Machine Learning Engineer Nanodegree

Capstone Project

Felipe Alves
September 15st, 2018

I. Definition

Project Overview

This project aims to provide a solution to Kaggle's TGS Salt Challenge. The Challenge consists in predicting if a target surface is salt or not.

Several areas of Earth with large accumulations of oil and gas also have huge deposits of salt below the surface. Knowing where large salt deposits are precisely is very difficult and it may lead potentially dangerous situations for oil and gas company drillers [1].

TGS is the company sponsoring this challenge and they have provided seismic images which this project will make use of.

Problem Statement

The problem stated by TGS (the company sponsoring this Kaggle challenge) is to build an algorithm that automatically and accurately identifies if a subsurface target is salt or not [1].

To solve the problem a Deep Learning model will be developed. The model is a Convolutional Neural Network algorithm, specially designed to learn how to segment images. The CNN will learn from the images provided by TGS. There are two sets of images, the ones that have a respective mask (ground-truth) which will be used to train a model and the ones without the ground-truth that will be used to test the model.

Metrics

The metric used to quantify the performance of both the benchmark model and the solution model is Intersection over Union metric. This is the metric used by Kaggle in this challenge.

Intersection over Union is an evaluation metric used to measure the accuracy of an object detector [2]. For this project, it will compare the ground-truth mask with the predicted mask. Intersection over Union can be mathematically defined as:

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$$

II. Analysis

Data Exploration

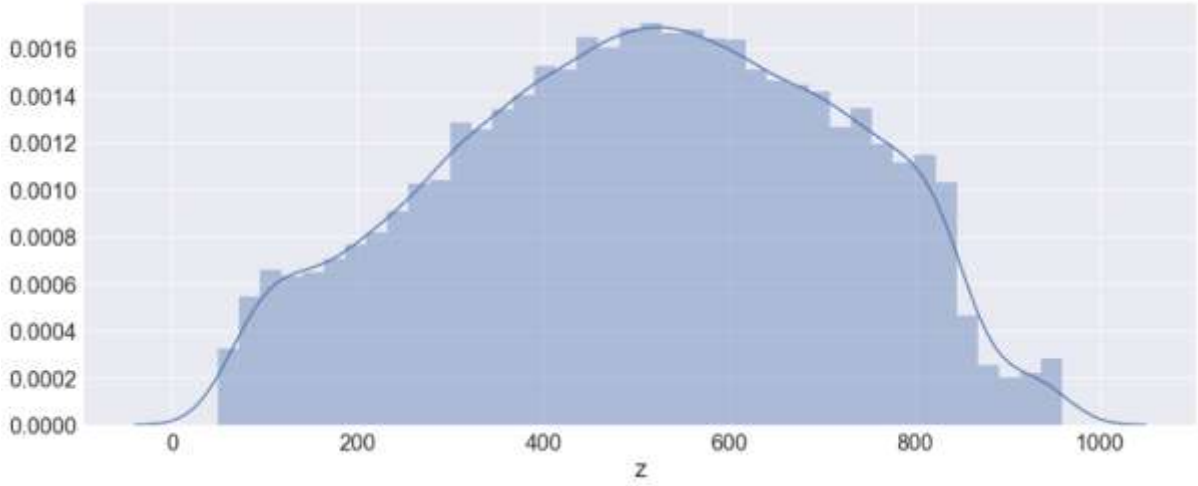
In this project, we will use the TGS Salt Identification Challenge dataset, available on Kaggle [1]. The dataset contains 26,000 images in total. Table 1 explains how they are distributed. The images are 101x101x3 PNG format, though they have 3 channels they are all greyscale.

Table 1 – Images from the dataset

Category	Number of Images	Description
Train	4,000	Images used for training the model
Mask	4,000	Images used as masks for training set (ground-truth)
Test	18,000	Images used for testing the model

Each image is named with a unique id that can be found in the "depths.csv" file, available together with the images. The "depths.csv" file also contains the seismic images depth. The figure below shows the histogram for depth feature with Seaborn's a kernel density estimate (KDE) [3].

Figure 1 – Depth distribution

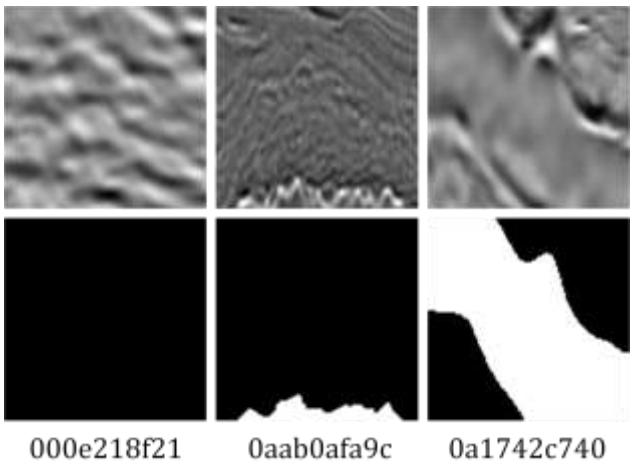


As it can be seen from the plot, the depths are in a range from 0 to 1000 and most images were taken around 500 units deep.

Exploratory Visualization

Seismic imaging is a tool that bounces sound waves off underground rock structures to reveal possible crude oil and natural gas bearing formations [4]. The input data contains 4,000 seismic images in training set and 18,000 images in testing set. As explained, the training images contain a respective mask, which is their ground-truth. Below are some examples of images found in the train set and their respective ground-truth.

Figure 2 – Different images and their masks



The white area in the masks shows where the salt is. Id 000e218f21 has no salt at all.

Algorithms and Techniques

There are two main techniques and two algorithms used in this project. The techniques are Semantic Segmentation and Transfer Learning. The two algorithms are U-net and Inception V3.

Semantic segmentation is one of the key problems in the field of computer vision [5]. It is the field of knowledge responsible for understanding the elements in an image in pixel level [6]. The goal of semantic image segmentation is to label each pixel of an image with a corresponding class of what is being represented [7]. The image below shows an example of semantic segmentation.

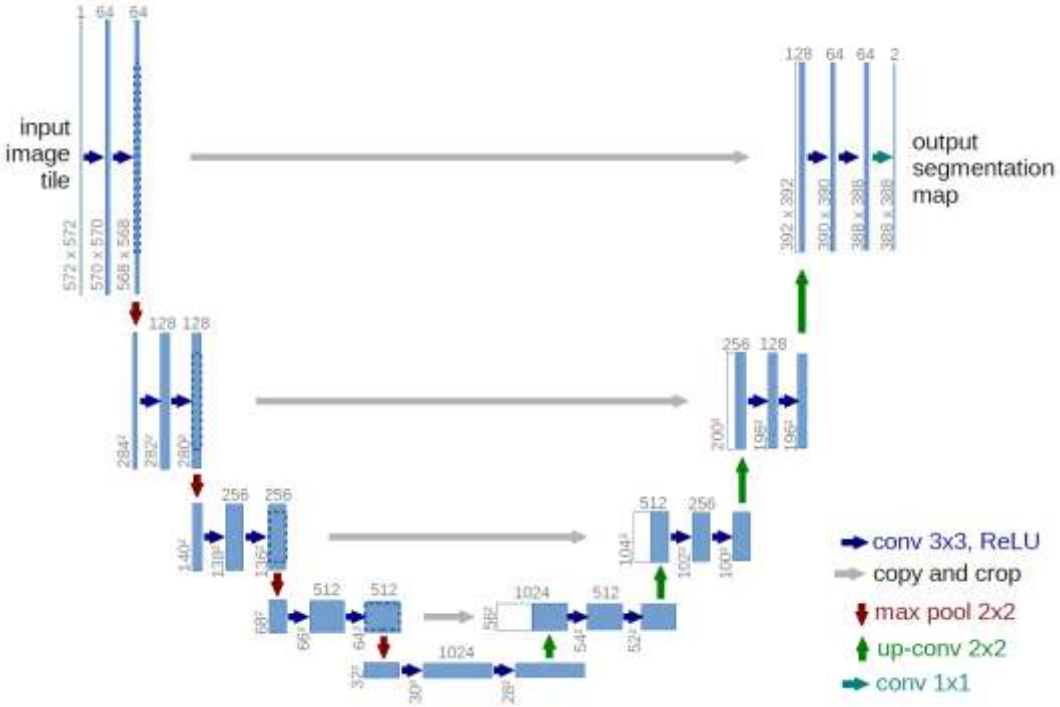
Figure 3 – Examples of semantic segmentation

Source: http://www.cs.toronto.edu/~tingwuwang/semantic_segmentation.pdf

In this project, semantic segmentation will be used to classify each pixel as having salt or not.

The algorithm used for semantic segmentation is U-net, an algorithm widely used for biomedical image segmentation. The image below shows U-net architecture.

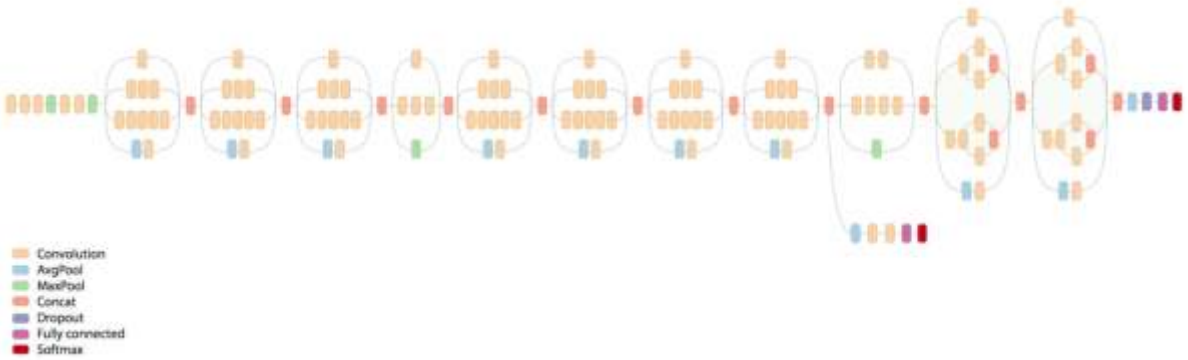
Figure 4 – U-net model



Source: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/

The second technique, Transfer learning, is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task [8]. This is useful because, as the name says, makes possible to transfer "knowledge" from a trained model to a new one. The image below shows Inception's V3 architecture.

Figure 5 – Inception V3 model



Source: https://becominghuman.ai/transfer-learning-retraining-inception-v3-for-custom-image-classification-2820f653c557

This project will use Inception V3 CNN with pre loaded weights from ImageNet dataset to transfer its knowledge to a new architecture. Inception's abilities in recognizing shapes, edges and shadows are expected to be helpful when feeding data to U-net perform seismic image segmentation.

Default parameters in the architecture are defined as the following: 224x224 is the image input size, patience is set to 5 to avoid overfitting and validation set is 10% of training set.

Benchmark

The benchmark model will be Jesper solution using only U-net [9]. His model uses only U-net and achieved an IoU score of 0.62. This project will try to enhance his model by using transfer learning from Inception V3 with pre loaded weights from ImageNet.

III. Methodology

Data Preprocessing

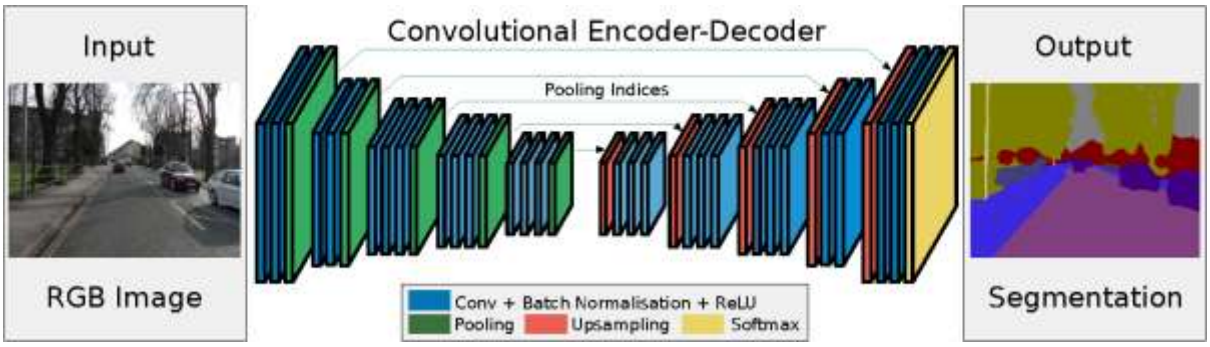
The data was preprocessed to resize all images to match Inception's V3 input size. They were scaled from 101x101 to 224x224 using *resize* function from *skimage.transform* module.

Implementation

To segment the images we will work with semantic segmentation methodologies as explained in section I – "Algorithms and Techniques" of this project.

The architecture for the final model is an encoder-decoder type. The encoder part is where you apply convolution blocks followed by a maxpool downsampling to encode the input image into feature representations at multiple different levels [10]. Figure 6 shows a representation of the encoder-decoder architecture.

Figure 6 – Encoder-Decoder Architecture



Source: https://www.semanticscholar.org/paper/SegNet%3A-A-Deep-Convolutional-Encoder-Decoder-for-Badrinarayanan-Kendall/673fbe8419f3444690175569febe29edfdadfd62

The encoder for this model is Inception V3. Include-Top was set to false to remove the final softmax layers, so the CNN will work as a feature extractor. The pre-loaded weights from ImageNet are already trained to identify a series of patterns in the image, and each layer can identify more elaborate patterns. Inception works with alternating convolution and pooling operations, progressively DownSampling feature maps [11]. Since we removed the final layer, Inception's last layer is Mixed 10, after that the model starts UpSampling.

The second part of our model is based on U-net. It consists of UpSample and concatenation followed by convolution operations. The decoder is connected after Mixed10 layer from Inception and starts expanding the feature dimensions to meet the same size with the corresponding concatenation blocks from the left [11].

The model will be trained on a p2.xlarge instance of Amazon EC2 (AMI) with epochs set to 30 and patience set to 5. The training will use IoU metric. The best weights for this problem will be saved on 'model-tgs-salt-1.h5' file. The training set will use 90% of the original set (3,600 images), the other 10% (400 images) are for validation.

To test the model, all the images will be resized to match the required input and they will be tested using the best weights generated by the training step. The resulting masks will then be resized back to 101x101 (original size) and be transformed into a csv file, ready for submission.

Refinement

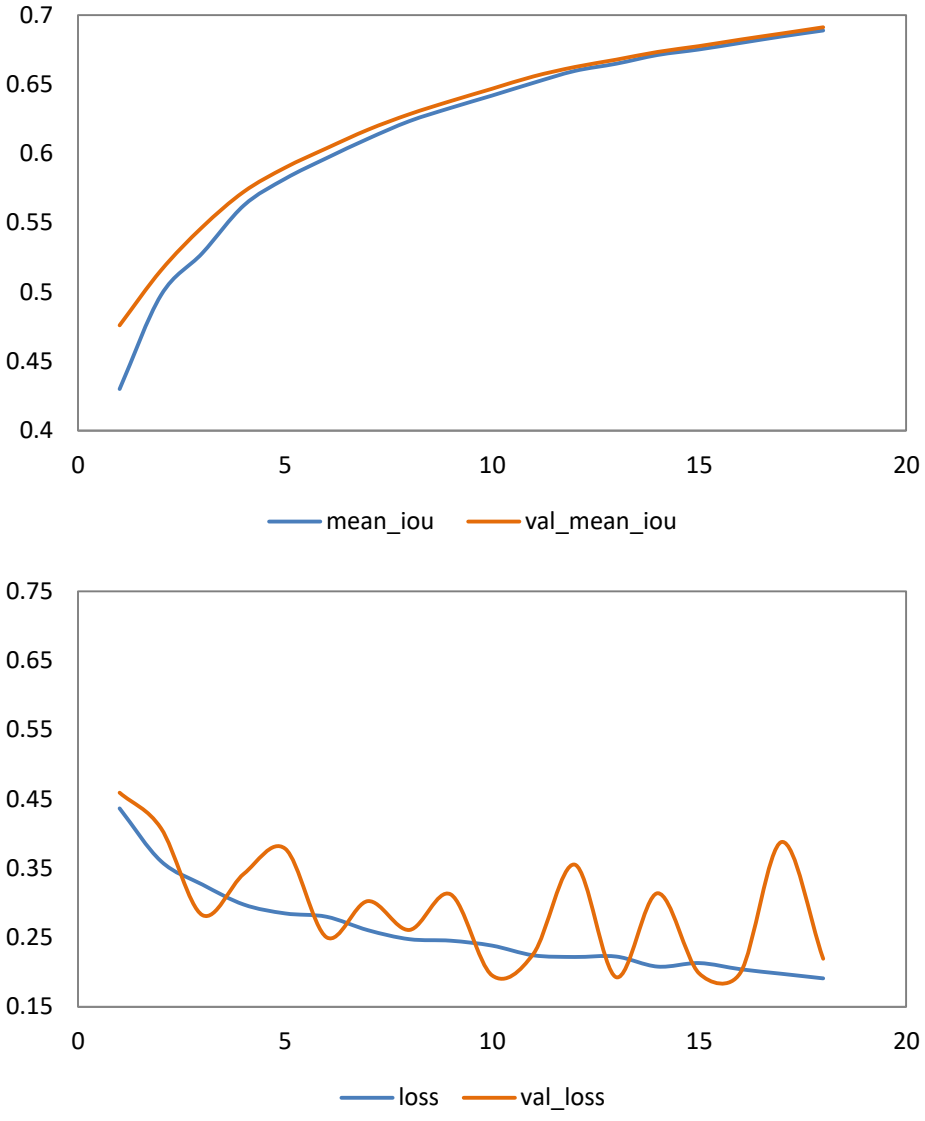
Due to computational cost, one of the big concerns for this architecture was to avoid overfitting. This was successfully achieved by setting parameter *patience* to 5.

IV. Results

Model Evaluation and Validation

The model was monitored using the IoU metric. Figure 7 shows the learning evolution.

Figure 7 – IoU and Loss metrics



The model was trained for 18 epochs before the early stopping parameter was met. From the figure, is possible to observe the learning curve going to 0.69. Both training and validation set metrics have the same curve profile, with very low variation.

For the loss plot, the training set is constantly decreasing, getting close to 0.15, but on validation set, the loss had a lot of variance.

After submitting the prediction to Kaggle, the model got a score of 0.634 for IoU metric. It proves that the model was able to learn using just 18 epochs of training and outperforming our benchmark. When comparing the train, validation and test set scores, is possible to see the model generalizes well to unseen data. The difference among the set's final score is 0.0572.

Justification

The following table shows the results for the model and the benchmark.

Table 2 – Results

Category	IoU
Model	0.634
Benchmark	0.620
No-Mask	0.380

This project's final model presented a result of 0.634, a slightly better result than our benchmark, which scored 0.620. In order to compare the models, a "no-mask" submission has a IoU score of 0.38 [12]. The "no-mask" submission allows us to affirm that both approaches have a result better than submitting a blank csv to the platform.

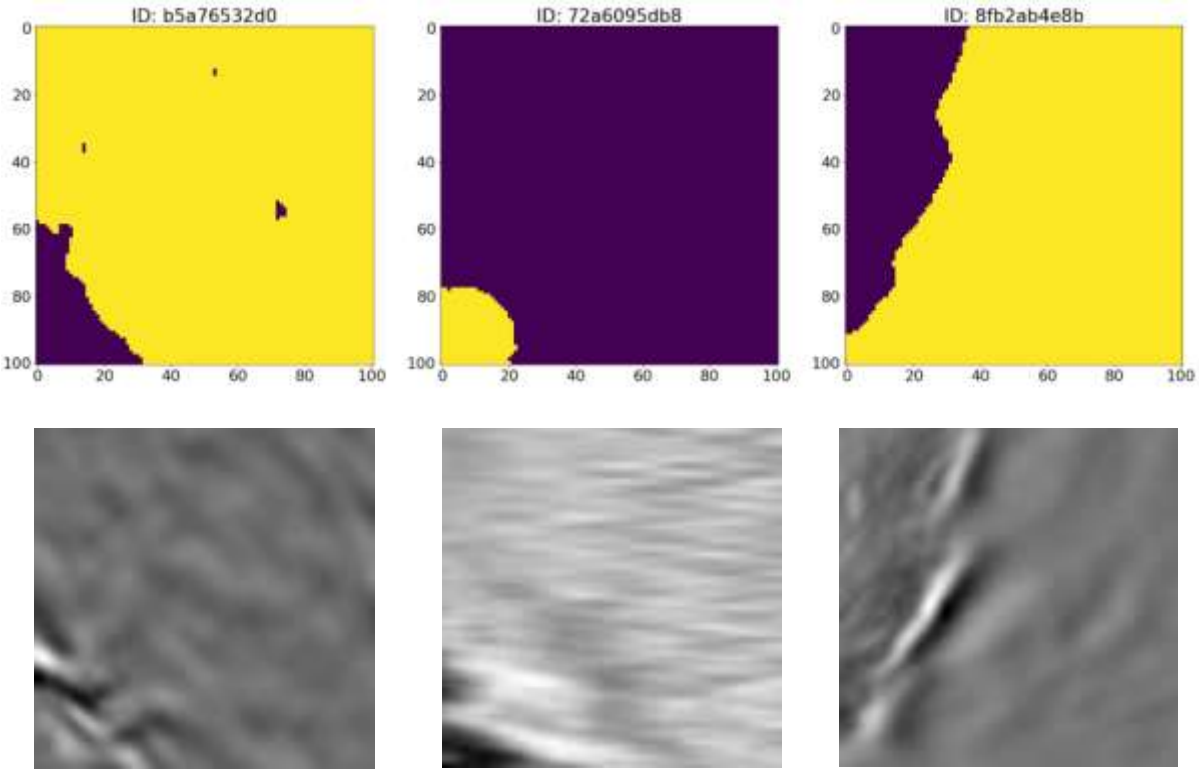
The training time for Inception V3 + U-net model was 3h12min. Since the encoder adds a lot of computing time for an IoU increase of just 0.014, the model does not have a good overall performance.

V. Conclusion

Free-Form Visualization

Figure 8 brings some predicted masks versus the original images.

Figure 8 – Predicted masks



The yellow part is the prediction for salt and the purple is the portion of the image without salt. It is possible to see that shadows – or darker areas – have a great influence in determining the masks. Also, the predictions for salt seem to be related with the smoothness of the area in the image.

Reflection

The model used in this project is an end-to-end solution that aims to explore the encoder-decoder architecture using transfer learning. Inception v3 and u-net were successfully implemented using ImageNet pre-loaded weights and the model was able to segment images for TGS Salt Challenge.

The *state of the art* techniques used in this project shows us that Deep Learning is a very interesting field of knowledge because it allows us to design solutions for a vast range of areas, such as geophysics, biomedicine and engineering.

The difficult aspect of this project was working with large amount of images, which required a powerful GPU and a lot of computational power. On the other hand, this is what makes Deep Learning such an intriguing matter: having the chance to explore new algorithms, neural net architectures and discover new solutions to complex problems makes it a fascinating challenge.

Improvement

Further studies might include image augmentation, use of depths and stratification. The model could also be tested with no pre-loaded weights to check if the IoU will increase.

There are also other backbones that could be tested, such as VGG16, VGG19, ResNet18, ResNet34, ResNet50, ResNet101, ResNet152, ResNeXt50, ResNeXt101, DenseNet121, DenseNet169, DenseNet201 and ResNet V2. Another suggestion is to use FPN as model decoder.

References

1 - https://www.kaggle.com/c/tgs-salt-identification-challenge

2 - https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

3 - https://seaborn.pydata.org/tutorial/distributions.html

4 - https://www.chevron.com/stories/seismic-imaging

5 - https://medium.com/nanonets/how-to-do-image-segmentation-using-deep-learning-c673cc5862ef

6 - http://www.cs.toronto.edu/~tingwuwang/semantic_segmentation.pdf

7 - https://www.jeremyjordan.me/semantic-segmentation/

8 - https://machinelearningmastery.com/transfer-learning-for-deep-learning/

9 - https://www.kaggle.com/jesperdramsch/intro-to-seismic-salt-and-how-to-geophysics

10 - https://medium.com/@keremturgutlu/semantic-segmentation-u-net-part-1-d8d6f6005066

11 - https://arxiv.org/pdf/1801.05746.pdf

12 - https://www.kaggle.com/osciiart/no-mask-prediction