Understanding the Amazon Rainforest using Neural Networks

Naveen Ketagoda, Christian Koguchi, Niral Pathak, Samuel Sunarjo

{nketagod, ckoguchi, nlpathak, ssunarjo}@eng.ucsd.edu

Abstract

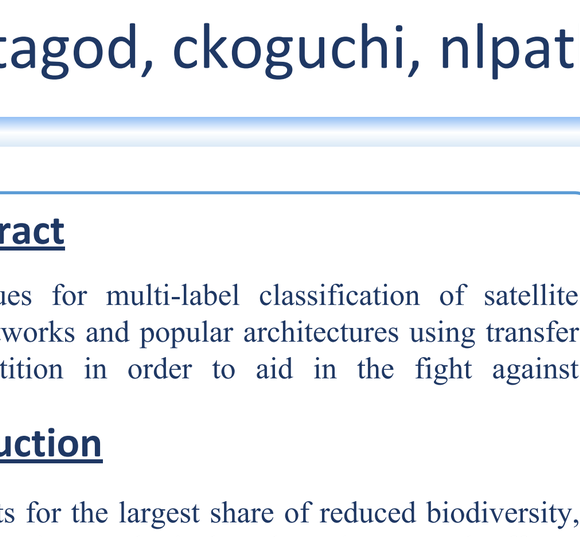
We apply modern machine learning techniques for multi-label classification of satellite imagery. Using custom convolutional neural networks and popular architectures using transfer learning, we participate in a Kaggle competition in order to aid in the fight against deforestation.

Introduction

Deforestation in the Amazon Rainforest accounts for the largest share of reduced biodiversity, habitat loss, climate change, and other devastating ecological and environmental effects. Understanding and tracking human deforestation and ecological changes over time can better help environmentalist and government efforts in responding to both human and natural forest loss. We attempt to aid in this effort by using satellite imagery data of the Amazon Rainforest to track the expansion of human deforestation efforts using modern computer vision techniques. The ability to label satellite data will allow us to "better understand where, how, and why deforestation happens all over the world - and ultimately how to respond" [1].

Data

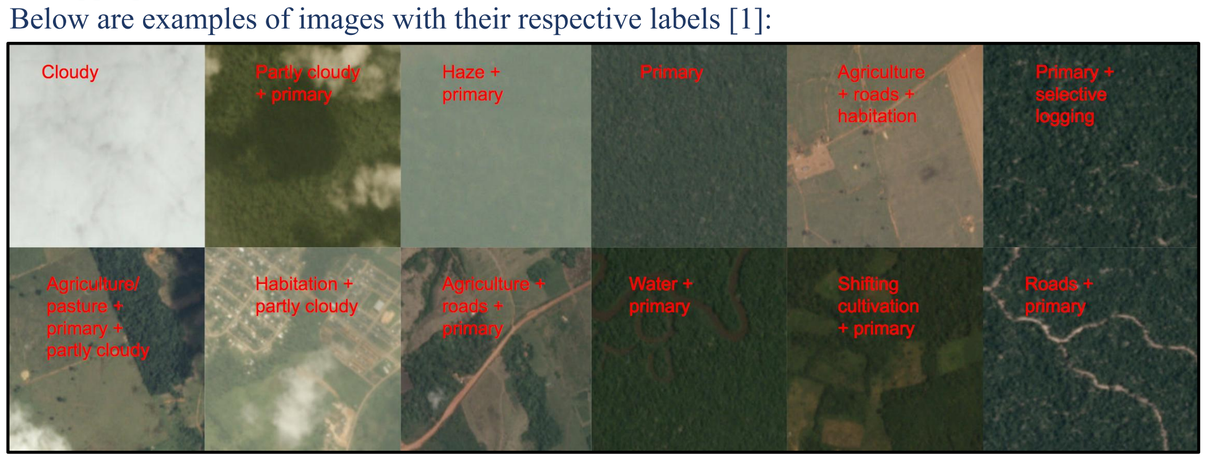
Satellite imagery is obtained from a Kaggle Competition sponsored by the company Planet [1]. The data, collected between January 1, 2016 and February 1, 2017, centers around the Amazon basin which includes Brazil, Peru, Uruguay, Colombia, Venezuela, Guyana, Bolivia, and Ecuador. Data is provided in a 3-channel JPG format (RGB) and a 4-channel GeoTiff format (BGRNir) which adds an extra near-infrared channel. We work with JPG images exclusively due to size constraints (average size: JPG = 15KB; TIFF = 538 KB). The training dataset consisted of 40479 labeled files, and the test dataset consisted of 61191 files. The images are 256 x 256 x 3 pixels [1].



There are 17 non-exclusive labels in the dataset which are provided in a CSV file with respective training image filenames:

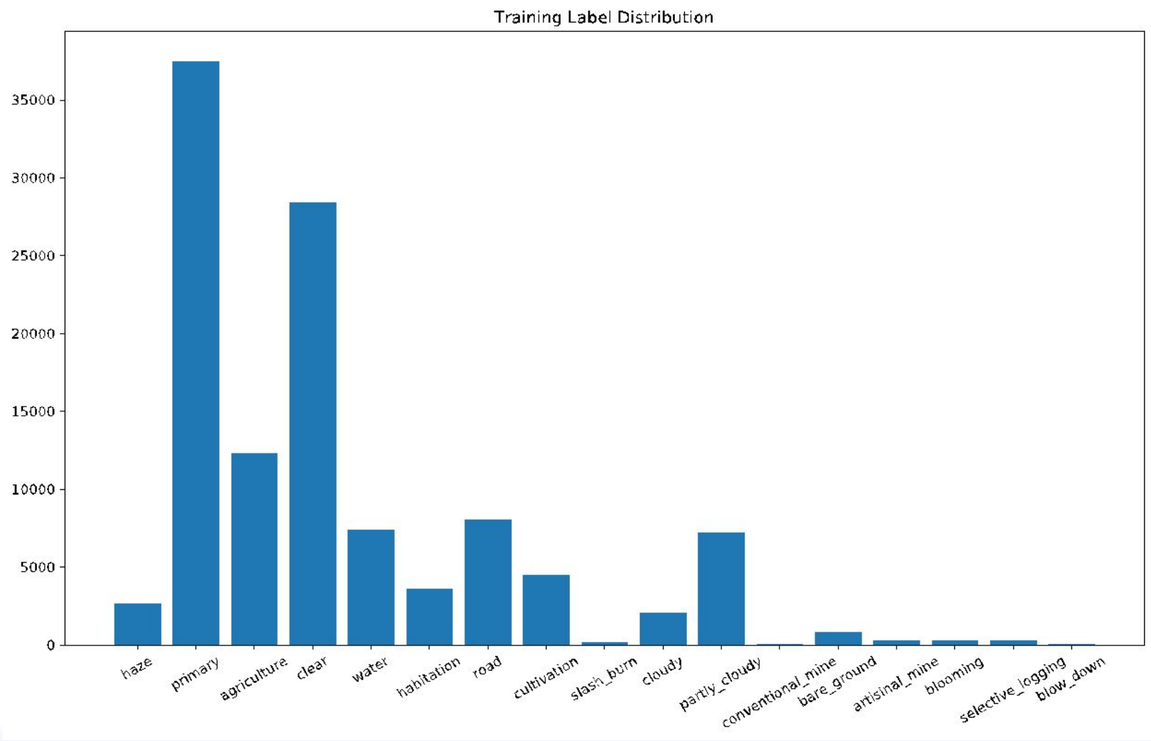
- 4 weather: clear, partly cloudy, haze, cloudy
- 6 land: primary, agriculture, water, cultivation, habitation, road
- 7 rare: slash burn, conventional mine, bare ground, artisinal mine, blooming, selective logging, blow down

Below are examples of images with their respective labels [1]:



We explore the data and find extreme class imbalance amongst the training labels. This will make training significantly harder. Classification accuracy is not meaningful, so evaluation metric is based on the F-Beta score which emphasizes precision (p) and recall (r) combining statistics about true positives (tp), false positives (fp), and false negatives (fn). Kaggle uses $\beta=2$.

$$F\beta = (1+\beta^2)\frac{pr}{\beta^2 p + r} \qquad r = \frac{tp}{tp+fn} \qquad p = \frac{tp}{tp+fp}$$

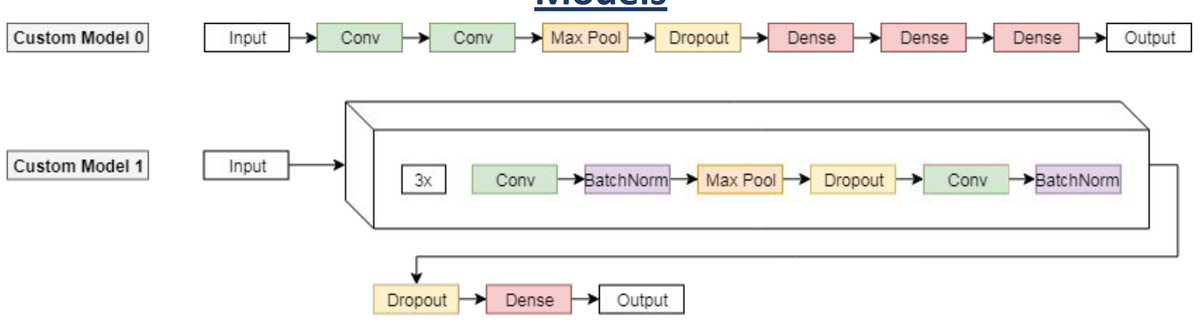


Objective

We use neural networks in order to approximate the function f for the multi-label classification problem $y = f(x)$ where x is a flattened input image, and y is a binary 17-dimensional vector where the binary value describes the presence of a label as the image descriptor. In order to learn this function f, we use Convolutional Neural Networks (CNNs). They are a "specialized kind of neural network for processing data that has a known, grid-like topology" [2]. CNNs have extremely practical applications in modern computer vision tasks. We can leverage the grid-like topology of the data and use the images as inputs into the CNNs in order to output the desired labels. However, to quickly train a robust CNN that generalizes well to unseen data, we preprocess our data as described below:

- Input images: resized to 96x96x3 normalized to [0,1], zero-mean, randomly rotated, randomly shifted up/down/left/right, randomly flipped horizontally/vertically
- Input labels: converted to 17-dimensional binary vector with 1 if label present else 0
- General: batch size = 128, training/validation split = 80/20, patience = 3 (prevents overfitting)
- Use binary cross-entropy loss and sigmoid activation on final layer

Models



We designed 2 custom networks (named custom_0 and custom_1) that are heavily inspired by Kaggle discussions. Custom_0 is a shallow network that trains quickly, and custom_1 is a deeper network that adds Batch-Normalization. Furthermore, we explore transfer learning with VGG16 and Xception which are popular architectures trained on the ImageNet dataset. Transfer learning is a technique that leverages learned weights in order to accelerate training and acts a feature extractor [3]. We perform fine-tuning with the architectures which means that we do not freeze the layers, but instead we allow weights to update throughout the network with backpropagation.

Results

Networks	Trainable Parameters	Validation F2	Kaggle F2
Custom 0	6,510,289	0.87931	0.87766
Custom 1	826,257	0.87987	0.87736
Transfer - VGG16	14,749,527	0.91042	0.90845
Transfer - Xception	21,120,319	0.92201	0.91918

Discussion

Note that Kaggle's top scores are 0.93. Recognize that the fine-tuned networks performed the best as expected. They are pre-trained and have already learned high-level features from ImageNet whereas our custom networks learned from scratch. Also, our custom_1 network performed nearly as well as custom_0 despite having a fraction of the parameters. This suggests that convolutional layers are in fact more suited for this computer vision task compared to dense layers (which exist in custom_0).

Future Work

With more time and resources, we would explore the use of the fully sampled dataset and could test our models on the 4-channel GeoTiff format. Furthermore, ensemble methods in which we train several models and have them vote on the output of testing examples typically leads to better generalization. This type of model averaging is also a form of regularization [2]. We would also like to explore Hinton's Capsule Networks as an alternative to CNNs and their applicability to this domain.

References

1. Planet: Understanding the Amazon from Space. https://www.kaggle.com/c/planet-understanding-the-amazon-from-space.
2. Ian Goodfellow,Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press. 2016. http://www.deeplearningbook.org.
3. A. Karpathy. Transfer Learning. https://cs231n.github.io/transfer-learning/.

Acknowledgements

We would like to acknowledge the neural network frameworks Keras, TensorFlow, and PyTorch. We further thank Amazon Web Services for free student tier for GPU computation which allowed us to train and test models quickly. Thanks to Professor Peter Gerstoft and Teaching Assistants Mark Wagner, Paolo Gabriel, and Nima Mirzaee for their continued encouragement and advice in the completion of this project.

UC San Diego

RESEARCH POSTER PRESENTATION DESIGN © 2015
www.PosterPresentations.com