Understanding the Amazon from Space
Deep Learning for Satellite Image Classification

Loren Amdahl-Culleton
Department of Mechanical Engineering
Stanford University
lkac@stanford.edu

Meredith Burkle
Department of Electrical Engineering
Stanford University
mnburkle@stanford.edu

Miguel Camacho Horvitz
Department of Computer Science
Stanford University
somosmas@stanford.edu

Abstract

Motivated by the burgeoning commercial and research interest in satellite images of Earth, we developed various models able to efficiently and accurately classify the content of such images. In particular, we trained deep convolutional neural networks (CNNs) to learn image features and used multiple classification frameworks including long short-term memory (LSTM) label captioning and binary cross entropy to predict multi-class, multi-label images. By fine-tuning an architecture consisting of pre-trained Inception-v3 parameters trained on ImageNet data together with the LSTM decoder, we were able to achieve 88.9% F2 test accuracy – well within five percent of the state-of-the-art ensemble models used by industry leaders.

1. Introduction

According to National Geographic, almost a fifth of the Amazon rainforest has been cut down in the last 40 years [18]. Estimates of the extent of the deforestation, however, are both difficult to ascertain and often inaccurate. Given this dearth of reliable and precise estimates, a comprehensive understanding of the natural and/or anthropogenic changes in the Amazon is still lacking [18].

Rudimentary methods for quantifying and characterizing deforestation – particularly in the Amazon – have proven inadequate for several reasons [20]. For one, the existing models often lack the ability to differentiate human-caused and natural forest loss. Secondly, these models often take in coarse-resolution imagery that does not allow the detection of small-scale deforestation and local forest degradation [20]. The prevalence of selective logging which only logs select tree species, for example, can conceal significant logging for low-resolution images [19].

Recent improvements in satellite imaging technology have given rise to new opportunities for more accurate quantification of both broad and minute changes on Earth, including deforestation. Indeed, Planet[1] (one of many smaller aero-astro companies entering the image analysis space) and its Brazilian branch, SCCON, collected novel satellite image chips with more than 10x the resolution of traditional Landsat and MODIS images and launched a Kaggle challenge in April of 2017 (titled *Planet: Understanding the Amazon from Space* [2]) in order to gain further insight into how and why Amazonian deforestation is occurring [20]. More specifically, the challenge is to label Planet's image chips with atmospheric conditions, land cover, and land use. The teams are provided with labeled training and testing images of the Amazon river basin taken from Planet Labs satellites; the chips (images) themselves are available on Kaggle in GeoTiff and JPG format, containing four bands (channels): RGB-NIR (RGB only for JPG) [20].

Given the dynamic labeling – with training images having between one and fourteen different labels – we will experiment with various multi-class, multi-label classifiers using state-of-the-art deep learning approaches borrowing from existing image classification model architectures.

At test time, our classification algorithm will take satellite images as input and will produce a dynamically-sized set of tags as output. Each image's predictions, i.e. its outputed labels, will be written to a submission file that enumerates the image name along with its predicted tags,

[1] https://www.planet.com/

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

as specified by the Kaggle competition submission rules, which will be evaluated using the F2 metric by Kaggle [3].

2. Related Work

Multi-label satellite image classification has been a task of interest ever since the first multi-spectral remote sensing imagery became available (to civilians) in the early 1970s. The overall approach has remained conceptually the same - record satellite observations, derive a feature vector for the image, run a classification algorithm, produce classification labels [2, 1]. In the earlier decades up through the 2000s, the most common approach was to extract feature vectors using methods such as image filtering, PCA, or clustering algorithms, and to then use simple regression-based classification thereafter [5, 1, 4]. Unfortunately, these early models were not particularly illuminating or powerful - over the period of 1989 to 2003, a study of multilabel satellite image classification algorithms was conducted, concluding that "there has been no demonstrable improvement in classification performance over 15 years [1]."

In the later 2000s, remote sensing-focused image processing communities began collaborating and communicating with artificial intelligence and computer vision researchers to advance techniques for classification [4, 2, 8, 3]. Hybrid methods that used a combination of manual tagging and primitive machine learning techniques became more popular [2]. For example, when sample land cover sites were available (called 'training sites'), the user would manually select representative samples for a given class or label [3, 4, 2]. Based on these training sites, a 'signature file' for the given class would be developed; predictions for new images would then be based on maximum-likelihood or minimum-distance estimation techniques [3, 4]. These supervised learning techniques were integrated into software platforms like ArcGIS, but if sample training sites were unavailable, it was also possible to use unsupervised techniques like k-means [6, 16]. In this case, the user would provide the number of clusters for the algorithm to group the pixels into. The user would then use those results to manually identify which clusters correspond to which classes (or merge duplicate clusters corresponding to the same class) [5, 6]. These techniques are still in use today, as ArcGIS (and other comprehensive remote-sensing analytics platforms) is the prevailing tool in many research communities [9, 16].

However, with increased computing power, more neural network-based frameworks have become feasible for use in large scale remote sensing data analysis, and researchers are increasingly using these approaches with dramatically improved results [15, 9]. State of the art techniques use a multitude of fully-connected and convolutional layers with non-linear activations to project the image data onto a predefined feature space that is then used to predict probabilities of given labels [13, 8, 9, 14]. This approach allows models to directly craft the image features based on the specific labels we want to predict (in essence combining the feature-extraction and classification steps from the original paradigm posed in the 1970s) [1, 15].

These CNN approaches take on a variety of different architectures. Some simply use a binary sigmoid cross-entropy loss (as opposed to softmax which is more common for single-label classification) [10, 7]. Rather than taking the `argmax` as a softmax-based model would, the binary sigmoid cross entropy loss focuses on deciding whether a given label applies or does not apply. The label is applied as such if its corresponding logit exceeds a certain probability threshold. A great deal of research has played with these thresholds and the benefits of including those as model parameters [10, 7], with considerable success. However, one issue with this method is that it ignores possible relationships and interdependencies between labels.

Other models have sought to address this problem, applying techniques rooted in statistics, NLP and other areas [11, 12, 8, 13]. For example, DeepSat uses a wide variety of CNN architectures to predict four or six different land cover classes [9]. Alternatively, multi-instance multi-label approaches use multi-layer perceptrons to extract regional features, to then pass to a second stage that is meant to capture connections between labels and regions as well as correlations between labels [12, 11]. A still more unique approach was proposed by Wang et al. that uses a CNN-Recurrent Neural Network (RNN) approach to extract image features (CNN) and then predict a series of labels (RNN) that captures inter-label dependencies while also maintaining label order invariance [8].

Our approach, described in the Methods section below, seeks to both evaluate and synthesize some of the more successful techniques within the scope of a multi-label satellite image classification problem, as well as develop novel infrastructures of our own that improve upon these works.

3. Methods

3.1. Models

The key classifiers used in our project include a Softmax Cross-Entropy (CE), a Sigmoid CE, and a joint Softmax-Sigmoid (SS) CE loss and evaluation as well as a LSTM cell structure. Each of these were used on top of baseline, transfer and fine-tuned pre-trained CNN models to exploit inherent structure in our dataset.

Softmax/Sigmoid Baselines. Most of the introductory networks seen in CS231n lectures and assignments are designed for single label classification; they make use of the

[3] https://www.kaggle.com/c/planet-understanding-the-amazon-from-space#evaluation

softmax CE loss and assign the most likely label out of a possible N labels based on that softmax result (grabbing the `argmax` across the final N-vector) [13]. As such, we began with this approach by designing a single-label classifier to attempt to predict weather labels for each image. Because each image in the dataset has one and only one weather label, we knew softmax classification could be applied. Of course, as our training images were by no means limited to one weather label, we soon moved towards a sigmoid classifier. The softmax loss scheme, while advantageous for single-label classification, is not as well-suited for multi-label classification [10, 8, 13] for a variety of reasons but most glaringly due to the fact that each label is no longer mutually exclusive in a multi-label classification task. The common fix for this is to alter the loss function from softmax CE loss to *sigmoid CE loss* [10]. Sigmoid CE is primarily used for binary classification, and so is able to evaluate which labels, out of N possible, should be set to 'on,' as opposed to selecting a single best label [10]. Indeed, we first attempted this approach, with success, using a small 2-Layer CNN to get baseline results (the architecture is a simplified version of a test model that appears in a PyTorch Starter Kit written by a fellow Kaggle challenge participant[28]). However, one issue with this approach is that it does not make use of unique structure within our dataset, which motivated our more advanced Softmax-Sigmoid joint loss approach.

$$Softmax(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{k=1}^{n} e^{x_k}}$$

$$Sigmoid(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{x}}}$$

Inception-SS. In order to exploit unique weather labels (every satellite image has one and only one weather label `clear`, `cloudy`, `partly_cloudy`, or `haze`) and multiple (zero to many) land use/land cover labels, we moved towards a hybrid loss function which applied single-label classification (softmax loss) for the weather tags and multi-label classification (sigmoid loss) for the rest of the land use and land cover tags[20].

After getting aforementioned 2-layer CNN baseline[28] off the ground, we realized we would need to leverage more advanced CNN architectures with transfer learning in order to develop the best model possible. As a result, we replaced our framework's back-end with GoogLeNet's Inception-v3 [17] trained on ImageNet images. Our initial approach, in line with the typical transfer learning paradigm of extracting and training on the final layer of a pretrained network [21], was to feed our images through pre-trained, but frozen Inception parameters and from there calculate the SS CE loss as described above. A figure depicting this model is shown in the purple box in 3. We will refer to this Sigmoid-Softmax with Inception-v3 pretrained features in later sections with the 'ISS-' prefix.

Inception-LSTM. This model was designed to exploit the label ordering in the training dataset as well as the fixed 'vocabulary' (only 17 possible labels). The idea was loosely inspired by Assignment 3's LSTM captioning problem, which trained an LSTM (Long-Short Term Memory cells)[30] based architecture on the Microsoft COCO Captioning dataset to be able to generate a predicted image caption one word at a time [22].

LSTMs follow a structure similar to that of a "Vanilla" RNN. Both architectures consist of cells (Fig.1) that make forward steps in both "time" and "depth"(Fig.2)[31].What differentiates an LSTM from an RNN network is the smooth flow of gradients represented by the red arrows in Fig 1. During back-propagation, the RNN network gradients have a tendency to explode or go to zero as the gradients are all connected through multiplication whereas the gradients passed backwards by the LSTM cell are connected through addition (thus avoiding the multiplicative relationship that leads to trouble in Vanilla RNN).

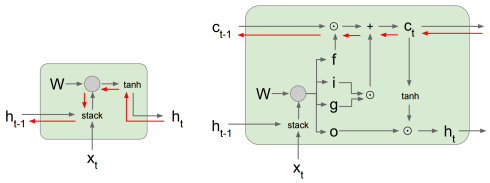

Figure 1. An RNN (left) and LSTM (right) Cell [31]

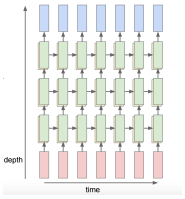

Figure 2. An LSTM Cell [31]

If we look at Fig. 2, the red rectangles represent input features, the green rectangles are LSTM cells, and the blue rectangles represent the scores (or predictions) of the LSTM network. As this network begins training and evaluating at time "0" (left of Fig 2), one can imagine how such a network produces ordered captions as it predicts a new word at each time step.

Although image captioning techniques are not typically applied to multi-label image classification [8, 13, 14, 9], we noted that the consistent ordering of the image labels in the training set (labels will always be seen in the order 'clear primary' as opposed to 'primary clear') as well as the comparatively low number of labels (and thus small "vocabulary") might perhaps make our problem better suited for RNN-based techniques. Using this framework, our labels would be considered as "captions," beginning with a

<START> token and ending with an <END> token. In the same way it probabilistically predicts any other label, the model will capture the dynamic sizing of the label sets by learning when to predict an <END> token. Overall, our architecture takes in image features generated by Inception-v3 (size 2048 vectors) and feeds into a fully-connected layer into an LSTM cell, and then into a temporal fully connected layer. From there, the temporal softmax weights predict a single label for a given timestep and the loss is propagated back through the network to update the model parameters. Figure 3 depicts this architecture in the blue box, showing the flow from the FC layer through to the temporal softmax classification step. After the model has been trained, predictions are formed by feeding new images through the model and generating "captions" (series of labels) one step at a time until an <END> token is predicted.

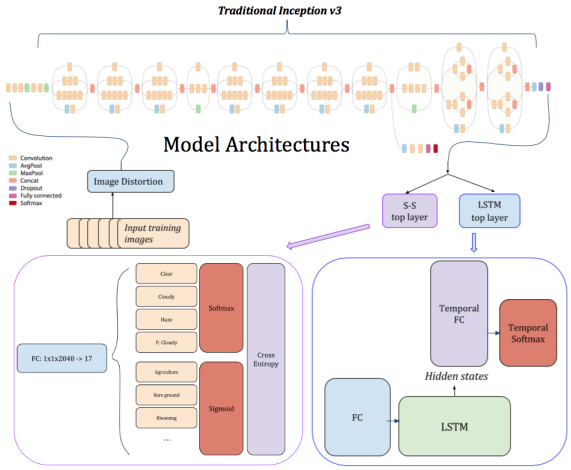

Figure 3. Model architecture for SS and LSTM.

4. Dataset and Features

4.1. Dataset

The Kaggle-provided datasets are divided into both `train-[type]` and `test-[type]` input images where the `[type]` is either `.tiff` or `.jpg` format. The data has a ground sample distance (GSD) of 3.7m and an orthorectified pixel size of 3m. Each image file is a 256x256 pixel (947.2m x 947.2m) "chip" which is sampled from a larger 6600x2200 pixel "Planetscope Scene [20]."

In total, Planet and Kaggle supplied 40,479 training images and their corresponding labels [20]. Of these $\sim$ 40K training images we split 4K off to be our validation set, and ultimately tested our models on the hidden set of $\sim$ 60K test images (via submission to the Kaggle challenge).

The 17 possible tags for each chip are divided into three categories: cloud cover, common labels, and less common labels (see example chips in 4). Each chip has exactly one cloud cover label taking one of four values: `cloudy`, `partly_cloudy`, `haze`, and `clear`. The common and less common labels, in turn, take zero or more values of `primary` (the label for primary rainforest), `water`, `habitation`, `agriculture`, `road`, `cultivation`, `bare_ground` and `slash_burn`, `selective_logging`, `blooming`, `conventional_mine`, `artisinal_mine`, `blow_down`, respectively [20]. Further worth noting is that the distribution of labels was quite unbalanced as evidenced in 5.

Figure 4. Labeled Chips

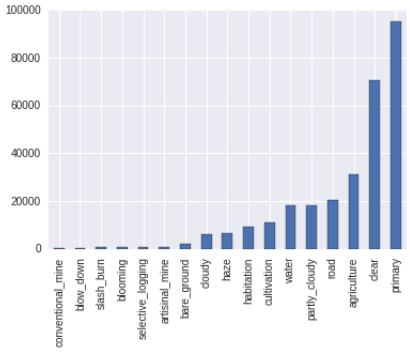

Figure 5. Label Occurences.

4.2. Preprocessing and Feature Extraction

Inception bottlenecks. Our choice of Inception-v3 as our back-end for transfer learning meant we had to follow certain preprocessing steps to ensure that our images would be similar in form to those that Inception was trained on (for example, certain networks have subtracted out the mean of the image, or have normalized the range of pixel values). Inception has a variety of relatively user-friendly resources that walk readers through transfer learning and retraining the network [23, 24, 25]. GoogLeNet has also written scripts that will feed JPG images through a preprocessing pipeline and up through the frozen section of the model for feature extraction [25]. These resulting 2048-vectors were the features we used for our model frontend. (Note: one issue we ran into was that the Kaggle-provided JPG images were in CMYK format, whereas Inception expects RGB. As a result our early models were aggressively mediocre since the model was having a lot of difficulty interpreting our 4-channel image.)

Data augmentation. Because our model architectures added layers onto the existing Inception-v3 structure (and hence increased the number of model parameters), we wanted to prevent our model from overfitting. In order to mitigate this issue, we distorted our training images. In particular, we experimented with random left-right image flipping, random up-down image flipping, random number of image rotations, random cropping of 5-10% of the image, random scaling of 5-15% of the image, and random brightness scaling (i.e. multiplication of pixel values by 5-10%).

By adding random distortions to the training images at each epoch we prevented the model from seeing the same images many times and generally succeeded in preventing overfitting as our validation and test accuracies show in the Experiments section below.

5. Experiments

5.1. Performance Metrics

Table 1 shows our F2 performance, calculated as follows:

$$F_2 = (1+\beta^2)\frac{pr}{\beta^2 p + r},$$

$$\text{where } p = \frac{tp}{tp+fp}, r = \frac{tp}{tp+fn}, \beta = 2$$

Variables *p* and *r* are precision and recall respectively, while tp/fp and tn/fn represent true/false positive and negatives. The final score is given as the mean F2 score over all examples in the test set.

Overall, we were able to achieve an F2 score within 5% of top teams in the Kaggle challenge, eventually reaching 245th on the leaderboard. Although the Kaggle challenge leaderboard is based on F2 accuracy accuracy alone, we will also look at loss plots, one-to-one accuracy, example predictions, and final layer logits to get a better sense of our model's strengths and weaknesses. With the 'one-to-one' metric we are simply referring to the percentage of matches among all 17 ground truth versus predicted possible labels. More precisely:

$$\text{one-to-one} = \frac{tp+tn}{tp+fp+tn+fn}$$

5.2. Performance Results

Following implementation of our more sophisticated models, both training and validation F2 scores increased uniformly by ∼36%, achieving a final highest score of **88.941%** with ISS-20. As our loss graphs show, after sufficient iterations, both our training and validation losses converged to comparable values.

As we can see in Table 1, even our baseline model performed with high one-to-one accuracy. In juxtaposition with the poor F2 scores for the baseline model, one-to-one accuracy showed itself as only a vague indicator of success early on.

While ISS-21 and ISS-19 achieved higher Validation F2 and One-to-One accuracy, ISS-20 gave us our best Test F2 result. Given the sparsity in the presence of most of the classes in the data set, our model does a very good job of generalizing and learning to accurately handle these rare class cases.

We have also included loss plots below, serving as a confirmation that our model was in fact training properly, and additionally indicated convergence of training and loosely indicated generalizability based on validation loss.

Model	One-to-one Accuracy / F2 Accuracy		
	Train	Val	Test
Sigmoid Baseline	90.0 / 64.0	91.0 / 65.0	- / -
LSTM	96.3 / 88.7	95.0 / 87.1	- / 87.212
ISS-7	92.0 / 79.0	93.0 / 74.0	- / -
ISS-19	95.5 / 88.9	95.6 / 88.9	- / 87.282
ISS-20	94.6 / 90.5	94.9 / 87.6	- / **88.941**
ISS-21	95.0 / 90.3	95.6 / 89.1	- / 88.731
I-FT	96.2 /88.2	96.6 / 88.9	- / -

Table 1. Accuracy Evaluation Results

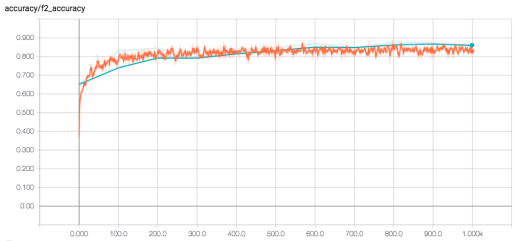

Figure 6. ISS-21 F2 Train (orange) + Validation (blue) Accuracy

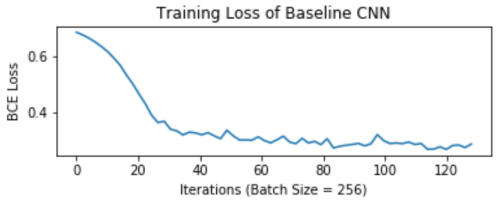

Figure 7. Baseline CNN Loss

5.3. Example Predictions

Visualizing predictions from both the ISS- and LSTM models gives insight into the predictive mechanisms of these different architectures and algorithms.

Figures 11, 12, and 13 show a few examples of image predictions generated by the Inception-LSTM model. The

Figure 8. ISS-7 Train (orange) + Validation (blue) Loss

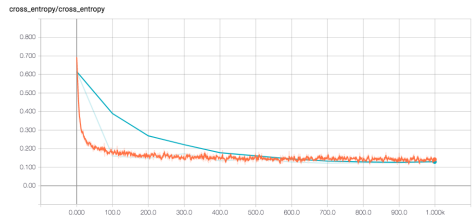

Figure 9. ISS-21 Train (orange) + Validation (blue) Loss

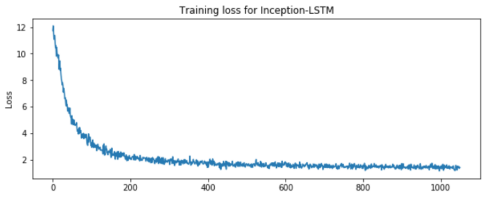

Figure 10. Inception-LSTM Loss

heat map plots on the right of the figure show the probability distributions at each time step; the darker the square, the higher the probability. Starting from the top, we see the first predicted label (i.e. darkest square), continuing until the last row, where the model predicts the <END> token. The first image/prediction pair display a relatively common label set, and we can see that the model is confident in its predictions. The second pair demonstrates an intermediate difficulty image, and also showcases the models ability to predict weather patterns. Despite the image showing similar characteristics to a haze or cloudy image, it correctly (if hesitantly) guesses partly cloudy. The final pair displays a more difficult label set, as we can see the model is not very confident in its predictions (i.e. for each prediction step, the difference between the darkest cell, second darkest cell, etc. is less obvious than in the previous two examples).

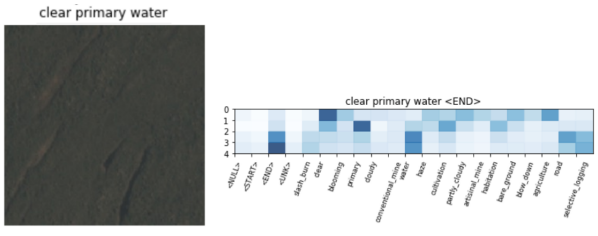

Figure 11. Example 1 LSTM Image and Prediction

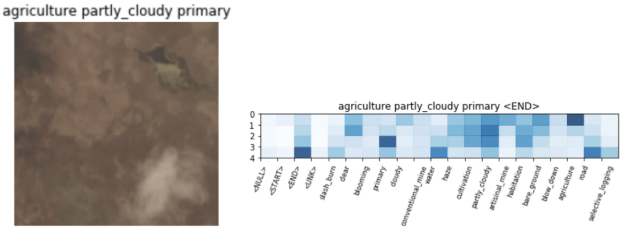

Figure 12. Example 2 LSTM Image and Prediction

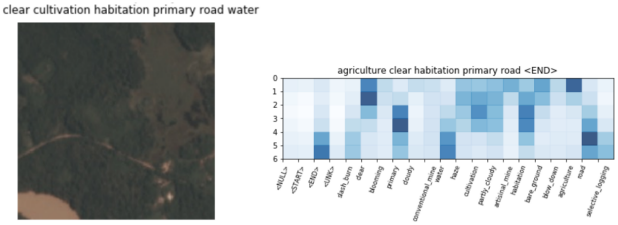

Figure 13. Example 3 LSTM Image and Prediction

We can also extract more general insights from these examples by looking at the other predictions the models might have made. For example, we see from the LSTM predictions that the model only considers predicting `cloudy` on the first timestep, this indicates that the model was able to figure out that if the image is cloudy, it will never have any other tags associated with it (everything else is obscured). We can also determine which examples the model struggles most with based on timesteps where it is a close call between labels - in Example 3 (Fig. 13) at timesteps 5 and 6, we can see that the probabilities for both road and water are comparable. Indeed, from the air, roads and bodies of water (usually rivers) tend to look like snaking brown lines of varying thicknesses, so it is not unsurprising that the model would struggle with these types of examples (this is also an example where incorporating the Near-IR channel would be beneficial).

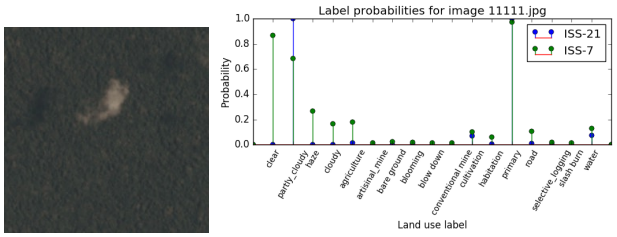

Figure 14. Example 1 ISS- Image and Predictions

The plots in Figures 14 and 15 demonstrate example predictions generated by ISS-21, our more successful Inception-SS model, and ISS-7, an intermediate model of a similar architecture. For each image, the plots show the probabilities with which the model predicts a given label. The first four categories on the x-axis correspond to weather labels, and the rest correspond to land use and land cover.

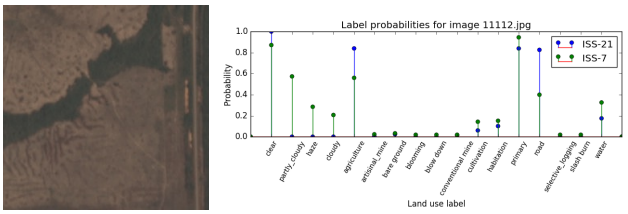

Figure 15. Example 2 ISS- Image and Predictions

The first image, 11111.jpg, gives an example of an image that was previously misclassified by ISS-7. Its ground truth label set is `partly_cloudy` and `primary`; but our old model predicted clear and primary instead. As we can see, the ISS-21 distribution is able to surpass ISS-7's hesitance and discern that the image's weather label is in fact partly cloudy.

The second image, 11112.jpg, is an example that shows that although ISS-21 is a vast improvement on ISS-7, it is still not perfect. The ground truth label set is `agriculture`, `clear`, `primary`, and `water`. Although the ISS-21 model is more confident in its prediction of clear and agriculture, it loses confidence in its labeling of primary and instead chooses the "road" label instead of "water." As mentioned in the discussion of LSTM example predictions, roads and water features do in fact look similar so this is a trickier example, but still shows areas where the model could be improved.

5.4. Hyperparameter Tuning

In order to achieve the performance metrics discussed above, diligent hyperparameter-tuning was a crucial step. We ran a variety of tests to evaluated the effect of hyperparameters on our various models, eventually submitting four Inception-based transfer learning models to the Kaggle leaderboard. The three ISS- models were run using two trained affine layers on top of the pretrained network into a softmax-sigmoid joint CE loss. The LSTM model submitted used a single trained affine layer, fed into an LSTM cell, along with a temporal fully-connected layer into a corresponding temporal softmax CE loss (3). The hyperparameters for each submission can be found in Table 2.

Model	Optimizer	LR	LRD	Dropout	Epochs
LSTM	Adam	0.0001	0.98	0.2-0.3	7
ISS-19	Adadelta	0.5	0.50	0.2-0.3	$\sim$3
ISS-20	RMSProp	0.0005	0.90	0.2-0.3	$\sim$3
ISS-21	Adam	0.0005	0.96	0.2-0.3	$\sim$3

Table 2. Inception Transfer Models

In addition to the parameters in 2, we also looked at transfer learning layers to train as well as number of training steps in order to optimize our model. Initially following a standard transfer learning approach, we kept the Inception pretrained waits up to a new final affine layer, which we trained ourselves. Trying various learning rates and learning rate decay schedules, we found that we could generally achieve rapid convergence. As shown in Fig. 16, the dark blue, light blue, and green schedules certainly converged by $\sim$500 steps, corresponding to less than two epochs on our training data of $\sim$32,000 images and training batch sizes of 100 images.

After reaching convergence with the lone affine layer, we experimented with training additional top layers, namely multiple affine layers with dropout, as well as fine-tuning the full *Inception* model (I-FT)[29] (i.e. the previously frozen model parameters, see the Inception architecture in 18). As shown in Fig. 17, we were able to decrease our loss with an additional top layer. Unfortunately, as shown in Fig. 19 when training the full Inception model, this modification did not significantly lower our loss.

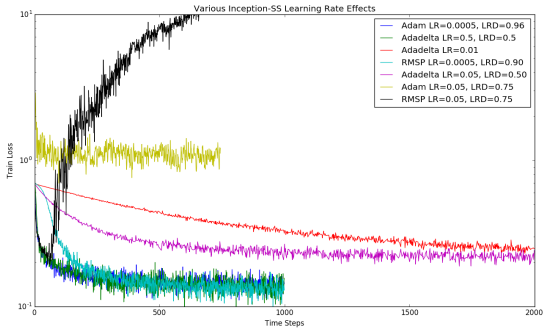

Figure 16. Hyperparameter Tuning.

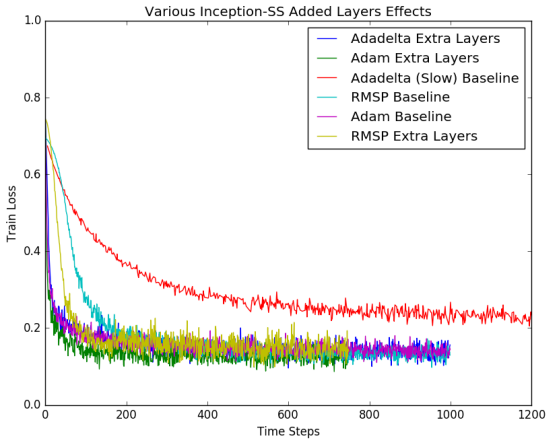

Figure 17. Impact of Additional (Top) Affine Layers.

-

type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

Figure 18. GoogLeNet Inception Layers.

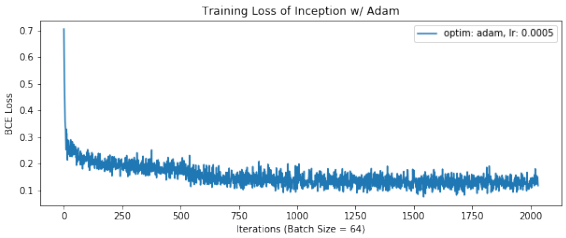

Figure 19. Fully-Finetuned Inception V3

Thresholds	F2 Accuracy
0.160	90.11
0.175	90.13
0.189	90.14
0.204	90.11
0.218	90.04
0.233	89.97
0.248	89.84
0.262	89.70
0.277	89.58
0.292	89.44
0.306	89.31
0.321	89.17
0.335	88.99
0.350	88.85
0.500	87.79
Optimized	**90.45**

Table 3. ISS-20 F2 Scores by Threshold Values. These F2 scores correspond to the accuracy when predicting labels over constant threshold values shown in the left column. The 'optimized', bottom row shows the corresponding F2 accuracy when choosing variable thresholding values by label.

5.5. Thresholding

Even after optimizing our models to achieve low loss results, we still observed poor F2 accuracy which is what our classifier was to be ultimately judged upon. In order to address this, we optimized our predictions post-training to improve F2 accuracy. Specifically, we did a greedy search along hundreds of discrete threshold values *per label F2 score* to accumulate individual 'optimized' thresholds for every label above which we would predict a label at validation/test time. Initially, we predicted a label if the outputted logits were over constant threshold of 0.5 (i.e. 50%) for all labels; however, as Table 3 shows, a constant threshold, especially 50%, was often a poor thresholding value.

Indeed, given that F2 accuracy penalizes false negatives more so than false positives it came as little surprise that lower thresholding values often gave better validation F2 accuracy scores. For test time predictions, we ultimately used per-label threshold values which (in our validation set experimentation) tended to outperform constant thresholding values. Table 3 below shows that the 'optimized' per-label thresholding gives a slight boost relative to the best constant threshold value.

In this case, the 'optimized' threshold values were:

cloudy ≥ 0.109, partly_cloudy ≥ 0.123, haze ≥ 0.154, clear ≥ 0.15, slash_burn ≥ 0.104, booming ≥ 0.1, primary ≥ 0.191, conventional_mine ≥ 0.1, water ≥ 0.286, cultivation ≥ 0.1, artisinal_mine ≥ 0.118, habitation ≥ 0.114, bare_ground ≥ 0.1, blow_down ≥ 0.164 agriculture ≥ 0.104, road ≥ 0.163, selective_logging ≥ 0.1.

6. Conclusions

With the current boom in satellite earth-imaging companies the obvious challenge lies in accurate and automated interpretation of the massive datasets of accumulated images. In this project, we tried to tackle the challenge of understanding one subset of satellite images – those capturing images of the Amazon rainforest – with the particular goal of aiding in characterization and quantification of the deforestation of this area.

Using pre-trained state-of-the-art models such as GoogLeNet's Inception architecture we were able to create architectures that exploited the structure of our dataset in multiple ways and achieved strong performance accuracy. Still, moving forward, there are still various milestones we wish to pursue. Specifically, we are currently working on exploiting the labeling (i.e. hierarchically predictions which exploit the weather label, common land type, then rare land type natural ordering), ensembling multiple optimized models including transfer models using ResNet and other pre-trained deep CNN algorithms, and leveraging the information in the `.tiff` files (specifically the Near-IR channel which tends to be very informative in remote-sensing applications). Overall, experimenting with and optimizing our suite of model frameworks served to be an illuminating and exciting final project, especially when applied to a topical and impactful real-world Kaggle challenge.

References

[1] G.G. Wilkinson. "Results and implications of a study of fifteen years of satellite image classification experiments." *IEEE Transactions on Geoscience and Remote Sensing* (Vol. 43, No. 3). 2005.

[2] Sunitha Abburu, Suresh Babu Golla. Satellite Image Classification Methods and Techniques: A Review. *International Journal of Computer Applications*, (Vol. 119, No. 8). 2015.

[3] Sayali Jog, Mrudul Dixit. Supervised classification of satellite images. *Conference on Advances in Signal Processing (CASP)*, 2016.

[4] George F. Hepner. Artificial neural network classification using a minimal training set. Comparison to conventional supervised classification. *Photogrammetric Engineering and Remote Sensing*, (Vol. 56, No. 4). 1990.

[5] Turgay Celik. Unsupervised Change Detection in Satellite Images Using Principal Component Analysis and k-Means Clustering. *IEEE Geoscience and Remote Sensing Letters*, (Vol. 6, No. 4). 2009.

[6] ArcGIS. What Is Image Classification? ArcGIS 10.5 Help Site, 2017.

[7] A. McCallum. Multi-label text classification with a mixture model trained by EM. *AAAI99 Workshop on Text Learning*. 1999.

[8] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, Wei Xu. CNN-RNN: A Unified Framework for Multi-Label Image Classification. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2285-2294.

[9] Saikat Basu, Sangram Ganguly, Supratik Mukhopadhyay, Robert DiBiano, Manohar Karki, Ramakrishna Nemani. DeepSat A Learning framework for Satellite Imagery. *Computer Vision and Pattern Recognition*, 2015. arXiv:1509.03602

[10] Rong-En Fan, Chih-Jen Lin. A Study on Threshold Selection for Multi-label Classification. *Advances in neural information processing systems*, 2017. A Study on Threshold Selection for Multi-label Classification

[11] Zenghai Chena, Zheru Chia, Hong Fua, Dagan Fenga. Multi-instance multi-label image classification: A neural approach. *Neurocomputing*. (Vol. 99). 2012.

[12] Zheng-Jun Zha, Xian-Sheng Hua, Tao Mei, Jingdong Wang, Guo-Jun Qi, Zengfu Wang. Joint multi-label multi-instance learning for image classification. *Computer Vision and Pattern Recognition*. 2008.

[13] Yunchao Wei, Wei Xia, Junshi Huang, Bingbing Ni, Jian Dong, Yao Zhao, Shuicheng Yan. CNN: Single-label to Multi-label. *Journal of LaTex Class Files* (Vol. 6, No. 1). 2014. arXiv:1406.5726

[14] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, Alan Yuille. Deep Captioning with Multimodal Recurrent Neural Networks (m-RNN). *Computer Vision and Pattern Recognition*. 2015. arXiv:1412.6632v5

[15] H. Bischof, W. Schneider, A.J. Pinz. Multispectral classification of Landsat-images using neural networks. *IEEE Transactions on Geoscience and Remote Sensing* (Vol. 30, No. 3). 1992. IEEE Link

[16] V. Mnih and G. Hinton. Learning to detect roads in high-resolution aerial images. *European Conference on Computer Vision (ECCV)*. 2010. Learning to Detect Roads in High-Resolution Aerial Images

[17] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. *Computer Vision and Pattern Recognition*. 2015. arXiv:1512.00567

[18] Scott Wallace. Amazon Rainforest, Deforestation, Forest Conservation. *National Geographic*. Farming the Amazon

[19] Robinson Meyer. Terra Bella and Planet Labs' Most Consequential Year Yet. *The Atlantic*, 2016. Terra Bella and Planet Labs' Most Consequential Year Yet

[20] Planet: Understanding the Amazon from Space. *Kaggle*. Challenge link

[21] Andrej Karpathy. Transfer Learning, 2017. CS231n: Transfer Learning

[22] CS231n Course Staff, CS231n Assignment 3. 2017. Assignment 3

[23] Tensorflow. Retraining Inception's Final Layer for New Categories. 2017. How to Retrain Inception's Final Layer for New Categories

[24] Radek Bartyzal. Multi-label image classification with Inception net. *Medium*. April 2, 2017. Multi-label image classification with Inception net

[25] Google Developers. Image Classification Transfer Learning with Inception v3. 2017. Transfer Learning with Inception v3

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition.arXiv:1512.03385v1, 2015.

[27] PyTorch ResNet

[28] Mamy Ratsimbazafy. Starting Kit for PyTorch Deep Learning, 2017.

[29] Justin Johnson. PyTorch Finetuning Example, 2017.

[30] Sepp Hochreiter, Jrgen Schmidhuber. Long Short-Term Memory, 1997.

[31] Fei-Fei Li, Justin Johnson, Serena Yeung. CS 231n Lecture Slides, *Lecture 10* , 2017.