**TGS Salt Identification Challenge: Final Report**

Segment salt deposits beneath the Earth's surface

Jeremy Lawson

**Abstract:**

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.

**Background:**

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.

**Objectives:**

* 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.
* Create predictions and visualizations using python.

**Development Platforms:**

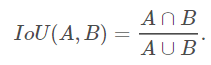
This project was developed on a Hp zbook G3 laptop. CPU: Intel Xeon E3-1505M v5 running at 2.8GHz with Turbo to 3.7GHz; Graphics: Nvidia Quadro M1000M, Intel HD Graphics P530; RAM: 32GB DDR4

**Proposed Visualizations:**

The proposed visualizations are below:

* + Salt Coverage (Data) Bar Chart
  + Depth Distribution Histogram
  + 3 Train History (display of loss and my\_iou\_metric)

**Functional Requirements:**

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:

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.

**System Architecture and Description:**

In the python script, a U-net model is built using a Resnet34 encoder and a standard decoder for semantic segmentation of images provided for the TGS salt segmentation problem. An 80-20 split of the train data is used for training and validation of the model. To increase the size of the training dataset, the training data is augmented by including images (and masks) that have been flipped along the x-axis.

Figure 1 and 2 show description of the Salt Coverage and the Depth Distribution from the datasets provided.

Fig. 1 Salt Coverage

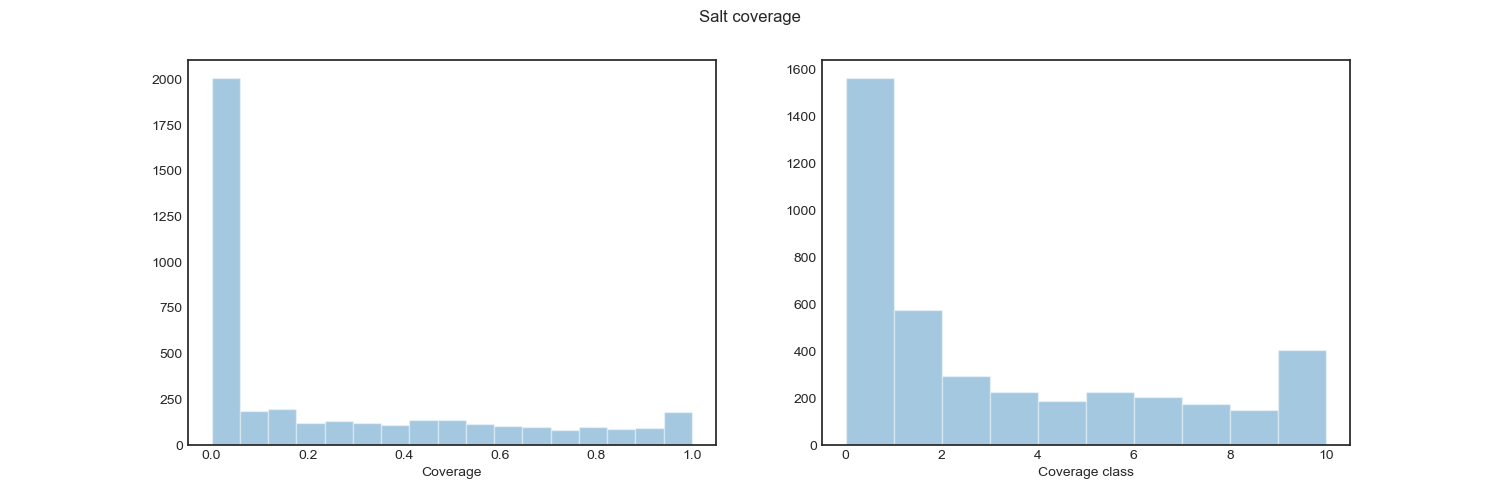
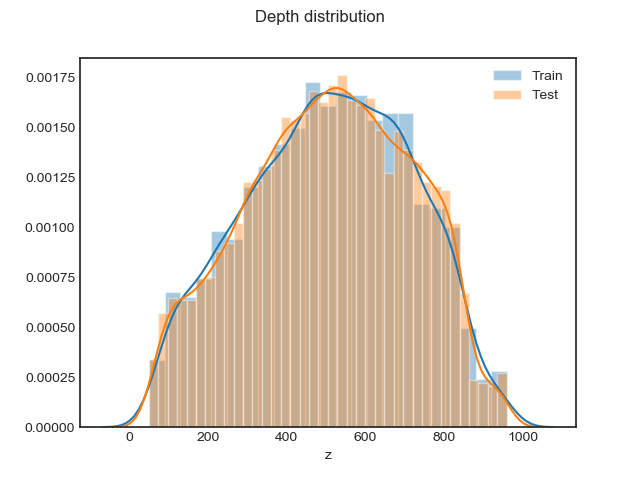


Fig. 2 Depth Distribution



**Experimental Analysis and Conclusions:**

The Adam optimizer is used to the train the model using a sequence of loss functions starting with the binary-cross entropy, binary-cross entropy plus dice loss, and finally the binary-cross entropy plus dice loss plus mean squared error. The objective of introducing these different loss functions is to gradually move the set of weight parameters to a part of weight-space where the masks can be better reproduced. In a way the sequential use of loss functions acts as a form of regularization.

Fig. 3 Train History 1

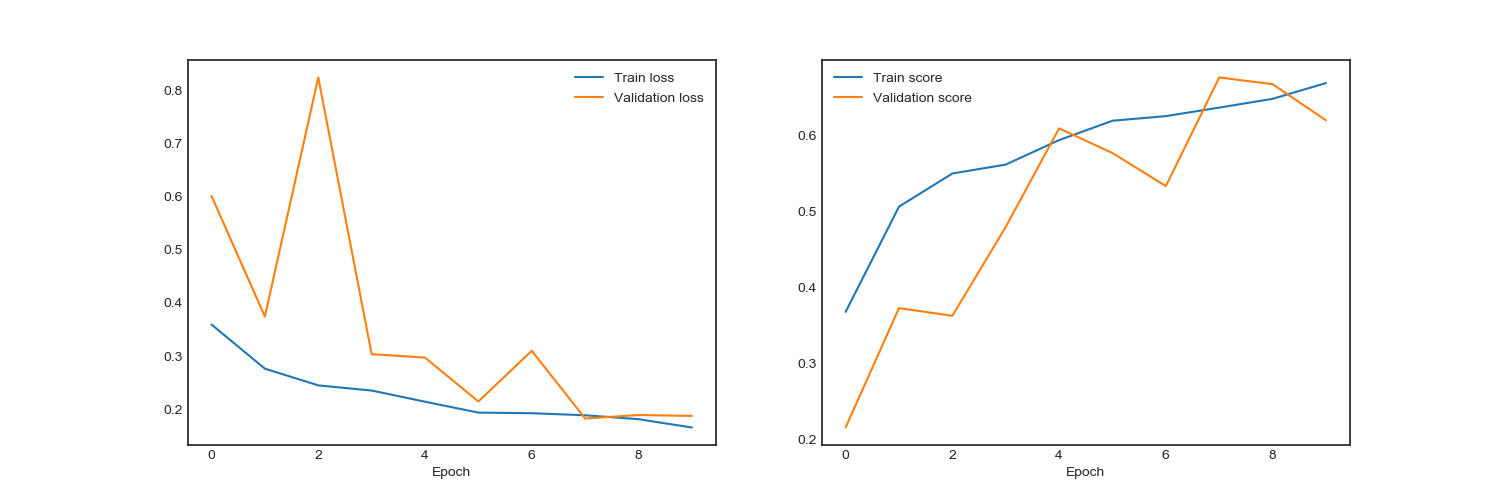


Fig. 4 Train History 2

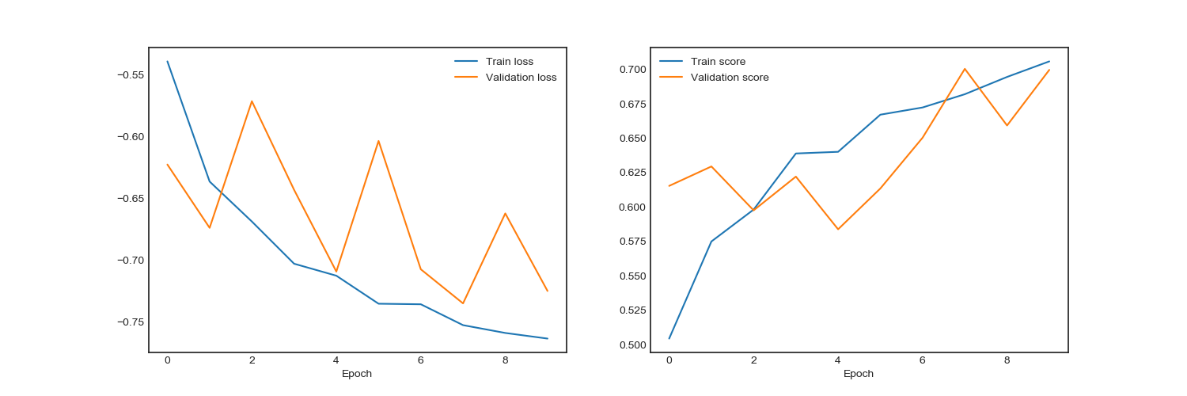
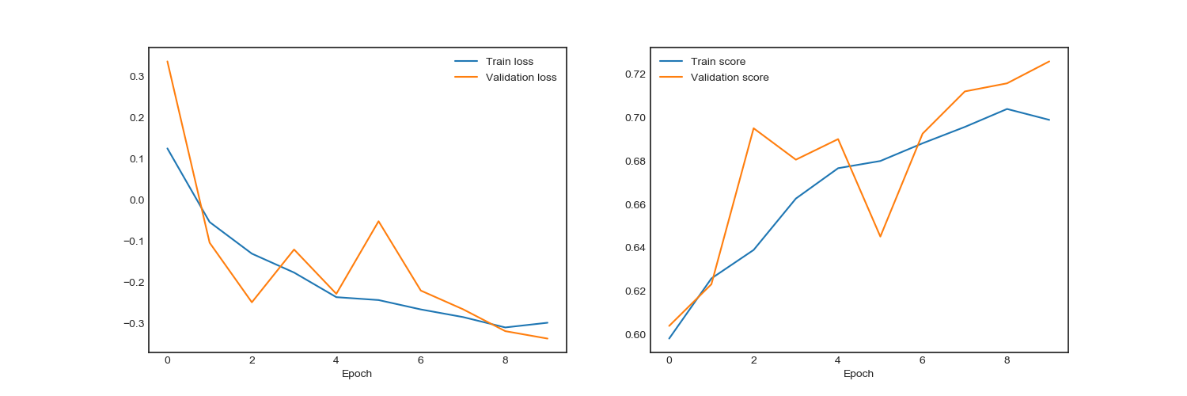
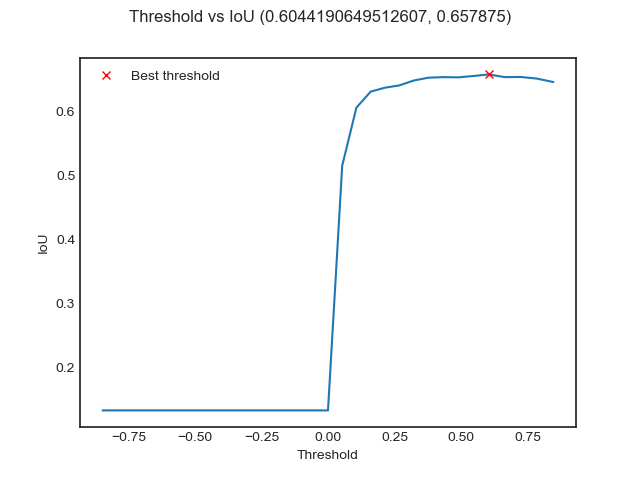


Fig. 5 Train History 3



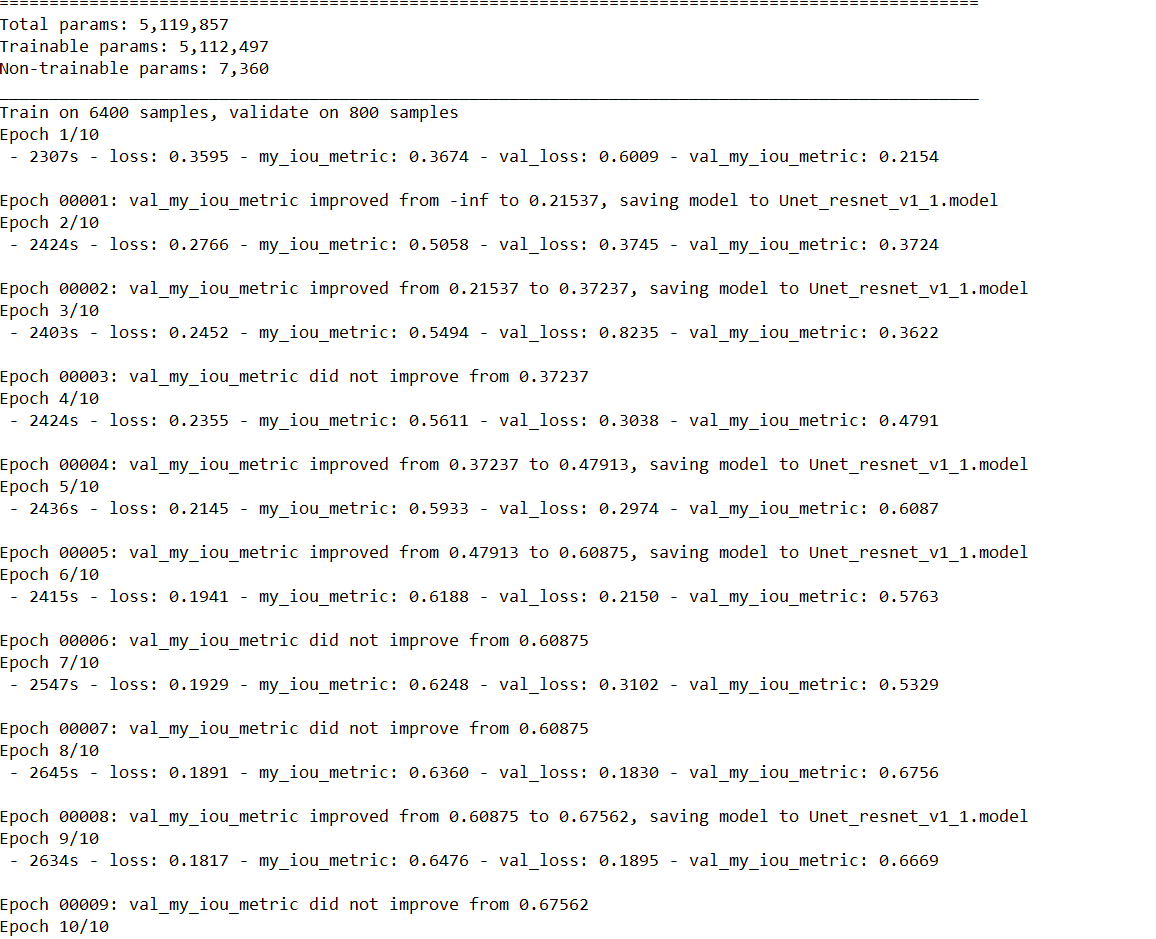
Since the performance of the model for this competition was evaluated in terms of the mean of the precision over a range of Intersection over Union(IoU) threshold values in the range 0.5 to 0.95, these values were evaluated for each IoU threshold and the optimal threshold value estimated from the validation data.

Fig. 6 Threshold vs IoU



Finally, mask predictions for the test data were made and their run-length encodings saved for submission. Below are results from the model and the score from the Kaggle competition. The IoU metric improved with each stage in the code however it did not get above 70% when ran against the full test dataset for the competition.

Table 1 – Model Results



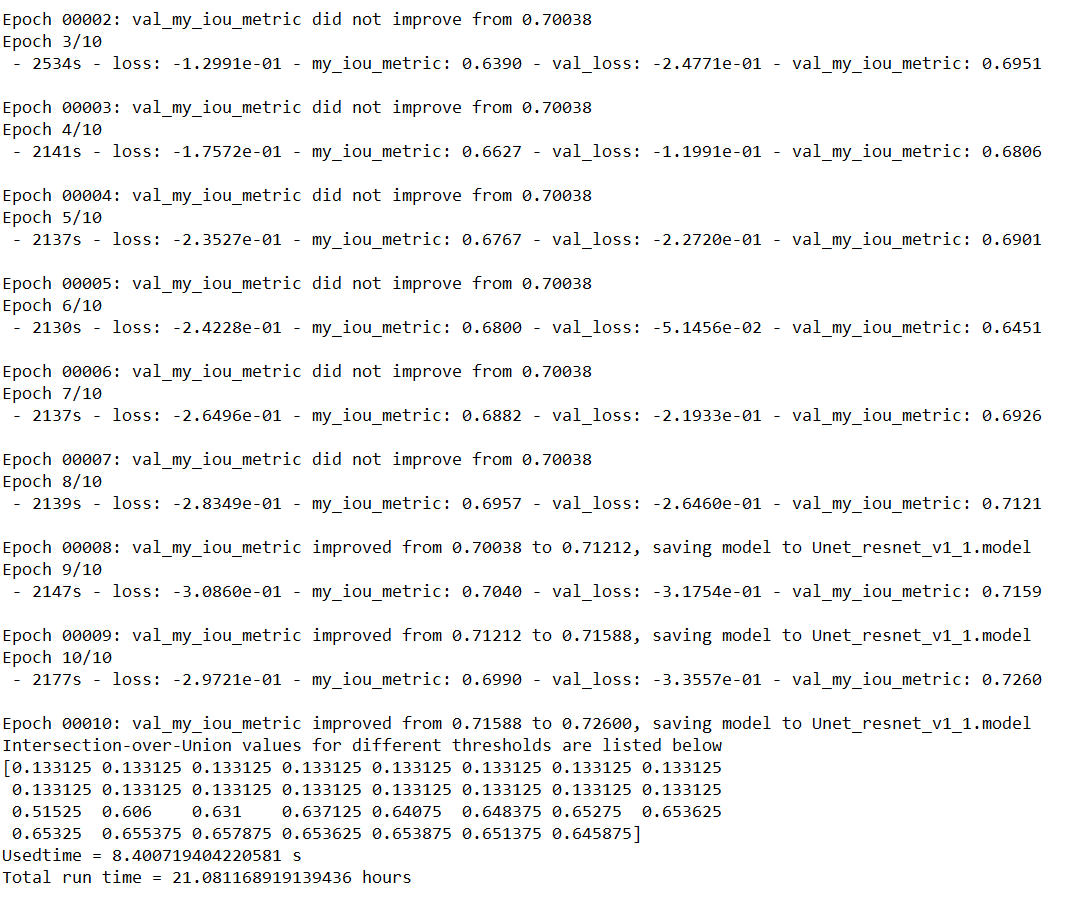


Fig. 7 – Kaggle Score

