DeepRootz: Classifying satellite images of the Amazon rainforest

Scott Longwell
Stanford University
Department of Bioengineering
longwell@stanford.edu

Tyler Shimko
Stanford University
Department of Genetics
tshimko@stanford.edu

Alex Williams
Stanford University
Neurosciences Program
ahwillia@stanford.edu

Abstract

The Amazon rainforest is a critical, yet highly threatened South American natural resource. The size and density of the forest make monitoring by land or low-air expensive and intractable. Recent advancements in satellite imaging techniques allow for rapid, high-resolution imaging of the entire forest. However, hand-labeling and monitoring of these images would be exceptionally expensive and error-prone. We present a deep convolutional neural network for labeling images of the Amazon rainforest with both atmospheric and terrestrial development labels. The model presented here is custom-built to work with four-channel (red, green, blue, near-infrared) images with a resolution of 3-5 meters per pixel. We discuss the challenges of training such a model and considerations for using the F2 metric as an accuracy statistic. We find that our model achieves an accuracy, as measured by the F2 metric, that is within five percent of the performance of the top model in the Understanding the Amazon from Space Kaggle competition.

1. Introduction

The Amazon rainforest is one of the planets most critical natural resources. The rainforest is one of the most biodiverse land areas on the planet. However, human encroachment on the forest threatens the habitat of many of the areas plant and animal species. Due to the interconnectivity of such an ecosystem, the effects of human development of the forest could have catastrophic effects.

In response to over-development and habitat loss, many of the surrounding nations have created special departments that work to monitor and limit human encroachment. However, the sheer size of the forest makes monitoring development from the ground or low-air intractable. One alternative to terrestrial surveillance techniques is observation by satellite. Unfortunately, given the resolution and scale of satellite imagery, hand-labeling features contained within the images is unrealistic for any single point in time, let alone for tracking development over time. Recently, the San Francisco-based company Planet Labs, Inc., which specializes in satellite imagery, launched a competition to algorithmically label satellite images with both atmospheric and terrestrial conditions, including labels pertaining to human activities such as mining and agriculture.

To overcome feasibility issues and efficiently label images on the fly, the Planet Labs team painstakingly labeled a training and test dataset of 256 square pixel chips. The training dataset was then released to the public for use in development of algorithmic labeling techniques while the test set was withheld for judging the techniques.

Successful algorithms correctly apply a single label for the atmospheric condition and at least one terrestrial label indicating the land utilization observed in the chip. By completing labeling computationally, the images of the rainforest can be categorized in near real time, greatly facilitating monitoring of human related activities. Because the consequences of illicit land use in the Amazon are frequently irreversible, rapid response is necessary to intervene properly and prevent the spread of development further into this irreplaceable resource.

Here we present a deep convolutional neural network for labeling images of the Amazon rainforest consistent with the rules outline in the Kaggle competition run by Planet Labs, Inc. Our model was constructed from the ground up and trained using the provided training dataset. The inputs to our model are 256x256 pixel, four channel images of the Amazon rainforest and the outputs labels of atmospheric and terrestrial conditions. Our model achieves an accuracy, as measured by the F2 statistic, that is within five percent of the top performing model in the competition.

2. Related work

With implications for national security [20], environmental monitoring [18], and conservation [3, 6], image segmentation and feature recognition in satellite images are active areas of research. In particular, the need for clean satellite imagery data suitable for training machine learning algorithms is so great that two large-scale datasets have recently been released [4]. Hopefully, datasets such as these

will spur development in the satellite imagery field, in a fashion similar the ImageNet dataset for object recognition [5].

2.1. Training deep networks

Extending depth usually improves the representational capacity of neural networks. However, as networks increase in size, they inevitably become more difficult to train. To address this problem, several strategies have been proposed. The first of these strategies is to use an initialization scheme that is congruent with an architecture of choice. Specifically, the initialized coefficients for all weights in the network should scale in such a way that vanishing or exploding signals are not propagated early in training. So-called Xavier initialization provides such a scheme that can be used with great success during the training of deep networks [7]. A second strategy to facilitate training a deep network is to implement residual layers. These layers serve to simplify the complexity of training by forcing the network to learn many smaller, simpler functions that accurately describe the residuals of an overarching function, instead of the whole function directly [8]. In tandem, the use of both Xavier initialization and residual layers represent a marked advance in both the speed and accuracy of deep neural networks.

2.2. Object recognition in aerial images

Efforts on satellite and other aerial image recognition tasks have already yielded dividends, with machine learning approaches able to recognize man-made features such as buildings and roadways [1, 14]. In the case of recognizing development patterns in rainforest imagery, this previous work is reassuring in that it proves man-made structures indicative of non-natural land use can be identified on an individual basis. To correctly label all the necessary images in the Planet Labs dataset, we need to extend this work to correctly recognize more general development patterns such as agriculture or mining.

2.3. Development pattern recognition

Beyond recognition of man-made structures and development patterns, convolutional neural networks have seen great success in the prediction of more abstract consequences of human development. For instance, Jean and colleagues were recently able to adapt a ImageNet trained network to accurately predict local poverty rates from satellite images of Africa [10]. This work indicates that network architectures, like those presented here, are capable of learning higher-order representations of the terrestrial development state.

In addition, pretrained networks, such as those trained on the ImageNet dataset have repeatedly demonstrated their value for such tasks [10, 13]. These networks perform better off the shelf, but tend not to incorporate the additional information present for spectra outside visible light (red, green, and blue channels).

Efforts have also been made to use deep convolutional neural networks to perform unsupervised feature extraction on satellite images [17]. These techniques, especially when performed with very deep networks, have been shown to outperform standard dimensionality reduction techniques such as PCA. Generally, the representational capacity of deep neural networks should be sufficient for the task of labelling the Planet Labs dataset.

2.4. Non-visible light imaging

Unlike other recognition tasks that rely on three channel images (red, green, and blue) many satellite images have more channels, corresponding to non-visible parts of the electromagnetic spectrum. These types of images are referred to as hyperspectral images, as they capture information from many narrow spectral bands, many from outside the visible light portion of the spectrum. These images have many uses from uncovering ancient, abandoned settlements [2] to mapping extent of seagrasses beneath the surface of the ocean [16]. In these applications, spectral information outside visible light is used to gather further information about the imaged area. Specifically, infrared spectra can be used to infer the density, age, and health of any vegetative ground cover. In fact, information from the red and near-infrared portions of the electromagnetic spectrum is enough to estimate the amount of chlorophyll present in imaged plants [15].

3. Dataset

Planet Labs, Inc. provided a training dataset for this competition of 40,479 16-bit, 256x256 4-channel tiff files (BGRN, for blue, green, red, and near-infrared). Unlike many other image recognition tasks that rely only on the visual red, green, and blue channels, the inclusion of the fourth, near-infrared channel, is likely necessary for top model performance. Vegetation, a key component of nearly all of the images, strongly reflects near-infrared light. Indeed the Normalized Difference Vegetation Index (NVDI), which is commonly used in remote sensing applications to determine the presence of green-vegetation, is defined as the difference between the near-infrared and red channel intensities divided by their sum. In contrast, most other image features suggestive of human development, such as roadways and buildings, tend to absorb light in the near-infrared spectrum. Therefore, the near-infrared channel is likely one of the most informative for labeling the images in the dataset.

Each pixel in these images represents an area on Earths surface of 3-5 square meters. Through a combination of crowdsourcing and expert analysis, these images were an-

Figure 1. Co-occurrence of atmospheric classes with terrestrial labels. Given an image had a row class, each element gives the probability that the image shares a corresponding column label.

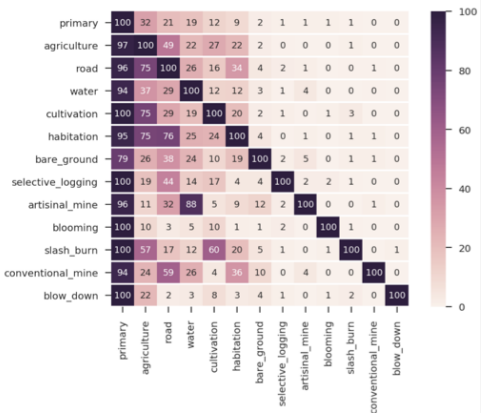

Figure 2. Co-occurrence matrix for terrestrial labels in the training dataset. Given an image had a row label, each element gives the probability that the image shares the corresponding column label.

notated with 17 tags, which indicate atmospheric conditions (specifically, cloud cover) as well as terrestrial conditions (e.g. mining, forestry, agriculture). These tags are assigned in a hierarchical fashion, with each image being assigned exactly one of 4 atmospheric classes and zero or more of 13 ground activity labels. In the case that an image is assigned a class of *cloudy*, indicating heavy cloud cover, no ground activity labels are applied.

Based on the annotation rules, there are $1 + 3(2^{13}) - 3 = 24574$ possible tag combinations. However, only 430 ($< 2\%$) are observed in the provided dataset. The clear atmospheric class is by far the most prevalent, while primary (for primary growth rainforest) labels over 90% of images. Both classes and labels exhibit a striking imbalance (Figure 2). Additionally, since each image can be tagged with more than one label, many labels are highly correlated (Figure 1). Imbalance and correlation within the labels in the dataset can present major problems during training as the network will observe some labels far more often than other labels. These issues can be mitigated either *a)* on the model's frontend by upsampling examples with rare labels or *b)* on the backend by weighting the loss higher for rare labels such that larger gradients are backpropagated.

Given the relatively small size of the dataset, we decided to perform basic data augmentation to increase the number of examples (see Methods section).

4. Methods

4.1. Preprocessing and data augmentation

The 40,479 labeled images were split 90-10 into a training set of 36,431 examples and a validation set of 4,048 examples. For the training set, a per-channel mean and standard deviation were calculated. Using this mean and standard deviation, all images were normalized as they were loaded into the model. During training, images were augmented using random 90° rotations and reflections. Whereas traditional images are usually oriented to align with gravity, satellite images have no obvious natural orientation, so we would expect 90° rotations and reflections to increase the effective dataset size 8-fold while not adding significant noise.

4.2. Basic convolutional network

As a general architecture template, we used a stem of several 2D-convolutional layers along with one strided 2D-convolution to reduce spatial dimensionality, then classified the flattened output of the stem with two fully connected layers to ultimately yield 17 score predictions which were mapped to values (0,1) using a sigmoid. During training, these 17 pseudo-probabilities were compared to the ground-truth labels via variations of a binary-cross entropy (BCE) loss function (see sub-section 4.4). The model was trained with the Adam optimizer [12] and a batch-size of 128. Weights were initialized with the Xavier-Glorot algorithm [7], batch normalization [9] was applied after all fully connected and standard convolutional layers, and inverted-dropout [19] (0.2 probability of dropout) was applied to the first fully connected layer. After an exploratory screen, we found a high-performing architecture and hyperparameter combination which is summarized in table 1 as model 008. The full model architecture is shown in section 8.

4.3. Deep residual network

While the basic convolutional architecture performed reasonably well, it struggled to overfit the training data. Furthermore, the single strided convolution (which reduced spatial dimensionality) did not cull the dimensionality of the flattened representation significantly: of the 21M parameters in the basic model, 99.8% were from the first fully connected layer. Thus, we implemented a deep, higher-capacity architecture with residual layers to ease model optimization and two additional strided 2D-convolutions to further reduce dimensionality. Every residual layer (ResLayer) was

composed of two 2D-convolutions, each with 2D-batch normalization and ReLU activation. Following the batch normalized output of the second convolution, the original input to the ResLayer was added before the final ReLU activation (such that each ResLayer learned a *residual* relative to its input). Overall, our deep residual network architecture was similar to that of the basic convolutional network, except the 2D-convolutions of the stem were replaced with several ResLayers. To classify the flattened output of the stem, two fully connected layers (including first layer 0.2 dropout) and BCE loss were used. With the reduced dimensionality of the flattened representation, the model size decreased greatly from 21M to 1.55M parameters (with 84.7% of the parameters in the first fully connected layer).

We initially trained the model with Adam, and observed that the model was prone to overfitting. This model is summarized as model 013 in table 1. Based on literature [11, 21], we opted to retrain this architecture with stochastic gradient descent with momentum and observed that the model was less likely to overtrain. This model is summarized as model 015 in table 1. The full model architecture is shown in section 8.

4.4. Loss function and imbalanced data

All model architectures output 17 values corresponding to the 17 labels associated with each image; a sigmoid nonlinearity was applied to ensure all final scores ranged between zero and one. A binary cross-entropy (BCE) loss independently to all atmospheric and terrestrial labels. Specifically, the BCE loss for each feature j is:

$$\ell_j = \frac{1}{N} \sum_i \left(t_i^{(j)} \log o_i^{(j)} + (1 - t_i^{(j)}) \log(1 - o_i^{(j)}) \right)$$

where $t_i^{(j)} \in \{0, 1\}$ is the ground truth label for image i on feature j, $o_i^{(j)} \in (0, 1)$ is the model estimate, and N is the number of images in the training set or minibatch. The full loss was the average BCE across features: $L = \frac{1}{17} \sum_j \ell_j$.

Since many of the terrestrial features have few positive examples, we experimented with re-weighting the loss function such that images with positive labels for rare classes had a larger gradient during training. Specifically, we define the weighted BCE loss as:

$$\ell_j = \frac{1}{N} \sum_i \left(\frac{t_i^{(j)} \log o_i^{(j)}}{f_j + \gamma} + \frac{(1 - t_i^{(j)}) \log(1 - o_i^{(j)})}{1 - f_j + \gamma} \right),$$

where $f_j = \frac{1}{N} \sum_i t_i^{(j)}$ is the proportion of images with a positive label for feature j and $\gamma > 0$ is a hyperparameter that controls the relative weighting of positive and negative examples. (As $\gamma \to \infty$ the weighted BCE loss approaches the unweighted BCE loss up to a constant scaling factor.)

The unweighted BCE loss equally penalizes false positive and false negative predictions from the model, while the weighted BCE loss differentially penalizes these errors based on the frequency of the label in the training data. Positive training examples ($t_i^{(j)} = 1$) are weighted more heavily for very rare labels ($f_j \approx 0$), and negative training examples ($t_i^{(j)} = 0$) are weighted more heavily for very common labels ($f_j \approx 1$). We found that a relatively large value for γ produced better performance in all models (Model 015 used $\gamma = 1$), but was generally not critical for performance.

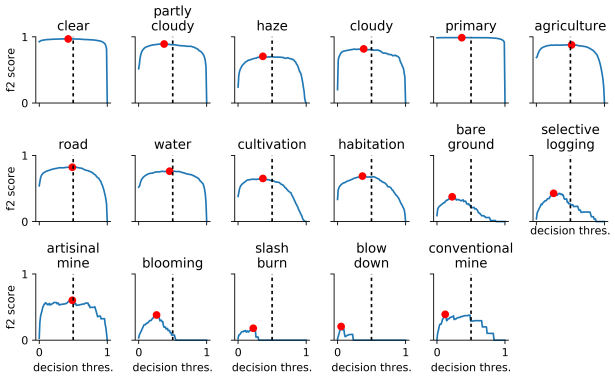

Figure 3. How decision thresholds for each feature effect the F2 score (model 015). The red point for each feature indicates the optimal threshold (determined from training set) that was used on the validation and test sets.

4.5. Optimizing Decision Thresholds

The weighted and un-weighted BCE loss functions were simple to implement and interpret. However, this loss was not directly related to the mean F2 score that was used to compare models in the Kaggle competition. Thus, at the end of each epoch we determined the decision threshold for each feature that maximized the F2 score on the training set. This is visualized for each feature in Figure 3; while a naive threshold of 0.5 (black dashed line) performed reasonably well when the γ hyperparameter was appropriately tuned, we obtained modest improvements in performance by tuning the decision threshold of the model post-training.

5. Results

After only ten epochs of training, the simple model architecture (section 4.2) achieved a validation and training F2 score of 0.88; suggesting that even relatively simple models that utilize the near-IR channel can perform quite well on this task. ction We trained the deep residual architecture (section 4.3) more extensively and achieved a validation F2 score of 0.9 (model 015). We submitted this model for evaluation on the Kaggle test set, and received a similar score of 0.898 suggesting that our model was not overfit.

-	model properties				performance (*mean F2*)		
model	architecture	params	optimizer	epochs	train	val	test
008	ConvNet	21.0M	Adam	10 / 10	0.88	0.88	-
013	ResNet	1.55M	Adam	22 / 53	0.94	0.89	-
015	ResNet	1.55M	SGD+M	104 / 135	0.91	0.90	0.898

Table 1. Summary of model architectures, hyperparameters, and performance. The epochs column denotes the epoch of maximal performance followed by the total number of epochs.

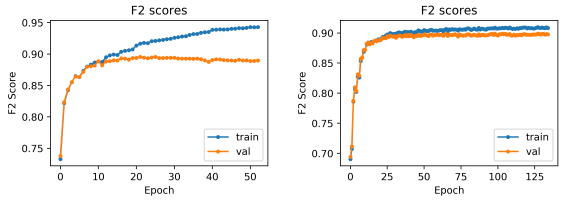

Figure 4. Mean F2 score on the training and validation image sets. *Left*, model 013 optimized with Adam. *Right*, model 015 optimized with SGD.

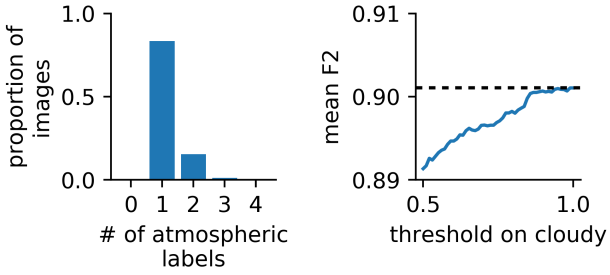

Figure 5. Assessing atmospheric labels in model 015. *Left*, number of atmospheric labels assigned to each image. *Right*, effect of nullifying other labels when a *cloudy* label is applied above a particular threshold.

5.1. SGD generalized better than Adam

We originally fit all models with Adam, an adaptive gradient method that rescales parameter updates by estimates of lower-order moments of the gradient [12]. However, doing so resulted in substantial overfitting — the F2 score of our model became higher on the training set than on the validation set (Figure 4, *left*). Recent work has suggested that Adam identifies solutions that are less generalizable (i.e. overfit) than SGD and other non-adaptive methods [21], especially with a large batch size [11]. Switching to SGD with momentum marginally improved performance on the validation set, and greatly decreased the gap between the training and validation set (Figure 4, *right*).

5.2. Ignoring atmospheric label constraints did not drastically hurt performance

One concern about our approach is that our models do not incorporate two constraints of the true problem. First, each image contains exactly one atmospheric label while our models can output multiple atmospheric labels. Second, any image labeled as *cloudy* has no other labels while our model can attach additional labels to these images. We therefore examined whether our models produced unrealistic predictions with respect to these constraints.

Our models assigned a single atmospheric label to the majority of images in the validation set, despite this not being a hard constraint (Figure 5, left). Interestingly, no images were assigned a single atmospheric label, and very few images were assigned more than two atmospheric labels. We found that replacing the multi-label predictions with a single atmospheric label (the one with maximal score) modestly decreased performance on the validation set and never increased performance. This makes sense because false positives are less harshly punished than false negatives, so assigning two atmospheric labels is a reasonable way for the model to hedge its bets.

Next, we investigated the effect of nullifying all labels other than *cloudy* for images labeled as such by our model. Again, since the F2 score does not penalize false positives as harshly as false negatives, we found no performance improvement. Figure 5 (right) shows the effect of nullifying other labels when the model outputs a score for *cloudy* that is above a threshold between zero and one half. Setting this threshold at one — i.e. the upper bound of model output — was optimal and matched the baseline performance (dashed line). Thus, it was never advantageous to use the *cloudy* label to nullify other labels. Overall, these results show that our choice to treat each label independently in the loss function was not problematic for our performance.

5.3. Qualitative analysis

We qualitatively characterized the output of our best performing model by plotting the co-occurence matrix of terrestrial labels (Figure 6); which closely matched the observed correlations in the ground truth dataset (Figure 2). This is notable since our loss function was applied independently to each label, meaning the model was not directly constrained to produce these correlations.

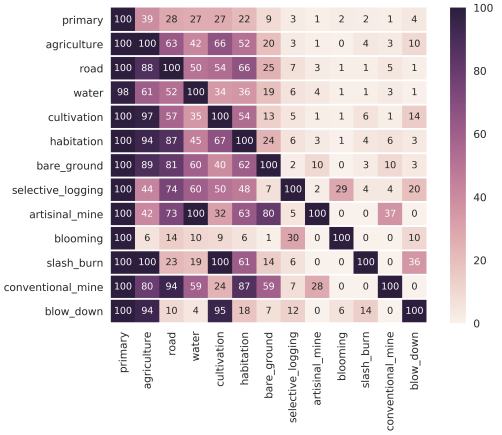

Figure 6. Co-occurrence matrix of label predictions, model 015. (Compare with Figure 2).

We then visually inspected images that our model incorrectly labeled (Figure 7). We were pleased to observe that many of the false negative images were not easily identified by our (admittedly non-expert) visual inspection. In addition, many false negative images contained significant cloud cover. We also examined and verified images that our model correctly labeled, which tended to be more recognizable (data not shown).

6. Discussion

Satellite image datasets are particularly amenable to use in machine learning contexts. These datasets provide rich (multi-channel) and diverse images that can be used to monitor land use through time. However, given the size and resolution of these datasets, annotation must be automated to monitor resources in near real time. We have leveraged the latest advances in convolutional neural networks to construct a deep learning model that is able to rapidly and accurately annotate satellite images of the Amazon rainforest.

Our algorithm has potential implications for environmental monitoring, where knowing and understanding human encroachment into the forest is key to halting illegal action and aiding in conservation efforts. By labeling images algorithmically, throughput is greatly increased and, consequently, response time can be decreased. We are able to discern a great variety of human activities, which has the potential discriminate between legal activities and illegal activities, since the visible signatures are different. Specifically, the dataset contains two labels that describe the mining operations: *conventional_mine* and *artisinal_mine*. Conventional mining describes large-scale mining operations that are more likely to be legally permitted. In contrast, artisinal mining describes smaller scale mining operations that are more likely to be illegally installed in the forest.

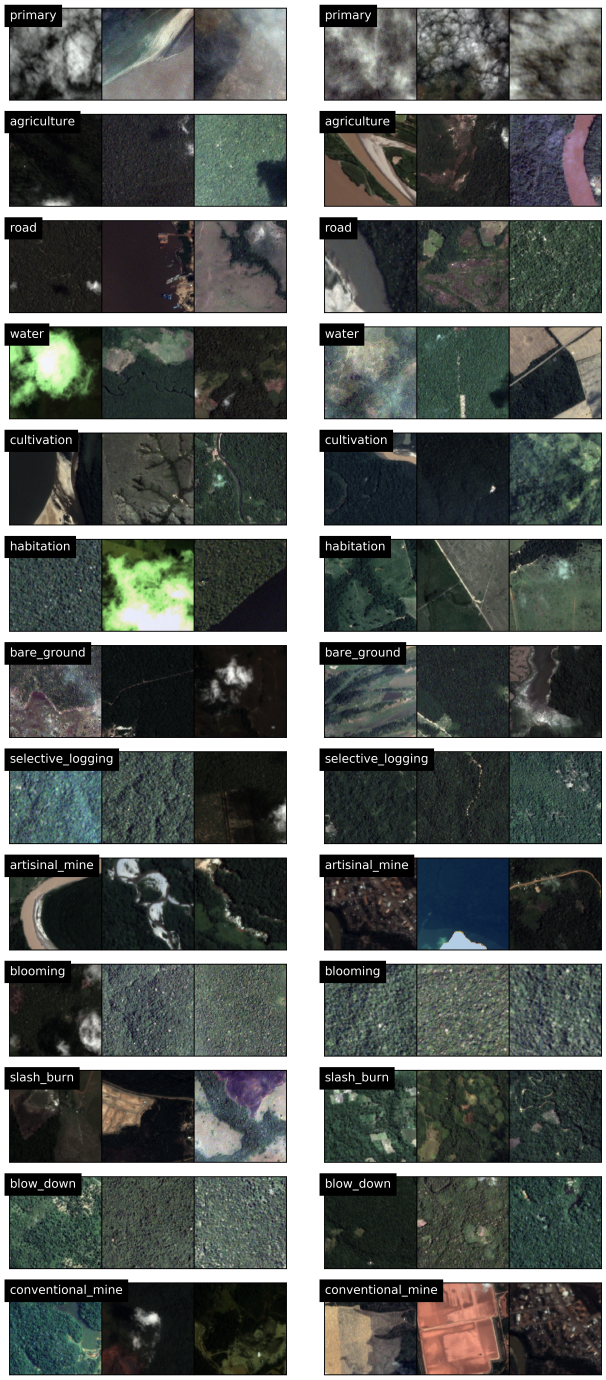

Figure 7. *Left*, false negative labels given by model 015 for each atmospheric and terrestrial feature. *Right,* false positive labels given by model 015 for each atmospheric and terrestrial feature.

Our model could be used direct law enforcement and conservation officials to only illegal operations, while allowing the conventional, legal operations to continue.

A primary concern when attempting to annotate satellite images is cloud cover. Cloud cover obscures the view of the ground, which can change the labels applied to a specific image. Our model considers each of four possible classes for cloud cover and applies a label or labels that it finds appropriate. While the ground truth labels forbid any additional labels on a cloudy image, we found that we did not need to impose this constraint on our model for good performance.

We crossed several hurdles while training our model. The first and primary challenge was that our dataset was inherently unbalanced. Specifically, some labels, such as *primary* for primary rainforest, appeared in the overwhelming majority of images, while others, such as *slash_burn* for farmland cleared by a slash and burn logging technique, appeared far more rarely. We attempted to correct for this imbalance by implementing a weighting correction over our loss function.

The second hurdle we faced was in attempting to curb overfitting. We initially trained our model using the Adam optimizer, which led to rampant overfitting to the training data as measured by an epoch-over-epoch increase loss on the validation dataset. Adam is known to be prone to overfitting, especially with large batch sizes, such as those employed during training of our model. Fortunately, we were able to correct this issue by switching to a standard stochastic gradient descent with momentum.

We initially found that a relatively simple model composed of two-dimensional convolutional layers and several fully-connected layers performed quite well on the satellite image dataset. However, we found that adding layers and residual connections to the network increased performance, but that the models became more prone to overfitting. By changing the optimizer, we were able to create a final model that had increased performance.

When measured using the F2 statistic, our final model performs well, coming within five percent of the top performing algorithm on the Kaggle leaderboard. However, several issues with the training dataset have been brought to our attention that could be negatively affecting our model's predictions. First and foremost, a discussion in the Kaggle forums (https://www.kaggle.com/robinkraft/issue-with-tif-files) has raised the issue of corrupted TIFF files that were distributed for model training. Additionally, several competitors have noted improved performance when using the JPEG formatted images for training. These findings call into question the integrity of the TIFF dataset, which we relied upon for training. This finding is unfortunate, because the TIFF files contain the near-infrared channel, whereas the JPEG images do not. The three-channel JPEG images are compatible many pre-trained architectures, which some competitors have used to great success, whereas the potentially more information-rich TIFF files are not. We attempted to build a custom architecture to harness the increased information content of the TIFF files, but training and performance have likely been hindered by the aforementioned issues with that particular dataset.

A second, but less specific issue is that of potential mislabeling in the "ground truth" dataset. Figure 7 shows false negative and false positive labels from our model on the left and right sides, respectively. In some of these images the section of the image giving rise to each ground truth label is not entirely clear upon human examination. It is possible that some of the images in the dataset were mislabeled, considering both the size of the dataset and the inexperience of the croudsourced labelers.

7. Conclusion

Here, we present a purpose-built deep convolutional neural network for classification of atmospheric conditions and labeling of terrestrial development using a satellite image dataset. This architecture utilizes all four channels of provided spectral information and performs within five percent of the top performing model on Kaggle at the time of submission. Alternative approaches, such as those pretrained on an orthogonal dataset, are more finely tuned than our current adaptation; however, by incorporating the near-infrared, our model likely has greater room for improvement with further hyperparameter tuning. This work demonstrates that a relatively simple architecture can achieve exceptional performance on satellite imaging labeling tasks, especially when all available information is fed through the network.

7.1. Future directions

We have several ideas to further improve the performance of our model. The first potential direction is to train a separate model for each of the individual labels in addition to a single model to classify the atmospheric conditions. Each binary label prediction can then be pooled with an individual atmospheric class to produce the labels for any given image. We believe that, while hyperparameter optimization may be far more difficult, allowing each network to fully specialize may lead to more accurate labeling.

A second strategy to improve performance is to create an ensemble of models that can collectively vote on the appropriate label set for an image. It is widely accepted that ensembles of models tend to outperform individual models for many tasks. It would be trivial for us to train several slightly different models and ensemble their predictions.

Finally, we would like to implement further data augmentation, including blurring, into the input preparation pipeline. A new input processing pipeline could be attached to our current model as well as any ensemble or multi-model frameworks. Further data augmentation would

also serve to potentially increase the dataset size, which also have a potentially positive impact on training.

8. Model Architectures

ConvNet

```
(0): Conv2d(4, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(1): ReLU (inplace)
(2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True)
(3): Conv2d(32, 32, kernel_size=(2, 2), stride=(2, 2))
(4): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True)
(5): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(6): ReLU (inplace)
(7): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True)
(8): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(9): ReLU (inplace)
(10): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True)
(11): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(12): ReLU (inplace)
(13): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True)
(14): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(15): ReLU (inplace)
(16): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True)
(17): Flatten ()
(18): Linear (524288 -> 40)
(19): BatchNorm1d(40, eps=1e-05, momentum=0.1, affine=True)
(20): ReLU (inplace)
(21): Dropout (p = 0.2)
(22): Linear (40 -> 17)
(23): Sigmoid ()
```

ResNet

```
ResLayer (
  (conv1): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True)
  (relu): ReLU (inplace)
  (conv2): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True))

(0): Conv2d(4, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(1): ReLU (inplace)
(2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True)
(3): Conv2d(32, 32, kernel_size=(2, 2), stride=(2, 2))
(4): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True)
(5): ResLayer ()
(6): ResLayer ()
(7): ResLayer ()
(8): ResLayer ()
(9): Conv2d(32, 32, kernel_size=(2, 2), stride=(2, 2))
(10): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True)
(11): ResLayer ()
(12): ResLayer ()
(13): ResLayer ()
(14): ResLayer ()
(15): Conv2d(32, 32, kernel_size=(2, 2), stride=(2, 2))
(16): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True)
(17): ResLayer ()
(18): ResLayer ()
(19): ResLayer ()
(20): ResLayer ()
(21): Flatten ()
(22): Linear (32768 -> 40)
(23): BatchNorm1d(40, eps=1e-05, momentum=0.1, affine=True)
(24): ReLU (inplace)
(25): Dropout (p = 0.2)
(26): Linear (40 -> 17)
(27): Sigmoid ()
```

References

[1] A. Albert, J. Kaur, and M. Gonzalez. Using convolutional networks and satellite imagery to identify patterns in urban environments at a large scale. *arXiv.org*, Apr. 2017.

[2] D. Alexakis, A. Sarris, T. Astaras, and K. Albanakis. Detection of Neolithic Settlements in Thessaly (Greece) Through Multispectral and Hyperspectral Satellite Imagery. 9(2):1167–1187, Feb. 2009.

[3] G. P. Asner, G. V. N. Powell, J. Mascaro, D. E. Knapp, J. K. Clark, J. Jacobson, T. Kennedy-Bowdoin, A. Balaji, G. Paez-Acosta, E. Victoria, L. Secada, M. Valqui, and R. F. Hughes. High-resolution forest carbon stocks and emissions in the Amazon. *Proceedings of the National Academy of Sciences of the United States of America*, 107(38):16738–16742, Sept. 2010.

[4] S. Basu, S. Ganguly, S. Mukhopadhyay, R. DiBiano, M. Karki, and R. Nemani. DeepSat - A Learning framework for Satellite Imagery. *arXiv.org*, Sept. 2015.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*, pages 248–255. IEEE, 2009.

[6] C. Fischer and I. Kakoulli. Multispectral and hyperspectral imaging technologies in conservation: current research and potential applications. *Studies in Conservation*, 51(sup1):3–16, Dec. 2013.

[7] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *Aistats*, 2010.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. Dec. 2015.

[9] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Feb. 2015.

[10] N. Jean, M. Burke, M. Xie, W. M. Davis, D. B. Lobell, and S. Ermon. Combining satellite imagery and machine learning to predict poverty. *Science*, 353(6301):790–794, Aug. 2016.

[11] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. Sept. 2016.

[12] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. Dec. 2014.

[13] D. Marmanis, M. Datcu, T. Esch, and U. Stilla. Deep Learning Earth Observation Classification Using ImageNet Pretrained Networks. *IEEE Geoscience and Remote Sensing Letters*, 13(1):105–109, 2016.

[14] V. Mnih and G. E. Hinton. Learning to Detect Roads in High-Resolution Aerial Images. In *Computer Vision – ECCV 2010*, pages 210–223. Springer, Berlin, Heidelberg, Berlin, Heidelberg, Sept. 2010.

[15] W. J. Moses, A. A. Gitelson, S. Berdnikov, and V. Povazhnyy. Satellite Estimation of Chlorophyll-a Concentration Using the Red and NIR Bands of MERIS - The Azov Sea Case Study. *IEEE Geoscience and Remote Sensing Letters*, 6(4):845–849, 2009.

[16] S. Phinn, C. Roelfsema, A. Dekker, and V. Brando. Mapping seagrass species, cover and biomass in shallow waters: An assessment of satellite multi-spectral and airborne hyper-spectral imaging systems in Moreton Bay (Australia). *Remote Sensing of Environment*, 2008.

[17] A. Romero, C. Gatta, and G. Camps-Valls. Unsupervised Deep Feature Extraction for Remote Sensing Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, 54(3):1349–1362, 2016.

[18] M. Shimada, T. Tadono, and A. Rosenqvist. Advanced Land Observing Satellite (ALOS) and Monitoring Global Environmental Change. *Proceedings of the IEEE*, 98(5):780–799, 2010.

[19] N. Srivastava, G. Hinton, and A. Krizhevsky. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine ...*, 2014.

[20] S. Voigt, T. Kemper, T. Riedlinger, R. Kiefl, K. Scholte, and H. Mehl. Satellite Image Analysis for Disaster and Crisis-Management Support. *IEEE Transactions on Geoscience and Remote Sensing*, 45(6):1520–1528, 2007.

[21] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht. The Marginal Value of Adaptive Gradient Methods in Machine Learning. May 2017.