Planet: Understanding the Amazon from Space

Bernardo Casares Rosa[1]

[1]bcasares@stanford.edu (06191000)

March 2018

Abstract

The project represents a solution to the multi-class classification competition from Kaggle: "Planet: Understanding the Amazon from Space" using convolutional neural networks (CNN) with Tensorflow.

1 Introduction

Deforestation contributes to reduced biodiversity, habitat loss, climate change, and other devastating effects. Understanding the location of deforestation and human activity on forests can help governments and local authorities to respond quickly and effectively.

The problem that will be investigated is the Kaggle data-set from planet lab: "Planet: Understanding the Amazon from Space". Kaggle Competition.

Planet lab is the largest constellation of Earth-imaging satellites and the objective is to correctly label 256 x 256 satellite images from the Amazon with several labels from atmospheric conditions, land cover, and use.

The current best model is a CNN with 16 blocks. For each block, the architecture is the following: 3x3 conv -> batch norm -> relu -> 2x2 maxpool. The last two layers are fully connected layers and the loss function is a modified loss function.

2 Previous Work

The project presented in the report below has been previously solved by other Stanford University students taking CS231N. Their work served as a starting point for the project. [5, 6].

Additionally, since the data-set is available to online, other people has done similar work in the past, and their experience was useful when trying different architectures for the project.

The Kaggle blog has an interview with the winner of the competition, and it was used as reference. [2].

3 Data-set

The data-set consists of 40,479 training images with labels and 61,192 test images with no labels (both in TIFF and JPG format). Since there is no way to validate the test samples (competition is closed), the training images and labels were divided into 90 %train 5% development and %5 test sets and used for the project. The TIFF images have lower

quality, therefore, only the JPG images were used.

Each image is of size (256, 256, 3), with the channels representing R, G, B.

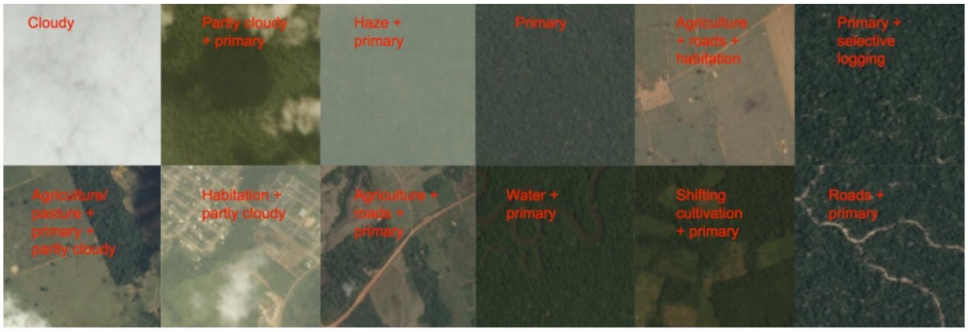

Figure 1: Example of Labeled Images (from the competition website)

3.1 Labels

Most of the images have several labels. However, there is no guarantee that the labeling is correct for all the images; scenes may either omit class labels or have incorrect class labels (detailed in the data description of the competition).

A histogram of the label frequency can be seen below.

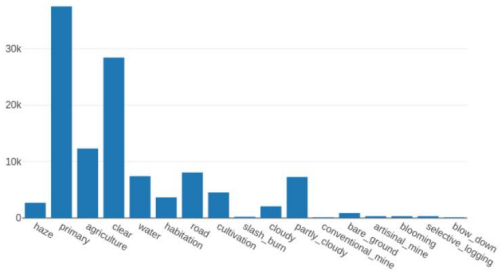

Figure 2: Histogram of Label Frequency (starter code Kernel [3, 4])

As can bee seen in Figure 2, the data-set is unbalanced. In many cases, people could solve the unbalance by performing data augmentation for the labels that appear less frequently. However, for the Amazon Rain-forest Data-set, when an image has an infrequent label, it is likely that the image also has a frequent label, making it almost impossible to reduce the unbalance.

In order to proceed with the analysis, it is important to know if the labels are co-related to each other. The heat-map below shows what percentage of the X label also has the Y label.

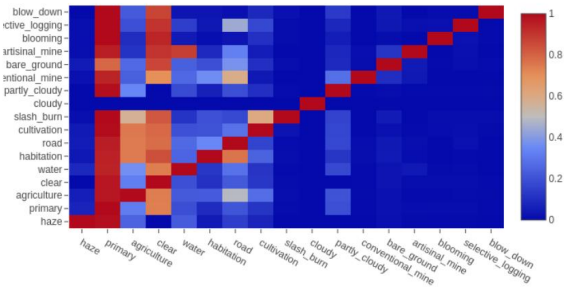

Figure 3: Co-occurrence matrix (starter code Kernel [3, 4])

The label "primary", which is shorthand for primary rain-forest, has the highest proportion of labels.

It is interesting to note that each image should have exactly one weather label, but the land labels may overlap. The weather labels are: clear, partly_cloudy, haze, and cloudy, and the land labels are: primary, agriculture, water, cultivation, habitation. The co-occurrence matrices plotted as heat map can be seen in the appendices.

4 Approach

The first step for this project was to build a basic CNN that would be able to train the model. Once a basic model was working, the next step was to perform data augmentation, try different architectures, fine tune and evaluate performance.

4.1 Basic Model

The basic model for this problem is the following:

1. Re-size the images from 256x256 to 64x64 for ease of learning (larger images take longer to train).
2. Build a simple CNN with the following architecture: 3x3 conv -> batch norm -> relu -> 2x2 maxpool.
3. Add two fully connected layers at the end.
4. Calculate the loss with sigmoid cross entropy for the 17 classes.

The basic model was trained trying to optimize accuracy.

4.2 Data Augmentation

Data augmentation included flipping, rotating, and transposing images with random probability.

Other data augmentation techniques, such as adding brightness and saturation to the images, were tried but some techniques did not improve performance.

After the basic model was working, and data some augmentation techniques proved to be effective, a more complex model was build. The main difference between the more complex model and the basic model was that several blocks were added to the CNN, where each block represents: 3x3 conv -> batch norm -> relu -> 2x2 maxpool. Additionally, batch normalization and momentum, and Xavier initializer for the weights, were used.

The results of the analysis with a batch size of 32, 10 epochs, 16 channels, using batch norm and momentum ($\beta = 0.9$) for different learning rates are the following:

Table 1: Initial Results

	Acc	Loss
$\alpha = 10^{-4}$	0.947156	0.136272
$\alpha = 10^{-3}$	0.950818	0.127826
$\alpha = 10^{-2}$	0.949481	0.129836

The initial results seem promising. However, for the Amazon Rain-forest Data-set, the most important evaluation metric is the F2 score. The F2 score evaluation metric will be described below.

4.3 Metric - The F2 Score

The F2 score is a way of combining precision and recall into a single score – like the F1 score, but with recall weighted higher than precision. The kaggle competition (when active) used the F2 score as an evaluation metric.

The F2 evaluation is as follows:

$$F2 = \frac{5}{N} \sum_{n=1}^{N} \frac{P_i R_i}{4P_i + R_i}$$

where P_i and R_i represent the precision and recall for each example. However, since Tensorflow calculates the precision and recall automatically, this project uses a slight modification of the F2 metric, defined as follows:

$$F2 = \frac{5PR}{4P + R}$$

where P and R represent the average precision and recall over all training examples.

A visual description of precision vs recall can be seen in Figure 4

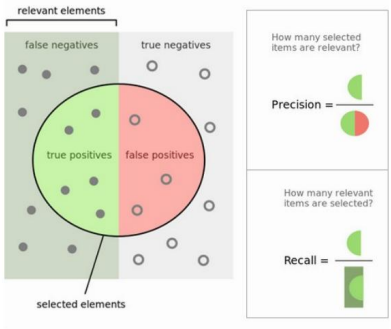

Figure 4: Precision vs Recall

One of the challenges of optimizing for the F2 score is that lower losses don't necessarily lead to higher F2 scores. Therefore, the models not only need to predict the label probabilities, but also select the optimum point to determine whether or not to select a label given its probability.

F2 penalizes false negatives more heavily than it penalizes false positives.

4.4 Loss and Regularization

Once the model was evaluated on the F2 score there was a significance decrease in performance. For that reason, the next technique was to change the loss function of the model.

The idea was to create a loss that would penalize miss-classified predictions of positive labels more heavily. Therefore the new loss function was defined as follow:

$$\mathcal{L} = -y * log(sigmoid(\hat{y})) * weight -$$

$$(1-y)log(1-sigmoid(\hat{y})) + \lambda|w|^2$$

$weight > 1$ increases the recall.

Note that regularization was also added, but not always used. Regularization was added because on some experiments there was a significant gap between training, testing, and evaluating results.

Before applying the new loss function, the result from the more complex model with the a learning rate of 10^{-3} has the following recall, accuracy, and F2 score:

Loss	Precicion	Recall	F2
0.1278	0.8813	0.8187	0.831

The results with the new loss function will be described in the next section.

5 Results

The synthesis of some of the experiments can be seen in the Table 2.

It is important to note that several values for the weight of the loss were attempted, and it is evident that the redefined loss was able to improve the recall significantly. However, there is a trade-off between precision and recall to increase the F2 score.

Finding the optimum point for the trade-off between precision and recall yield our best model.

Table 2: Synthesis of Results

	Experiment	recall	accuracy	precision	loss	F2
1	Preprocess	0.805	0.952	0.898	0.123	0.822
2	Loss weight$_{1.5}$	0.758	0.935	0.841	0.659	0.773
3	Loss weight$_{4}$	0.888	0.935	0.766	0.721	0.860
4	Loss weight$_{16}$	0.961	0.896	0.624	0.897	0.867
2	Loss weight$_{16}(BestRegularization)$	0.929	0.868	0.722	0.937	0.884
5	Loss weight$_{64}$	0.967	0.813	0.473	1.370	0.800
6	Resnet$_{18}$	0.882	0.902	0.656	0.977	0.825
7	Resnet$_{34}$	0.888	0.862	0.556	1.016	0.793
8	Resnet$_{50}$	0.869	0.900	0.653	1.103	0.815
3	Resnet$_{101}$	0.940	0.924	0.705	0.774	0.882

*The results in the table above correspond to different fine tuned models, and only the best performing results are on display. Disclaimer: Additional models were trained after the poster presentation, so the table of results may vary. Specifically, better parameters of regularization for certain models were attained.

5.1 Resents

Once we had our best performing model in place, an additional effort was made to try to compare the results from our architecture to architectures that have proved to be useful in image recognition. For this purpose, several configurations of resents were tried and the results are also in Table 2 [1] [7]. Hyper-parameter tuning was only performed for $Resnet_{101}$ which gave us the second best result.

6 Discussion

When fine tuning a CNN, or any other deep learning model, it is important to remember what is the metric that we are trying to optimize.

In our specific case, although the initial results seem promising, when changing the metric the performance decreased.

Changing the loss function significantly improved the results. However, there was a trade off that was achieved after several iterations and fine-tuning.

From previous work, Resnets appeared to have worked better for other people in similar work. Nonetheless, in my case, simpler models outperform more complex models. Additional work could be needed to fine tune the resnets, or sometimes simpler models might work better.

7 Future Work

Future work for this project includes trying and fine-tuning different CNN architectures such as DenseNets, and revisiting Resnets. There is always room for improvement in deep learning problems.

Acknowledgements

The author wish to sincerely thank the CS230 course staff for for all the support during all the stages of the project. A special thank to Lucio Dery for all the guidance and support.

[1] The architecture for the resents came from an implementation found online, and was adapted for this project [7].

References

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

[2] Kaggle Blog from Winner: http://blog.kaggle.com/2017/10/17/planet-understanding-the-amazon-from-space-1st-place-winners-interview/

[3] Started Code from Kaggle: https://www.kaggle.com/bcasares/getting-started-with-the-data-now-with-docs/edit?unified=1

[4] Started Code from Kaggle: https://www.kaggle.com/anokas/data-exploration-analysis

[5] Jeff Pyke. Understanding the Amazon from Space with Convolutional Networks. http://cs231n.stanford.edu/reports/2017/pdfs/914.pd

[6] Sneh Kudli, Steven Qian, Benjamin Pastel. Kaggle Competition: Understanding the Amazon from Space. http://cs231n.stanford.edu/reports/2017/pdfs/913.pd

[7] Resnet from Github: https://github.com/tensorflow/models/tree/master/o

8 Github Repository

https://github.com/bernardocasares/CS230_Final_Project