Machine Learning Engineer Nanodegree

Capstone Report

Amitabha Chakravarty
18 July 2017

Kaggle Competition - **Planet: Understanding the Amazon from Space**

Definition

Project Overview

Our planet is loosing valuable resources every minute. Every minute an area of forest the size of 48 football field is disappearing from Amazon basin due to deforestation. Deforestation has caused a large scale of devastating effects – reduction of biodiversity, habitat loss, climate change to name a few. To counter the effect of loss of such forest area we need data that will pin point areas of human encroachment. This will help local governments and local stakeholders to respond quickly and effectively.

For this purpose, daily imagery from Planet, designer and builder of world's largest constellation of Earth-imaging satellites, will be used along with its Brazilian partner SCCON. They want Kagglers to label the satellite image chips with atmospheric conditions and various classes of land cover and land use. Resulting algorithms will help the global communities understand better where and how deforestation is happening and ultimately how to deal with it.

Problem Statement

Image (chip) data is collected from Planet's full-frame analytic scene products using 4 band satellites in sun-synchronous orbit. The data is labeled using Crowd Flower platform and a mixture of crowd-sourced labor. There are class labels like 'Cloudy', 'Partly cloudy + Primary', 'Shifting cultivation + primary' and so on. There are a significant number of cloudy scenes with complete to partial cloud coverage and also hazy conditions. There are also clearly shown images of 'Primary Rain Forest', 'Water (River and Lakes)' and 'Habitation' etc. The job is to come up with a data analysis model and use these labeled data for training purpose. There will be a set of images with no label and the model has to be tested against the unlabeled data.

The goal of this project would be to develop and algorithm to detect the type of scene represented by the testing images with varying atmospheric conditions and various classes of land cover/land use. There are several approaches to analyzing an image and classify the scene based on prior knowledge. One such approach would be finding PCA (Principal Component Analysis) components and classify using SVM (Support Vector Machine). However, with recent

success of deep learning, Convolution Neural Network (CNN) is another very effective technique to analyze image and classify its content based on prior learning. We will go with CNN approach here for its high success rate.
CNN would be used to train and detect the correct label for the ground scenario represented by the amazon image chip. CNN outperforms many image recognition algorithms in ImageNet dataset, The CNN algorithm consists of many convolution operations followed by pooling operation to feed finally to fully connected layer. Finally, the prediction comes out of the classification layer connected next to the fully connected layer.

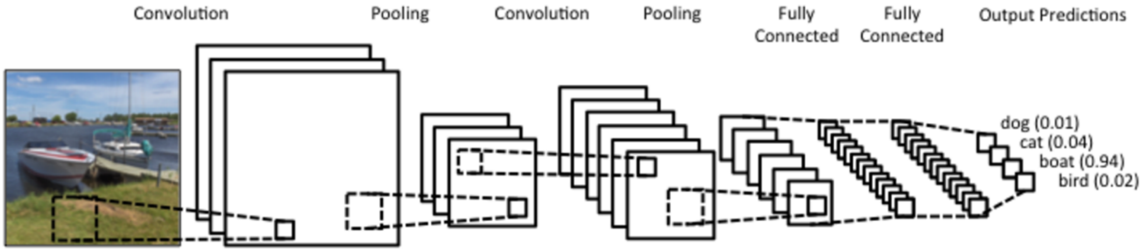


Image courtesy - http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

Metrics

Two kind of metrics will be utilized. One is the ‘Accuracy’ of predicted label against validation data. The other is the ‘F1 Score’ which is defined as follows –

F1 = 2 * (precision * recall) / (precision + recall)

This score is based on the ‘true positive’ and ‘false positive’ as calculated as part of validation.

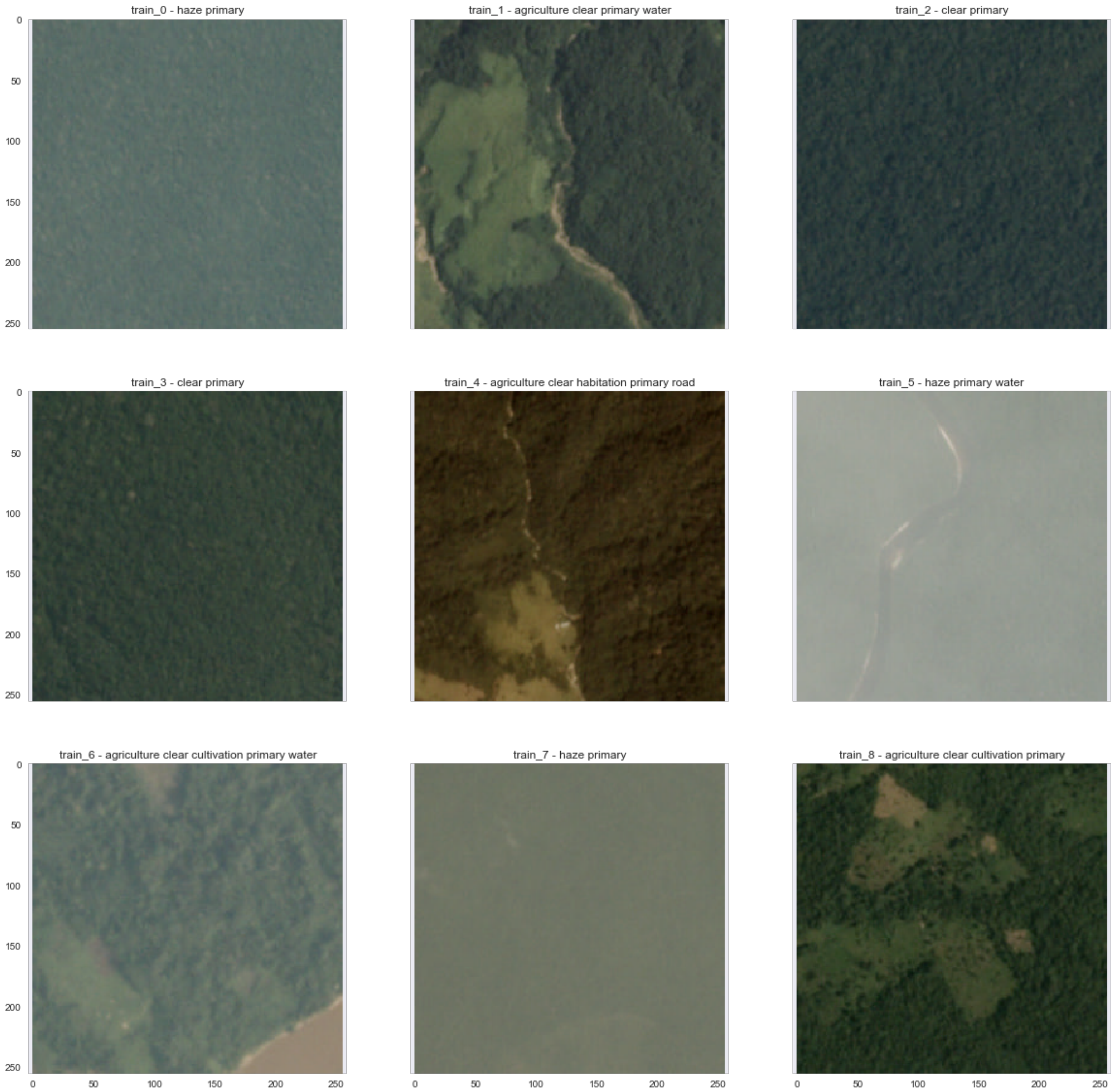
Analysis

Data Exploration

We have around 40000 images for training and 40000 images for testing. There are both jpg and 4-band tiff images for training and testing. Each image is a 256x256 pixels corresponding to a real area of 221.7 Acres. For training the corresponding labels are presented whereas for testing we need to predict the labels.

The majority of the data set is labeled as "primary", which is shorthand for primary rainforest, or what is known colloquially as virgin forest. There are also areas representing 'Water', 'Agriculture', 'Road', 'Cultivation' and so on.
The labels are presented as applicable to the ground reality with multiple keywords like – 'Agriculture/pasture + primary + partly cloudy'.
There are two kinds of images - a "hard" and an "easy" set. The easy set contained scenes that are easier-to-identify labels like primary rainforest, agriculture, habitation, roads, water, and cloud conditions. The harder set of data was derived from scenes to represent shifting cultivation, slash and burn agriculture, blow down, mining, and other phenomenon. A disclaimer is mentioned on the competition page that some training labels could be wrong as there could be labeling error. Overcoming the inaccuracy would be a challenge for this particular completion. There is also significant cloud covering for some of the images. Detecting scenario on ground in presence of cloud would be particularly challenging.

Below are some of the images from the training set. One observation can be made that these images represent scenes that can tolerate horizontal or vertical flips. This can be one method of augmenting the training data set to overcome generalization.


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

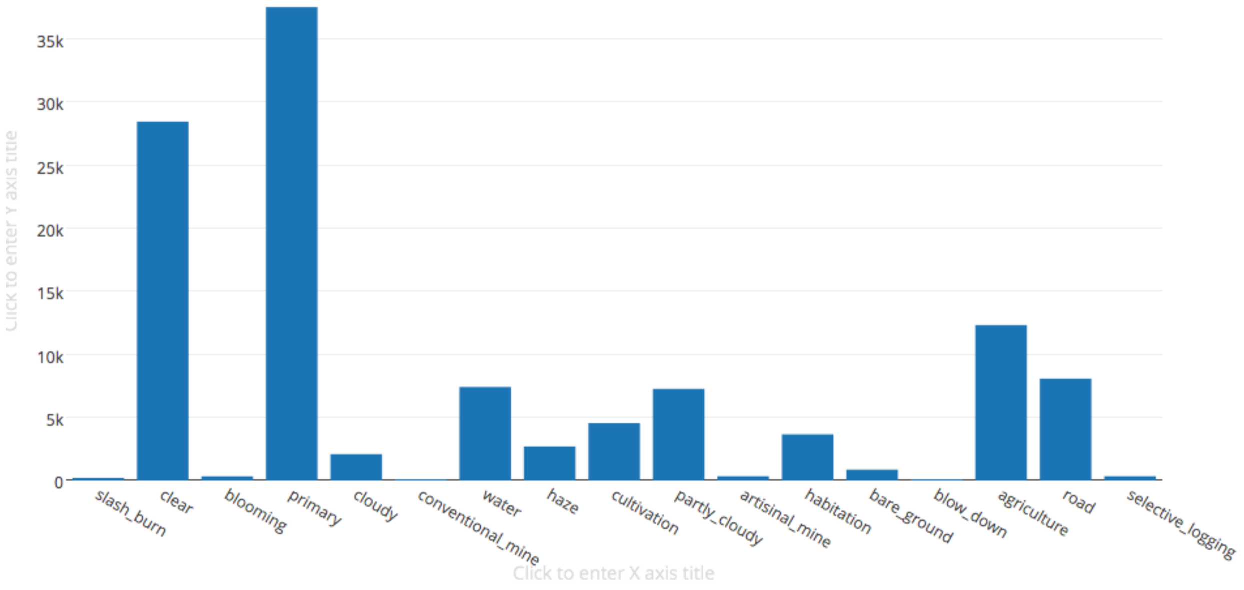
Exploratory Visualization

We have analyzed the input training labels and found that there are following unique 17 levels.

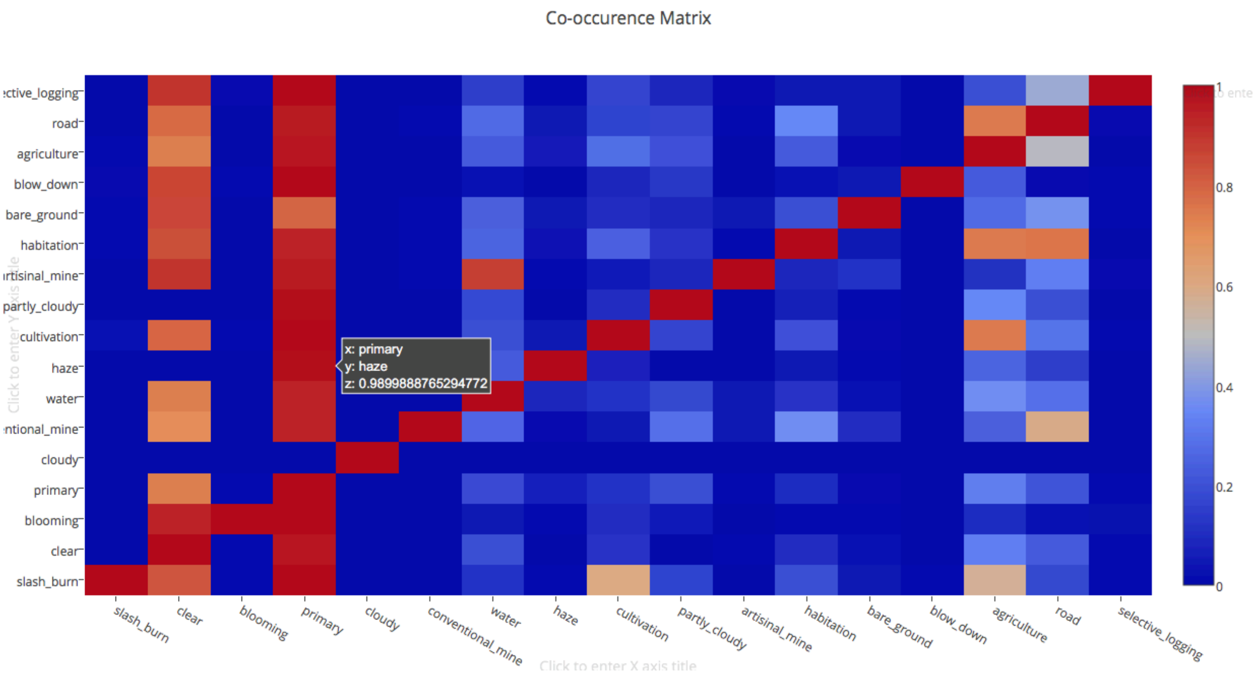
0: 'agriculture',
1: 'artisinal_mine',
2: 'bare_ground',

3: 'blooming',
4: 'blow_down',
5: 'clear',
6: 'cloudy',
7: 'conventional_mine',
8: 'cultivation',
9: 'habitation',
10: 'haze',
11: 'partly_cloudy',
12: 'primary',
13: 'road',
14: 'selective_logging',
15: 'slash_burn',
16: 'water'

Next we have plotted the relative frequency of occurrence of each label as a bar plot as shown below.



One more observation is made about the relative presence of one label in presence of another. This is done by calculating the co-occurrence matrix of all labels as shown in the heat-map below. This basically shows what percentage label A is present when there is already label B. We can see from following heat-map that ‘primary’ is present with almost all other labels except when it is ‘cloudy’. Similarly, we see most of other labels when label ‘clear’ is present which also makes sense. We are seeing also a moderate relation between ‘cultivation’ or ‘agriculture’ with ‘slash burn’ and that is a tale-tell proof of deforestation.



Algorithms and Techniques

Benchmark

Each Kaggle competition has a leaderboard where competitors' scores are arranged in descending order. We looked at the range of results obtained by the leading competitors and arrived at a benchmark score for the algorithm.

The benchmark score was published when the proposal was made for this project. When we published the project proposal we had following observations.

After looking at the leaderboard for current competition at https://www.kaggle.com/c/planet-understanding-the-amazon-from-space/leaderboard we saw that about 700 teams were competing and here were some of the score ranges –

~400 of them scored 90% or more
~350 scored 91% or more
~215 scored 92% or more
~56 scored 93% or more – TOP Tier with highest score being 93.385%

We chose a benchmark score of 91% as half of competitors have crossed that mark. We needed to better that score to consider our model as successful.

The leaderboard has changed since we arrived at the benchmark. Now there are about 900 teams and around 450 teams are above score 0.91. The highest score still remains 0.93. So, our assumption of crossing at least 0.91 to be in the upper 50th percentile still remains valid.

Methodology

Data Preprocessing

Our Convolutional Neural Network will process images through multiple layers and finally produce a classification label. However, it is not a single class. From the training data it is seen that multiple labels are applied against each image. Here are some samples –

```
image_name                                       tags
0    train_0                               haze primary
1    train_1            agriculture clear primary water
2    train_2                              clear primary
3    train_3                              clear primary
4    train_4  agriculture clear habitation primary road
```

So, we have to predict more than one class instead of just one. We have adopted something similar to one-hot-encoding for the labels. We have converted the tags to a fixed length sparse vector where only the corresponding places of labels that are found we have to insert 1, otherwise there is 0 by default.

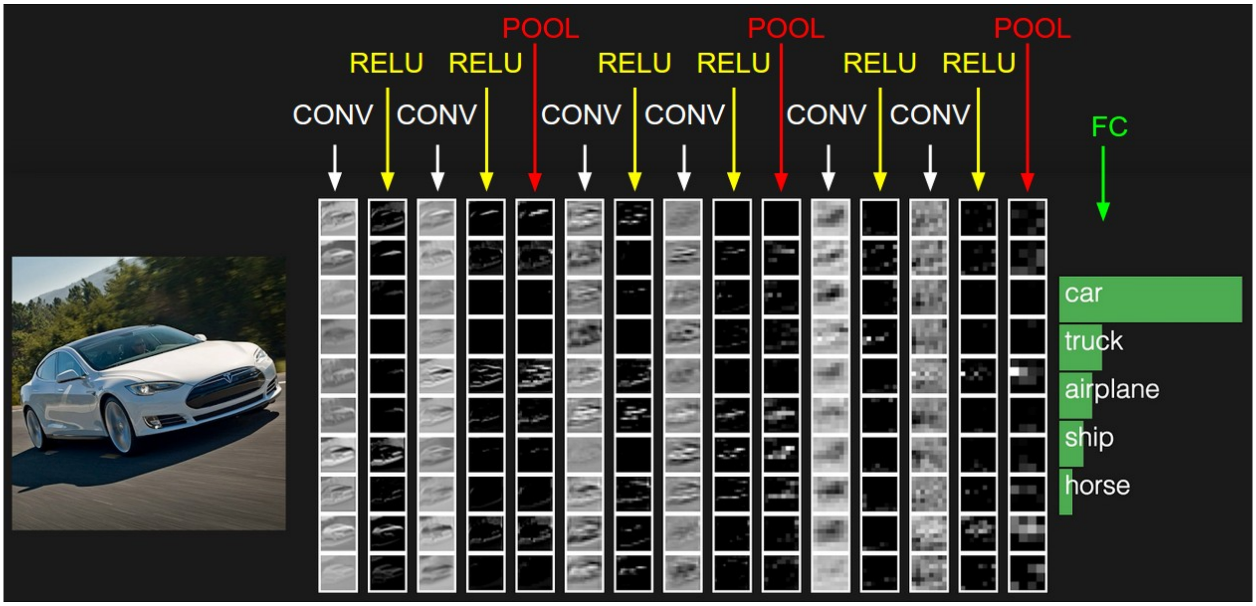
For example – we can have following vector for train_0 image –

```
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  1.  0.  0.  0.  0.]
```

where 1 in 11th place and 13th place indicate presence of labels 'haze' and 'primary'.

Implementation

The RGB 3 channel input is applied to each layer consisting of a convolution, max-pooling and RELU activation. This is followed by a dropout layer. This comprises a typical processing layer for a fixed output size. We have applied output size of 32, 64, 128 and 256 being followed by a fully connected layer of size 512. The output layer consists of as many output as there are number of labels (17). Each neuron's coefficients are computed during training using the training data. Basically, training is done through back propagation algorithm with gradient descent. Following figure depicts the multiple layer of processing that takes place.



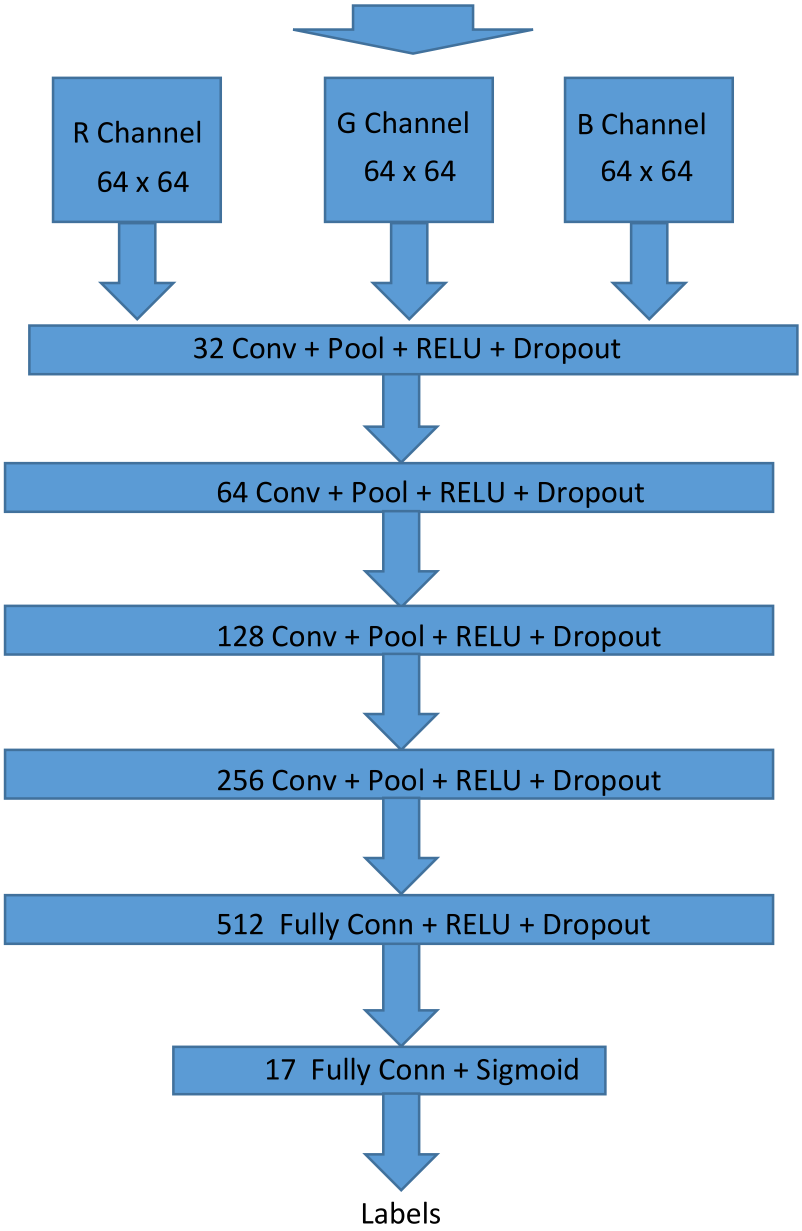
Convolutional Networks (ConvNet) uses mainly three type of layers – CONV (convolutional) +POOL (pooling), FC (fully-connected) and Classification layer. Image goes from Input layer through these processing layers finally to output or classification layers.

Input: At the very start of processing we have images with sizes as 256x256. Since our computation capability is limited, we have resized the images to 64x64. The pixel values are normalized between 0 and 1 (by dividing with 255). There are 3 channels of input data for R, G and B.

Convolutional + RELU+ Pool + Dropout Layer: This is the filtering layer where we apply 2D convolution of dimension 3x3 over the image area where each filter will compute a dot product between their weights and a small region they are connected to in the input. We have Convolutional layers using small filters (3x3) and stride S = 1 and 'same' padding that results in padding the input such that the output has the same length as the original input. This results in volume of output [64x64x32] as there are 32 filter to begin with. The output of each element is applied RELU (Rectifier Linear Unit) activation function. POOL layer will perform a down-sampling operation along the spatial dimensions (width, height). Random dropout is applied with a fixed retain-rate to aid regularization. We have a repetition of this (Convolution + RELU + Pool + Dropout) layer with varying number of filters starting with 32 and then 64, 128 and 256.

Fully Connected (FC) Layer: This layer will reduce the size of input data to the size of classes that the CNN is trained for by combining output of CNV layer with different weights. Each neuron at the output of the CNV layer will be connected to all other neurons after weighted properly, Similar to CNV layer, weight of these taps in FC layer is found though back propagation algorithm.

Classification Layer (CL): This is the final layer of the CNN that converts the output of FC to probability of each object being in a certain class. We have chosen 'sigmoid' as the final activation.


R Channel
64 x 64
G Channel
64 x 64
B Channel
64 x 64
32 Conv + Pool + RELU + Dropout
64 Conv + Pool + RELU + Dropout
128 Conv + Pool + RELU + Dropout
256 Conv + Pool + RELU + Dropout
512 Fully Conn + RELU + Dropout
17 Fully Conn + Sigmoid
Labels

For implementation we have used Keras library (https://keras.io) and TensorFlow (https://www.tensorflow.org/) backend to implement each layer. For computation we have used GPU NVidia GeForce GTX 1060 installed as a graphics card on a Linux desktop.

Refinement

Threshold Calculation: The output labels are not 1 or 0 but output of sigmoid function which can range from 0 to 1. To arrive at a label, we have to compare the output to a predefined threshold and arrive at a decision about presence of a label. We have adopted an algorithm that will choose the threshold which will maximize the F1 Score computed with that threshold. We have found a threshold of 0.2 (for all labels) provides maximum F1 score (0.93274).

Results

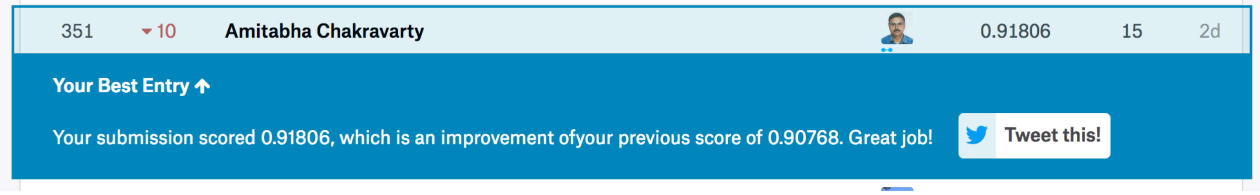
Model Evaluation and Validation

For the learning we have optimized the cross-entropy of the predicted vs actual label-vector. Data is analyzed in batches and best models is check-pointed. We have used K-Fold technique of split and validation to separate training set from validation set and rotate this process multiple times over the whole training set.

We have experimented with parameters like 'learning-rate', 'pooling frequency', convolution strides and k-value, number of layers etc. to arrive at the optimal model as shown above.

We have obtained F1 Score of 0.93274 and accuracy of 0.57313 while iterating the model over the whole training set.

The testing dataset was used to predict final score for the derived model and we have achieved 91.8%

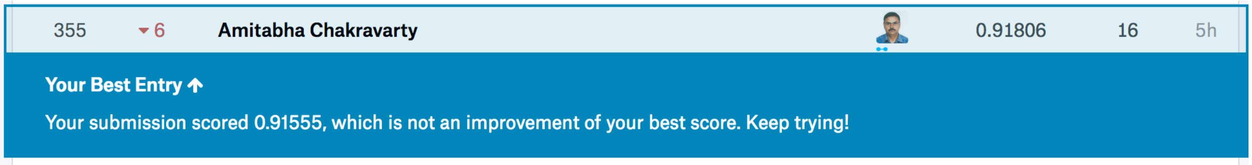


351	▾10	Amitabha Chakravarty	0.91806	15	2d

Your Best Entry ↑

Your submission scored 0.91806, which is an improvement ofyour previous score of 0.90768. Great job!

Model Robustness: We have subjected the input data to transformation like 'vertical-flip', 'horizontal-flip', height and width 'shift' etc and trained the model from scratch again. We subjected the model to same test data as before and obtained a score of 91.5 % (see below). So it proves that small perturbations (changes) in training data or the input space does not affect the results significantly.



355	▾6	Amitabha Chakravarty	0.91806	16	5h

Your Best Entry ↑

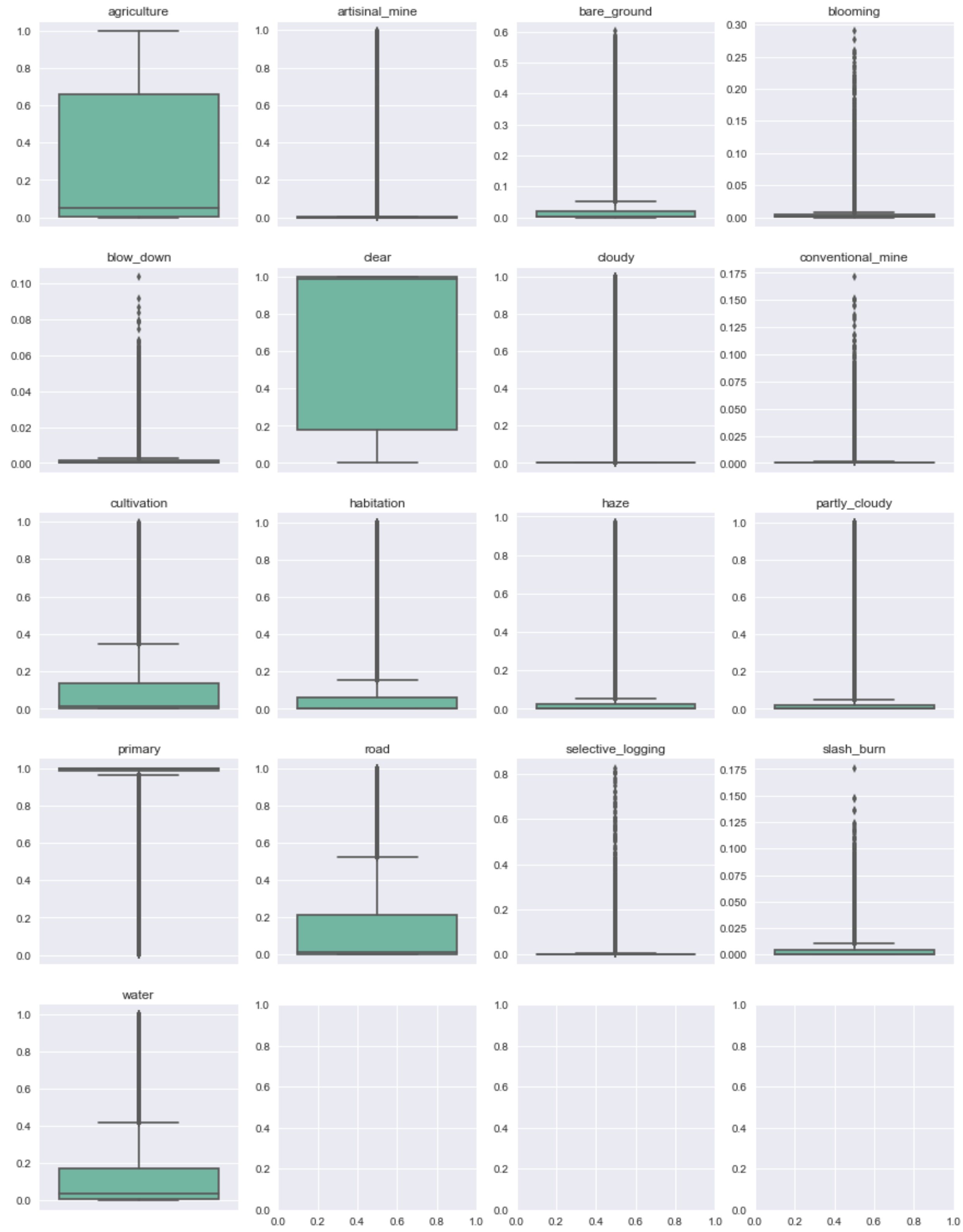
Your submission scored 0.91555, which is not an improvement of your best score. Keep trying!

Since the benchmark score goal was set as 91% our result of 91.8 % turned out to be stronger. However, there is still room for improvement but that depends on computing resource limitations.

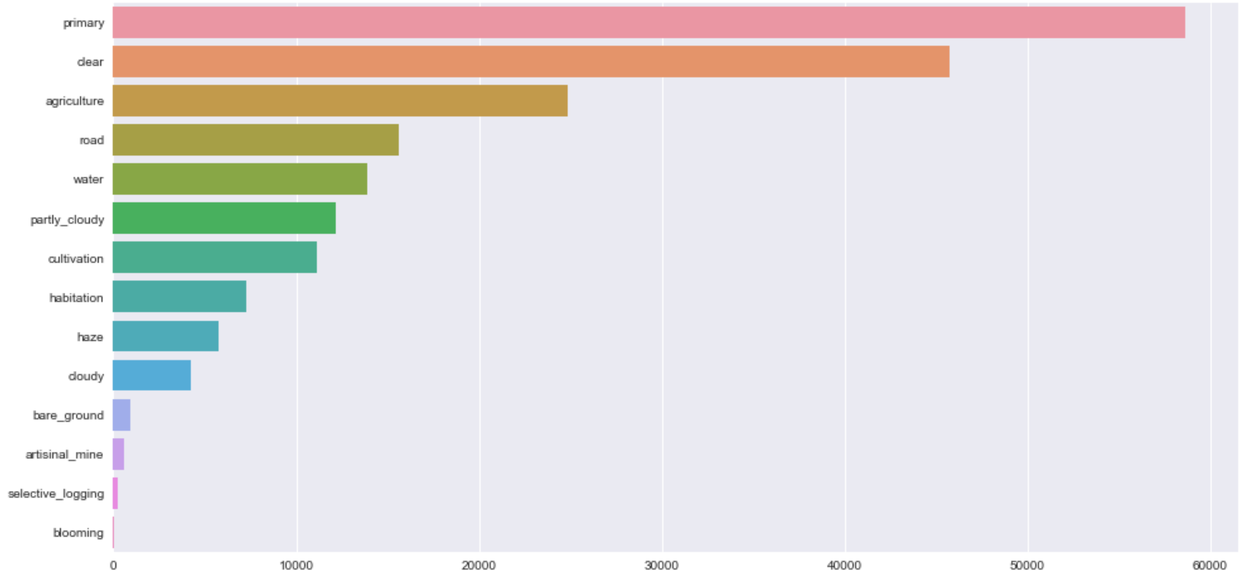
Conclusion

Free-Form Visualization

The output level for each prediction is between 0 and 1 and following box plot represents relative frequency of each label with median and inter-quartile range clearly indicated. For example, ‘primary’ is always present and it is hovering around 1.0. So is ‘clear’ label. ‘Cultivation’, ‘road’ and ‘water’ have moderate presence (0.1 to 0.2). These predictions are all in expected line.


agriculture
0.0
0.2
0.4
0.6
0.8
1.0
artisinal_mine
0.0
0.2
0.4
0.6
0.8
1.0
bare_ground
0.0
0.1
0.2
0.3
0.4
0.5
0.6
blooming
0.00
0.05
0.10
0.15
0.20
0.25
0.30
blow_down
0.00
0.02
0.04
0.06
0.08
0.10
clear
0.0
0.2
0.4
0.6
0.8
1.0
cloudy
0.0
0.2
0.4
0.6
0.8
1.0
conventional_mine
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
cultivation
0.0
0.2
0.4
0.6
0.8
1.0
habitation
0.0
0.2
0.4
0.6
0.8
1.0
haze
0.0
0.2
0.4
0.6
0.8
1.0
partly_cloudy
0.0
0.2
0.4
0.6
0.8
1.0
primary
0.0
0.2
0.4
0.6
0.8
1.0
road
0.0
0.2
0.4
0.6
0.8
1.0
selective_logging
0.0
0.2
0.4
0.6
0.8
slash_burn
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
water
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

The distribution of predicted labels are also evident from bar plot that is drawn below. Here the maximum length is occupied by ‘primary’ followed closely by ‘clear’. Then we have moderate presence of ‘agriculture’, ‘road’ and ‘water’. Important entries are seen at the bottom of the chart. The ‘selective logging’ and ‘blooming’ are detected and are also in line with expectation with their lower spread. This is also seen in the bar plot obtained from the input training data.



Reflection

The good part of using ConvNet is that it fits the image data fairly well and with a choice of good number of layers the classification turns out to be pretty accurate. However, there is always the contradiction on how much accuracy we can achieve with the given limitation of computing power. With 6GB GPU card we have been successful in processing 64x64 size of images but it failed miserably with Out of Memory error when image size was increased to 128x128.

Another improvement that we tried was to augment input data set with transformed images like ‘vertical/horizontal flip’, ‘height/width shift’ and so on. We have used ImageDataGenerator of Keras library which turned out to be pretty slow in fitting the model in real-time. Finally we tried one offline generation of entire training set and then running the K-Fold training on the generated images. However this gave almost same result as before (slightly lower in fact).

Improvement

There is still scope of improvement as we can increase the image size with better GPU card. We can also try running inline generation of image data using ImageDataGenerator on a more powerful machine. We can even think about using a better model like ‘Inception’ for this project which of course will take more processing power. With the current limitation of computing power and the given data size the current ConvNet model turned out to be a good compromise with effective score of 91.8% on Kaggle Leaderboard.

References

- Kaggle Competition - Planet: Understanding the Amazon from Space
https://www.kaggle.com/c/planet-understanding-the-amazon-from-space

- Convolutional Neural Networks (CNNs / ConvNets) - by Andrej Karpathy (karpathy@cs.stanford.edu) - http://cs231n.github.io/convolutional-networks/

- https://en.wikipedia.org/wiki/Gradient_descent

- https://en.wikipedia.org/wiki/Stochastic_gradient_descent

- https://en.wikipedia.org/wiki/Backpropagation

- https://www.tensorflow.org/