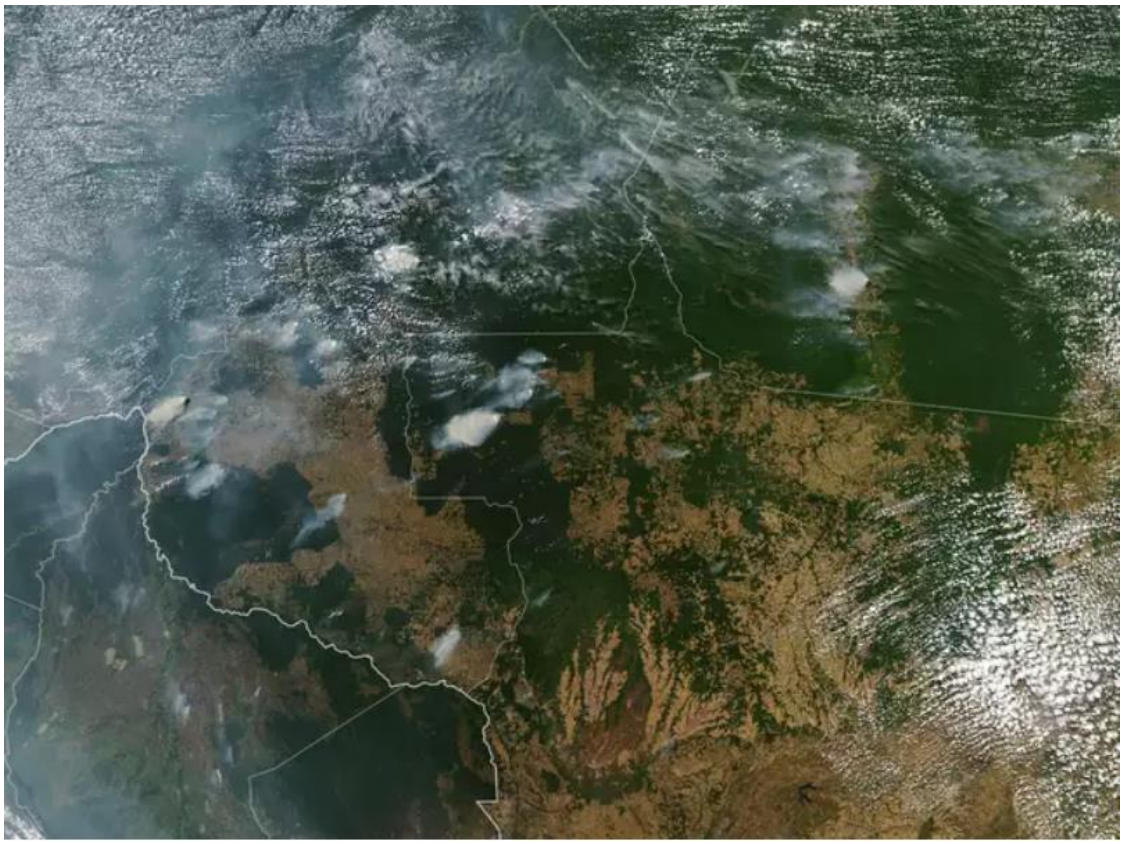
Springboard Capstone Project 2 – Machine Learning

Amazon Rainforest Image Classification



Connor McAnuff

November 27, 2019

1. Machine Learning

1.1 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		

1.2 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.

1.3 Dataset

Dataset formatting is necessary to put the image information into the format required for CNN 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.

1.4 Convolutional Neural Networks

1.4.1 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.

1.4.2 Base Model

The base model was run for 50 epochs. The loss and Fbeta over epochs are shown in Figure 1. 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.

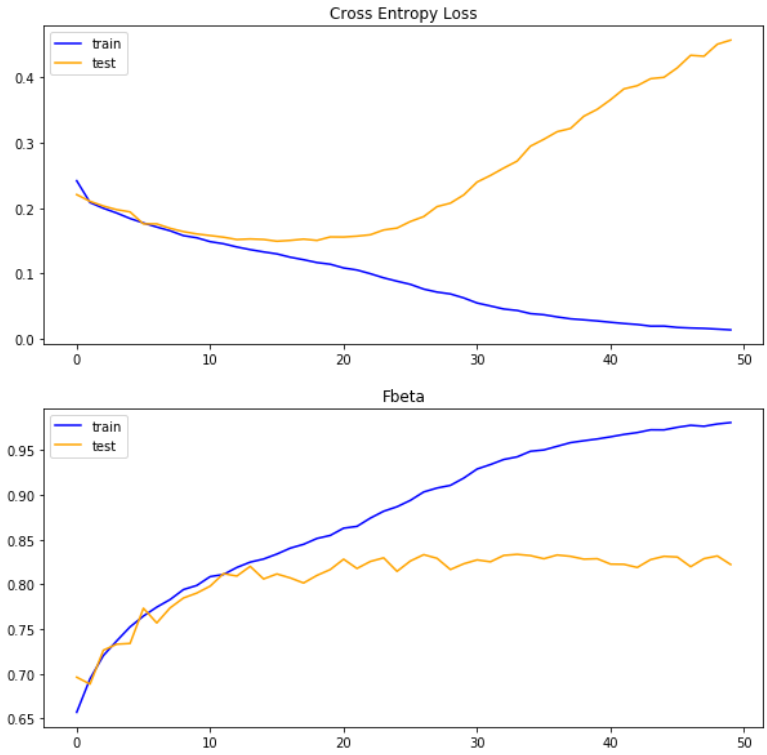


Figure 1: Base model loss and Fbeta over epochs.

1.4.3 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 2. 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.

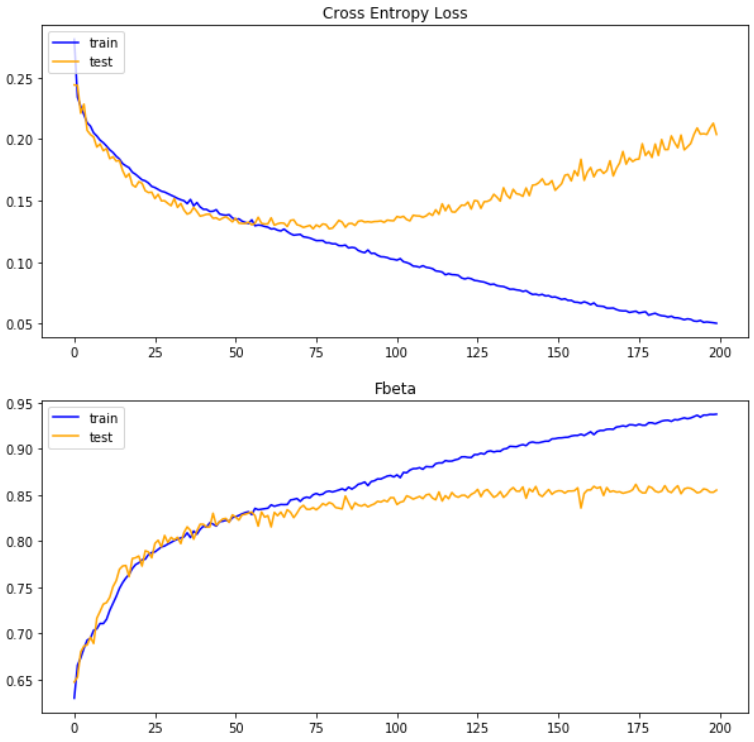


Figure 2: Dropout model loss and Fbeta over epochs.

1.4.4 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 3. 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.

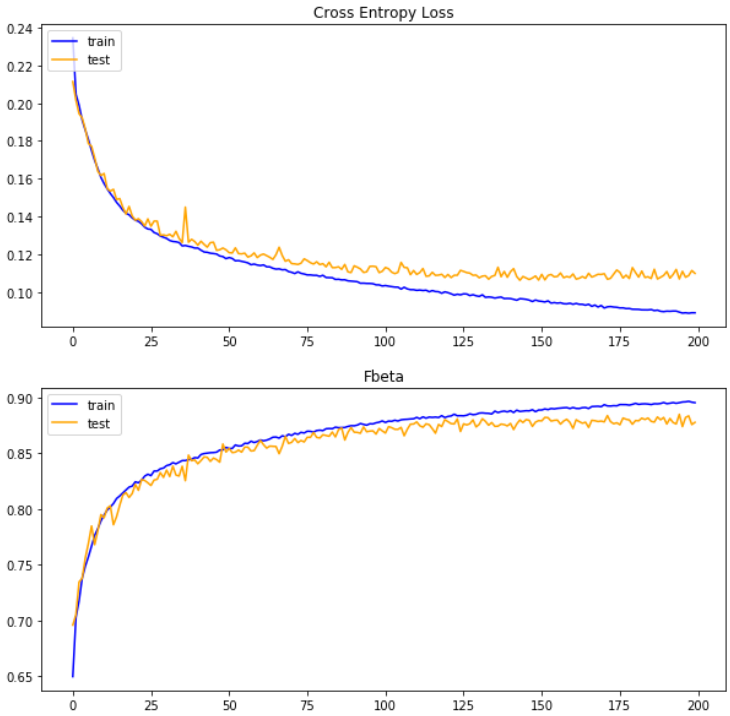


Figure 3: Augmentation model loss and Fbeta over epochs.

1.4.5 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 4. 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.

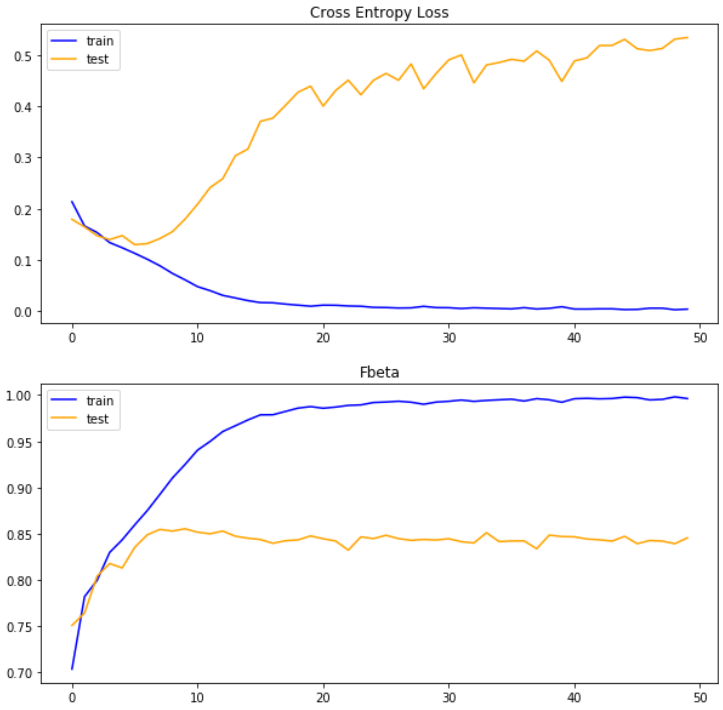


Figure 4: Adam model loss and Fbeta over epochs.

1.4.6 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 5. 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.

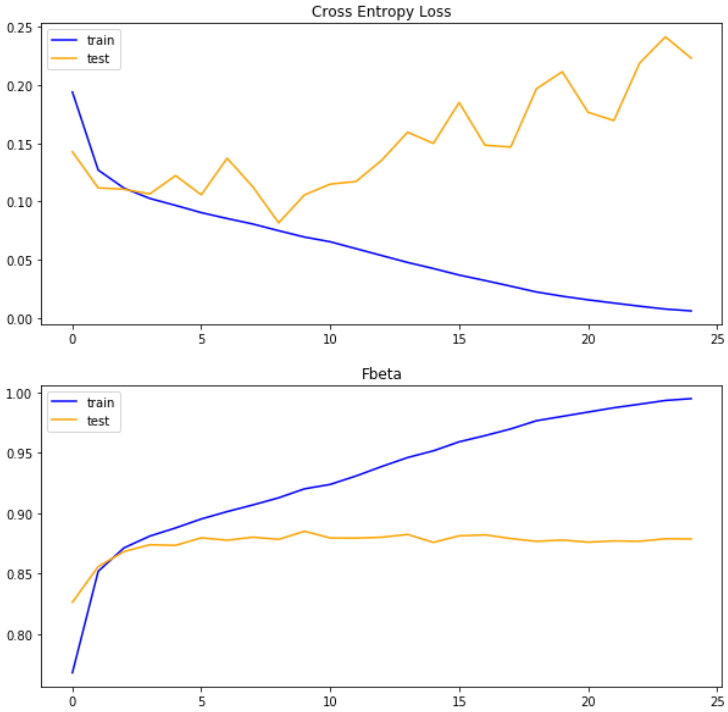


Figure 5: Pre-trained model loss and Fbeta over epochs.

1.4.7 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

1.5 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 6), recall (Figure 7), and Fbeta (Figure 8) have been calculated for each tag.

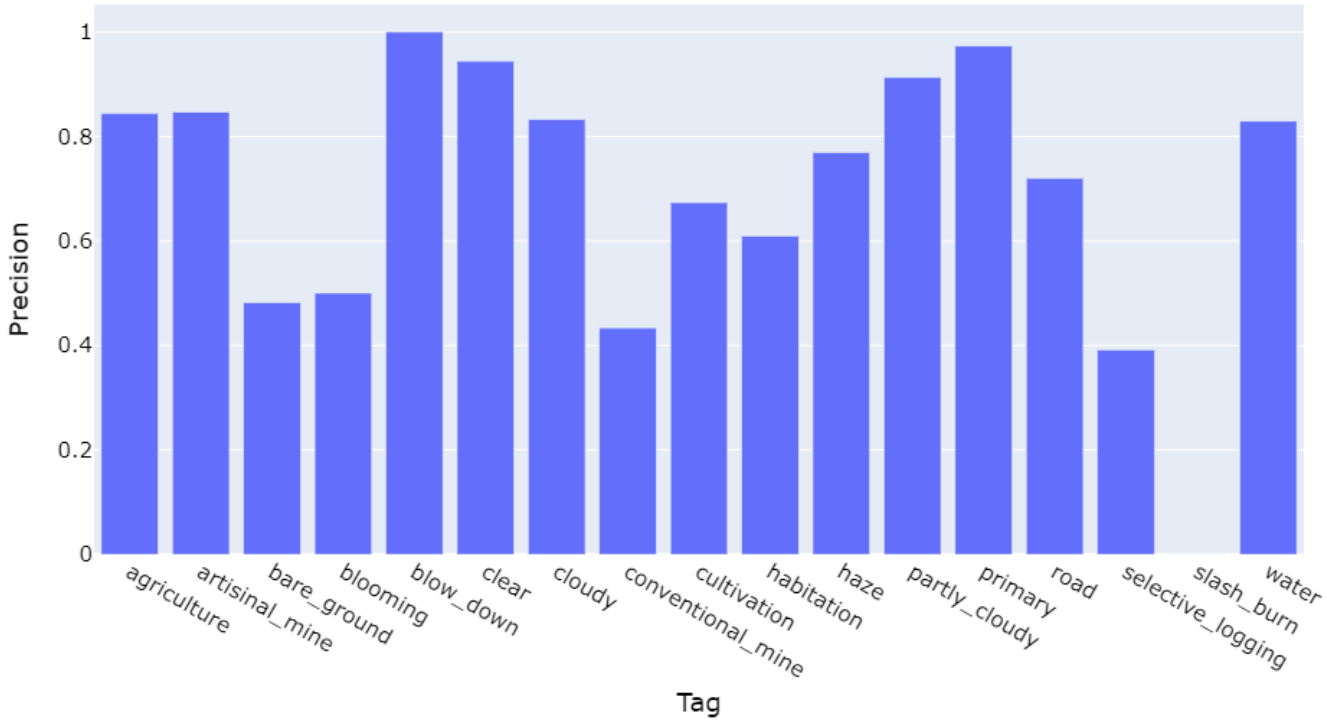


Figure 6: augment_model precision by tag.

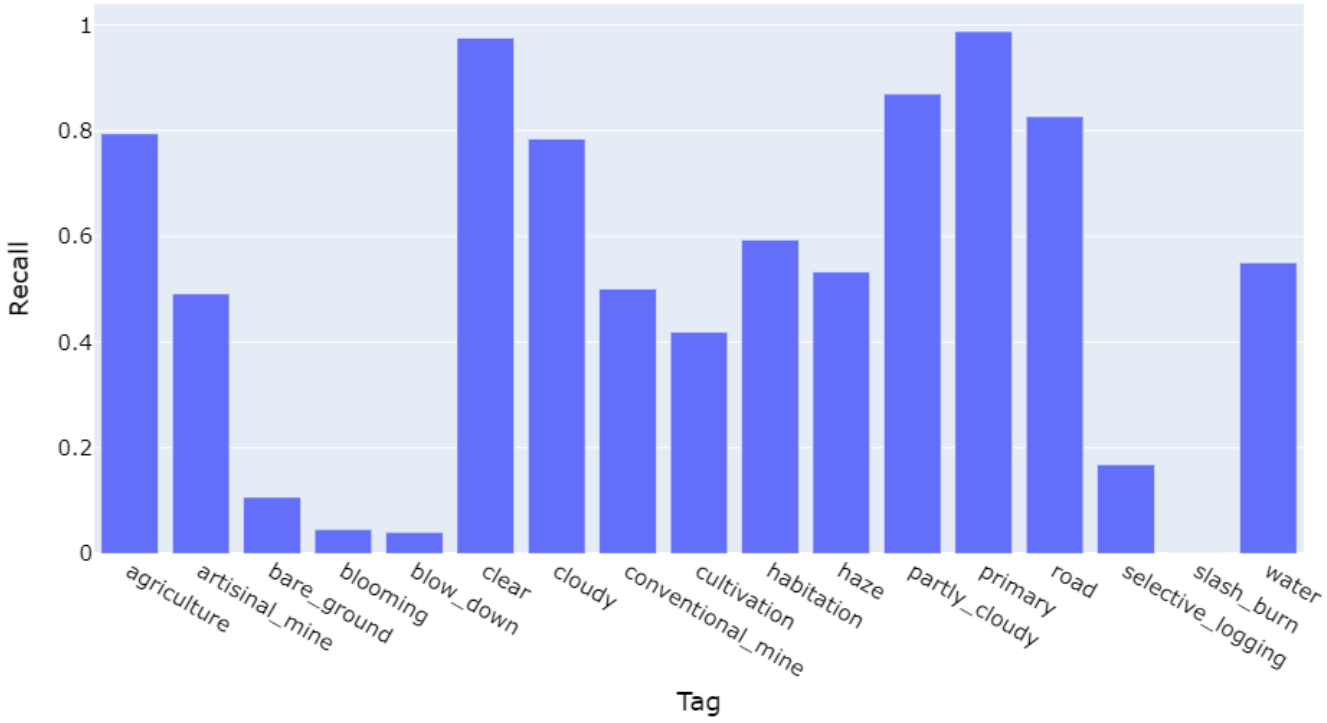


Figure 7: augment_model recall by tag.

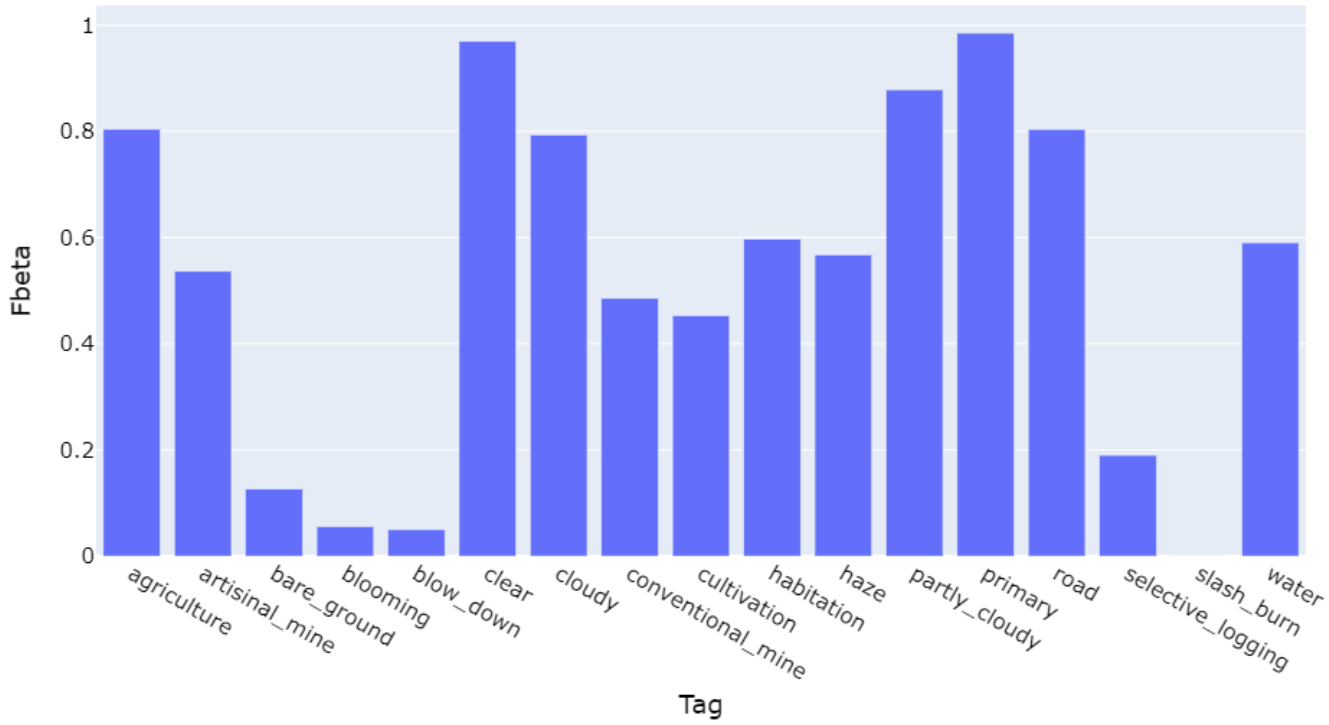


Figure 8: augment_model Fbeta by tag.

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 9, Figure 10, and Figure 11. 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.

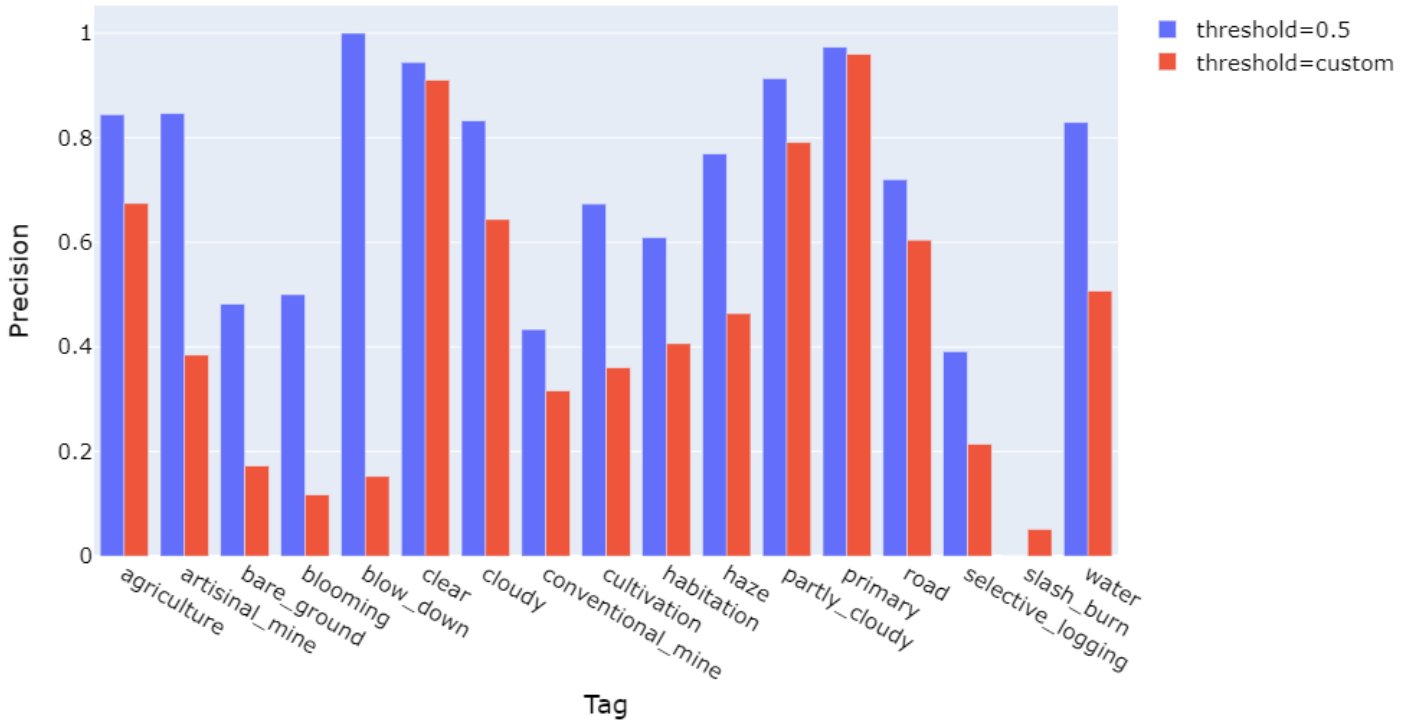


Figure 9: augment_model precision by tag using best thresholds and a threshold of 0.5.

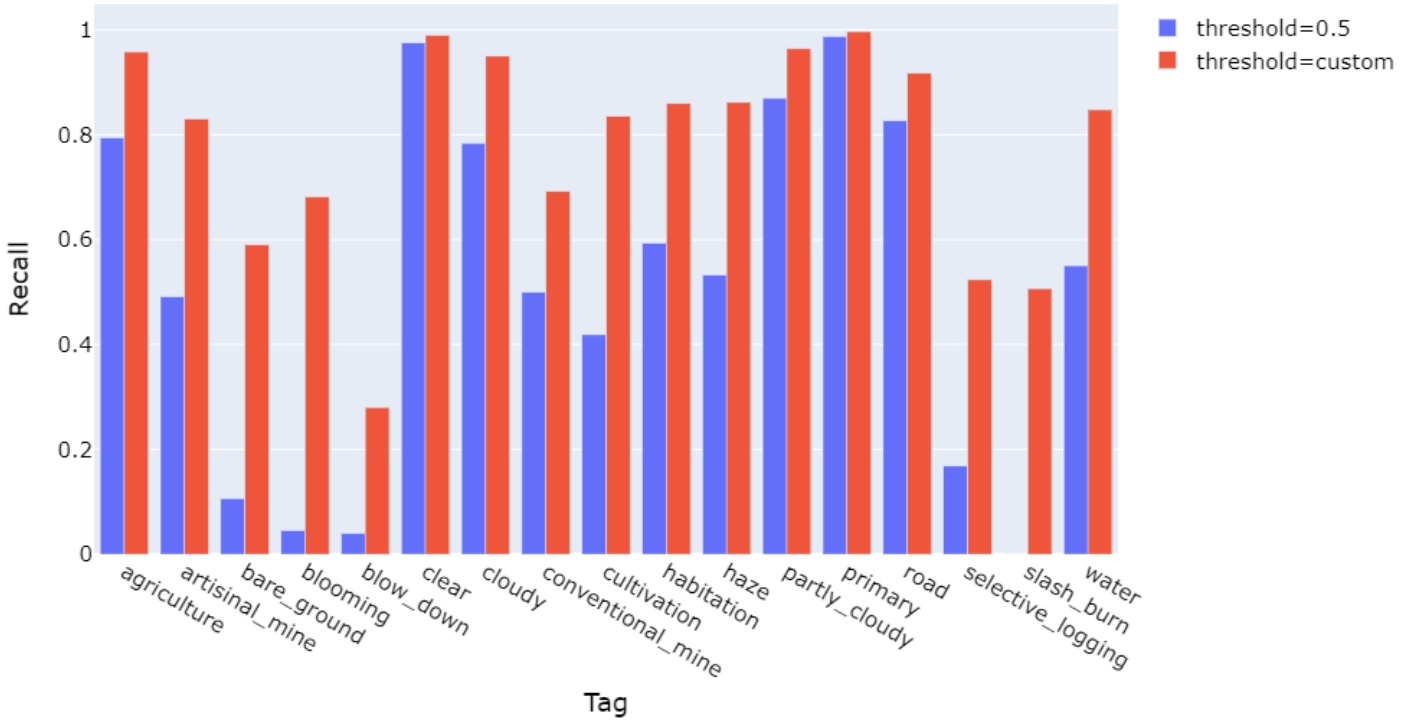


Figure 10: augment_model recall by tag using best thresholds and a threshold of 0.5.

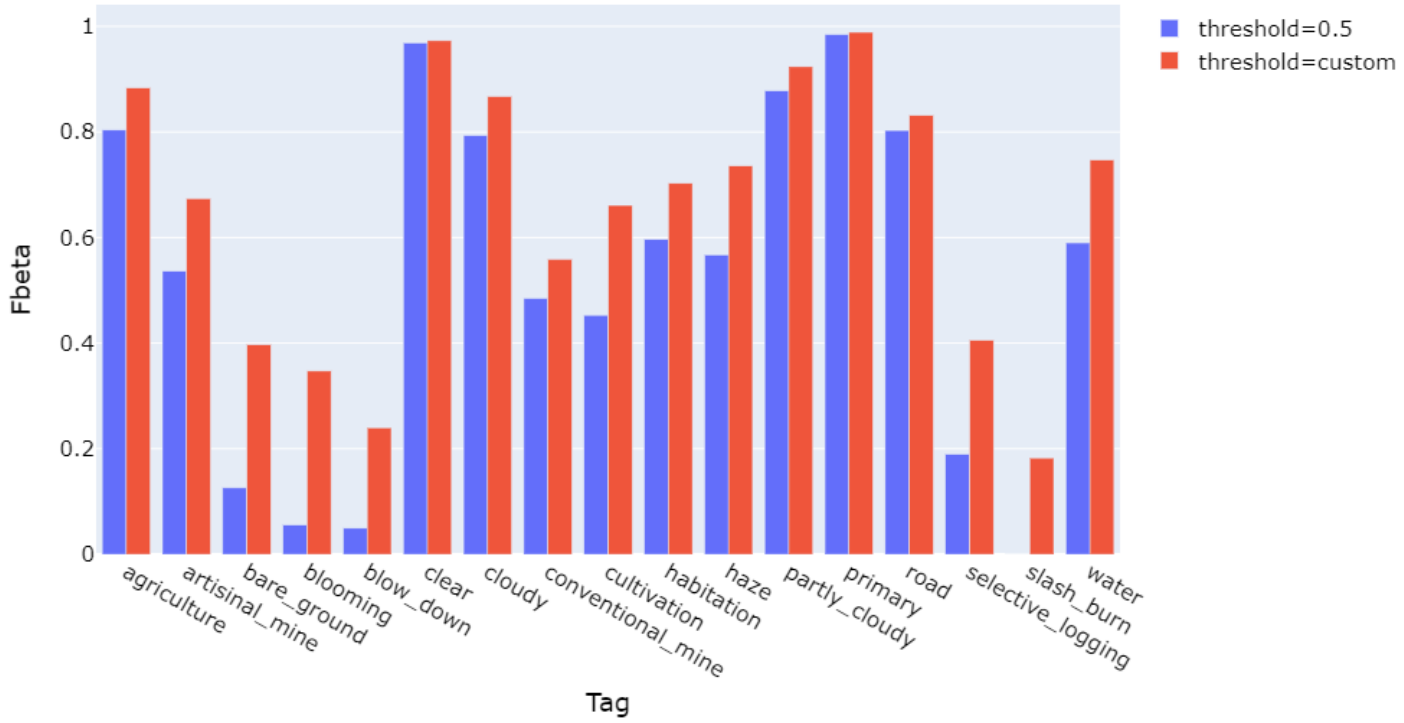


Figure 11: augment_model Fbeta by tag using best thresholds and a threshold of 0.5.

1.6 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.

1.7 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.

2. 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].