Machine learning in geoscience: using deep learning to solve the TGS Salt Identification challenge

Marcelo Guarido*, Junxiao Li* and Raúl Cova*

ABSTRACT

Deep learning, or neural networks, contain a widely range of applicability, that goes from regression of business analyses to treats identification on medical images. In this paper, we successfully applied an U-net based image semantic segmentation to identify salt bodies using only seismic images from the TGS Salt Identification Challenge. The process is simple applied with moderate computer requirement for a small set of images, but it can grow exponentially as more images are included. In the end, we could train a model that gives a 0.8 score on the *IoU* metric.

INTRODUCTION

Subsurface salt bodies and layers are a challenging obstacle in the Oil & Gas prospection, either on seismic imaging (Zhang et al., 2009), than for drilling (Beltrão et al., 2009; Chatar and Imler, 2010). Misinterpretation of the salt boundary depth usually leads to higher costs on the production step.

To minimize budget loss during the drilling, much effort is expend in the salt interpretation, that can be done manually by an expert using different geophysical attributes, such as seismic images and gravity maps (Buur and Kühnel, 2008). Salt interpretation is closely related to the seismic image resolution, and the main challenge is due to the salt movement that results in steeply-dipping complex structures (Davison et al., 2014). Whiteside et al. (2012) use the RTM-based DIT (reverse time migration based delayed imaging time) to update the velocity model along the salt and sub-salt areas. Reasnor (2007) show the importance on selecting the proper algorithm to migrate the seismic data, and the top of the salt auto-picking, while Zhang et al. (2009) explain how the salt interpretation requires a balance of knowledge of geology and geophysical methods. However, our goal in this paper is to use previous interpreted surveys patterns on new ones.

In the medical sciences, deep learning methods are used to recognize different types of disease and anomalies. Shen et al. (2017) present a gathered analysis of different applications of deep learning in medical imaging, helping identify, classify and quantify patterns, and how they enhance the medical diagnosis. Pham et al. (2000) did a similar analysis on earlier years, but focused on image segmentation (classification of each pixel of an image). Each of the pixels are classified by semantic segmentation methods, such as FCN - or *Fully Convolutional Networks*, also know as FCNN, *Fully Convolutional Neural Networks* - on different types of images (Long et al., 2014). FCN combined with FC-ResNets (Fully Convolutional Residual Networks) is used to enhance the classification resolution on MRI (magnetic resonance imaging) images (Drozdzal et al., 2018), while Zhao et al. (2018) combines FCN with CRF (Conditional Random Fields) for brain tumor diagnosis.

*CREWES - University of Calgary

Image segmentation by deep learning is a so powerful tool that it is even used for video object segmentation (Liu et al., 2018). All the listed methods have a similar background: they are based on the *U-net* structure (Ronneberger et al., 2015), an encoding and decoding method focused on the pixel classification.

For this work, a combination of the U-net structure with FC-ResNets is used to classify salt location using seismic images from the TGS Salt Identification Challenge in the Kaggle website. The provided dataset contains part of seismic sessions on PNG format and their *masks* (another image containing the salt classification information, where 0 means **no salt** and 1 means **salt**), originally interpreted and marked by humans, to create a machine learning model to predict salt on unmarked seismic sessions.

DEEP LEARNING

Deep learning is any neural networks configuration with two or more hidden layers. In this section, we will talk about the fundamentals of neural networks, and what is behind the convolutional neural networks (CNN). We will follow the explanation development as presented by Shen et al. (2017).

Feed-Forward Neural Networks

Neural networks are mathematical models that, when given visual illustration, resemble the structure of neural systems in the brain. The simplest example of a neural networks is the *perceptron*, as shown on figure 1a, which is composed of the input and output layers only, and is understood to be a linear model. As, in general, the input layers is not counted, this model is considered to be *single-layered*.

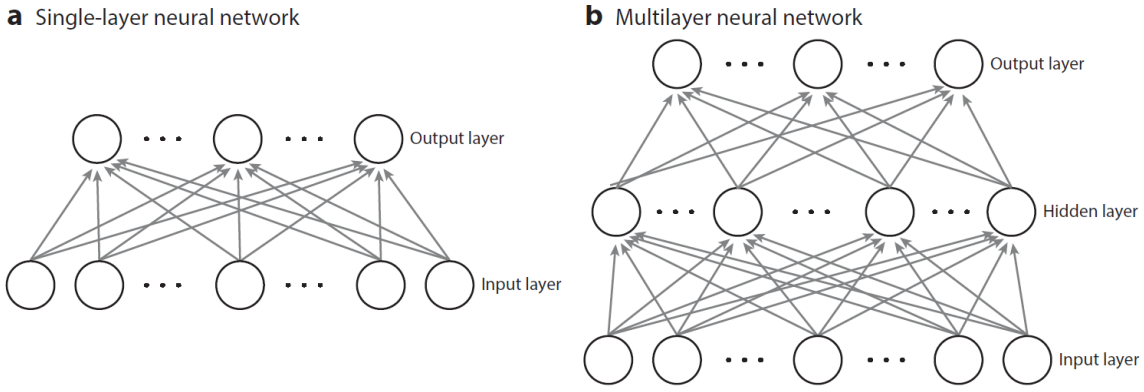

FIG. 1. Representations of a single (a) and a multi-layered (b) neural networks. Figure from Shen et al. (2017).

To enhance the power of the neural networks model and make it suitable for non-linear problems, *hidden layers* are included between the input and output layers (figure 1b). Each element of the layers are *fully connected* to the elements of the neighbors layers, but the elements of one layer have no connection to each other. Let's consider a two-layer neural network (one hidden layer and one output layer. The input is not counted). Given an input vector $\mathbf{v} = [v_i] \in \Re^D$, the output unit y_k is represented by the following estimation function:

$$y_k(\mathbf{v};\Theta) = f^{(2)}\left(\sum_{j=1}^{M} W_{kj}^{(2)} f^{(1)}\left(\sum_{i=1}^{D} W_{ji}^{(1)} v_i + b_j^{(1)}\right) + b_k^{(2)}\right) \tag{1}$$

where the superscript numbers denote the layer index, $f^{(\#)}$ denotes the nonlinear activation function, as the sigmoid, for example, M is the number of hidden units, and $\Theta = \left\{\mathbf{W}^{(\#)}, \mathbf{b}^{(\#)}\right\}$ is the parameter set. As the final estimation of equation 1 follows a forward direction (from input to output), it is referred as a *feed-forward neural network*.

In practice, the neural network method wants to minimize, for N observations, the error function:

$$E(\Theta) = \sum_{i=1}^{N} (v_i - y(\Theta)_i)^2 \tag{2}$$

The optimization is done by learning the best parameter set Θ. As the error function from equation 2 is highly nonlinear, there is no analytical solution for the parameter set. Instead, the gradient descent algorithm is used to update the parameters iteratively, by evaluating the optimization at the gradient $\nabla E(\Theta)$. For a feed-forward neural network, the gradient descent is applied by means of error back-propagation (evaluate E with the current parameter set, and apply the gradient descent backwards). The gradient descent for Θ optimization is written as:

$$\Theta^{(\tau+1)} = \Theta^{(\tau)} - \eta \nabla E\left(\Theta^{(\tau)}\right) \tag{3}$$

where τ is the iteration index and η is the learning rate (or step length). The process of equation 3 is repeat until equation 2 is minimized (reaching a stop criteria).

Convolutional Neural Networks

In the neural network described above, the input data is in a vector form. However, for this work, we want to analyze 2D images. An image could be vectorized and used on a regular neural network scheme, but the structural information from neighbors pixels would be lost.

Convolution Neural Networks (CNNs) are a variation of regular neural networks by having convolutional and pooling layers (figure 2) in blocks to detect local features at different positions of the input data (or input map in a deeper layer). Those blocks are repeated a few times and then are followed by fully connected layers, that work as a regular neural network, and the final output is a classification of the image (as a car, for example), or a list of probabilities of object classification.

Figure 3 shows a closer look at the feature learning of the figure 2. The convolutional layer, as said before, detect local features (such as horizontal lines, vertical lines, edges,

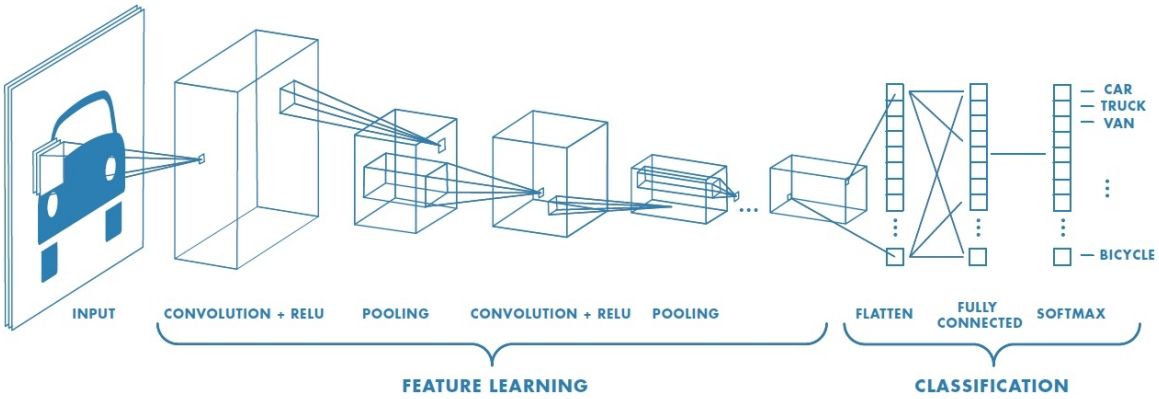

FIG. 2. Schematic representation of the convolutional neural networks. Figure from the MathWorks website.

etc) by convolving the input image (or the input feature map, the output of a previous convolutional layer) $\mathbf{A}_i^{(l-1)}$ (the $i-th$ feature map of the layer $l-1$) with the kernel $k_{ij}^{(l)}$, to generate a new feature map $\mathbf{A}_i^{(l)}$, as shown below:

$$\mathbf{A}_i^{(l)} = f\left(\sum_{i=1}^{M^{(l-1)}} \mathbf{A}_i^{(l-1)} * k_{ij}^{(l)} + b_j^{(l)}\right) \tag{4}$$

where $M^{(l-1)}$ is the number of feature maps in layer $l-1$, $b_j^{(l)}$ is a bias parameter, f is a non-linear activation function, and $*$ indicates a convolution.

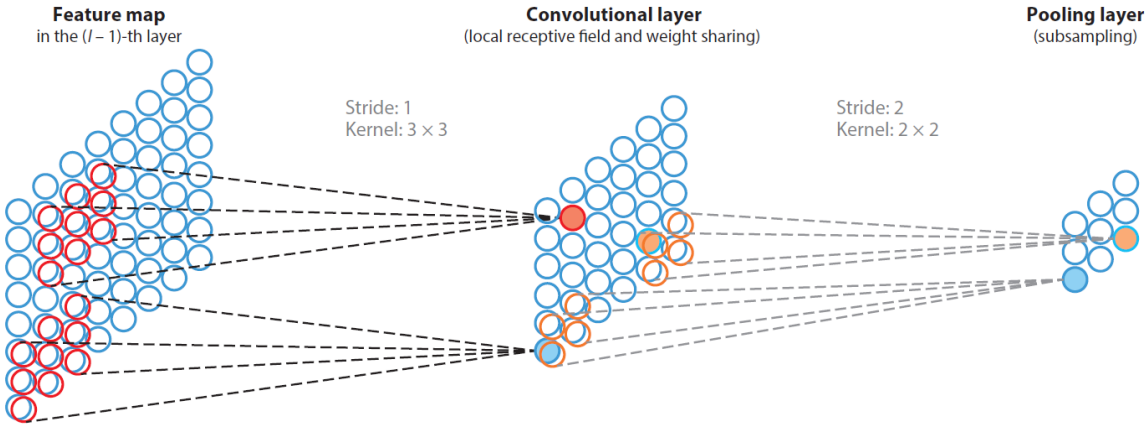

FIG. 3. A convolutional block. Each feature map is convolved to different kernels, and then down-sampled with a pooling layer. Figure from Shen et al. (2017).

A pooling layer comes after the convolutional layer in order to downsize the size of each feature map by grinding it and selecting a representative value of each cell (maximum, average, minimum, etc). This helps reduce the amount of parameters and computation of the network, and also helps to avoid overfitting.

The final step is the fully connected layers, that are similar to a simple neural network, and the output is a list of classification probabilities of the object.

Semantic Segmentation

CNNs are very powerful for imaging classification, but it classifies the whole image as a single object. For the salt identification challenge, an object inside the image needs to be detected and classified. In that point we enter in the *semantic segmentation* group of algorithms. Now the goal is to classify each pixels of the input image.

A possible way to solve this problem is to remove the fully connected and the pooling layers from the CNN scheme of figure 2, in a way that the convolved images always keep with the same size as the input. The output would be a matrix of probabilities of object classification. The problem with this solution is its cost. Deeper we chose the network to be, more memory is required in an exponential rate.

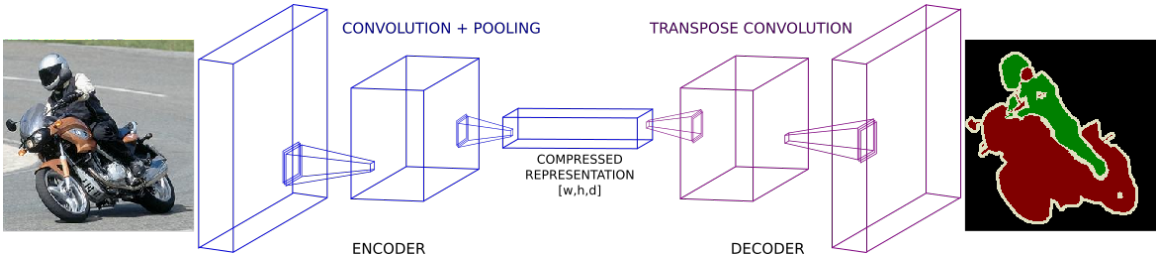

FIG. 4. Semantic segmentation scheme following an U-net shaped network. It is composed by three main components: encoder (downsize), compressed representation (center of network), and decoder (upsize). Image from https://sthalles.github.io/deep_segmentation_network.

Ronneberger et al. (2015) proposed a new structure for the CNN to classify pixels, but cheaper than series of convolutional layers keeping the size of the feature maps the same as the input image. This new structure divided in three main components: (1) *encoder*, where the convolutional blocks downsize the feature maps, (2) *compressed representation*, a dense vector of small detected features of the input image, and (3) *decoder*, composed by transposed convolutional layers, that has the goal to increase the feature maps spacial resolution so the output has the same dimensions as the input image. The output is expected to be a vector of matrices of probabilities to classify each object in the picture, i.e, if we have a image with three objects we want to classify (cat, dog and bird), the output will be a vector of three matrices, one with the probabilities (for each pixel) of the cta classification, one matrix with the dog probabilities, and another one for the bird. This proposed structure is *U shaped*, as shown in figure 4, and receives the name of *U-net*.

For the salt identification, the classification is binary: 0 means no salt and 1 means salt. So, the output of the U-net is a single matrix of probabilities (from 0 to 1) of the pixel be salt (1). To determine if the pixel is salt, we need to decide a threshold of probabilities where above is salt and below is not. The most common is to chose the threshold at 0.5, but it can be optimized by selecting one that reduces the score metric over a validation set.

Score Metric

A very common way to analyze the quality of semantic segmentation predictions is the *Jaccard IoU* (intersection over union) metric (Kosub, 2016). On the validation set, for the salt identification problem, take one of the known image classification matrix (of

0s and 1s), also called *mask*, **A**. The idea is to compare the mask **A** with the predictions (with the threshold applied) **B**. The IoU metric is the ration of the area of overlap (same classification) of **A** and **B**, with the area of union of the two matrices:

$$IoU = \frac{\mathbf{A} \cap \mathbf{B}}{\mathbf{A} \cup \mathbf{B}} \tag{5}$$

From equation 5, the score will have values from 0 to 1, where the later is a perfect prediction.

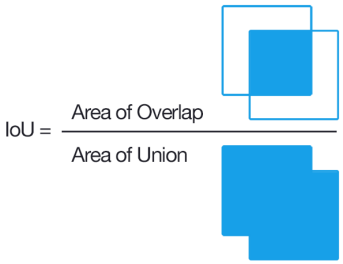

FIG. 5. Visual representation of the IoU score equation. Image from the PyImageSearch website.

Figure 5 gives a visual representation of the score metric IoU. In this project, the average IoU is computed over a validation set (here, 20% of the total number of images), for different values of probability threshold. The optimized threshold is the one with the highest score (used later as shown in figure 10).

SALT IDENTIFICATION

In this section, we present the data and some analyses of pixels count and distributions, image augmentation, ending up with the results over the validation set.

The Data

The data used in this paper is the train data of the TGS Salt Identification Challenge from the Kaggle website. For the competition, it is provided to sets of data: train and test. Each of them contain the seismic images on *PNG* format and their masks (pixel classification), also as *PNG* files. The competition also provides the depth, in feet, of each image, but with no explanation if it is the depth of the top, center, or bottom of the image. Below, is the list of files provided:

- *CSV* file with the depth, in feet, of each image from train and test sets
- Train folder:
  - 4000 *PNG* image files of parts of one or several seismic sessions
  - 4000 *PNG* mask files (pixel classification)
- Test folder:

  - 12000 *PNG* files of seismic images

- *CSV* file with the submission format

For the competition, the goal is to create a machine learning model by training it on the train set, then compute the predictions on the test set, create a submission file and upload it to the Kaggle website. For this paper, we will use only the 4000 images from the train set.

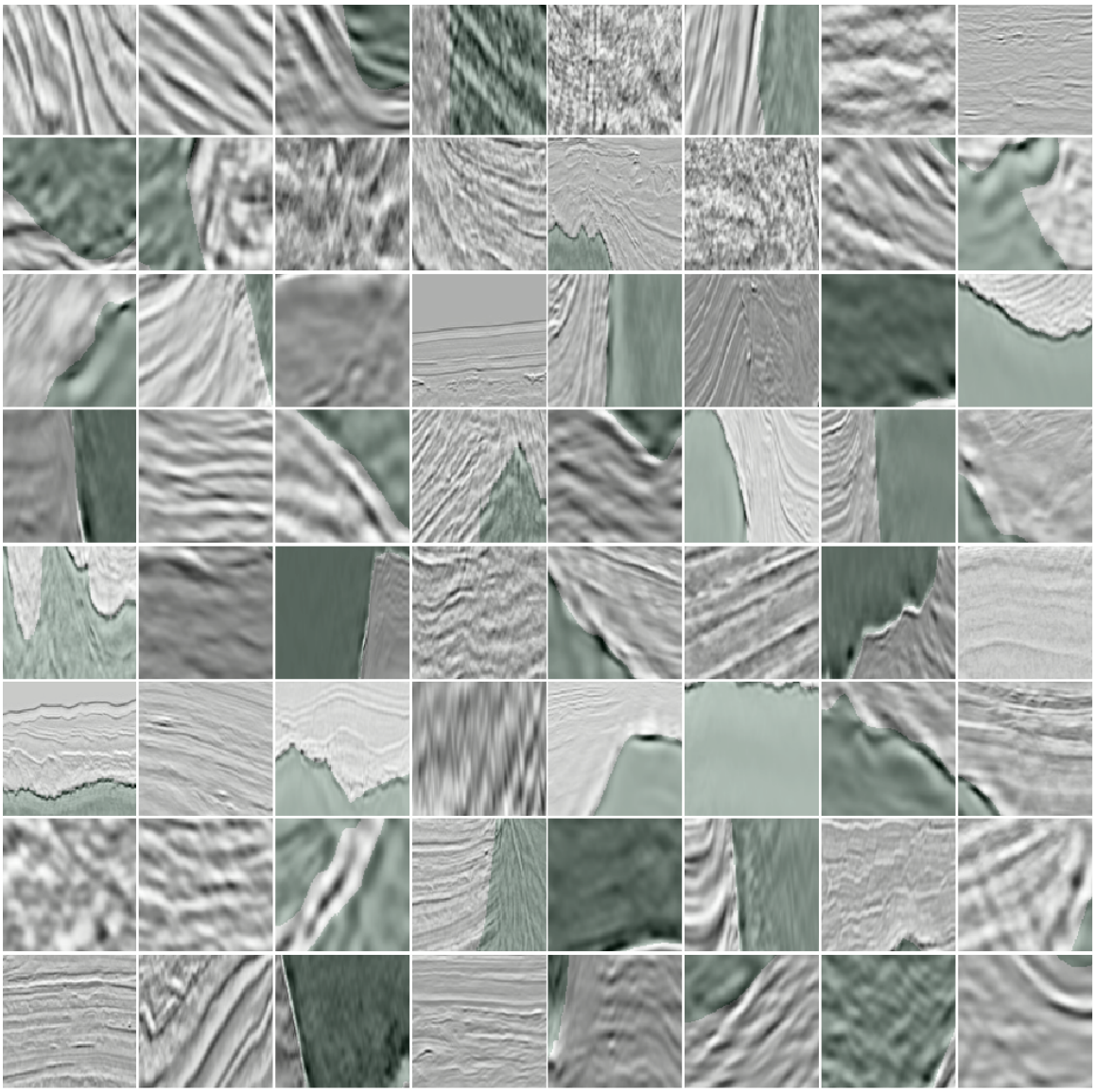

FIG. 6. Random seismic batch images in gray scale overlapped with the salt classification (green) at each pixel.

To decide with machine learning model to use, we have to take a look at the data. Figure 6 shows the overlap of 64 seismic (grayscale) and mask (green) images from the train set. The first observation is that we can not be sure that the images contain the same frequency content, or if they have same depth length or spacing. But on some of the images, we identified some water bottom reflections, suggesting marine data. In other images, we detect

an upside down water bottom, so the given images can be flipped and/or rotated. We also noticed that for some images, the classified salt does not look to follow any structural trend in the seismic image, suggesting that, for those images, it is cause by some interpolation during the salt interpretation.

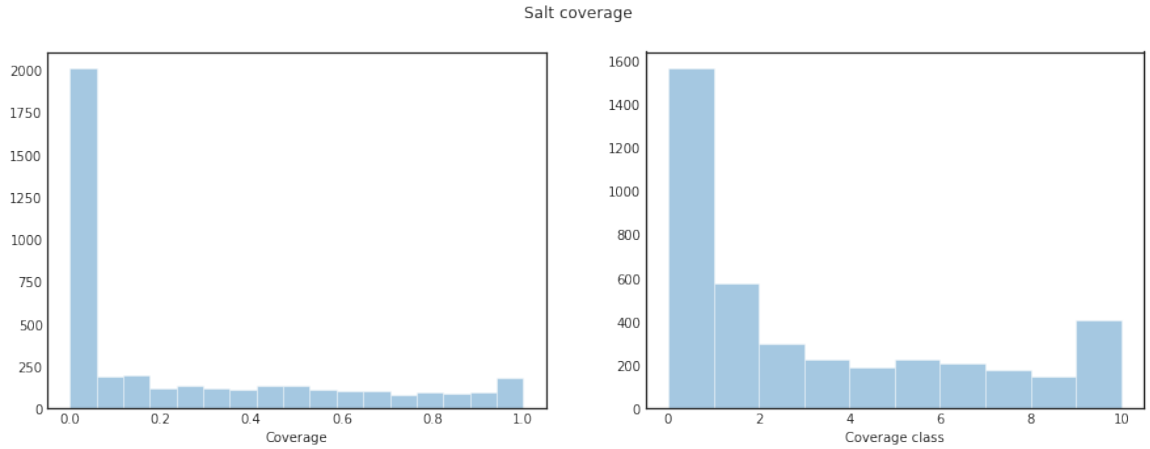

FIG. 7. Number of images with salt coverage proportion with random bin size (left) and with 10 bins named as class groups (right).

Figure 7 shows the number of images with salt coverage proportion with random bin size (left) and with 10 bins named as class groups (right). Images with any pixel classified as salt are the majority, and the count tends to decrease as the coverage increase, except for when it is close to 100%.

Image Augmentation

Using only the train set to create a deep learning model and validate it forces us to split the data set into train and validation sets. We selected the rate of 0.8 for the training set, letting us with 3200 images to train the model and 800 to validate it. 3200 is not a large amount of data to train deeper models. A way to minimize this issue is to do *image augmentation*.

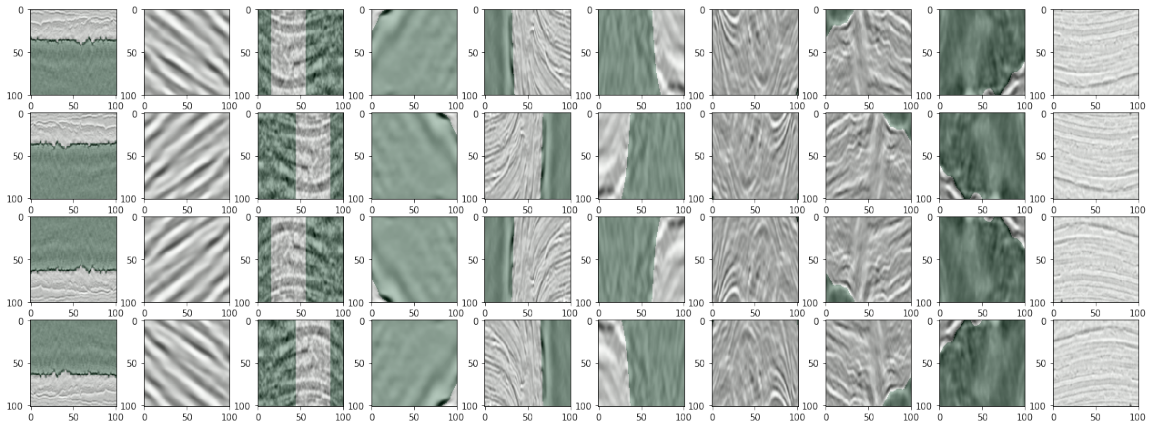

FIG. 8. Original images (top) and augmented images (three bottom rows).

The idea of image augmentation is quite simple: create new images and masks by flipping, rotating, and/or translating the original image. Figure 8 is te result of such augmentation. The original image is on the top row. The second row are the images flipped on the horizontal direction. The third row are the images flipped upside down. And the last row are the images flipped on the horizontal direction and also upside down. So, from 3200 training images, we end up with 12800 images. This method is not as good as having more independently different images, but it can help increase the accuracy of the predictions.

Training the Model and Predictions

To create and train a U-net model for the salt identification, we used the package *Keras* (Chollet et al., 2015) in *Python*. Keras is backed by *Tensorflow* (Abadi et al., 2015), a powerful deep learning package widely used for research. Both packages have support to run on *GPU*s, and are more stable when used with *CUDA* (NVIDIA Corporation, 2007).

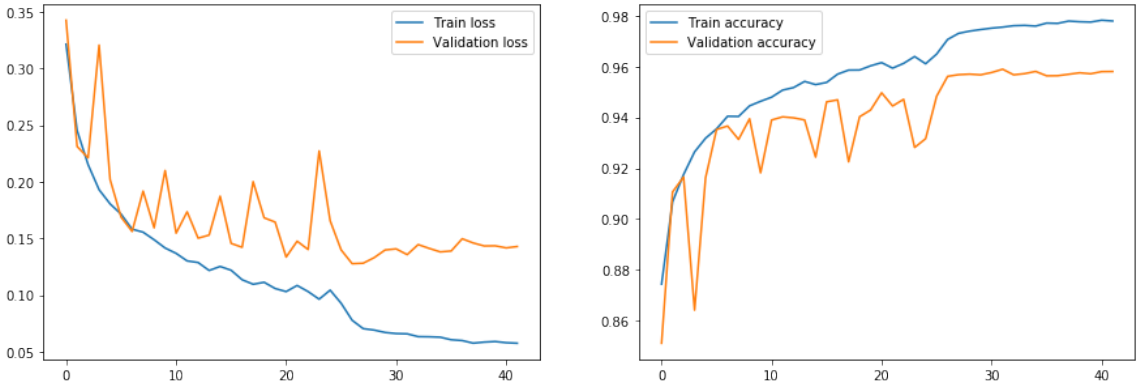

FIG. 9. Plots of the loss function (left) and accuracy (right) per iteration of the train (blue line) and validation (orange line) sets.

Figure 9 shows the predictions results (loss on the left, accuracy on the right) of the train (blue) and validation (orange) sets. Training stopped after 42 iterations with an accuracy on the predictions over the validation set close to 96%, which is impressive. However, the probability threshold used for those predictions is 0.5, and is not optimized.

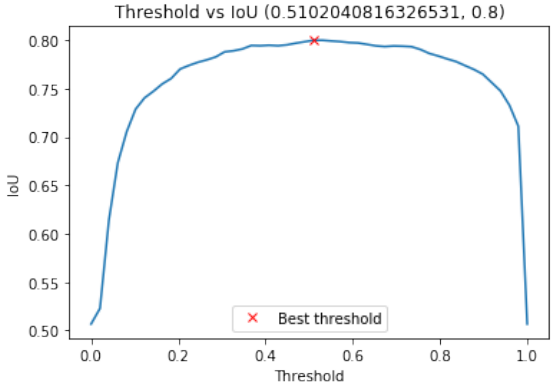

FIG. 10. *IoU* score calculated for different probability threshold values on the validation set.

The optimized probability threshold is chosen by calculating the IoU score for differ-

ent threshold values (figure 10) on the validation set. The best threshold found was 0.51, leading to an IoU score of 0.8.

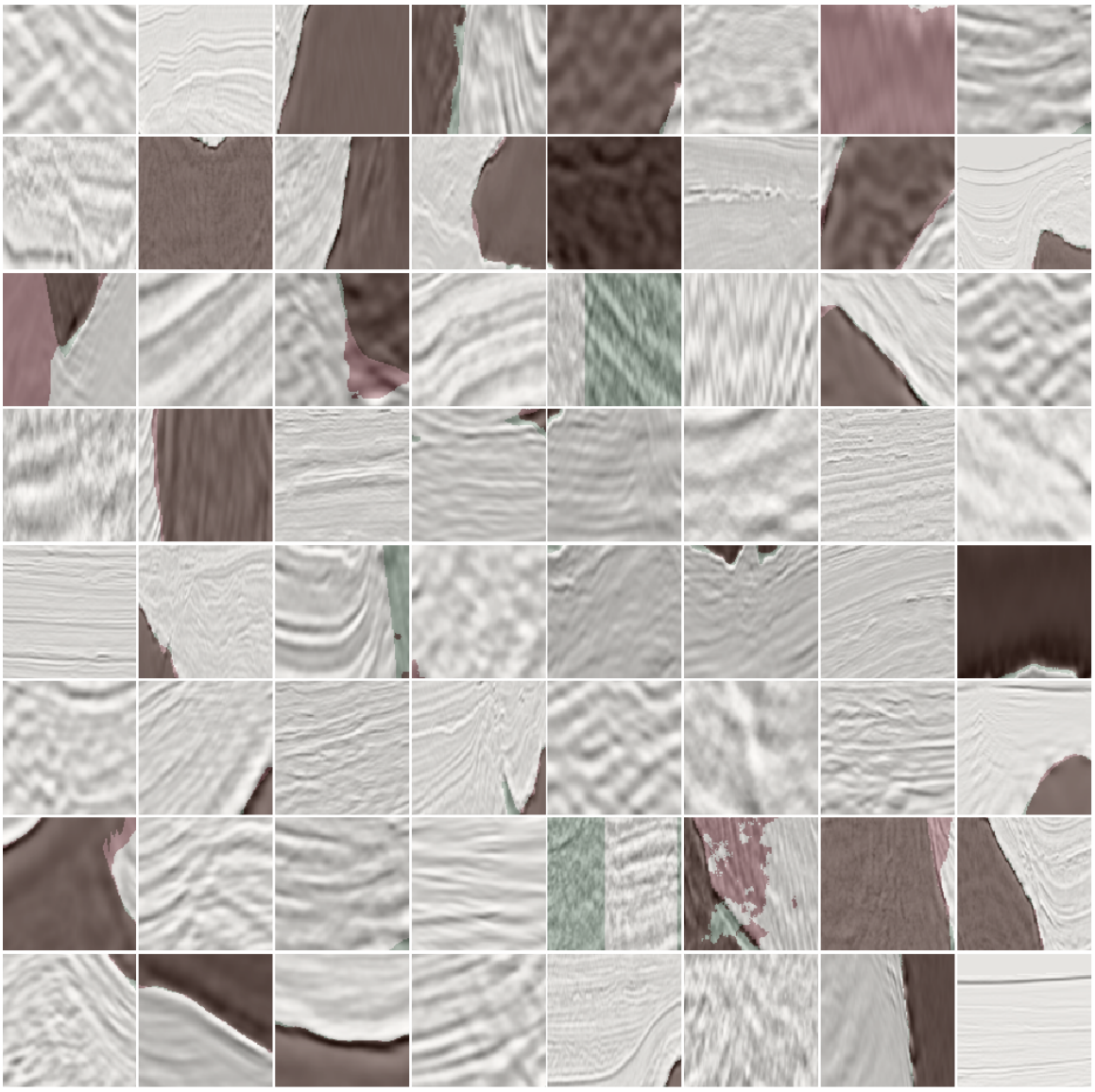

FIG. 11. Seismic overlapped by the original masks in green, predictions in red, and intersection in brown.

The salt identification predictions can be visualized on figure 11, where the seismic images are overlapped by the original masks in green, predictions in red, and intersection in brown. We noticed that no salt is predicted when the original mask does not follow any seismic structural trend. We can also find some areas where the salt is misclassified. However, in most cases, predictions match the original masks, even following complex trends.

Predictions could be improved with larger computational power. To train the model, the 12800 images could not be used at once, due to lack of memory. Instead, the model was trained with batches containing 32 images each, decreasing the accuracy of the gradient during the back-propagation process. Another possibility would be if we could do some

transformation on the seismic images in order to increase the contrast of salt areas.

CONCLUSIONS

Widely used in medical images, image segmentation using deep learning algorithms are powerful tools to identify specific objects in the image. We successfully applied an U-net based semantic segmentation algorithm to identify salt bodies on seismic images from the TGS Salt Identification Challenge, obtaining an IoU score of 0.8.

We observed that predictions accuracy are improved with image augmentation, creating new images by rotating, flipping, and/or translating the original image. We could quadruplicate the number of training images with this process, helping us to train a deeper model with the available data.

To train the model with the available images, that can be considered to be a small set, required moderate computer power, with the use of GPUs. However, the computer requirement can grow exponentially as more images are used to train the model.

ACKNOWLEDGMENTS

The authors thank the sponsors of CREWES for continued support. This work was funded by CREWES industrial sponsors and NSERC (Natural Science and Engineering Research Council of Canada) through the grant CRDPJ 461179-13, and the financial support from Canada First Research Excellence Fund.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X., 2015, TensorFlow: Large-scale machine learning on heterogeneous systems, software available from tensorflow.org.
URL `http://tensorflow.org/`

Beltrão, R. L. C., Sombra, C. L., Lage, A. C. V. M., Netto, J. R. F., and Henriques, C. C. D., 2009, Ss: Pre-salt santos basin - challenges and new technologies for the development of the pre-salt cluster, santos basin, brazil: Offshore Technology Conference, **January 1**.

Buur, J., and Kühnel, T., 2008, Salt interpretation enabled by reverse-time migration: GEOPHYSICS, **73**, No. 5, VE211–VE216.

Chatar, C., and Imler, M. D., 2010, Overcoming a difficult salt drilling environment in the gulf of mexico: A case study: Society of Petroleum Engineers, **January 1**.

Chollet, F. et al., 2015, Keras, `https://github.com/fchollet/keras`.

Davison, I., Jones, I. F., and Waltham, D., 2014, Seismic Imaging of Salt Diapirs: Problems and Pitfalls, 1332–1336.

Drozdzal, M., Chartrand, G., Vorontsov, E., Shakeri, M., Di Jorio, L., Tang, A., Romero, A., Bengio, Y., Pal, C., and Kadoury, S., 2018, Learning normalized inputs for iterative estimation in medical image segmentation: Medical Image Analysis, **44**, 1–13.

Kosub, S., 2016, A note on the triangle inequality for the jaccard distance, `1612.02696`.

Liu, Z., Wang, L., Hua, G., Zhang, Q., Niu, Z., Wu, Y., and Zheng, N., 2018, Joint video object discovery and segmentation by coupled dynamic markov networks: IEEE Transactions on Image Processing, **27**, No. 12, 5840–5853.

Long, J., Shelhamer, E., and Darrell, T., 2014, Fully convolutional networks for semantic segmentation: CoRR.

NVIDIA Corporation, 2007, NVIDIA CUDA Compute Unified Device Architecture Programming Guide: NVIDIA Corporation.

Pham, D. L., Xu, C., and Prince, J. L., 2000, Current methods in medical image segmentation: Annual Review of Biomedical Engineering, **2**, No. 1, 315–337.

Reasnor, M. D., 2007, Salt interpretation practices for depth imaging in the gulf of mexico: The Leading Edge, **26**, No. 11, 1438–1441.

Ronneberger, O., Fischer, P., and Brox, T., 2015, U-net: Convolutional networks for biomedical image segmentation, *in* Medical Image Computing and Computer-Assisted Intervention (MICCAI), LNCS, Springer, 234–241.

Shen, D., Wu, G., and Suk, H.-I., 2017, Deep learning in medical image analysis: Annual Review of Biomedical Engineering, **19**, No. 1, 221–248.

Whiteside, W., Guo, Z., and Wang, B., 2012, Automatic RTM-based DIT scan picking for enhanced salt interpretation, 3295–3299.

Zhang, Q., Chang, I., and Li, L., 2009, Salt interpretation for depth imaging - where geology is working in the geophysical world, 3660–3664.

Zhao, X., Wu, Y., Song, G., Li, Z., Zhang, Y., and Fan, Y., 2018, A deep learning model integrating fcnns and crfs for brain tumor segmentation: Medical Image Analysis, **43**, 98–111.