Foundations of Computer Vision

A Project Report on:

Understanding Deforestation in Amazon Basin using CNNs

Name: Ayush Soni (as2425@g.rit.edu)

I. Introduction:

Every year, an estimated 18 million acres of forest cover is lost from various parts of the world, according to UN Food and Agriculture Organization. Deforestation in Amazon Basin accounts for the largest share among them, leading to reduced biodiversity, habitat loss, climate change and other detrimental effects. Planet Labs Inc. has built and deployed the largest constellation of Earth-imaging satellites, collecting daily imagery of the entire land surface at 3-5 meter resolution, providing the highest quality of earth-imaging ever produced. They have provided satellite image chips of the Amazon Basin, both with and without labels corresponding to various atmospheric conditions and land cover/land use, on the Kaggle platform. The aim of the Kaggle challenge[1] is to build an efficient and reliable algorithm to automate the labeling of new satellite images using existing data, which can eventually help the global community better understand where, how, and why deforestation happens all over the world - and ultimately how to respond.

This problem can be framed as a multi-label, multi-class classification problem, which is discussed in detail in the Implementation section. For extracting image features and generating labels, a Convolutional Neural Network architecture called Squeeze-and-Excitation Resnet-154 is used, which is current the state-of-the art on the ImageNet classification challenge.

II. Overview of Dataset:

As stated earlier, the data set used in this project was taken from the Kaggle challenge at [1]. The total size of data is approximately 53 GB, consisting of Images and labels as described below:

- Images: There are around 40k(21 GB) labeled and 61k(32 GB) unlabeled images in the data set. These images are provided in both JPG and TIFF formats, however, the images in TIFF format contain extra information related to the Near Infra-red data from satellite that is missing in the JPG images. To make use of this extra information, we use the TIFF files rather than JPG files. Each TIFF file is of shape (256, 256, 4), where the first three channels are Blue, Green and Red channels respectively, and the fourth channel corresponds to Near Infra-Red(NIR) image data. Each of these images can have one or more labels corresponding to them, which are described below.
- Labels: The labels for all training images are provided in the ‘train.csv’ file. This file contains 17 unique labels, that describe various aspects of an image such as type of Cloud Cover, Haze, Primary Rain Forest, Water (Rivers & Lakes), Habitation, Mining, Agriculture, etc. Below is a complete list of unique labels in the file:

    1. 'slash_and_burn’
    2. 'cloudy'
    3. 'partly_cloudy'
    4. 'blooming'
    5. 'primary'
    6. 'haze'

7. 'conventional_mine'
8. ‘clear’
9. ‘road’
10. ‘selective_logging’
11. ‘agriculture’
12. ‘water’
13. ‘habitation’
14. ‘artisinal_mine’
15. ‘blow_down’
16. ‘cultivation’
17. ‘bare_ground’

Some random labeled samples from the training data set are as below:

If our algorithm can correctly label images in the testing set, these labels can be used to understand where and how deforestation has happened. For example, if an image has label ‘cultivation’ or labels denoting some specific type of cultivation, such as ‘blooming’, ‘selective_logging’, ‘blow_down’, ‘slash_and_burn’, using the location corresponding to that image(not included in data set, but Planet maintains such information in its databases), we can find where deforestation has happened. Also, the label names themselves give information about the cause of deforestation as well. Similar inferences can also be drawn from other labels such as ‘conventional mine’, ‘artisinal_mine’, ‘road’, ‘agriculture’ and ‘habitation’, as all these correspond to man-made constructions that cause deforestation. If such labels are absent in an image, that area can be considered to be unaffected by human intervention yet. Such images may have labels like ‘primary’(Primary Rain Forest), ‘water’ and ‘clear’.

III. Implementation:

As can be seen from the samples of labeled images, there can be one or more labels for every image. Hence, this is a Multi-label, Multi-class classification problem. Python, with Pandas, NumPy, OpenCV and Tensorflow were used to develop the project. The implementation process can be divided into three parts:

i. Data Pre-Processing:

The TIFF images were first normalized to 32-bit floating point numbers in the range [0, 1] using OpenCV. The labels were encoded into one-hot vectors, where the size of each vector was 17, equal to the number of labels.

These image-label pairs were then serialized to strings, and stored as Tensorflow Sequence protocol buffers in binary format, in multiple TFRecord files(shards). TFRecord is a proprietary Tensorflow file format. Performing these steps gives substantial boost in reading speed, as reading from TFRecord files is very fast.

All these operations were done using a python script that dumps the final data in the form of multiple TFRecord files.

ii. Data Reading and Augmentation:

A high-performance input pipeline was created using Tensorflow Data API. Using this pipeline, the data is read parallelly using multiple threads from multiple TFRecord shards. This pipeline also performs following tasks on the input data:

- Randomly flips the images along horizontal and vertical directions, with a probability of 0.5 each. Such flipping doesn't change the labels of the satellite image chips, but provides good data augmentation for our model.
- Randomly splits the data stream into train and validation streams, with 80%-20% train-validation ratio.
- Randomly shuffles the training data stream on every epoch
- Repeats both the streams infinitely while reading

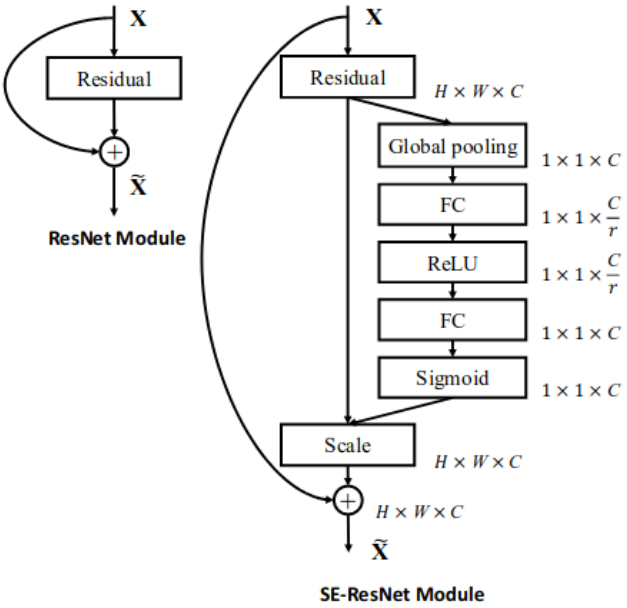
The outputs of this pipeline are the inputs for our model.

iii. Model Development and Training:

Architecture overview:

In this model, a SE ResNet-154 architecture, as described in [2] is used for extracting features from images. The SE Resnet-154, apart from having Residual connections, also consists of the Squeeze-and-Excitation blocks following all the Convolutional blocks. A schematic of a Squeeze-and-Excitation block as given in [2] is shown on next page. These blocks can be thought of as being made of two steps: Squeeze, followed by Excitation. First, Global Average Pooling over a convolutional block is performed, thereby reducing the dimensions from $H \times W \times C$ to $1 \times 1 \times C$. The dimensions are further reduced to $1 \times 1 \times \frac{C}{r}$ using a fully-connected layer, where r is an integer decided empirically, and $r = 16$ in this case. A ReLU activation is applied on the output. These three steps are collectively known as the "Squeeze" steps. The output of ReLU layer is then applied to another fully-connected layer, followed by a Sigmoid activation, bringing back the output to size $1 \times 1 \times C$. The C scalars in this vector act as the weights corresponding to the

respective *C* feature maps in the convolutional block. These steps are known as the “Excitation” steps. Finally, a channel wise multiplication of these scalars with respective feature-maps gives us the weighted convolutional block. The parameters in both the fully-connected layers are learned along with other parameters of the network. This introduces a small amount of extra computation per convolutional block, but gives a substantial boost in accuracy of most CNN architectures its applied to. A Resnet-154 with SE blocks is the current state of the art on the ImageNet classification task.

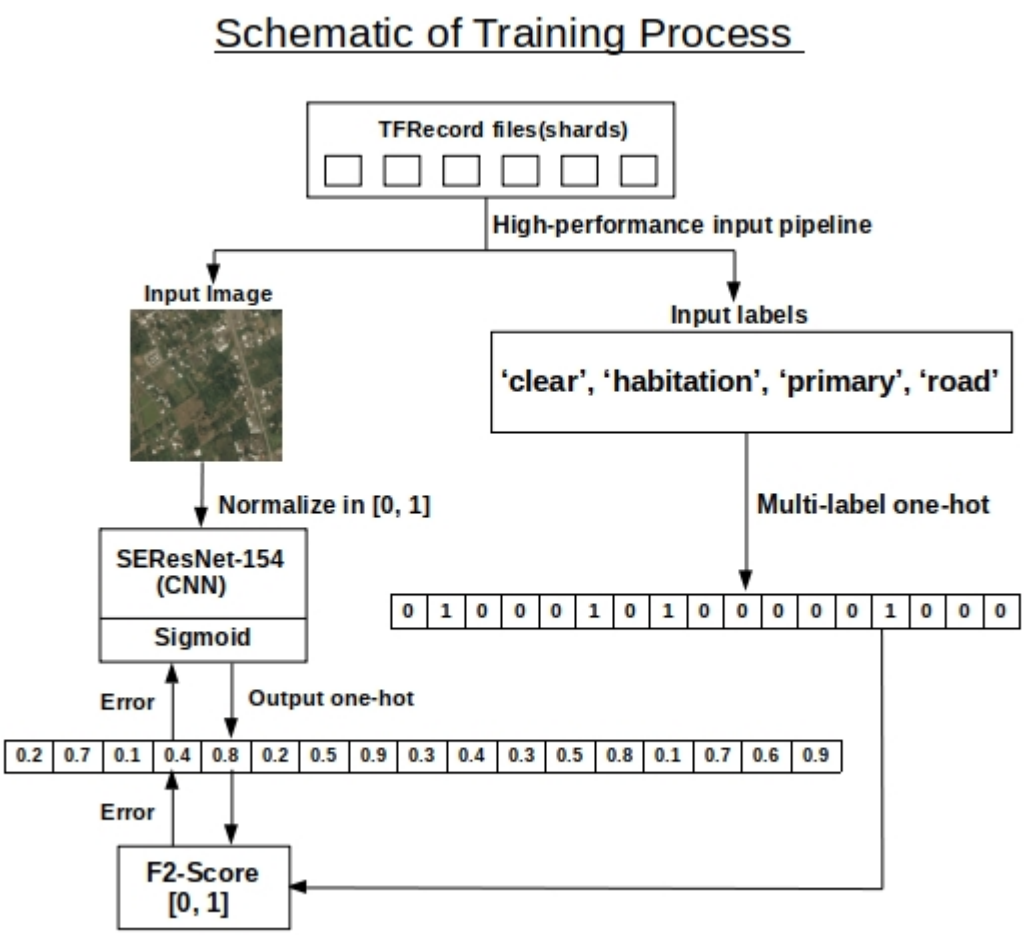


A Squeeze-and-Excitation Block in Resnet-154

Model Implementation:

Here, an existing implementation of SE ResNet-154 developed using Keras high-level API of Tensorflow is taken from [3] and used. The last layer of the model is replaced with a dense layer consisting of 17 units, and a Sigmoid activation is applied to it which gives probabilities of all the 17 labels as outputs. Note that in mulitlabel classification problem, Sigmoid activation has to be used instead of Softmax, since Softmax has a high tendency of fitting single labels to the images, while Sigmoid keeps the probabilities of individual labels independent of each other.

Furthermore, F2-score, as recommended by the authors of the Kaggle challenge is used to determine error between output one-hots and ground-truth one-hots, which is propagated back through the network for training. The F2 score is discussed further in the Results section. The schematic on the next page clearly describes the steps of the training process. Note that in the diagram, the one-hot encoding and and normalization steps are shown after creation of TFRecord files for information, just to show that these two transformations are performed. In reality, these two steps are performed before creating TFRecord files, and the data taken from pipeline is directly fed to the model.



As can be seen from the diagram, the optimization problem is to maximize the F2-score on the input samples while training. The validation was performed once per epoch and the model weights were saved every time validation loss improved. The training setup and the results after training are discussed in the next section.

IV. Results:

Training Setup: The model was trained for 50 epochs with Adam optimizer and a batch size of 24. The learning rate was varied between training, ranging from 1e-3 to 1e-5. Training was performed on Google Colaboratory, which provides an Ubuntu 18.04 based Docker image, with 12 GB of RAM and a Tesla T4 GPU with 16 GB Video Memory. Nvidia CUDA and CuDNN libraries are pre-installed in the Docker image, and are used by Tensorflow for faster training.

Evaluation Metric: F2-score, as recommended by the challenge authors, is used for evaluation on this task. Its described using below equations:

$$(1+\beta^2)\frac{pr}{\beta^2 p + r} \quad \text{where} \quad p = \frac{tp}{tp+fp}, \quad r = \frac{tp}{tp+fn}, \quad \beta = 2.$$

-[1]

where *tp*, *fp*, and *fn* are the number of *true-positives*, *false-positives* and *false-negatives* respectively. *p* and *r* are *precision* and *recall* values, and β is a constant which is equal to 2 for F2-score.

This metric ranges from [0, 1], with higher score indicating better accuracy.

Testing Results: As discussed before, there is a separate Test data, the accuracy on which is calculated by Kaggle servers based on the uploaded results. The model in this project converges on the Train/Validation data as expected, and achieves an **F2 Score of 90.3%** on the Test data.

Examples: Two examples of ground-truth vs output labels of Validation images are shown below.

Actual: agriculture, partly_cloudy, primary, water
Output: agriculture, partly_cloudy, primary, water, slash_and_burn

Actual: primary, road, habitation, clear
Output: primary, road, blow_down, habitation

References:

[1] Kaggle Challenge: www.kaggle.com/c/planet-understanding-the-amazon-from-space/overview/data

[2] Squeeze-and-Excitation Networks: arxiv.org/pdf/1709.01507.pdf

[3] SEResNet-154 Tensorflow Keras Implementation: www.github.com/titu1994/keras-squeeze-excite-network