

# Machine Learning Advanced Nanodegree

## Capstone Proposal for Classifier for Kaggle's Amazon from Space competition

Vivek Pothina

December 21<sup>st</sup>, 2018

### I. Definition


#### Project Overview




A few decades ago, Image Classification was a difficult task for computers, but now using Deep Learning we can empower satellites to label satellite images. "Planet: Understanding Amazon from Space" is competition in Kaggle. This is a multi-label classification problem too. Using a simple neural network to solve this problem would be slow and ineffective. CNN/Convnet has been used to use the spatial information to capture features of the satellite images such as cloudy or rain forest features.

#### Project Statement



The goal of this project is to create a model that can label all the satellite image chips. Deep learning techniques have become popular in the last few years as they have beat several other machine learning algorithms on various machine learning challenges. Convolutional Neural Networks are one such example, minates in the field of image classification and object detection.

This specific problem is a competition in Kaggle. The challenge is to solve multi-label classification. We begin by downloading the dataset from Kaggle from  <https://www.kaggle.com/c/planet-understanding-the-amazon-from-space/>. Then, we preprocess the data. The data is split into train, validation and test set. Then, we build a convolution model, and train the train set with it. Then, we evaluate performance using validation set and test set. Then, tweak the hyper parameters to get better performance from the model.

## Metrics

To measure the performance of the model, a validation set was created initially from the training set. Performance was measured on validation set every epoch and both log-loss and accuracy were measured.

$$accuracy = \frac{\text{number of correct predictions}}{\text{size of dataset}}$$

As this is multi-label classification, the loss function is,

$$-\frac{1}{N} \sum_{n=1}^N [y_n \log(\hat{y}_n) + (1 - y_n) \log(1 - \hat{y}_n)]$$

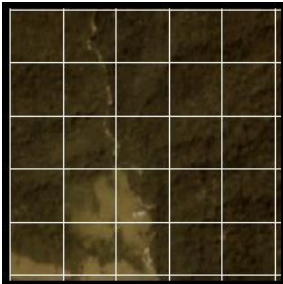
Where  $N = 17$  i.e. the number of classes/labels in the dataset,  $y_n$  is the label vector and  $\hat{y}_n$  is the prediction vector.

## II. Analysis

### Data Exploration

The data provided in Kaggle competition is present in train-jpg.tar.7z and the labels of these images are present in the train.csv file. The dataset consists of around 40479 images which belong to one or more classes among 17 classes. The images are in jpg format, each of shape (256, 256, 4). The color space is RGB + near infrared band. Each image has a unique id and the labels are given in train.csv file. The validation set has been created with a split of 80% from training set.

Label: agriculture clear habitation primary road  
Shape: (256, 256, 4) Max: 255, Min: 0



## Distribution of target class

```
{'haze': 2696, 'primary': 37512, 'agriculture': 12314, 'clear': 28430,
'water': 7410, 'habitation': 3659, 'road': 8070, 'cultivation': 4546,
'slash_burn': 208, 'cloudy': 2088, 'partly_cloudy': 7260,
'conventional_mine': 99, 'bare_ground': 861, 'artisinal_mine': 338,
'blooming': 331, 'selective_logging': 339, 'blow_down': 100}
```

No. of Labels: 17



The above plot shows the number of images labelled for each class. We can see that the labels which tell the condition of weather (clouds mostly) have been labelled for most images.

## Algorithms and Techniques

A neural network is made of input, output layers and many hidden layers. Output of one-layer passes to the next layer; an operation can be done in each layer.

Convolutional Neural Network is no different, it consists layers of convolution, maxpooling, dropout and fully connected layers (typical neural network layer)

Convolution layer: Convolution is performed on an image using a filter with strides and padding. Every pass through this layer modifies the shape of input.

Maxpooling layer: Maxpooling is an operation which is done on the filter on which convolution is taking place. Like the name sounds, maxpool takes the maximum of the values in the filter.

Fully Connected/Dense Layer: Dense layer is basic neural network layer connecting all the nodes of the previous layer.

Dropout: Dropout was a technique introduced to reduce overfitting. During training, dropout is used with a probability, using this probability, data tuples are dropped from the training input.

Using these layers, convolutional neural network identifies features from images. One or more layers of each type is used, the model is trained and evaluated on a validation set. Based on the accuracy and loss, the hyper parameters are tweaked. The model has been tested on the test set.

## Benchmark

One kaggler achieved accuracy of 93%. The link to his notebook is :

<https://gist.github.com/EKami/33ec0172590ab9f2e3a6b757c9f9dcb4>

The model was run on 20 epochs and has tried learning rates = 0.01, 0.001.

## III. Methodology

### Data Preprocessing

Before building the model, preprocessing of data needs to be carried out. These are the steps that were carried out during this project:

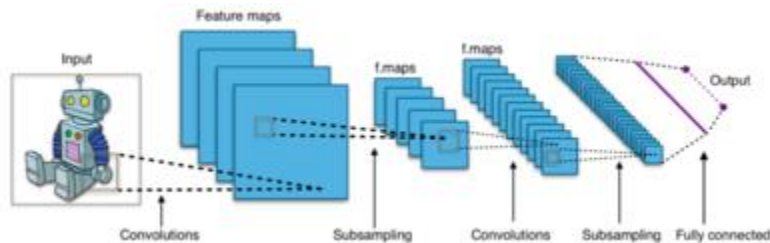
1. The images were resized to 128 x 128 pixels.
2. All three channels were used during training process as the satellite images are color images.
3. The images were normalized by dividing every pixel in every image by 255.
4. Lastly, the images were split into training, validation and test sets.

## Implementation



Below is the brief description of CNN and different layers of CNN as specified in a Wikipedia article.

*“In machine learning, a convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery.”*

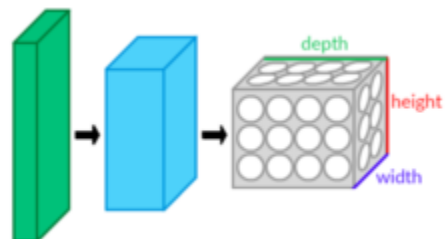


*A CNN consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers and normalization layers.*

*Description of the process as a convolution in neural networks is by convention. Mathematically it is a cross-correlation rather than a convolution. This only has significance for the indices in the matrix, and thus which weights are placed at which index.*

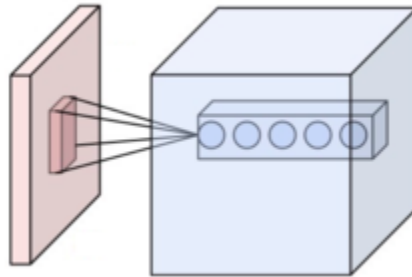
### Convolutional

*Convolutional layers apply a convolution operation to the input, passing the result to the next layer. The convolution emulates the response of an individual neuron to visual stimuli. Each convolutional neuron processes data only for its receptive field.*



*Although fully connected feedforward neural networks can be used to learn features as well as classify data, it is not practical to apply this architecture to*

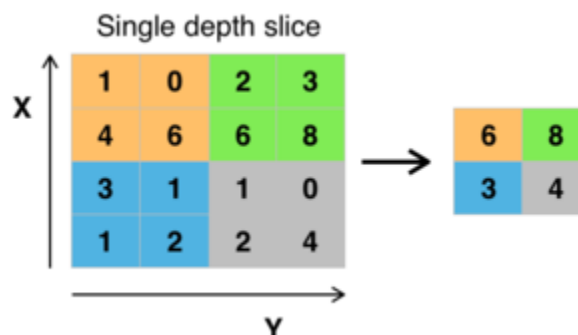
images. A very high number of neurons would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable.



For instance, a fully connected layer for a (small) image of size  $100 \times 100$  has 10000 weights for each neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters. For instance, regardless of image size, tiling regions of size  $5 \times 5$ , each with the same shared weights, requires only 25 learnable parameters. In this way, it resolves the vanishing or exploding gradients problem in training traditional multi-layer neural networks with many layers by using backpropagation.

## Pooling

Convolutional networks may include local or global pooling layers, which combine the outputs of neuron clusters at one layer into a single neuron in the next layer. For example, max pooling uses the maximum value from each of a cluster of neurons at the prior layer. Another example is average pooling, which uses the average value from each of a cluster of neurons at the prior layer].



## **Fully connected**

*Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP).*

## **Weights**





*CNNs share weights in convolutional layers, which means that the same filter (weights bank) is used for each receptive field in the layer; this reduces memory footprint and improves performance.*



Following are steps carried out for training a CNN:

1. The entire dataset is pre-processed and split into train, validation and test sets.
2. A simple CNN model is created.
3. First layer is Convolutional layer with inputs 32 and filter size as 3,3
4. Next layer is maxpooling layer with pool size as 2,2
5. It is followed by another Convolution layer of inputs 48 and filter size of 3,3
6. Next layer is maxpooling layer with pool size as 2,2
7. It is followed by another Convolution layer of inputs 64 and filter size of 3,3
8. Next layer is maxpooling layer with pool size as 2,2
9. Till now the layers have used relu as their activation function and padding is 'same'.
10. Next layer is dropout layer with 0.7 as probability.
11. Next the model is flattened as a layer and then, fully connected layer of inputs 512 is present with activation function as relu.
12. The output layer is a fully connected layer with sigmoid as activation function.
13. The model is compiled with Adam optimizer and 'binary\_crossentropy' as the loss function.
14. The model is trained for 10 epochs with batch size of 128.

## Refinement



To get the initial result  a simple CNN architecture was built and evaluated.  This resulted in decent loss. The hyperparameters were  tuned a couple of times. Drop out layer was added to account for overfitting. Training was initially carried out with 20 epochs  which was later reduced to 10 epochs. Batch size was not changed, using Google Colab for this project helped a lot in the computation aspect of this project.

## IV. Results

### Model Evaluation and Validation

During model generating and development, a validation set was used to evaluate the model. A test set was also created initially, finally the model was evaluated on this test set. Accuracy of 0.905385 achieved.

### Justification

The final solution has been thoroughly analyzed.  The results of this project didn't cross the project discussed in Benchmark section  but the final solution is significant enough to have solved the problem. The model has a decent accuracy based on leaderboard of Kaggle.

## V. Conclusion

### Free-Form Visualization