Classify Amazon Rainforest Pattern Using Satellite Data

Yao Weng

Abstract

In recent decades, scientists have studied the deforestation patterns using satellite data, and indicate that Amazon rainforest deforestation is rapidly accelerating. Kaggle together with Planet, designer and builder of the worlds largest constellation of Earth-imaging satellites, hold a competition to apply satellite data to track the human footprint in the Amazon rainforest. The labeled satellite image chips are provided by Planet and its Brazilian partner SCCON (Santiago & Cintra Consultoria), I will develop an algorithm to classify the image using what I have learned in Machine Learning Engineer Nanodegree Program.

1 Domain Background

Satellites are very powerful tools to detect deforestation of rainforest.The National Institute for Space Research (INPE) have developed a near real-time deforestation detection system (DETER) and it has helped Brazils government to reduce its deforestation rate by almost 80% since 2004 [1]. In April 21th 2017, Kaggle launched a competition of Understanding the Amazon from Space. It encourages competitors to apply machine learning algorithms to label satellite image chips with atmospheric conditions and various classes of land cover/land use [2]. I choose this topic as my capstone project for two major reasons. First, it is worthy to develop an algorithm to precisely classify the rainforest satellite images, as it will help scientists to do offline study of the deforestation pattern, rate and then better understanding the problem. Secondly, a fast algorithm can be applied in the real-time system which can help scientists or governments to quickly respond to the on-going deforestation.

2 Problem Statement

The chips (image) were derived from Planet's full-frame analytic scene products using our 4-band satellites in sun-synchronous orbit and International Space Station orbit. They were labeled using the Crowd Flower platform and a mixture of crowd-sourced labor and Berlin and San Francisco teams of Planet, the whole processing is time consuming and costly.

The labels can broadly be broken into three groups: a) atmospheric conditions, b) common land cover/land use phenomena, and c) rare land cover/land use phenomena.

An image may have one and potentially more than one atmospheric label, and zero or more common and rare labels, shown in Fig. 1. The task is to assign a chip a set of target labels, which can be considered as a multi-label classification.

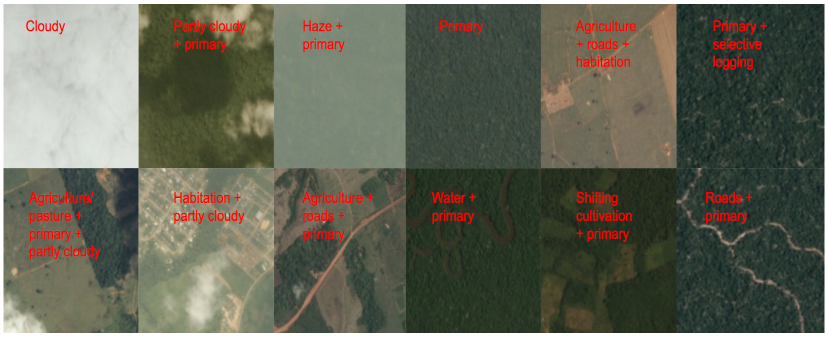
Cloudy
Partly cloudy + primary
Haze + primary
Primary
Agriculture + roads + habitation
Primary + selective logging
Agriculture/ pasture + primary + partly cloudy
Habitation + partly cloudy
Agriculture + roads + primary
Water + primary
Shifting cultivation + primary
Roads + primary

Figure 1: Sample chips and their labels.

3 Datasets and Input

Data [2] are composed of 40479 labeled training images and 61191 unlabeled images for final submission. The file size of an original image is $256 \times 256 \times 3$, I resize it to $128 \times 128 \times 3$. The labeled data is splitted into 80% training set and 20% testing set. Fig 2 and Tab. 1 show the label distributions of the training and testing data.

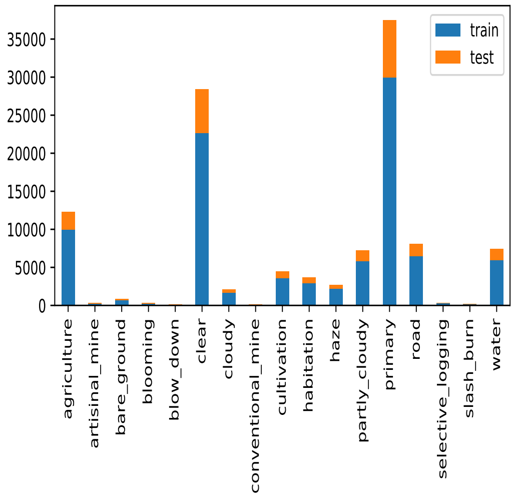

Figure 2: Labels distribution between training and testing

Label	total	train	test
cloudy	2089	1686	403
partly cloudy	7261	5844	1417
haze	2697	2177	520
clear	28431	22676	5755
primary	37513	29977	7536
water	7411	5941	1470
habitation	3660	2927	733
agriculture	12315	9950	2365
road	8071	6490	1581
cultivation	4477	3560	917
bare ground	862	698	164
slash burn	209	171	38
selective logging	340	285	55
blooming	332	260	72
conventional mine	100	81	19
artisinal mine	339	268	71
blow down	98	75	23

Table 1: Label classification of training/testing data.

4 Solution and Statement

There are two main methods to solve multi-label classification problem. One is to transform the multi-label problem into single label problems(s), but the transformed dataset grows large with high label cardinality and cannot model dependencies between labels. The other one is to adapt a single-label algorithm to produce multi-label outputs. In this competition, I will adapt conventional neural network with multiple outputs.

4.1 Loss Function

In single label classification, we use softmax function to squash the values of a vector in the range $[0, 1]$ that add up to 1. In multi-label case, a sigmoid function Eq. 1 is used to predict the outputs, which indicates the class probabilities.

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{1}$$

Binary cross entropy function is used as loss function Eq. 2,

$$\begin{aligned} \mathcal{L}(\theta) &= -\frac{1}{n} \sum_{i=1}^{n} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)] \\ &= -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} y_{ij} \log(p_{ij}) \end{aligned} \tag{2}$$

where i sums over all samples and j sums over all classes, y is the sample label and $p_{ij} \in (0, 1)$,$\sum_j p_{ij} = 1 \forall i, j$ is the prediction for a sample.

4.2 Measurement Metrics

Evaluation of multi-label algorithm is hard mostly because of partial correctness of a prediction. One trivial way around is to ignore partially correctness and extend the accuracy used in single label case for multi-label prediction, see Eq. 3. It represents the average percentage matching among 17 labels.

$$\text{accuracy} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{I}(y^i = y^i_{\text{pred}}). \tag{3}$$

In this kaggle competition, submissions are evaluated by mean F_2 score, see Eq. 4,

$$F_2 = (1 + \beta^2) \frac{pr}{\beta^2 p + r}, \quad \beta = 2. \tag{4}$$

Precision p is the ratio of true positives to all predicted positives (true positive + false positve), and recall r is the ratio of true positives to all actual positives (true positie + false negative), where

- True positive: label_i is correctly identified as label_i
- False positive: label_j is incorrectly identified as label_i $(i \neq j)$
- True negative: label_j is not indentified as label_i $(i \neq j)$
- False negative: label_i is not identified as label_i

F_2 weights recall higher than precision, because we would rather predicate a false alarm of deforestation than fail to detect a real one. The mean F_2 score is formed by averaging the individual F_2 scores for each row in the test data.

4.3 Baseline Model

Convolutional neural network (CNN) has demonstrated excellent performance on many complex visual tasks. Compared with regular neural network, CNN has layers not connecting to every single pixel of the input image. It learns an image from the combination of features like, edges, shape, corner, color *etc.* We call these features as filters, and each filter only learns its neighbor in a small range. CNN has 3 major components:

- Convolution layer: we slide the filters (weight matrices) over the complete image, and minimize the loss function to learn the weights. The filters contain features extracted from the original image, for example, one filter might learn edge and another one learn shape *etc.* In general, the initial layer extract more generic features, when network goes deeper, the filters will learn more complex features.
- Pooling or Sub Sampling: we reduces the dimensionality of weight matrices but retains the most important information. There are different spatial pooling types: max, average, sum *etc.*
- Fully Connected Layer is a multi layer perceptron that uses a softmax activation function in the output layer for classification.

My baseline model is a two layers of CNN with three fully connected layers. Each fully connected layer is followed by a pooling layers with drop probability 0.3. Early stopping is used as regularization to avoid overfitting. Fig. 3 shows the architecture of baseline CNN. Data is trained in 31 epochs with batch size of 128. Fig. 5 shows accuracy, loss and F_2 score over 31 training epochs, where the tagging threshold of prediction is 0.5. However, multi-label is different from single label classification, in single label classification, we select label with highest score as prediction. In multi-label classification, the threshold for each label might be different, thus, I tune the label threshold to optimize F_2 of each label over the training data. For each label, I calculate F_2 scores using 20 different thresholds that evenly spaced in range $[0.01, 0.99]$, then pick the one which maximizes the F_2 score.

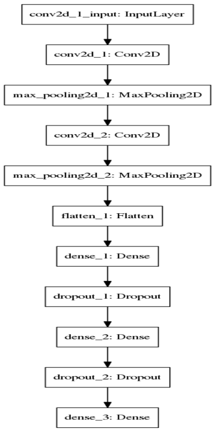

Figure 3: Tensorboard: baseline CNN architecture.

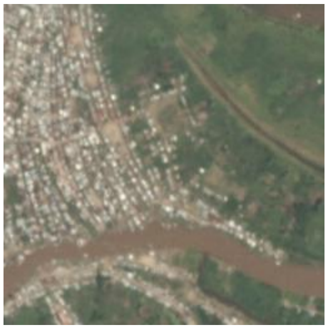

Figure 4: True Label: agriculture clear habitation primary road water.
Prediction: agriculture clear habitation primary road

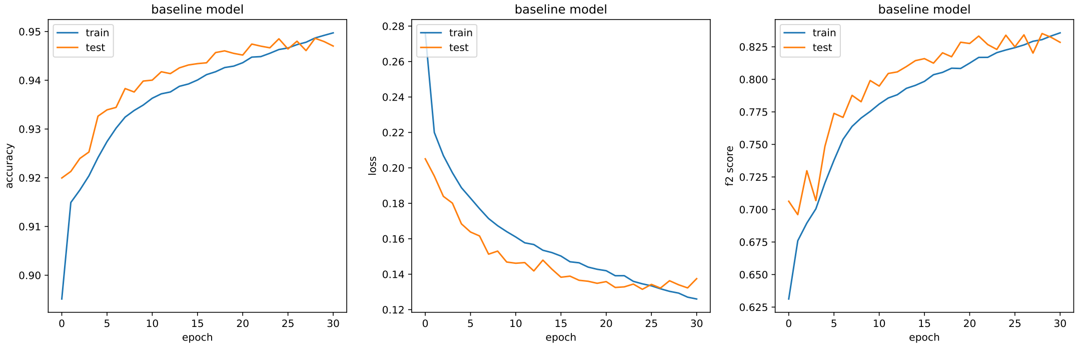

Figure 5: Accuracy, loss and F_2 score in each epoch

Label	threshold	training F_2	testing F_2	threshold	training F_2	testing F_2
cloudy	0.5	0.8396	0.7991	0.2679	0.8725	0.8184
partly cloudy	0.5	0.9195	0.8347	0.2679	0.9406	0.8626
haze	0.5	0.5515	0.5428	0.1647	0.7614	0.7124
clear	0.5	0.9724	0.9556	0.2163	0.9801	0.9714
primary	0.5	0.9826	0.9831	0.2679	0.9886	0.9883
water	0.5	0.3104	0.2636	0.1647	0.6741	0.6577
habitation	0.5	0.4006	0.3403	0.1647	0.7038	0.6499
agriculture	0.5	0.8062	0.7661	0.2163	0.8708	0.8346
road	0.5	0.6362	0.6158	0.2163	0.8104	0.7823
cultivation	0.5	0.1559	0.1097	0.1647	0.6312	0.6056
bare ground	0.5	0.0	0.0	0.0616	0.3804	0.2814
slash burn	0.5	0.0	0.0	0.0616	0.2018	0.0375
selective logging	0.5	0.0	0.0	0.0616	0.4332	0.1975
blooming	0.5	0.0	0.0	0.0616	0.3157	0.2193
conventional mine	0.5	0.0	0.0	0.0616	0.1933	0.1773
artisinal mine	0.5	0.2096	0.0865	0.2163	0.7410	0.4225
blow down	0.5	0.0	0.0	0.0100	0.1123	0.0812

Table 2: Baseline model: label tagging threshold, and label F_2 score of training/testing

Tab. 2 compares F_2 from using fixed threshold 0.5 with those from using optimized threshold, which are improved especially for rare features. I submit the predictions using baseline model, F_2 score is 0.82752 and 0.87329 for using fixed threshold and optimized threshold respectively. (Tab. 5 lists the F_2 of training/testing data), sample submission benchmark from Kaggle is 0.67485.

4.4 Transfer Learning

Transfer learning involves taking a pre-trained neural network and adapting the neural network to a new different data set and thus accelerate our deep learning model. There are two major methods in transfer learning: retraining and fine tuning, depending on the size of new data set and the similarity of new data to the original data. There are four main cases, see Tab. 3.

	size of new data	similarity to original data
case 1	small	similar
case 2	small	different
case 3	large	similar
case 4	large	different

Table 3: Transfer learning

In transfer learning, we replace the last fully connected layer with a layer matching the number of classes in the new data set.

- case 1: Freeze all the weights from the pre-trained network except the last layer, train the network to update the weights of the new fully connected layer.
- case 2 : Remove most of the pre-trained layers near the beginning of the network, retrain the weights of final layer like case 1
- case 3 : Fine tune the network by initializing weights except the last layer with the pre-trained weights and retrain the entire neural network.
- case 4: Retrain the network from scratch with randomly initialized weights or try fine tuning.

4.4.1 Inception v3 Architecture

GoogleNet won the ILSVRC 2014 challenge by pushing the top-5 error rate below 7%. The great performance is achieved by a sub-network called Inception. Inception architecture, see Fig 6, stacks multiple Inception modules to a deep depth, and each module is also ”wide” and architected to recognize features at multiple length scales. It includes 3×3 and 5×5 convolutions in each stacked module, as shown in Fig 7. However, these convolutions are expensive, and it is getting worse when repeatedly stacked them in a deep depth. To reduce dimensionality, a 1×1 convolution is stacked in front of the expensive 3×3 and 5×5 convolutions respectively, see Fig. 8.

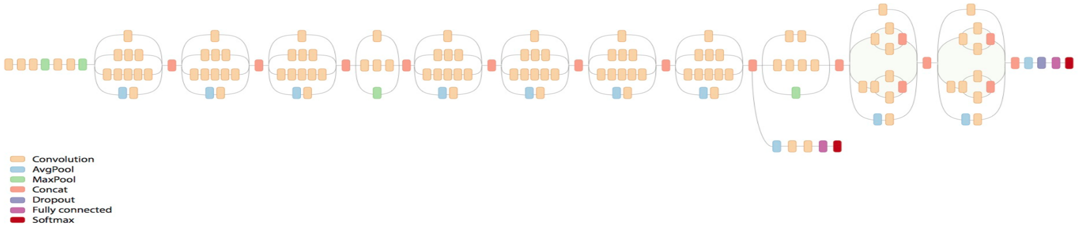

Figure 6: Inception v3 Architecture.

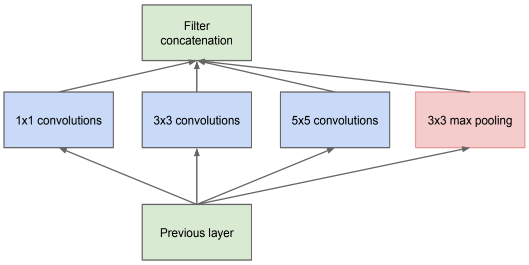

Figure 7: Naive inception module [3].

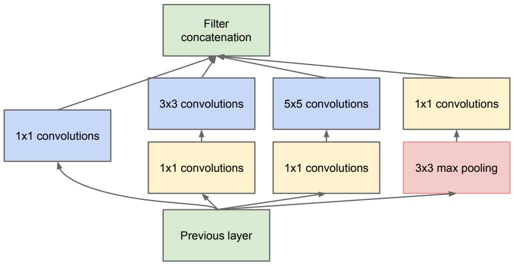

Figure 8: Inception module [3].

4.4.2 Retraining And Fine Tuning

The amazon satellite image data set is medium, however, is different from ImageNet which used to train InceptionV3, VGG16 *etc.* I load InceptionV3 model from keras, pop up the final layers and use a fully-connected layer in shape of $1 \times 1 \times 17$ as the final layer, where 17 is the total number of classes of amazon images. ImageDataGenerator of keras is used to generate batches of augmented tensor image in realtime training, the images are randomly rotated, and horizontally/vertically shifted. Fig. 9 shows the augmentation strategy applied on original image in Fig. 4. Adam optimizer [4] is chosen to minimize the loss function Eq.2 with a constant learning rate of 0.0005.

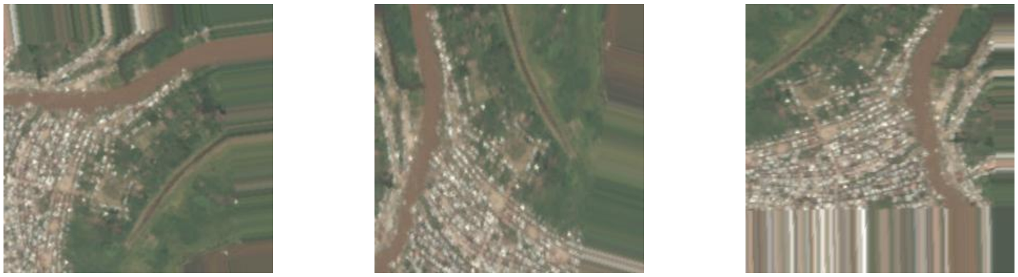

Figure 9: Augmented figures of Fig. 4.

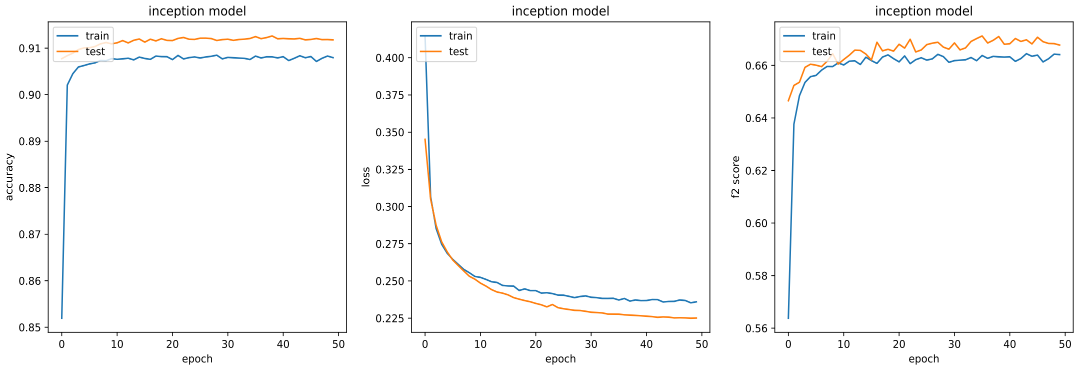

Figure 10: Transfer learning: accuracy, loss and F_2 score in each epoch

I freeze all initial weights from training ImageNet except those in the last fully connected layer, retrain the network in 50 epochs and save the checkpoint. Fig. 10 shows metrics in each epochs, all of which are worse than baseline model. Then I use the weights from retraining InceptionV3 as the initial weights and train the weights in all layers using 20 epochs with a smaller learning rate of 0.0001. In baseline model, optimized threshold gives a better prediction and higher F_2, therefore, I use the same method to find label thresholds.

Fig. 11 shows an improvement of accuracy, loss and F_2 score. Tab. 4 shows the tagging threshold and F_2 of labels for training and testing data, from which F_2 score is improved for all labels and the improvement for rare features, like blow down, slash burn *etc.* is noticable.

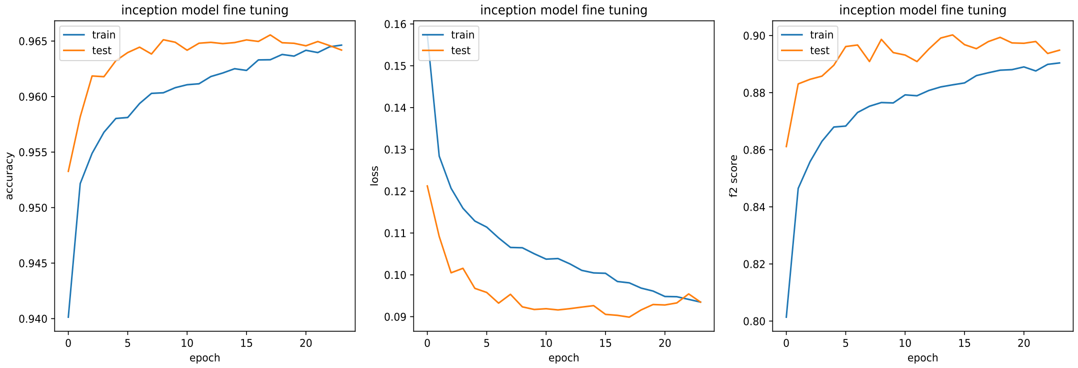

Figure 11: Fine tuning InceptionV3: accuracy, loss and F_2 score in each epoch

Label	tagging threshold	training F_2	testing F_2
cloudy	0.16474	0.9051	0.8832
partly cloudy	0.21632	0.9520	0.9275
haze	0.16474	0.8115	0.7598
clear	0.16474	0.9809	0.9731
primary	0.31947	0.9910	0.9921
water	0.16474	0.8271	0.8158
habitation	0.16474	0.7671	0.7433
agriculture	0.16474	0.9028	0.8936
road	0.21632	0.8673	0.8489
cultivation	0.16474	0.6957	0.6842
bare ground	0.06158	0.5228	0.4480
slash burn	0.06158	0.3113	0.3109
selective logging	0.11316	0.5300	0.4218
blooming	0.11316	0.4292	0.3167
conventional mine	0.21632	0.7160	0.5376
artisinal mine	0.21632	0.8656	0.8514
blow down	0.11316	0.4669	0.3901

Table 4: Fine tuning InceptionV3: label tagging threshold, and label F_2 score of training/testing

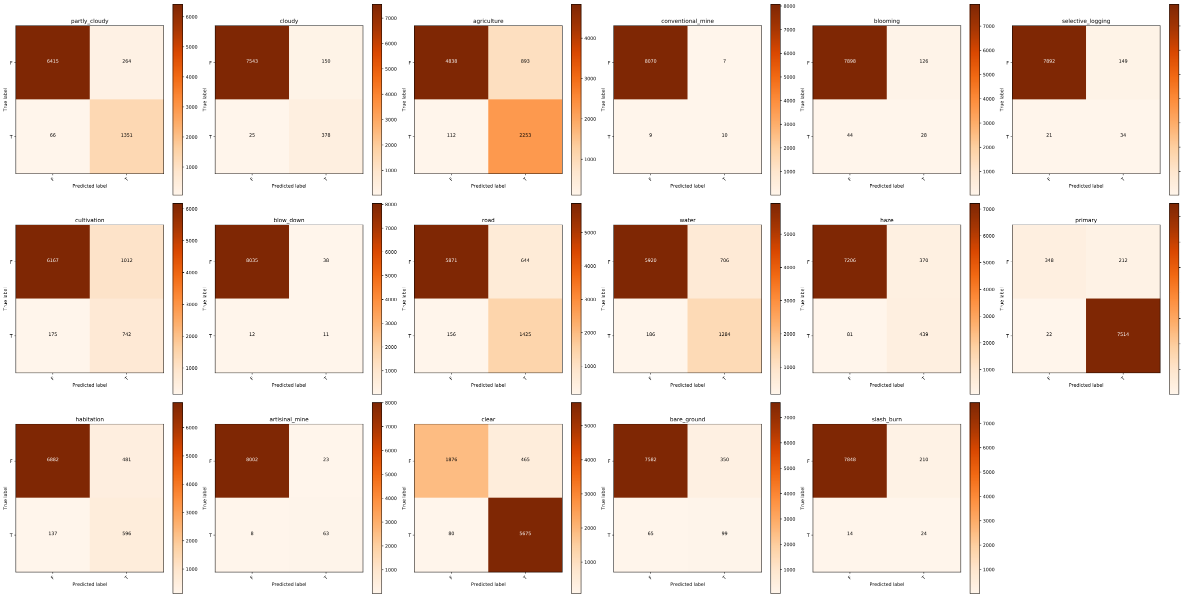

Figure 12: Fine tuning InceptionV3: confusion matrices of labels calculated with testing data.

5 Conclusion

5.1 Justification

Tab. 5 shows the F_2 scores of training/testing data and final submission of unlabeled data using baseline model and fine tuning InceptionV3. F_2 score is improved by $\sim 5\%$ when using transfer learning on InceptionV3, which demonstrates the power of transfer learning using deep CNN. My model provides a good prediction (F_2 is around 90%) of common features like 'primary', 'clear', 'cloudy', and a relatively good prediction (F_2 is around 80%) of some potential deforestation sign like 'agriculture', 'habitation' 'road'.

5.2 Reflection

However it is still behind the highest F_2 score 0.93318 in Kaggle lead board, my rank is 404 of total 938 groups. Fig. 12 shows label confusion matrices calculated using testing data. For rare features, like 'bare ground', 'blooming' and 'selective logging', false negative is relatively large, it indicates that my model is more likely to predicate false for rate features.

	F_2 training	F_2 testing	F_2 final submission
Baseline (fixed threshold 0.5)	0.852129	0.834331	0.82752
Baseline (optimized thresholds)	0.894239	0.874115	0.87329
Fine tuning InceptionV3	0.928658	0.920617	0.91796

Table 5: F_2 scores of training/testing data and final submission of unlabeled data for baseline model and fine tuning InceptionV3 model.

There are several ways to improve this multi-label learning.

- Label distribution is unbalanced, F_2 and accuracy of rare features might be improved by generating more augmented images with rare labels.
- Apply transfer learning on other pre-trained models like ResNet, VGG16, VGG19 *etc.*, and compare the results with InceptionV3.
- Apply ensembing method, which depends on the predictive power of several models to make final prediction.

This project is my first Kaggle competition, I am excited to explore and optimize different machine learning models. It is a valued experience to leverage what I learned in Machine Learning Nano Degree with the real life problem, and understand the powerful form of machine learning – 'Deep Learning'.

References

[1] Gabriel Popkin. Satellite alerts track deforestation in real time. *Nature*, 530(52):392–393, February 2016.

[2] https://www.kaggle.com/c/planet-understanding-the-amazon-from-space.

[3] Yangqing Jia etc. Christian Szegedy, Wei Liu. Going deeper with convolutions. *Computer Vision and Pattern Recognition, IEEE Conference*, (INSPEC Accession Number: 15523970), June 2015.

[4] Jimmy Ba Diederik P. Kingma. Adam: A method for stochastic optimization. *ICLR*, 2015.