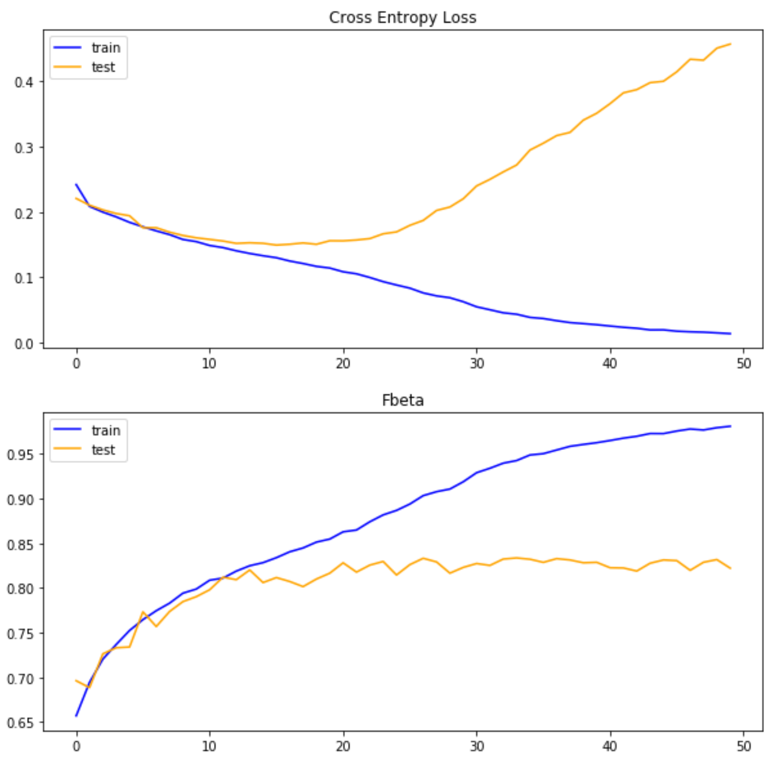


Figure 16: Base model loss and Fbeta over epochs.

4.4.3 Dropout Regularization

Dropout regularization is used to reduce overfitting and improve generalization error. Ensemble models are known to reduce overfitting, and dropout regularization simulates a single model having many network architectures by randomly dropping out nodes during training. For models with dropout regularization implemented, a dropout layer was added after each convolutional layer block with a fraction of 0.2 of the input units being dropped. A final dropout layer is added after the fully connected layer with a fraction of 0.5 of the input units being dropped.

The dropout model was run for 200 epochs. The loss and Fbeta over epochs are shown in Figure 17. The model resulted in a test data loss of 0.235 and Fbeta of 0.855. The model reaches its maximum Fbeta at approximately 50 epochs.

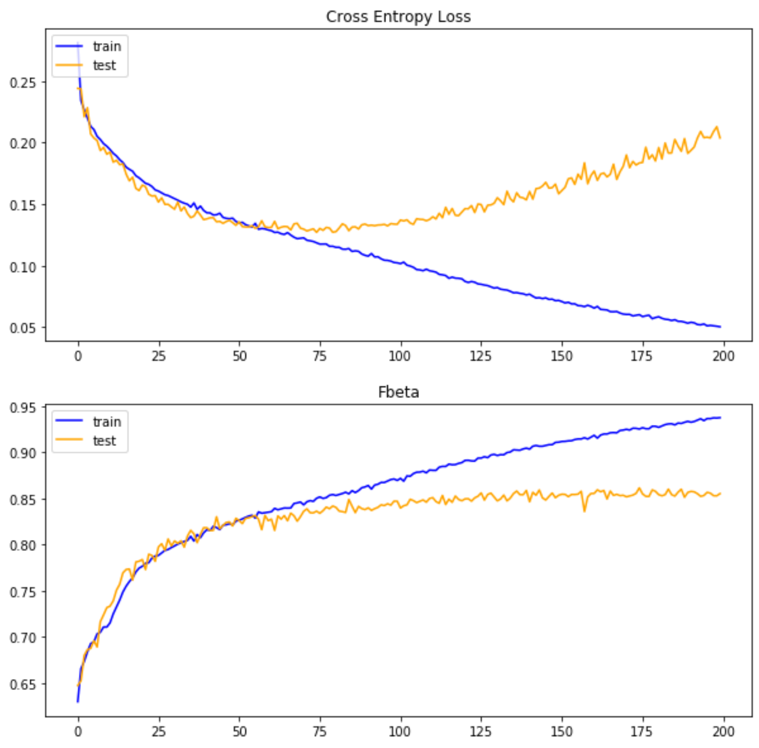


Figure 17: Dropout model loss and Fbeta over epochs.

4.4.4 Image Data Augmentation

Image data augmentation is used to artificially increase the size of the training dataset by creating modified versions of the raw images. It can help a model's ability to generalize and acts as a regularization method. For models with image data augmentation implemented, the train data generator is set to flip images horizontally and vertically, as well as rotate the images.

The augmentation model was run for 200 epochs. The loss and Fbeta over epochs are shown in Figure 18. The model resulted in a test data loss of 0.126 and Fbeta of 0.878. The model reaches its maximum Fbeta at approximately 125 epochs.

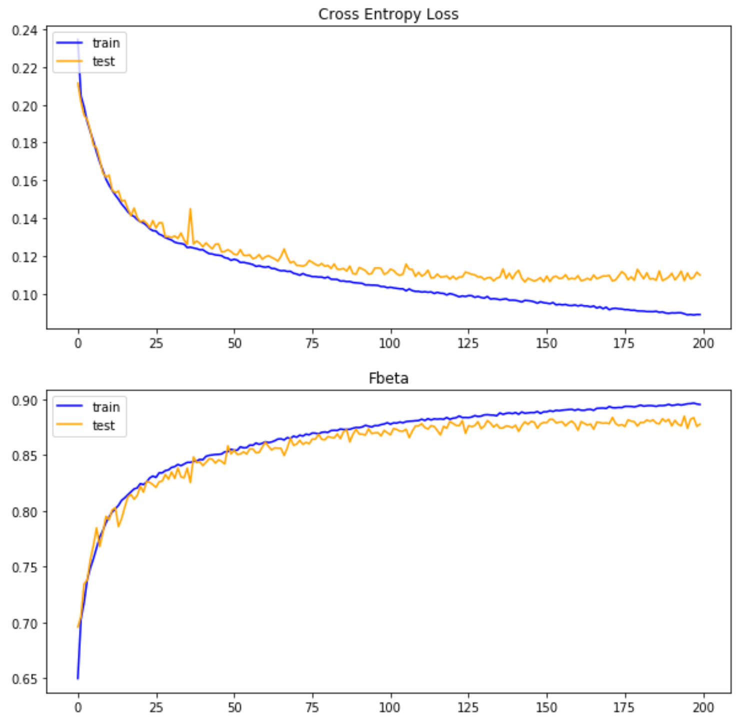


Figure 18: Augmentation model loss and Fbeta over epochs.

4.4.5 Adam Optimizer

The Adam optimizer implements an adaptive learning rate. Classically, standard Stochastic Gradient Descent optimization employs a constant learning rate for updating network weights. The Adam optimizer allows for the learning rate of each network weight to be adapted as learning unfolds. Benefits of using the Adam optimizer are:

- Computationally efficient.
- Little memory requirements.
- Well suited for problems that are large in terms of data and/or parameters.
- Appropriate for problems with very noisy and/or sparse parameters.

The Adam model was run for 50 epochs. The loss and Fbeta over epochs are shown in Figure 19. The model resulted in a test data loss of 0.711 and Fbeta of 0.845. The model reaches its maximum Fbeta at approximately 5 epochs.

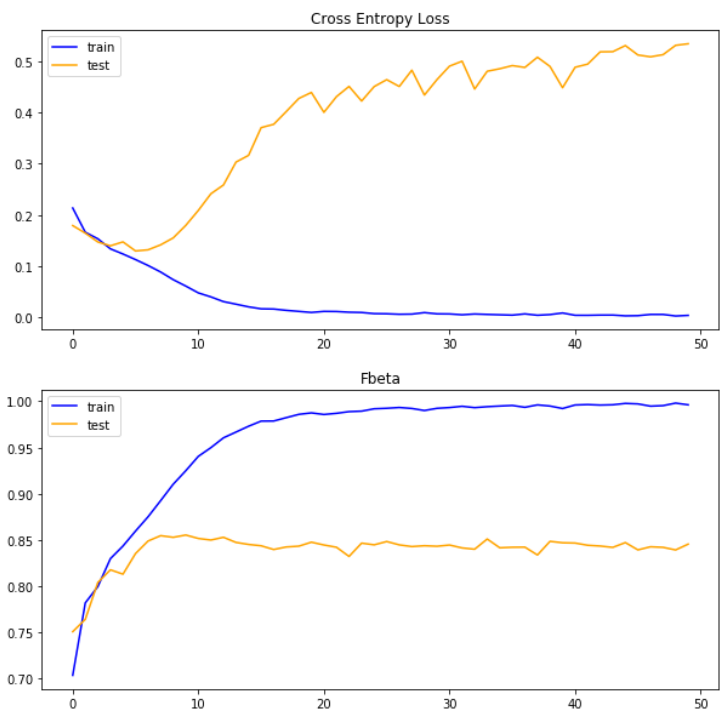


Figure 19: Adam model loss and Fbeta over epochs.

4.4.6 Pre-trained Model

Transfer learning involves using all or parts of a model pre-trained on a similar task. The VGG-16 model achieved top results in the ImageNet photo classification challenge at the time it was trained. The VGG-16 model was loaded and modified to be able to classify images from the Amazon Rainforest dataset. The original fully connected layer was replaced with a layer that interprets the model output and makes predictions for our purposes. The data generator must be modified to know the mean pixel values from the ImageNet training dataset as the model expects images to be centered.

The pre-trained model was run for 25 epochs. The loss and Fbeta over epochs are shown in Figure 20. The model resulted in a test data loss of 0.325 and Fbeta of 0.878. The model reaches its maximum Fbeta at approximately 2-3 epochs.

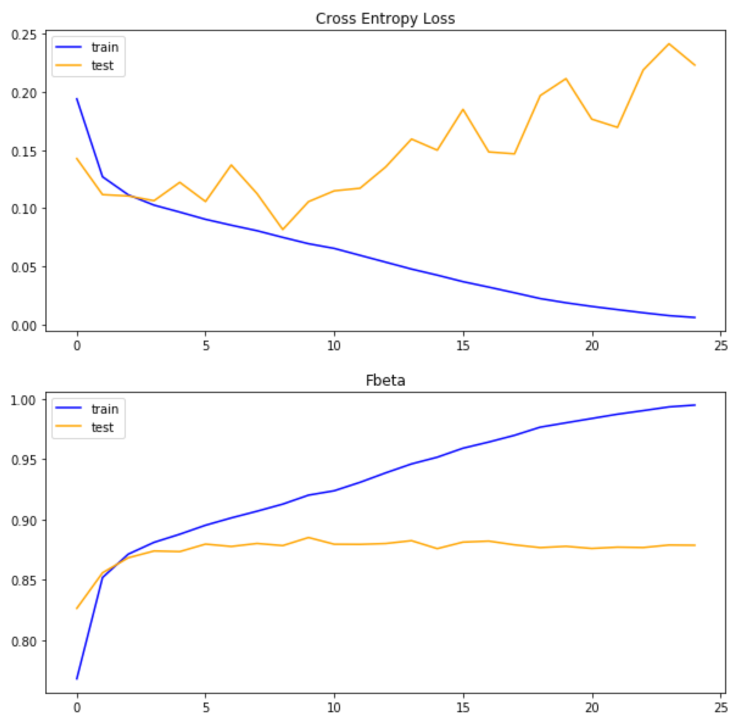


Figure 20: Pre-trained model loss and Fbeta over epochs.

4.4.7 Model Comparison

In addition to the models described previously, models with varying combinations of optimizer, regularization, and image augmentation were built and evaluated. The results of all models are given in Table 2. The base model is outperformed by all other models. Dropout regularization and image augmentation models performed better individually than when combined. The top performing models in terms of Fbeta score are (1) transfer model, (2) augmentation model, and (3) adam augmentation model.

Table 2: CNN model evaluation criteria.

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

4.5 Less Common Tag Prediction

The metric Fbeta is calculated as the mean Fbeta for all observations. While the metric gives an overall measure of model performance, it does not give information regarding how well each of the tags are predicted. Using the augment_model, precision (Figure 21), recall (Figure 22), and Fbeta (Figure 23) have been calculated for each tag.

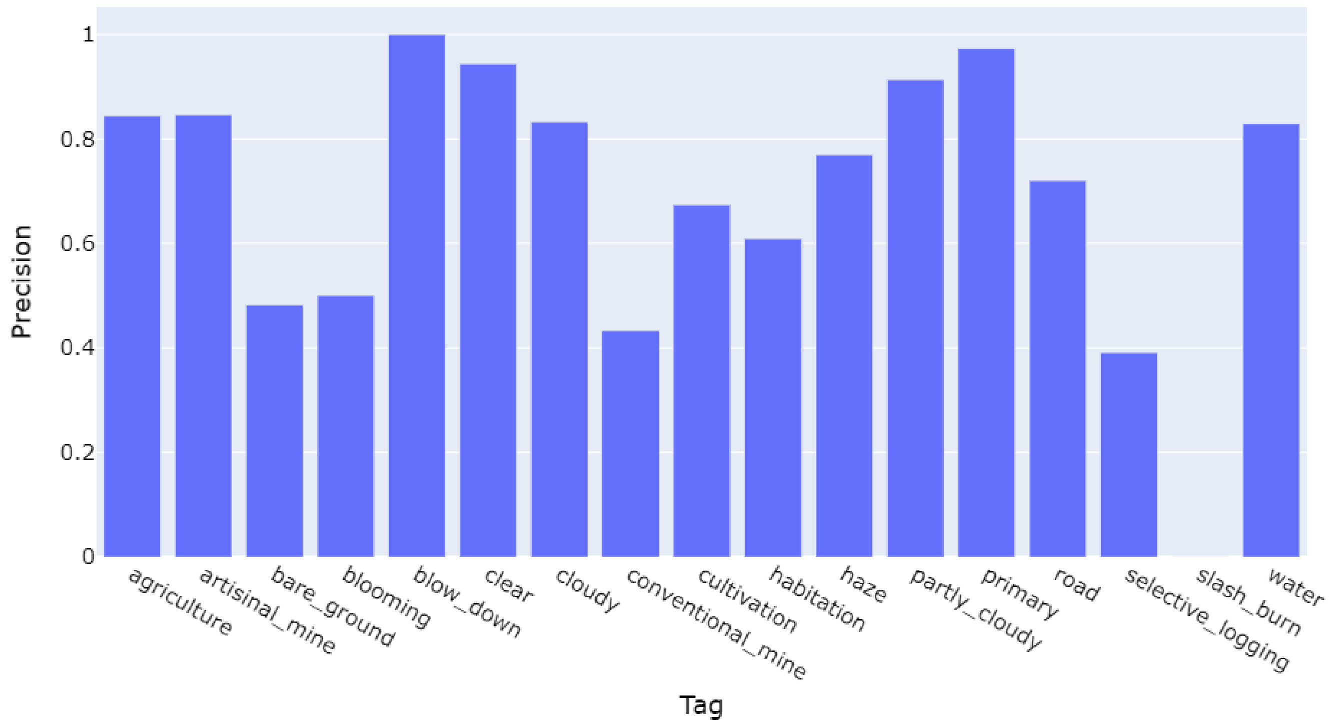


Figure 21: augment_model precision by tag.

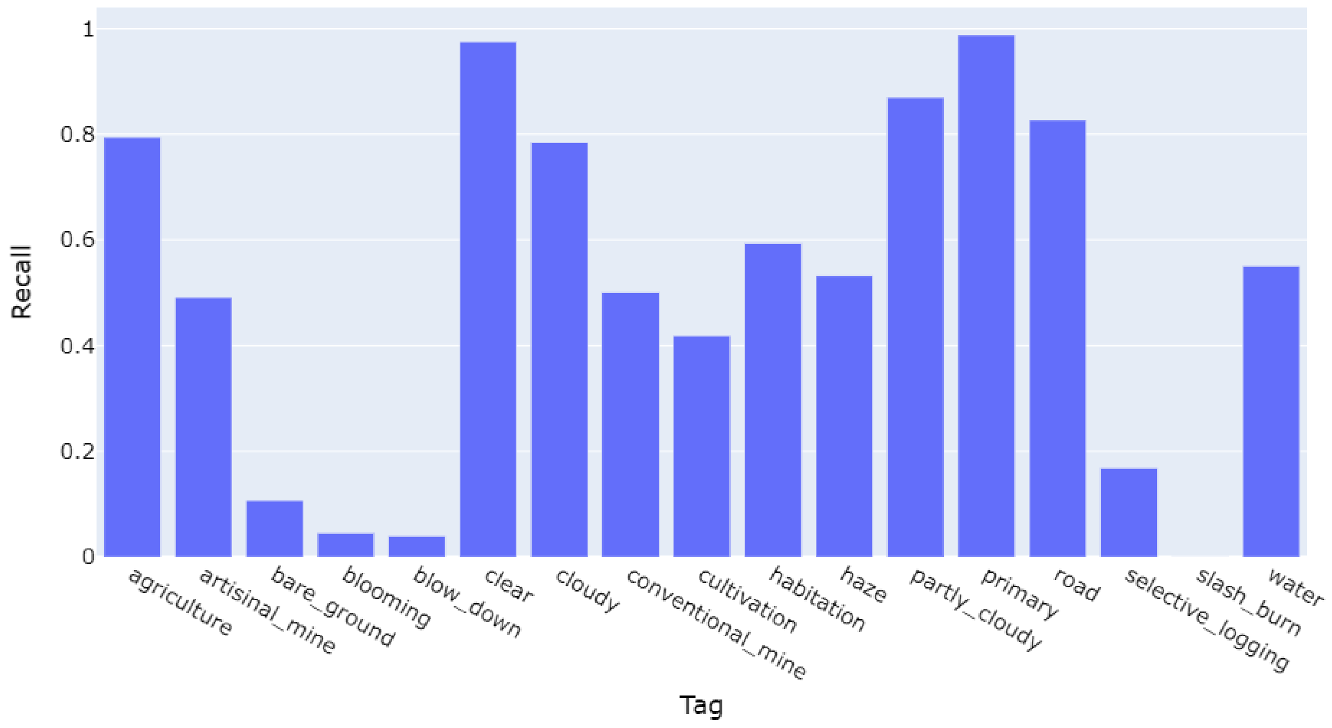


Figure 22: augment_model recall by tag.

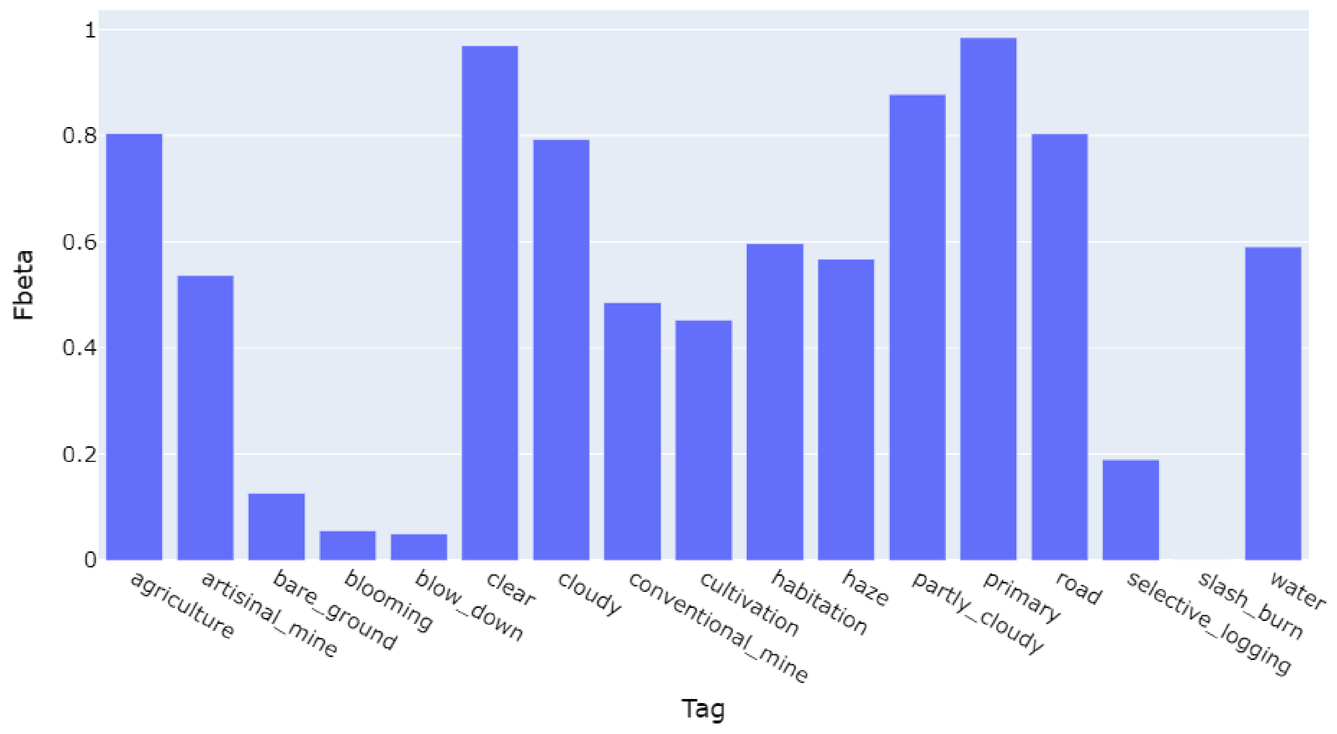


Figure 23: augment_model Fbeta by tag.

Fbeta is highest for tags primary (0.985), clear (0.967), partly cloudy (0.878), agriculture (0.804), and cloudy (0.793). Fbeta is lowest for tags slash_burn (0), blow_down (0.050), blooming (0.056), bare_ground (0.126), and selective logging (0.190). The low Fbeta scores are generally a result of low recall, meaning there are many false negatives. The tags with low Fbeta scores are the less common tags – each have fewer than 1000 observations. Slash_burn and blow_down have only 209 and 101 observations, respectively. The model has failed to accurately determine features for these tags and thus cannot effectively predict them in new images. The same trend is seen for all models.

The CNN models do not simply predict the presence or absence of a tag as 1 or 0, rather they predict a probability between 0 and 1. The threshold value for considering a prediction to be the presence of a tag can be adjusted to increase recall, precision, or Fbeta. To perform this optimization, the target metric can be calculated for a set of thresholds and the threshold with the maximum metric is taken to be the best cut-off value. The precision, recall, and Fbeta calculated using the best threshold for each of the tags are compared to those calculated when using a threshold of 0.5 in Figure 24, Figure 25, and Figure 26. As expected, the Fbeta increases for all tags increases when using the best threshold, however many tags have a drastic decrease in precision as a trade-off for an increased recall and Fbeta.

The optimal threshold to be used for prediction would depend on the desired outcome of the predictions – the benefits of increasing recall by allowing for a decrease in precision would have to be weighed against the costs. Still, the Fbeta scores for the less common occurring tags are drastically lower than those of the more common tags. A final model specific to slash_burn tagged images has been constructed to address this issue.

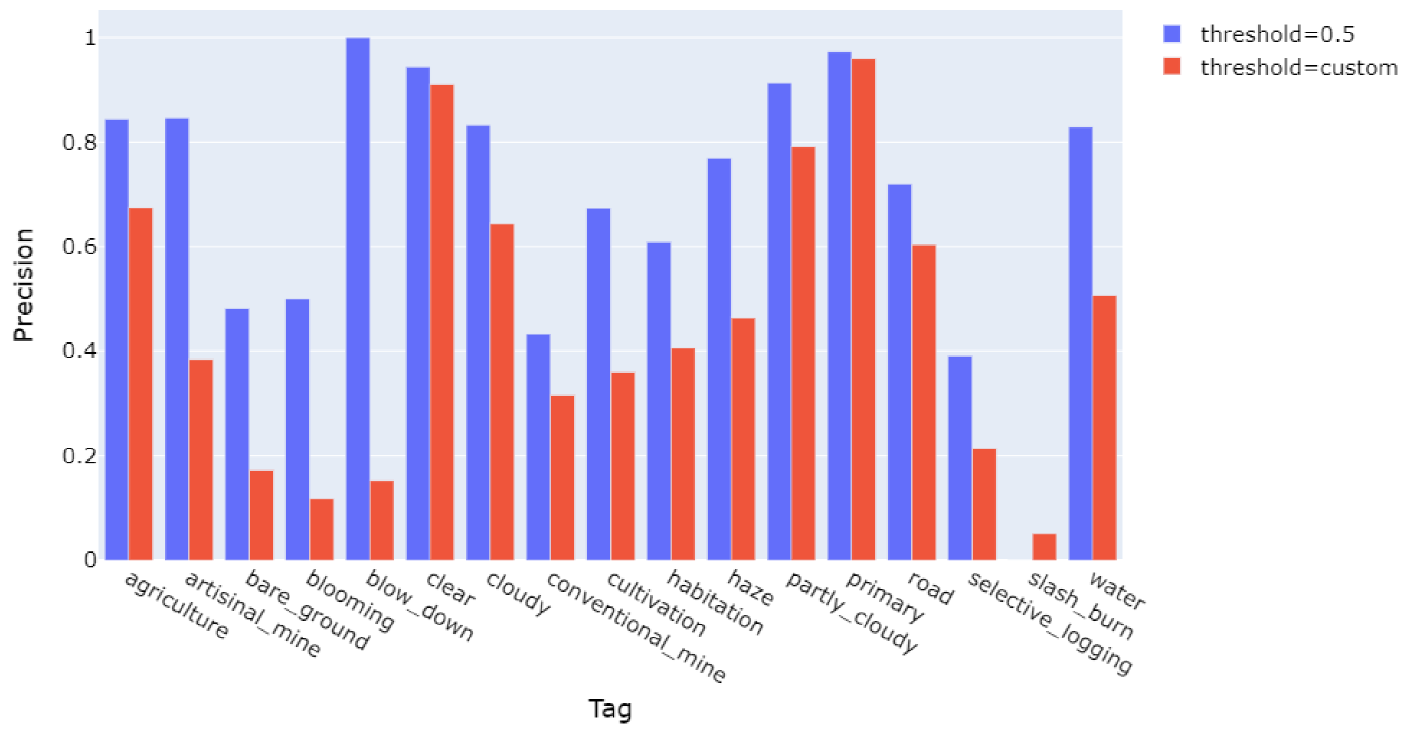


Figure 24: augment_model precision by tag using best thresholds and a threshold of 0.5.

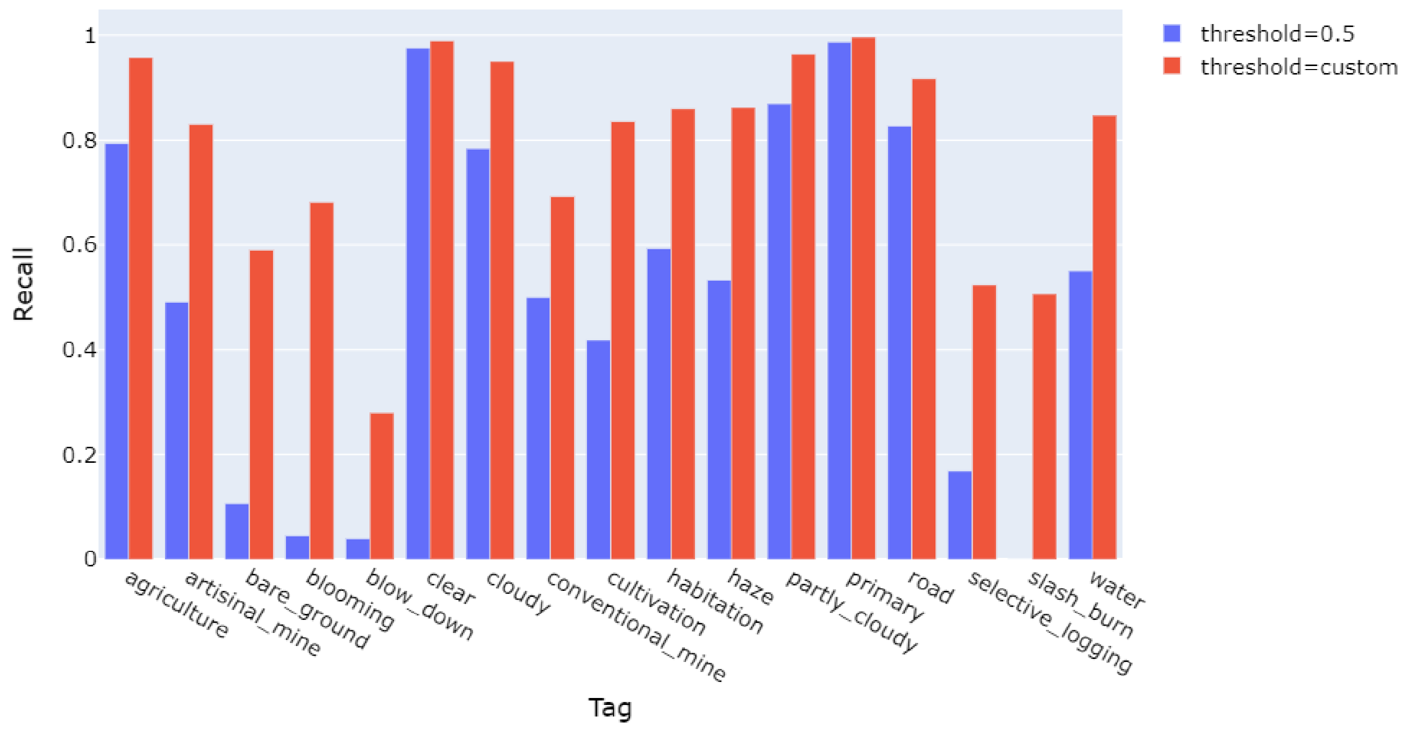


Figure 25: augment_model recall by tag using best thresholds and a threshold of 0.5.

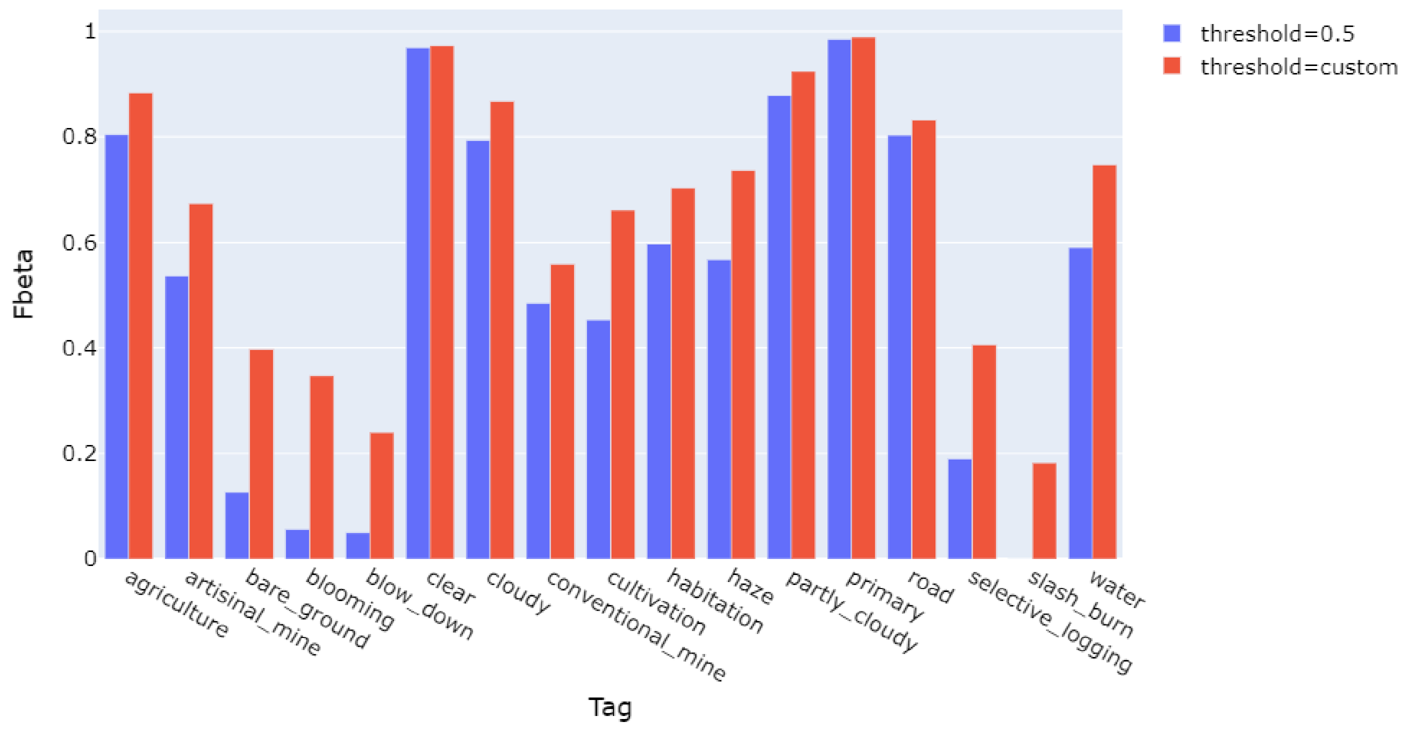


Figure 26: augment_model Fbeta by tag using best thresholds and a threshold of 0.5.

4.6 Slash and Burn Specialized Model

To address the issue of less common tags being predicted poorly, a specialized model to predict slash_burn tagged images has been implemented as a proof of concept for a set of models specific to each tag that will drastically improve predictive performance. The augment_model with a threshold of 0.5 has an Fbeta score of 0 for predicting slash_burn images. When the threshold is optimized, the Fbeta score becomes 0.182 with a precision of 0.05 and recall of 0.5. These results are poor and the model does not have value in predicting slash_burn images.

There are 209 images with tag slash_burn. The dataset for this model has been artificially expanded using data augmentation:

- Train data:
  - 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 data:
    - 63 raw images tagged with slash_burn
    - 437 random raw images from original dataset not tagged with slash_burn.

The output shape of the model is set to 1, as we only need the model to predict whether each observation should be tagged with slash_burn or not.

Training and testing the model results in a loss of 0.630 and Fbeta of 0.608. This Fbeta result is a drastic improvement for predicting slash_burn tags. The result can further be improved by optimizing the threshold. Setting the threshold to 0.477 as opposed to 0.5, the Fbeta score increases to 0.614 with a recall of 0.810 and precision of 0.313. The precision is still low, however recall has been determined to be of more importance for detecting forest destruction.

A final experiment with image augmentation has been performed for which the images are equalized prior to training and testing. The pixel histograms are 'stretched' horizontally to use the entire pixel intensity spectrum. This method increases contrast within each image and may help the model better identify features within the image. This experiment resulted in a slight decrease in Fbeta, indicating that histogram equalization does not benefit the dataset in terms of predictive performance.

4.7 Conclusions

- 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.
- 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.
- A specialized, one-vs-all model for predicting slash_burn tags was trained using a dataset specific to slash_burn images. The size of the dataset was increased using image augmentation. 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. The same type of specialized model could be built for each of the less common tags to increase Fbeta.
- Image equalization did not increase the predictive performance of the specialized slash_burn model.

5. References

[1] Encyclopaedia Britannica, "Amazon Rainforest," 27 08 2019. [Online]. Available: https://www.britannica.com/place/Amazon-Rainforest. [Accessed 2019].

[2] Pachamama Alliance, "Effects of Deforestation," 2019. [Online]. Available: https://www.pachamama.org/effects-of-deforestation. [Accessed 2019].

[3] Planet, "Understanding the Amazon from Space," 2017. [Online]. Available: https://www.kaggle.com/c/planet-understanding-the-amazon-from-space/overview/description. [Accessed 2019].

[4] Planet, "Understanding the Amazon from Space - Data," 2017. [Online]. Available: https://www.kaggle.com/c/planet-understanding-the-amazon-from-space/data. [Accessed 2019].