Land-Use and Deforestation in the Brazilian Amazon

Using Satellite Imagery and Deep Learning for Environmental Conservation

Cameron Bronstein
DSI Capstone

Hi all.

The topic of my project was to Land Use and Deforestation in the Brazilian Amazon.

The goal is to show the application of deep learning for environmental conservation.

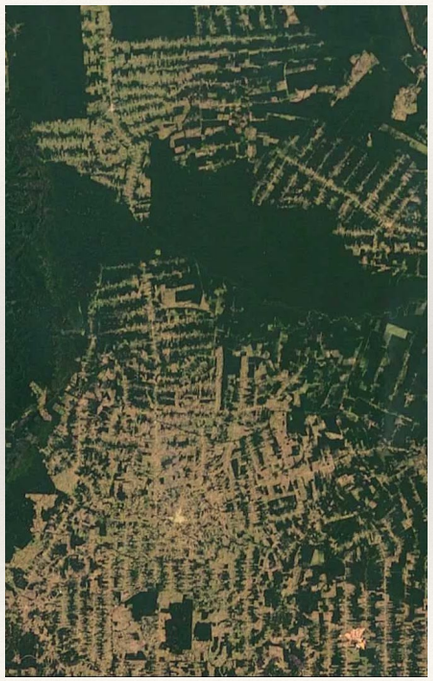
Agenda

- Introduction
- Data Collection with Planet API
- Exploratory Data Analysis & Image Preprocessing
- Model Building & Evaluation
- Recommendations & Future Steps

Today I give you an introduction to the problem, talk about Data collection, exploratory data analysis and image preprocessing, modeling techniques, model performance, and my recommendations for future steps.

Introduction

- The Amazon rainforest is an eco-climate system vital to regulation of global climate, via oxygen production and carbon sequestration.
- Facing huge deforestation pressure from illegal logging and cattle ranching.
- Over 20% of the land cover has been deforested.
- Nearly 8000 square km logged from Aug 2017 - July 2018



Fishboning in Rondônia

The idea of an ecological-climatic (or acclimate) system is most applicable to the Amazon. It is vital for the regulation of global climate, generating huge amounts of oxygen and carbon capture through photosynthesis.
However, it is under severe pressure of deforestation - (five times the size of London in less than one year)
Because the total land area is so massive (> 5 million sq km), illegal logging can be hard to detect.

Introduction

- Planet, a satellite imaging company, has the largest fleet of satellites in the world.
- In Brazil, Planet partners with a conservation consultancy (SCCON) to monitor land coverage in the Amazon with remote sensing, providing high resolution imaging that can distinguish human-caused from natural forest loss.
- Currently, no in place methods exist for automated classification of images into a multitude of land use and land cover categories.

Planet has over 300 machines image the entire planet, every day.

In Brazil, Planet provides images to monitor local ecosystems. Yet currently, they do not have a method for classifying images into categories based on land use or land cover.

Images must be reviewed by peopled through mundane visual ID and data entry.

Introduction

Problem Statement:

- Using Satellite Imagery from Planet, build a method to classify images by weather, land use and land cover types.
- Effective models will correctly predict images with multiple land-use types.

Applications:

- Real-time identification to direct conservation organizations and local government to stop illegal land use activities.
- This would automate the currently tedious human-verified image labeling and data entry system.

The problem is to build an automated method that can identify

Data Pipeline with Planet API

- To generate images, the first task is to specify and Area of Interest (AOI).
- Create compound filter: AOI and time period.
- Specify item type (Satellite type) and assets (analytic, visual, etc.)
- Different Item types have different resolutions and contain different associate metadata.

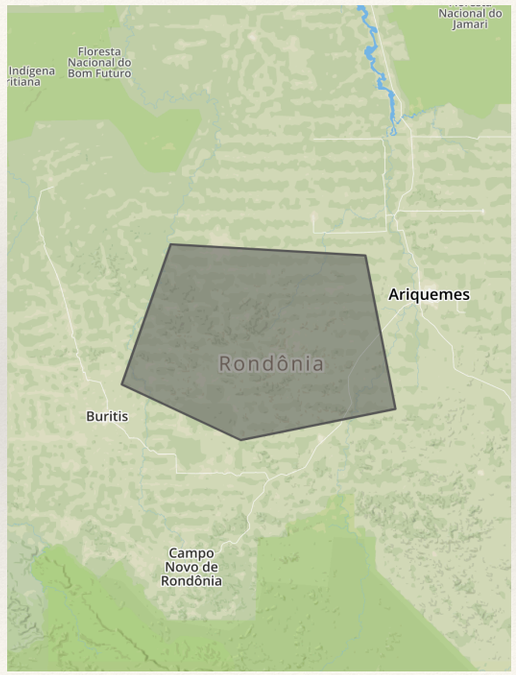
Now I am going to talk about how to generate data from satellite imagery.
Planet maintains a database of their daily images organized by geography and time.

So to generate images, I could specify an Area of Interest and create a filter on geography and time. I could then choose from numerous satellites and image types depend on the product or associated metadata I want.

With metadata, you can calculate metrics like Vegetation Index, which is informative for long term ecological monitoring.

Data Pipeline

- AOIs can be really large!
- API requests include images that contain fragments that lie within the AOI.
- Images need to be augmented, rotated, cropped significantly to be processable by a neural network.



AOI in Rondônia, Brazil

AOIs can be really large (Hundreds of KM - sq'd)!

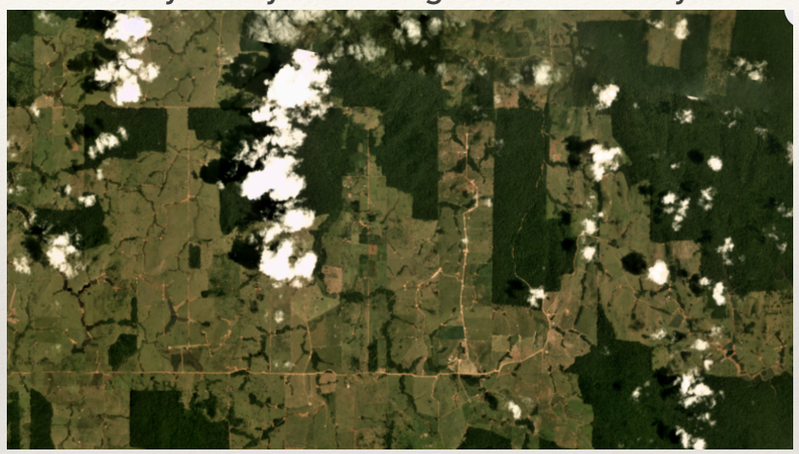
For that reason, the api does not generate an exact clip of your AOI - but rather, images associated with that AOI, or containing fragments of your AOI.

Thus, images need to be augmented and cropped to some standardized format before they can be used to train a model or evaluated.

I did use any pre-generated image for these reasons. Also, the images need to be labeled.

Image Labels

Party cloudy - Road - Agriculture - Primary



Land Use and Land Cover Labels

Primary Forest - Road - Water - Agriculture - Habitation - Bare Ground - Cultivation - Blooming - Artisanal Mine -Selective Logging - Slash and Burn - Conventional Mine - Blow Down

Weather Labels

Clear - Haze - Cloudy - Partly Cloudy

So that brings us to the images. I generated this image on planet's visual
"explorer", and it is a good example of what the project dataset looks like.
In this image, you can clearly see partly cloudy - road - primary forest - agriculture.
So our images all contain labels like the one above - these labels were entered by people, and are necessary for training. You can see some of the categories on this slide. Images only have one weather label, but can have multiple land use labels.

The Data

Provided by Planet and Kaggle

Dataset

- 40,000 training images
- 60,000 test images

Image Specs

- Red: 610 - 700 nm
- Green: 500 - 590 nm
- Blue: 420 - 530 nm
- Near Infrared: 700 - 1000 nm

Resolution

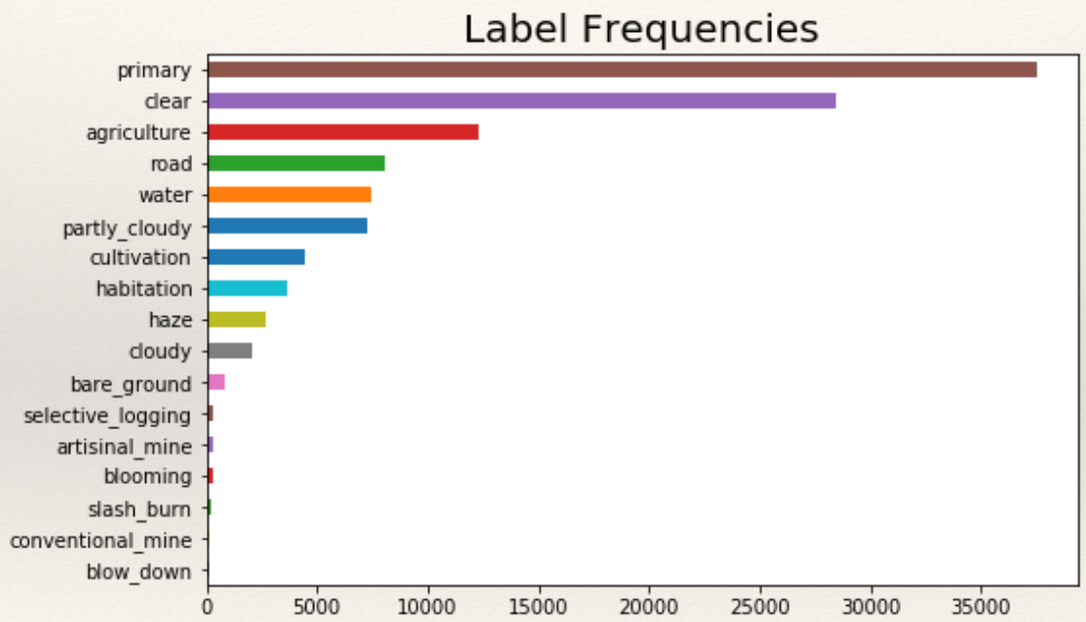
- 256 x 256 Pixels
- 3 meter resolution
- 768 m^2 per image

That brings us to the data. For the purpose of this project - Planet prepared a dataset of over 100,000 images that were preprocessed for analysis and posted on Kaggle.

The dataset is divided into a labeled training set and an unlabeled, large validation set.
The images are from the Planetscope satellite fleet and are in a geotiff format. This format preserved geolocation data in the image rendering, which has applications down the line.
The images contain 4 analytic colorbands (Red, Green, Blue, and NIR), which all contain different information about the images response to different ranges of light wavelengths.

The images are 256 pixels square, so at 3 m resolution per pixel, represent land areas of about ¾ sq km.

Understanding the Labels



This brings us back to the labels.
This is a multi-label classification problem, such that each datapoint (image) can contain multiple labels.
And as you can see, there is pretty extreme imbalances in the dataset. This presents a challenge such that it will be more difficult to predict the classes that are infrequently seen.

It also poses a problem with evaluating result - models could be accurate even if they failed to predict every “rare” label. So we need to use a different metric (f - score), that includes a weighted penalty for falsely predictions (more on that later).

Image Pre-Processing

- Neural networks “learn” by recognizing patterns in image features. For each image, a feature is a pixel.
- However, if a certain weather or land type only appears in a certain region of an image, the neural network will not recognize the exact same type if it is in a different location in the image.

True Image

Vertical Flip

Each image has 256^2 features

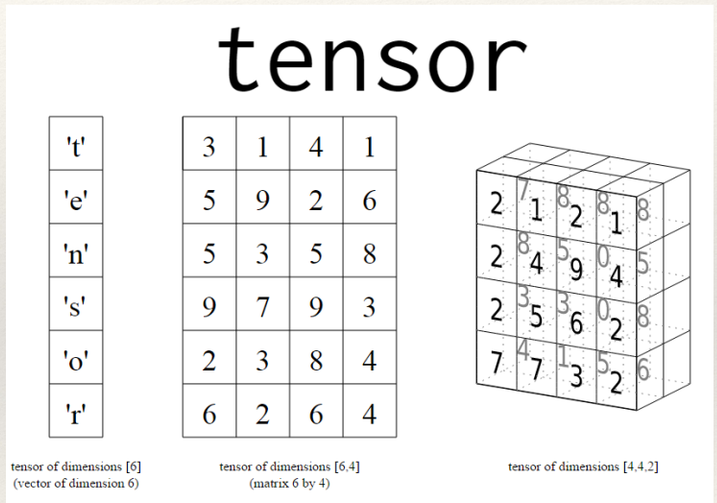
Neural networks “learn” by recognizing patterns in image features. For each image, a feature is a pixel.
However, if a certain weather or land type only appears in a certain region of an image, the neural network will not recognize the exact same type if it is in a different location in the image.

In an ideal circumstance, you could show a network the same image with features in every possible location or size in the image, and it would detect the same feature each time.

We can approximate this by randomly choosing from a suite of augmentations when feeding the training images into the network.
e.g. show flipped image
Flipping (vertical/ horizontal), Rotating + Filling

Final Image Conversion

- Inputs to the model: Image converted to tensor
- Tensor - a multi-dimensional array.
- Each image is a tensor (128 x 128 x 3).
- Only included RGB bands.



Regardless of the flipped, images are all processed in the same way.
This is how the neural network processes image data.
Images are tensors in the shape of their pixelated area, time the number of color bands they contain. For modeling, I used the RGB bands. Including the fourth band has computation costs and is not always shown to improve results (plus ImageDataGenerator only takes 3…)

The data gets randomly augmented and sent into the model in batches, so you can think of each batch as a tensor with additional dimension.

CNN Explained

- Convolutional Neural Networks (CNN) are best for image processing.
- Convolutional layers apply a filter to the image and transform its values and dimensions.
- As the network is trained, the calculations are adjusted for each transformation so that the network converges on the appropriate output for each possible prediction type.

- Convolutional Neural Networks (CNN) are best for image processing.
- CNNs are often deep network contain multiple layer types.
- Convolutional layers apply a filter to the image and transform its values and dimensions.
- Densely connected layers connected every input node to ever output node. These are best place near the end of the network.
- These transformations allow the network to process the images more efficiently, and learn certain characteristics across each layer as the layer approaches an output.
- As the network is trained, the calculations are adjusted for each transformation so that the network converges on the appropriate output for each possible prediction type.
- This way, they can generalize well across a complexity of inputs.

Train - Validate - Test

- We only have so many truly labeled images, so we have to make full use of them to train and test our model.
- Randomly splitting the labeled data into training and testing allows building and validating multiple types of models before making final predictions.
- Train the model across multiple epochs (full pass of the data through the network).

Cross validation (we only have so many truly labeled images, have to make full use of them) on the training set.

Iteratively and randomly splitting sets into training and testing can give you an aggregate prediction of model performance that accounts for randomization bias.

Can’t do typical stratified sampling because some class-combinations only have 1 sample.

Modeling Evaluation

- Model outputs 17 “probabilities” – one for each label category.
- Thresholds allow us to convert to binary values.
- If probability > threshold: prediction = 1
- How do we determine the right thresholds?

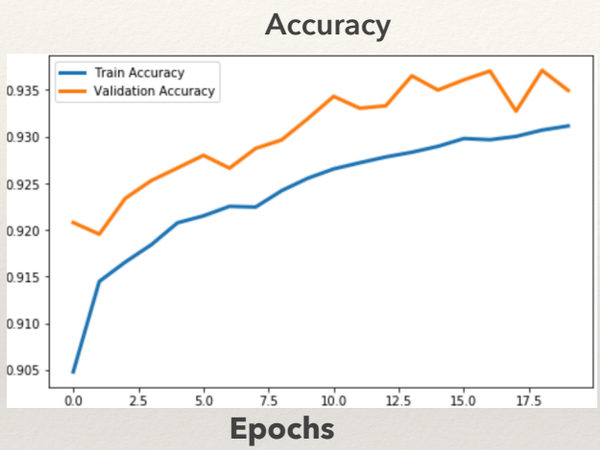
Should thresholds be the same for each category?

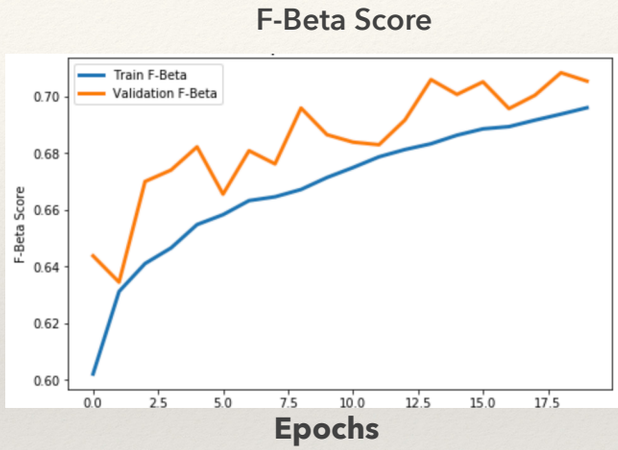
Probably not, since some categories have so few images…

So, we train, evaluate thresholds variations with validation data, converge on a threshold.

Then, retrain the network on the full dataset and make a final test prediction.

Model Results





- Model performance continues to increase through successive Epochs.
- F-Beta score follows similar trends to accuracy, but is reduced and suffers from instability.
- Train score are more stable but lower than validation.

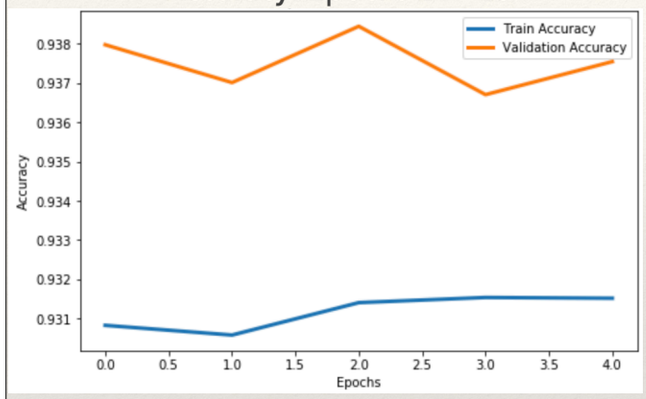
Initially, we see upon first

What is loss? Loss is a measure of error

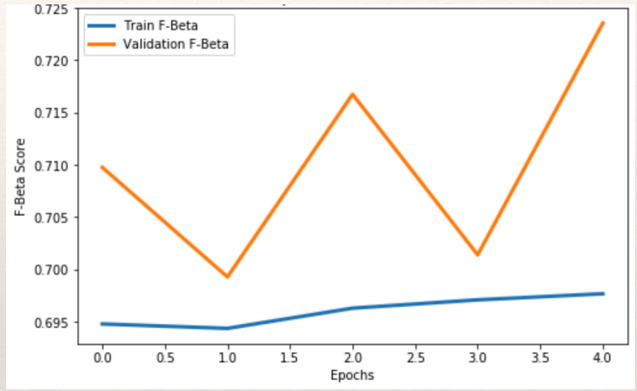
Why is model underfit? Too much dropout?

Model Results

Accuracy: Epochs 21 - 25



F-Beta Score: Epochs 21 - 25



- Upon additional Epochs (re-initialized with saved weights), we still see slight improvements in training performance but increased instability.
- This is evidence of "overfitting"

Final Accuracy: 0.938

Final F-Beta:
0.846 Validation
0.833 Test | 0.834 with Thresholds

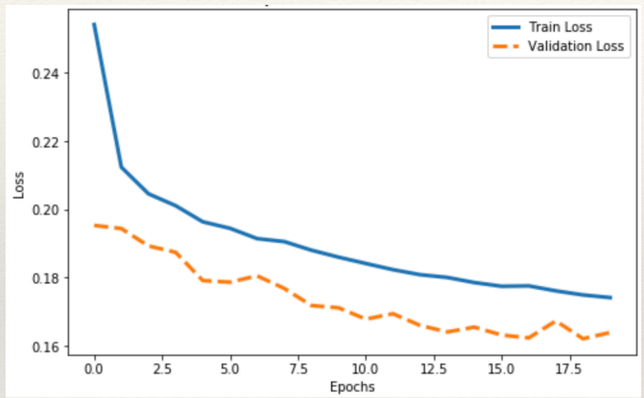
It's not exactly overfitting, but the network is past convergence.

Could do more batch normalization … or add residual layers

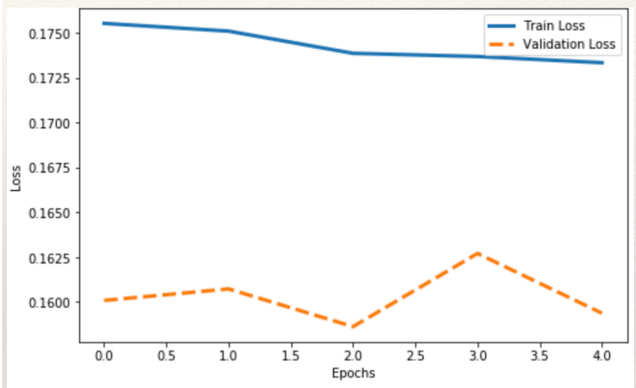
The original dataset has more "noise" in that it has more augmentation..

Model Results

Loss: Epochs 1 - 20



Loss: Epochs 21 - 25



- Loss follows an inverse pattern to our other metrics.
- Shows general improvements with increasing epochs and instability after ~ epoch 22.

Initially, we see upon first

What is loss? Loss is a measure of error

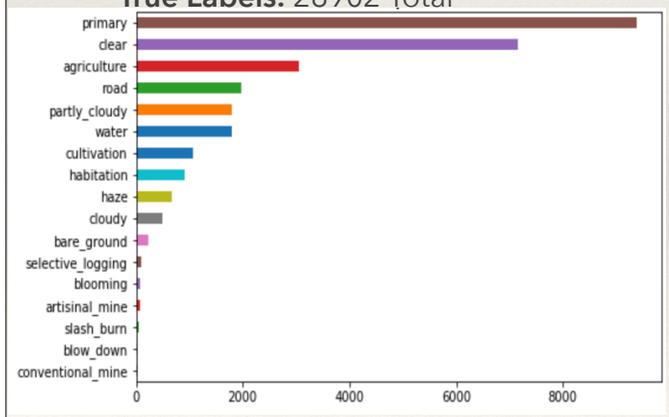
Why is model underfit? Too much dropout?

Validation Labels Investigation

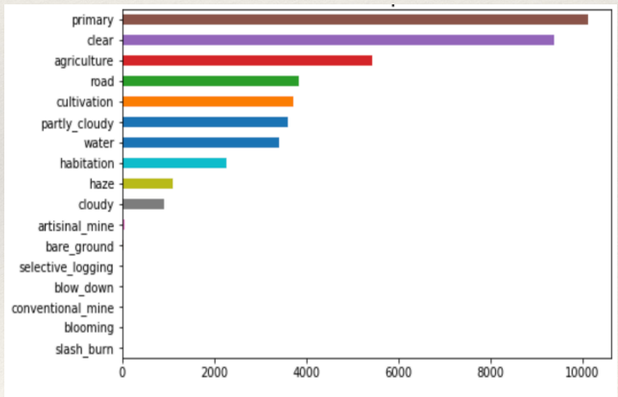
Validation Data: 10120 Images

- The model over-predicts labels
- This may account for the instability in F-Beta Score.

True Labels: 28902 Total



Predicted Labels: 43914 Total



Initially, we see upon first

What is loss? Loss is a measure of error

Why is model underfit? Too much dropout?

Recommendations

- Invest in powerful GPU work-center.
- Continue training model with bootstrapped images (varying augmentations of low frequency label combinations).
- Increasing Network complexity.
- Consider practicality of performance - resource tradeoff.
  - Better performing models = more time & money.
- Partner with scientists for future conservation applications.

Ensemble multiple neural networks that are better at predicting different particular labels.

Use linear regression to converge upon an optimal prediction for each label to find an aggregate predictions.

One regression for each label. Features are predictions for that label from each model. Then regression some intermediate value and that becomes the new probability for each datapoint.

Experiment with haze removal to improve predictions on images that are hidden behind the haze. Remove noise in images themselves.

Modify network to produce augmented images that paint each class a different color

Could then construct a larger map and quantify area of different features across broader geographical ranges.

Acknowledgments

- Matthew Brems: seamless introduction to CNNs
- Riley Dallas: introduction to batch processing.
- David Yerrington: troubleshooting linux, server issues and model.
- Tucker Allen: endless guidance and project scope adjustment suggestions to keep my vision within the bounds of planet earth.
- Isa Cuervo: presentation suggestions.
- All of my classmates for being awesome and helping me learn SO MUCH during the last 12 weeks!

Questions & Discussion

THANK YOU !

Model Structure

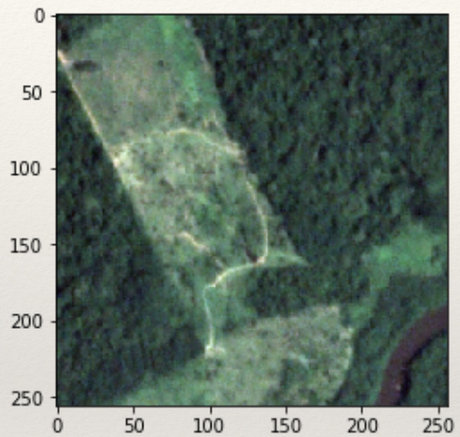
- C32 - Pool - C64 - Pool - Dense 128 - Dense 17
- Batch Normalization scales data to a .
- Dropout eliminates a random 50% of the nodes at each layer every epoch.
  - This reduces tendency for the neural network to overfit on the training data.

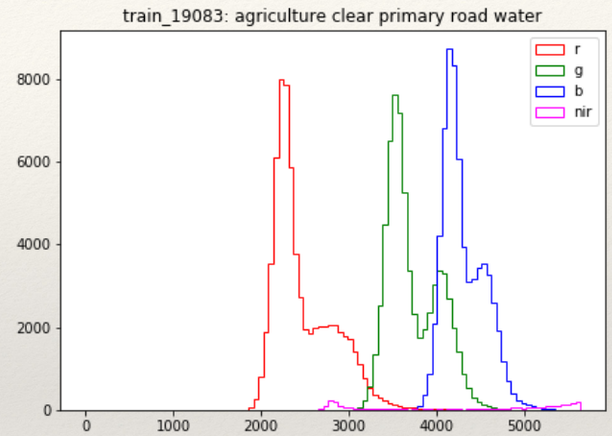
The network consists of two stacked convolutional layers with increasing size, with a densely connected layer before output nodes. This allows the network to account for increasing complexity after the initial layers do basic color and edge detection.
This network is like a mini-version of much deeper and more successful ne
Because we have 17 possible labels, our networks output layer is 17 nodes.

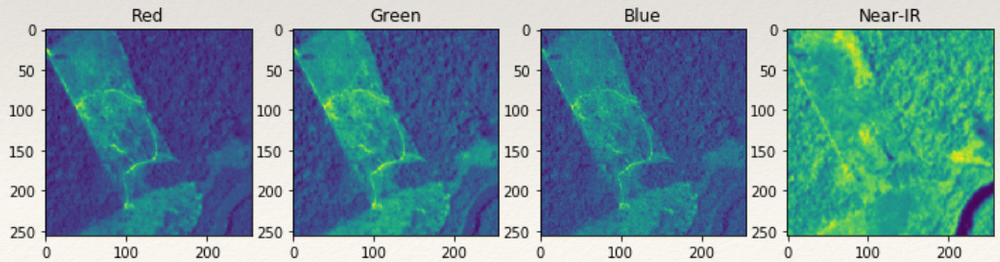
Batching

- Test data: 21 GB + Training data: 30 BG = OOM
- Need to do batch processing through and through, keeping consistent randomization across Kfold testing
- Thankfully, we have generators! Keras has a nice one to do all of this.

Understanding Image Data







Here, we graph the histogram of the four color bands. The histogram on the right depicts the number of pixels at each given digital number (conversion of color based on bit number). As you can see, there is a very low spectral response to IR (pink values are very sparse) - this is seen in the IR only band, with the intensity of color seen in the water.

This is valuable for detecting water (but alas, we don't use the four bands).

Modeling Process cont.

- Regularization techniques
  - Early stopping
  - Dropout
  - Batch normalization at multiple steps