Classify Amazon Rainforest Pattern Using Satellite Data

Yao Weng

Abstract

In recent decades, scientists have studied the deforestation patterns using satellite data, and indicate that Amazon rainforest deforestation is rapidly accelerating. Kaggle together with Planet, designer and builder of the worlds largest constellation of Earth-imaging satellites, hold a competition to apply satellite data to track the human footprint in the Amazon rainforest. The labeled satellite image chips are provided by Planet and its Brazilian partner SCCON (Santiago & Cintra Consultoria), I will develop an algorithm to classify the image using what I have learned in Machine Learning Engineer Nanodegree Program.

1 Domain Background

Satellites are very powerful tools to detect deforestation of rainforest.The National Institute for Space Research (INPE) have developed a near real-time deforestation detection system (DETER) and it has helped Brazils government to reduce its deforestation rate by almost 80% since 2004 [1]. In April 21th 2017, Kaggle launched a competition of Understanding the Amazon from Space. It encourages competitors to apply machine learning algorithms to label satellite image chips with atmospheric conditions and various classes of land cover/land use [2]. I choose this topic as my capstone project for two major reasons. First, it is worthy to develop an algorithm to precisely classify the rainforest satellite images, as it will help scientists to do offline study of the deforestation pattern, rate and then better understanding the problem. Secondly, a fast algorithm can be applied in the real-time system which can help scientists or governments to quickly respond to the on-going deforestation.

2 Problem Statement

The chips (image) were derived from Planet's full-frame analytic scene products using our 4-band satellites in sun-synchronous orbit and International Space Station orbit. They were labeled using the Crowd Flower platform and a mixture of crowd-sourced labor and Berlin and San Francisco teams of Planet, the whole processing is time consuming and costly.

The labels can broadly be broken into three groups: a) atmospheric conditions, b) common land cover/land use phenomena, and c) rare land cover/land use phenomena. An image may have one and potentially more than one atmospheric label, and zero or more common and rare labels, shown in Fig.1. The task is to assign a chip a set of target labels, which can be considered as a multi-label classification.

Figure 1: Sample chips and their labels.

3 Datasets and Input

Data [2] are composed of 40479 labeled training images and 61191 unlabeled images for final submission. The file size of an image is $256 \times 256 \times 3$, Tab.1 shows the label distributions of the training data.

Atmospheric	cloudy	2089
	partly cloundy	7261
	haze	2697
	clear	28431
Common Land	primary	37513
	water	7411
	habitation	3660
	agriculture	12315
	road	8071
	cultivation	4477
	bare ground	862
Rare Land	slash burn	209
	selective logging	340
	blooming	332
	conventional mine	100
	artisinal mine	339
	blow down	98

Table 1: Label classification of training data.

4 Solution and Statement

There are two main methods to solve multi-label classification problem. One is to transform the multi-label problem into single-abel problems(s), but the transformed dataset grows large with high label cardinality and cannot model dependencies between labels. The other one is to adapt a single-label algorithm to produce multi-label outputs. In this competition, I will adapt conventional neural network with multiple outputs.

4.1 Loss Function

In single label classification, we use softmax function to squash the values of a vector in the range $[0, 1]$ that add up to 1. In multi-label case, a sigmoid function Eq.1 is used to predict the outputs, which indicates the class probabilities. I set 0.5 to the threshold of tagging a class, if the predicated class probability is greater than this threshold, the image is labeled with this class.

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{1}$$

Binary cross entropy function is used as loss function Eq.2,

$$\begin{aligned} \mathcal{L}(\theta) &= -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)] \\ &= -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij}\log(p_{ij}) \end{aligned} \tag{2}$$

where i sums over all samples and j sums over all classes, y is the sample label and $p_{ij} \in (0, 1), \sum_j p_{ij} = 1 \forall i, j$ is the prediction for a sample.

4.2 Measurement Metrics

The F_2 score is used to evaluate the submission, see Eq.3

$$F_2 = (1 + \beta^2)\frac{pr}{\beta^2 p + r}, \beta = 2 \tag{3}$$

where Precision p is the ratio of true positives to all predicted positives. Recall r is the ratio of true positives to all actual positives. F_2 weights recall higher than precision, the mean F_2 score is formed by averaging the individual F_2 scores for each row in the test data.

4.3 Baseline Model

Convolutional neural network (CNN) has demonstrated excellent performance on many complex visual tasks. My baseline model is a two layer CNN with three fully connected layers. Each fully connected layer is followed by a pooling layers with drop probability 0.3. Fig.2 shows the architecture of baseline CNN. Training data is splitted to training set (80%) and testing set (20%), the image is resize to $128 \times 128 \times 3$. Data is trained in 50 epochs with batch size of 128. I submitted the predictions using baseline model, the F_2 score is 0.82752. The best F_2 score in the competition is 0.93423, and the sample submission benchmark from Kaggle is 0.67485. Fig.3 shows one misclassification in common land feature, the baseline model failed to tag 'water'.

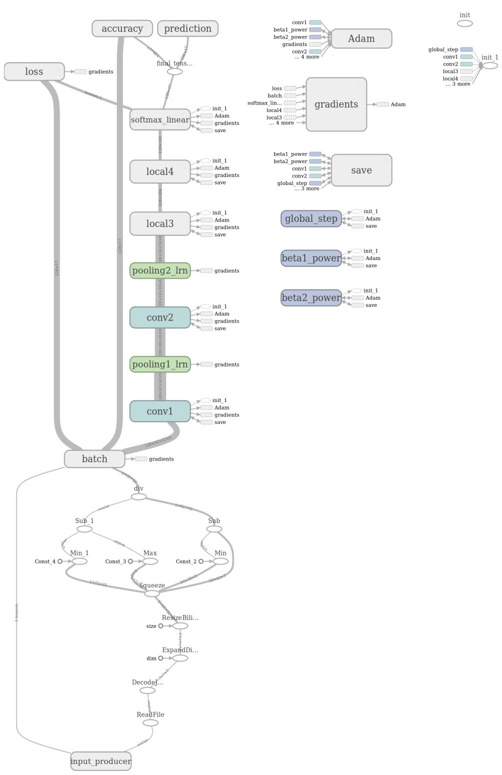

Figure 2: Tensorboard: baseline CNN architecture.

Figure 3: True Label: agriculture clear habitation primary road water. Prediction: agriculture clear habitation primary road

4.4 Transfer Learning

Transfer learning involves taking a pre-trained neural network and adapting the neural network to a new different data set and thus accelerate our deep learning model.There are two major methods in transfer learning: retraining and fine tuning, depending on the size of new data set and the similarity of new data to the original data. The amazon satellite image data set is not big, however, is different from ImageNet which used to train InceptionV3, VGG16 etc. In this capstone project, I will apply transfer learning on InceptionV3 in the following steps

- Use opencv to read images, and divide each pixel by 255 to make them in range $[0, 1]$
- Split the labeled dataset, 80% as training and 20% as testing
- Load InceptionV3 model from keras and pop up the final layers. Use a fully-connected layer in shape of $1 \times 1 \times 17$ as the final layer, where 17 is the total number of classes of amazon images
- Use ImageDataGenerator of keras to generate batches of augmented tensor image in realtime training. I will try random rotation, and random horizontal/vertical shifts.
- Choose Adam optimizer [3] to minimize the loss fuction Eq.2.

- Initial weights of InceptionV3 are from training ImageNet. I will freeze all initial weights except those in the last fully connected layer, retrain the network to update them and save the checkpoint. I will save the plot of loss vs number of epochs, and choose a proper epoch number to avoid overfitting. Another choice is to use EarlyStopping in keras to stop training when validation loss isn't decreasing anymore.
- Fit testing data using saved model and calculate mean F_2 score.
- Apply fine tuning on the retrained model, update weights of all layers and save the model. Compare mean F_2 score from using fine tuning model with that from retrained model.

References

[1] Gabriel Popkin. Satellite alerts track deforestation in real time. *Nature*, 530(52):392–393, February 2016.

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

[3] Jimmy Ba Diederik P. Kingma. Adam: A method for stochastic optimization. *ICLR*, 2015.