CS230

Engineer Level Automated Interpretation of Geothermal Well Logs

Ahinoam. Pollack*
Department of Energy Resources Engineering
Stanford University
ahinoamp@stanford.edu

Abstract

Every geothermal engineer needs to learn well log interpretation skills. A well log is a series of measurements taken by various sensors along a wellbore. It is one dimensional data taken along a depth axis (a "depth-series"), measuring for example the temperature per depth. Like doctors, engineers need to look at these logs to diagnose subsurface properties. For example, the locations where faults intersect the wellbore ("feed zones"), the reservoir pressure, and the reservoir temperature. I have implemented a multi-layered convolutional neural network to automatically diagnose sets of temperature and pressure well logs. The algorithm achieves results similar to those of a professional engineer. This algorithm enables the interpretation of many well logs in just a few seconds.

1 Introduction

Geothermal energy is a renewable energy source that uses heat from the subsurface to generate electricity. Operating a geothermal energy facility involves drilling into the subsurface, analyzing the properties of the subsurface and then using that analysis to make important decisions regarding the facility. One aspect of this analysis is interpreting well logs. A well log is a series of measurements taken by various sensors along a geothermal wellbore. It is one dimensional data taken along a depth axis (a "depth-series"), measuring for example the temperature per depth, see Fig. 1. In this study, I have implemented an algorithm to automatically predict subsurface properties from the well logs using a convolutional neural network. There are three inputs into the algorithm:

1. Temperature log, with shape (200, 1)
2. Pressure log, with shape (200, 1)
3. The derivative of the temperature log, with shape (200, 1).

There are five outputs from the algorithm:

1. Depth of feed zone 1.
2. Depth of feed zone 2.
3. Reservoir temperature.
4. Reservoir pressure.

*Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

5. Depth of reservoir pressure.

A single example of inputs and output can be seen in the example well logs on the right in Fig. 1.

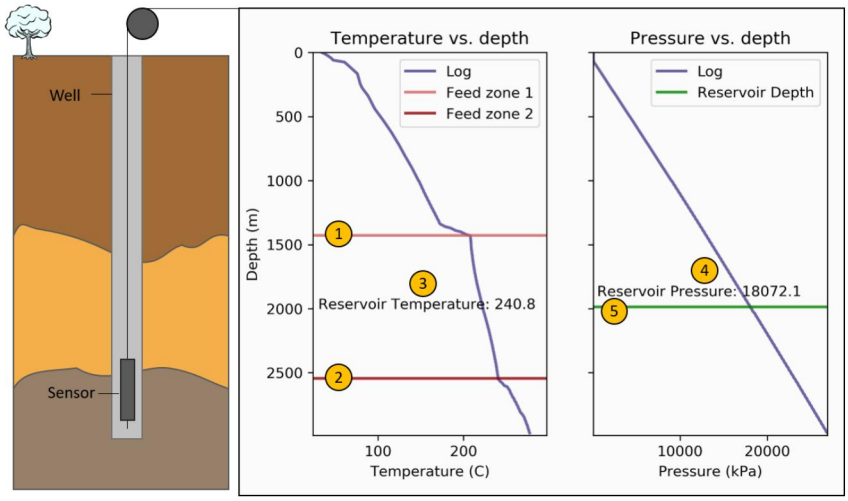

Figure 1: (left) a sensor collects subsurface information from a wellbore. (right) An example of data collected by the sensor as a function of depth (the log data) and the five characteristics determined by a reservoir engineer based on the well data.

2 Related work

The task of this paper is identification of geothermal reservoir properties using neural networks. A literature review has revealed no prior research on this particular task. A search of "deep learning" and "geothermal" in Google Scholar yields no relevant results. The use of machine learning in geothermal is mostly relegated to classical algorithms such as classification and regression trees, see for example [1]. In general, the geothermal energy industry is relatively small in size and research budget and therefore less research has been done on deep learning applications.

Well logs are also used in the oil and gas industry. Neural Networks have been applied successfully to automated analysis of well logs for oil and gas applications. For example, estimating rock lithology and permeability from well-logs [2, 3, 4, 5]. For a more complete summary of the applications of neural networks in oil and gas applications, see Pulton [6].

Most of these papers, though, predict a "depth-series" output from a set of "depth-series" inputs. In this work, however, the goal is to predict only five summary statistics that are dependent on both the values and pattern of the data.

3 Dataset and Features

For this project, I have only 21 real well logs from my research site in Nevada (confidential location). Therefore, I wrote an algorithm to create synthetic well data that mimics the real data (though it is much less noisy). I used the algorithm to generate 10,000 training samples.

I split the data into three sets as follows:

- Training data: 10,000 generated (synthetic) temperature and pressure series, and temperature derivative, of length 200. The data was normalized to have values between 0 and 1. Shape: [10000,200,3].
- Validation data: 12 real sets of temperature, pressure and temperature derivative series, used for hyper-parameter tuning. The data was resampled to be of length 200 and normalized to have values between 0 and 1. Shape: [12,200,3]
- Testing data: 9 real sets of temperature, pressure and temperature derivative series, resampled to be of length 200 and normalized to have values between 0 and 1. Shape: [9,200,3].

4 Methods

A convolutional neural network (CNN) is a form of a deep neural network that has gained attention as Krizhevsky et al[7] used it to successfully classify images into 1000 different classes. CNNs include a convolution operator as part of their framework and are particularly well suited for spatial data. The kernel size of the convolution operator leads to neurons that perform calculations using only local neighborhoods of pixels. In addition, the parameter sharing attribute of CNNs allows for a lower number of parameters than having fully connected layers between all the pixels [8].

I have implemented a multi-layered neural network with the following architecture shown in Fig. 2. The network has the following steps:

1. Input: takes as input three “depth-series” logs of temperature, pressure, and temperature gradient
2. Initial Convolution Block: a block that has a one-dimensional convolution, followed by an optional batchnorm, an activation function, an optional max pooling layer, and an optional dropout layer.
3. Middle Convolution Block: a block that is identical to the previous one, except does not take an "input shape" parameter. This block is repeated multiple times depending on the value of the hyperparameter "number of convolution blocks".
4. Flat layer: a layer that is just a flattened 1D version of its input.
5. Fully connected layer: a fully connected layer with five variables: the depths of the feed zones, the reservoir temperature, the reservoir pressure and the depth at which the reservoir pressure is known.

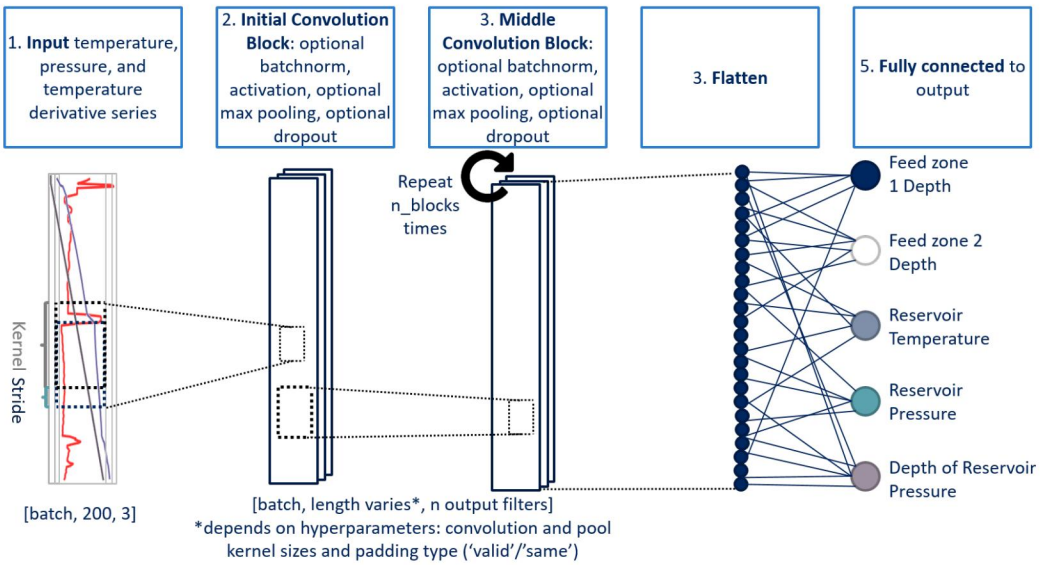

Figure 2: General architecture.

The cost function for this model was mean squared error. An additional recorded metric was the mean absolute error. The optimizer algorithm used was Adam, and the learning rate had an exponential decay. The algorithm recorded the model state that had the lowest mean absolute validation error. This early stopping is a form of regularization to avoid over-fitting. The architecture was implemented in Keras [9] with a TensorFlow backend [10].

5 Hyper Parameter Fitting

There were 25 possible values for the architecture presented in the above section. Therefore, I wrote an algorithm to sample 200 random combinations of the hyperparameters. I then used the validation

errors from those realizations to test the sensitivity of the hyperparameters to the error level and help fine-tune the hyper parameter search. I calculated the sensitivity using distance based generalized sensitivity analysis [11], and the results of that analysis are shown in Fig. 3(left). The results show that the most important parameter is the number of convolution blocks. The right side of Fig. 3 shows that a higher number of convolution blocks can significantly lower the validation error. Based on these results, another two hundred realizations were run with a higher possible number of convoltion block. In the end, the best model, with the lowest validation error, had four convolution blocks, 'valid' padding, and an initial kernel size of 12.

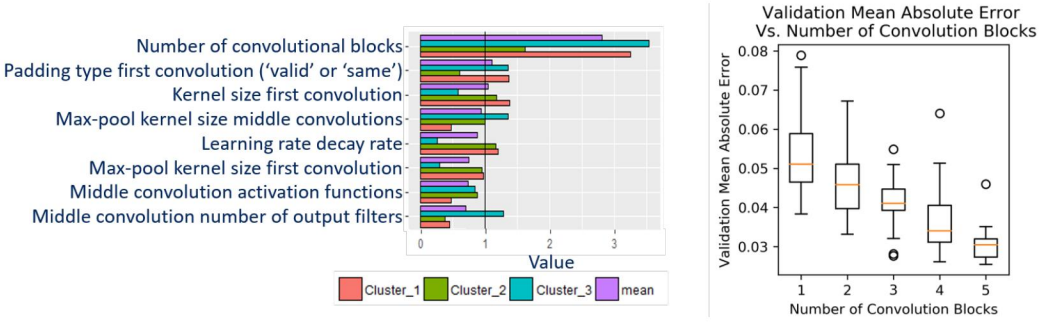

Figure 3: (left) Sensitivity of the validation mean absolute error to the hyper parameters. (right) Influence of the number of convolution blocks on the validation mean absolute error.

6 Results

The best model found during the process of hyper parameter tuning was used to predict the reservoir characteristics for the validation and testing data sets. Fig. 4 shows the best prediction result and the worst prediction result by this model. As can be seen, the best result almost completely overlaps with the true data.

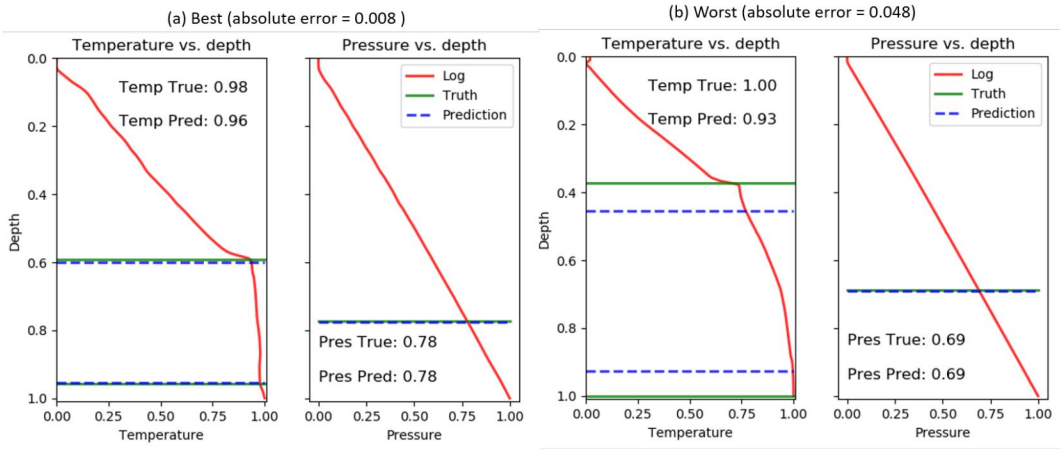

Figure 4: (a) The best and (b) worst prediction by the chosen model.

The training, validation and test errors for the chosen model are shown in the table below. These results shows that there is still some improvements to be made in the algorithm. The bias is fairly low and does not need much improvement. There is a significant gap between the testing error and validation error. The hyper-paramer tuning process seems to have slightly overfit the the validation data. The gap between the tranining and validation error is not due to overfitting of the training error but rather to a dissimlarity between the smooth training data and the more noisy real data. This was checked by creating an additional training-validation set composed of unseen training data. The model performed well on the training-validation set.

Training Error	Validation Error	Testing Error
0.008	0.020	0.030

Table 1: Training, validation and testing error.

7 Future and Current Work

After performing error analysis, I saw that many of the errors of the algorithm occurred in well logs that were slightly ambiguous even to professional engineers. An engineer could give several acceptable answers to some of the data. Therefore, I decided that the CNN algorithm should also be able to provide several possible answers. If one of the possible answers predicted by the algorithm was the true answer, then the prediction would be counted as correct. I started working on the above suggestion, using a similar architecture as that presented in the Methods section above. However, instead of a fully connected layer in the end with five properties, there are several possible outputs that branch out:

1. An output that detects the first feed zone. This output layer is a (200,1) layer followed by a softmax activation. Turning this into a classification problem.
2. A similar output that detects the second feed zone.

Similar to Google Maps, the algorithm then suggests feed zones with the highest post-softmax values, but chooses only the highest value in any given neighborhood. This is the custom validation metric:

- Run a window with a certain kernel size along the softmax output. The window keeps only values that are the highest within the window. This makes sure that the suggested feed zone locations are not too close to each other and don't overlap.
- Find the maximum value from the softmax output. Then, only keep the values that are more than a threshold times the maximum.

This algorithm can lead to a prediction of only one feedzone in case the softmax output strongly indicates only one location. When the softmax output has several high value candidate locations, then the results present several of the high value candidate locations so that the engineer can choose amongst them. Fig. 5. on the right shows an example of relative confidence of the algorithm in the solution, providing only two interpretations. In Fig. 5. on the left, the algorithm suggests many interpretations. This method will allow for a semi-automated approach: the algorithm solves the easy scenarios, and provides possible solutions for the complicated scenarios.

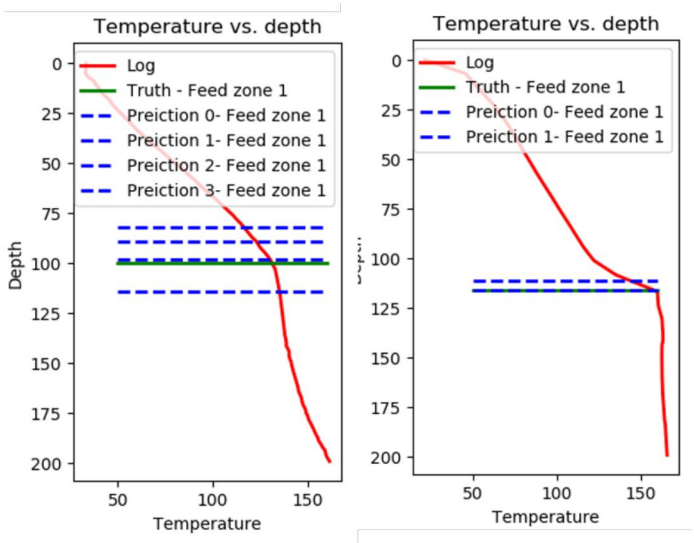

Figure 5: A CNN model that provides multiple predictions for the input.

8 Contributions

Single team member. No outside contributions except for helpful suggestions from the TA Jay Whang and my adviser.

References

[1] Joe Iovenitti, Fletcher Hank Ibser, Matthew Clyne, Jon Sainsbury, and Owen Callahan. The Basin and Range Dixie Valley Geothermal Wellfield, Nevada, USA—A test bed for developing an Enhanced Geothermal System exploration favorability methodology. *Geothermics*, 63:195–209, 2014.

[2] Kexiong Wang and Laibin Zhang. Predicting formation lithology from log data by using a neural network. *Petroleum Science*, 5(3):242–246, 2008.

[3] Lianshuang Qi and Timothy R Carr. Neural network prediction of carbonate lithofacies from well logs, big bow and sand arroyo creek fields, southwest kansas. *Computers & Geosciences*, 32(7):947–964, 2006.

[4] Mohsen Saemi, Morteza Ahmadi, and Ali Yazdian Varjani. Design of neural networks using genetic algorithm for the permeability estimation of the reservoir. *Journal of Petroleum Science and Engineering*, 59(1-2):97–105, 2007.

[5] Guoyin Zhang, Zhizhang Wang, and Yangkang Chen. Deep learning for seismic lithology prediction. *Geophysical Journal International*, 215(2):1368–1387, 2018.

[6] Mary M Poulton. Neural networks as an intelligence amplification tool: A review of applications. *Geophysics*, 67(3):979–993, 2002.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[8] K Parthy. CS231n Convolutional Neural Networks for Visual Recognition. Technical report, 2018.

[9] François Chollet et al. Keras. `https://keras.io`, 2015.

[10] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[11] Ognjen Grujic. *DGSA, an R package*, 2016. R package version 1.0.