Machine Learning Engineer

CAPSTONE PROPOSAL

Amartya Kalapahar

Date: 26/09/18

TGS SALT IDENTIFICATION CHALLENGE

Domain Background

Several areas of Earth with large accumulations of oil and gas *also* have huge deposits of salt below the surface.
But unfortunately, knowing where large salt deposits are precisely is very difficult. Professional seismic imaging still requires expert human interpretation of salt bodies. This leads to very subjective, highly variable renderings. More alarmingly, it leads to potentially dangerous situations for oil and gas company drillers.

Problem Statement

To create the most accurate seismic images and 3D renderings, TGS (the world's leading geoscience data company) is hoping Kaggle's machine learning community will be able to build an algorithm that automatically and accurately identifies if a subsurface target is salt or not.The goal is to segment regions that contain salt.

Datasets and Inputs

Seismic data is collected using reflection seismology, or seismic reflection. The method requires a controlled seismic source of energy, such as compressed air or a seismic vibrator, and sensors record the reflection from rock interfaces within the subsurface. The recorded data is then processed to create a 3D view of earth's interior. Reflection seismology is similar to X-ray, sonar and echolocation.

A seismic image is produced from imaging the reflection coming from rock boundaries. The seismic image shows the boundaries between different rock types. In theory, the strength of reflection is directly proportional to the difference in the physical properties on either sides of the interface. While seismic images show rock boundaries, they don't say much about the rock themselves; some rocks are easy to identify while some are difficult.
There are several areas of the world where there are vast quantities of salt in the subsurface. One of the challenges of seismic imaging is to identify the part of subsurface which is salt. Salt has characteristics that makes it both simple and hard to identify. Salt density is usually 2.14 g/cc which is lower than most surrounding rocks. The seismic velocity of salt is 4.5 km/sec, which is usually faster than its surrounding rocks. This difference creates a sharp reflection at the salt-sediment interface. Usually salt is an amorphous rock without much internal structure. This means that there is typically not much reflectivity inside the salt, unless there are sediments trapped inside it. The unusually high seismic velocity of salt can create problems with seismic imaging.

The data is a set of images chosen at various locations chosen at random in the subsurface. The images are 101 x 101 pixels and each pixel is classified as either salt or sediment. In addition to the seismic images, the depth of the imaged location is provided for each image.

The dataset can be downloaded from here : https://www.kaggle.com/c/10151/download-all

The dataset contains 3 csv files and 2 zip files namely:

1. depths.csv : This file contains the depth underground (in feet) of each image. It has 2 columns namely id(A unique image identifier) and z(depth in feet). There are in total 22001 unique values in this file.
2. train.zip : This file contains the set of images required for training purpose. It includes both image and salt segmentation mask.

3. test.zip : This file contains set of images required for prediction.
4. train.csv : A helper file that shows the training set masks in run-length encoded format. It has two columns namely id(a unique image identifier) and rle_mask. There are in total 4001 unique values in the file.
5. sample_submission.csv : A sample submission showing the correct submission format. This file also contains 2 columns namely id(a unique image identifier) and rle_mask.

Solution Statement

A deep learning algorithm will be developed using Tensorflow/Keras and will be trained with training data. Specifically a CNN will be implemented in Tensorflow/Keras. Predictions will be made on the test data set and will be evaluated.

Benchmark Model:

The model with the Public Leaderboard score of 0.231 will be used as a benchmark model. Link to the kernel :https://www.kaggle.com/christofhenkel/keras-baseline/notebook

Attempt will be made so that score reaches above 0.5 in the leaderboard.

Evaluation:

This competition is evaluated on the mean average precision at different intersection over union (IoU) thresholds. The IoU of a proposed set of object pixels and a set of true object pixels is calculated as:

$$IoU(A,B) = \frac{A \cap B}{A \cup B}.$$

The metric sweeps over a range of IoU thresholds, at each point calculating an average precision value. The threshold values range from 0.5 to 0.95 with a step size of 0.05: (0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95). In other words, at a threshold of 0.5, a predicted object is considered a "hit" if its intersection over union with a ground truth object is greater than 0.5.

At each threshold value t, a precision value is calculated based on the number of true positives (TP), false negatives (FN), and false positives (FP) resulting from comparing the predicted object to all ground truth objects:

$$\frac{TP(t)}{TP(t) + FP(t) + FN(t)}.$$

A true positive is counted when a single predicted object matches a ground truth object with an IoU above the threshold. A false positive indicates a predicted object had no associated ground truth object. A false negative indicates a ground truth object had no associated predicted object. The average precision of a single image is then calculated as the mean of the above precision values at each IoU threshold:

$$\frac{1}{|thresholds|} \sum_t \frac{TP(t)}{TP(t) + FP(t) + FN(t)}.$$

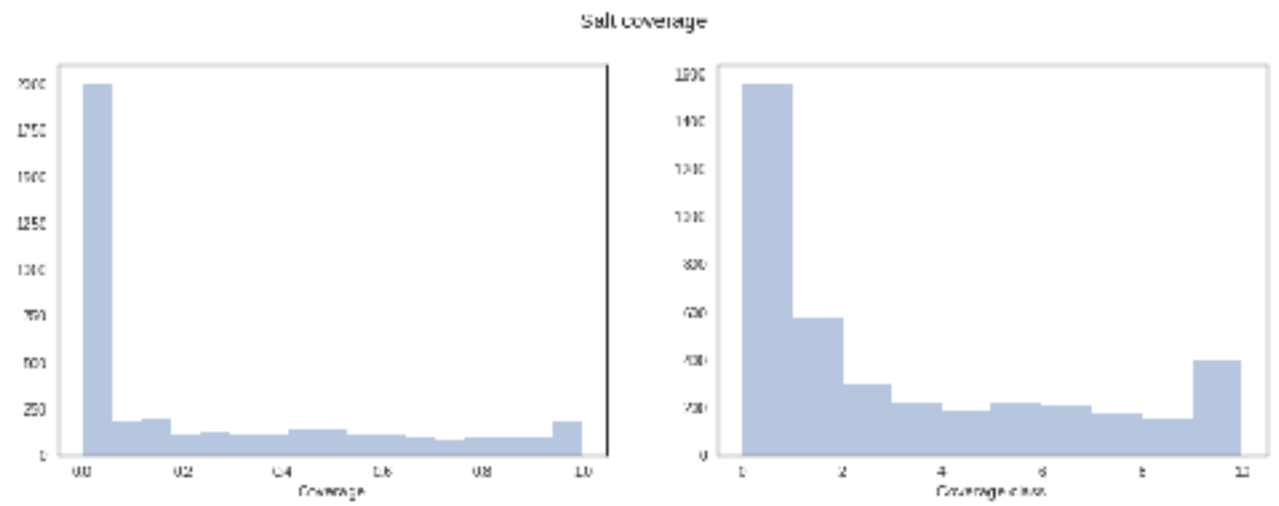
Lastly, the score returned by the competition metric is the mean taken over the individual average precisions of each image in the test dataset.

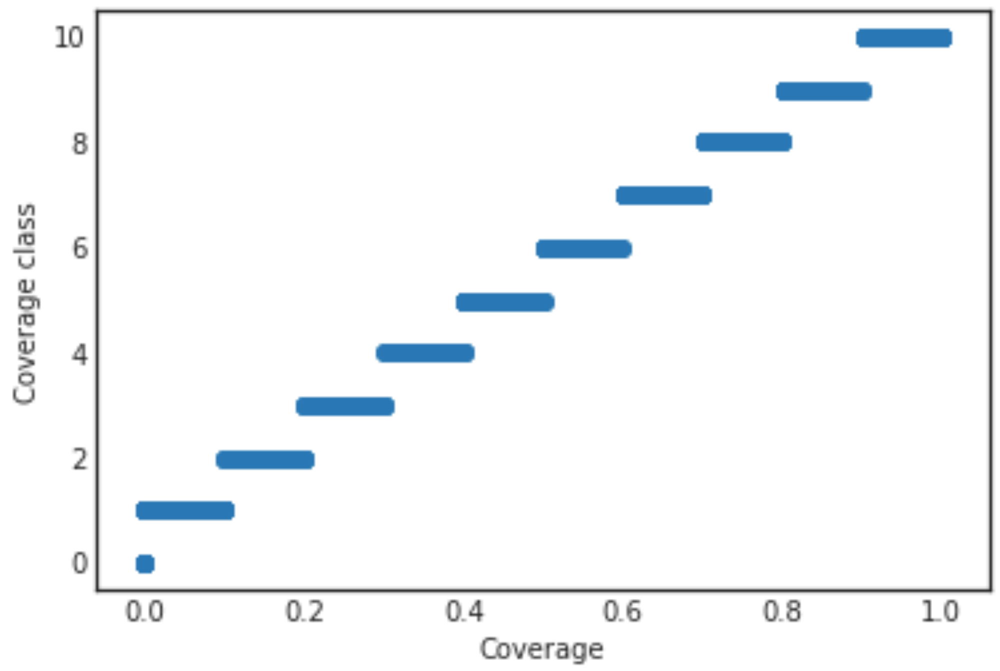
Project Design:

CNN class of deep learning algorithm can be employed for this problem. Initially data exploration will be carried out to understand the data in a more accurate way.This will help preprocess the data and can end up with better predictions. After this necessary preprocess functions will be implemented , data will be randomized and CNN will be implemented in Tensorflow/Keras. Finally necessary predictions on the test data will be carried out and these will be evaluated.

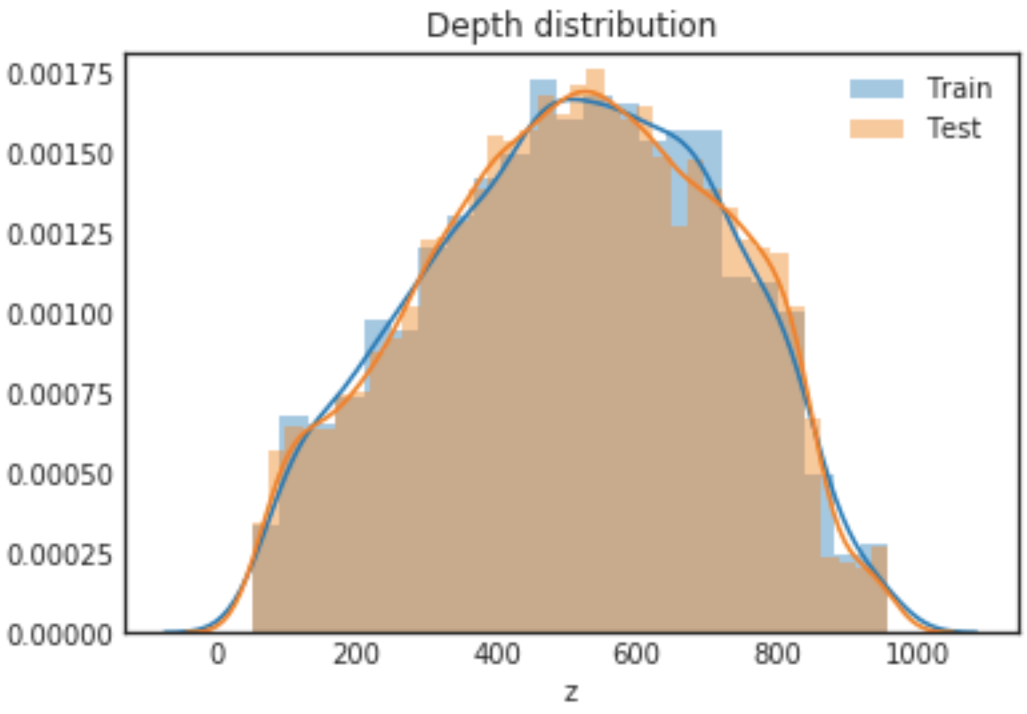
Some data visualizations:

Calculating the salt coverage and salt coverage classes:

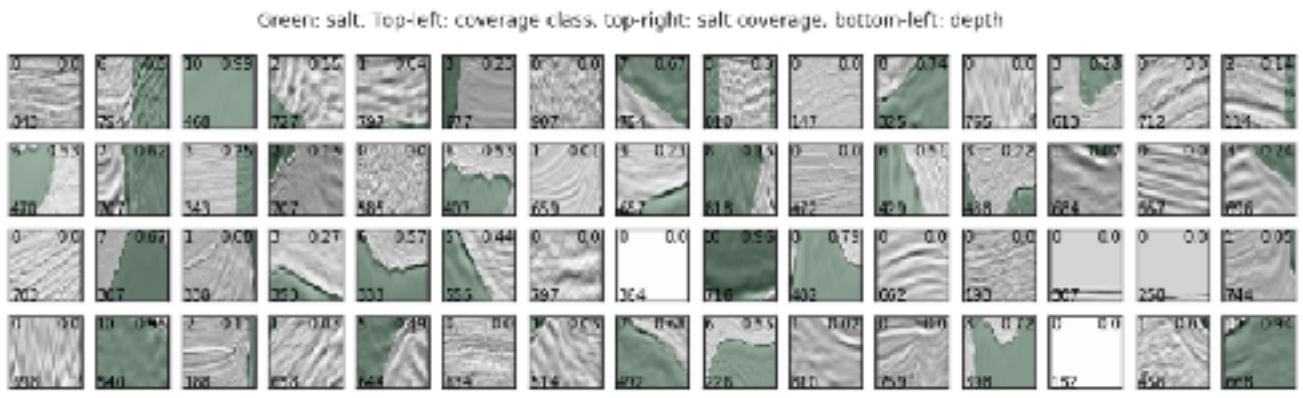




Plotting the depth distributions:



Showing some example images:


Green: salt. Top-left: coverage class. top-right: salt coverage. bottom-left: depth