Classifying Depositional Environments in Satellite Images

Alex Miltenberger and Rayan Kanfar

Department of Geophysics
School of Earth, Energy, and Environmental Sciences
Stanford University

1 Introduction

Subsurface mapping and geologic understanding is of key importance for resource exploration and exploitation. Part of our research interest is to develop geologically driven interpolation tools that incorporate geological realism with well logs and geophysical information. The current state of the art is to use statistical methods such as kriging between subsurface data points, which has no bearing in geologic structures. In this project, the goal is to classify clastic depositional environments, particularly, deltas, rivers, and lakes, using supervised machine learning methods on satellite images. The input to our algorithms are one of two types: average features extracted from each image and the images themselves. We compare the results of multiclass support vector machines, softmax, a simple neural network, and a convolution neural network to classify images of deltas, rivers, and lakes. This project could serve as the first step in developing geologically driven interpolation methods for subsurface mapping.

2 Data

2.1 Collection

Google Earth Engine using Landsat 8 satellite was used to manually select depositional environment of interest, namely, deltas, rivers, and lakes. We extracted four bands: the red, green, and blue bands for true-color images and the near-infrared band. The manually selected images were downloaded using Pythons Earth Engine library.

For each image a polygon was created around the depositional environment of interest. The Python script takes all the satellite images collected by Landsat 8 that overlap the polygon of interest and filters through the available images to select a single, high-quality image to download. Only images recorded between June 2014 and June 2016 were considered. The remaining filtering includes removing nighttime images, removing cloudy images, masking the remaining clouds, and finally clipping to the geometry desired. The images for the red, green, blue, and near-IR bands are then downloaded as custom-formatted text files and named according to the depositional environment they represent. The text files were then converted to Matlab-compatible formats for future use.

The discrete cosine transform was used to reformat all Matlab-compatible images to 250×250 pixels. A visual representation of this workflow is shown in Figure 1.

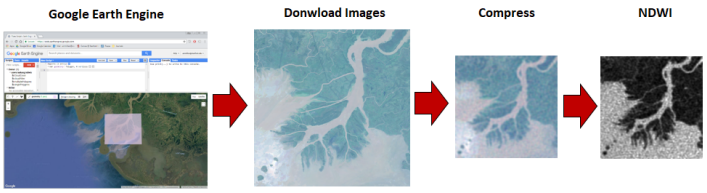


Figure 1: Overview of the workflow for collecting and preprocessing the data for the project.

2.2 Feature Extraction

Features were extracted from the four downloaded bands and an additional normalized difference water index (NDWI) image. NDWI provides an estimate of how much water is contained in a single pixel. This well-established index measures the difference between green and infrared reflectance and normalizes the value by the sum of the two reflectances, as illustrated in Equation 1.

$$NDWI = \frac{green - NIR}{green + NIR} \tag{1}$$

Higher NDWI values generally indicate the presence of water and lower values can be interpreted as less likely to contain water. Equation 1 was applied to the corresponding compressed bands for all images to create an NDWI image. Figure 2 shows twenty of the NDWI images used in the models.

The features extracted were designed to be representative of the entire image they represent. For example, we included the mean red value for a particular image rather than all pixels in the red band of that

Table 1: List of single image features used in the supervised learning models and one of the neural networks.

	Red	Green	Blue	NIR	NDWI
Mean Value in Image	✓	✓	✓	✓	✓
Median Value in Image	✓	✓	✓	✓	✓
Mean Edge Value					✓

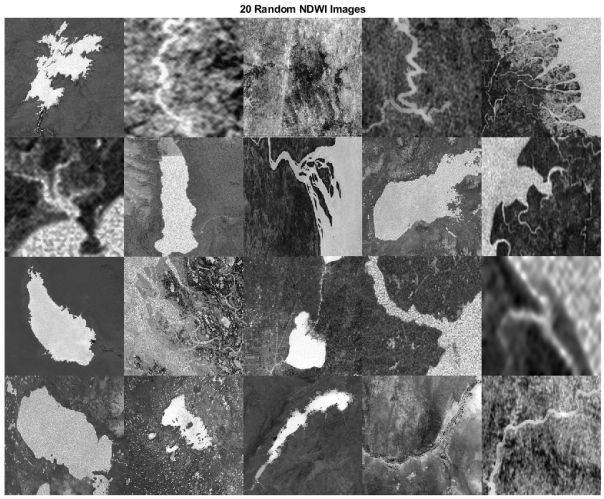


Figure 2: A collection of randomly selected NDWI images from the dataset.

image. This significantly reduces the dimensionality of the data and also avoids problems with shifting and rotation of the images. Table 1 summarizes the 11 features extracted from each image. The mean and median values in each band, including NDWI, were extracted, composing 10 of the features. The final feature, the mean edge value, was included for NDWI primarily to help with classifying deltas, since deltas have many more water pixels on the edges of the image than lakes and rivers.

3 Methods

3.1 Basic Supervised Learning Models

A total of five simple supervised learning models are presented here. Four models are variants of support vector machines and the fifth is a simple softmax regression. Each of these five models used the 11 single representative features shown in Table 1 as the feature space.

Multiclass Support Vector Machine

Since support vector machines are inherently binary classifiers, we used three separate support vector machines with two different encoding schemes: one-vs-one and one-vs-all. Figure 3 shows the one-vs-one encoding scheme.

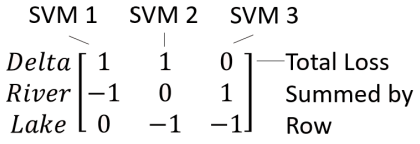


Figure 3: One vs. One Encoding matrix

The one-vs-all encoding scheme is similar to one-vs-one, but the zeros in the one-vs-one scheme are replaced with -1. Each support vector machine is trained to minimize the standard hinge loss function,

$$J_j = \sum_{k=1}^{3} L_{kj} = \sum_{k=1}^{3} \max\{0, 1 - t_{kj} y_{kj}\} \tag{2}$$

whereJ_j is the total loss for SVM j, L_{kj} is the loss for the k^{th} class of the j^{th} SVM, t_{kj} is the encoding for that class and SVM, and y_{kj} is the prediction from the j^{th} support vector machine.

Once the three SVMs in the model are trained the classification is determined by summing the total loss produced by the three SVM's for each possible classification. The class which yields the smallest total loss is chosen as the prediction. Mathematically, this is written as

$$\hat{k} = \arg\min_{k} \sum_{j=1}^{3} L_{kj} \tag{3}$$

where $\hat{k}$ is the predicted class.

Four multiclass SVM variants are presented in the results. The different multiclass SVMs are varying combinations of encoding schemes and kernels. The two kernels used are linear (unkernelized) and a Gaussian radial basis function.

Softmax Regression

The final supervised learning method used in this project was a simple softmax, or multinomial logistic, regression. The softmax regression is mathematically represented as

$$P(y = k|x) = \frac{\exp(\theta_k^T x)}{\sum_{i=1}^{3} \exp(\theta_i^T x)} \tag{4}$$

where $P(y = k|x)$ is the probability that the features x belong in class k, and θ_i is the weights for i^{th} class. The predicted class is the one which is most probable.

3.2 Neural Network Models

Three deep learning models are evaluated. The first two are Simple Neural Networks (SNNs) with a single hidden layer and the third is a Convolutional Neural Network (CNN).

The architecture used for the simple neural network consists the input layer, one hidden layer with 300 neurons, and an output layer with three neurons. The difference between the two simple neural networks is the features. The first SNN uses the 11 features in Table 1 and the second uses each pixel of the NDWI image as a feature. The architecture of the convolution neural network includes an input layer, three sets of convolutional, normalizing, subsampling pools, and ReLu layers, and finally an output layer. Figure 4 shows the architecture and the forward propagation of three random images through the network for illustrative purposes. The intermediate images are the features extracted by the CNN after each ReLu layer.

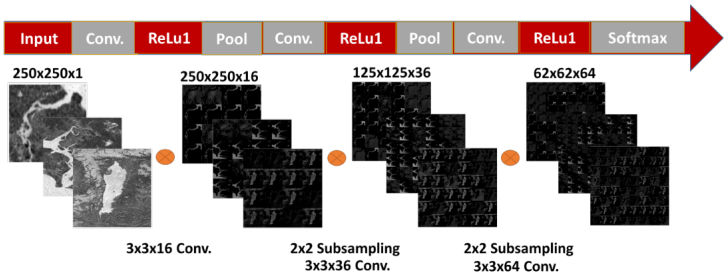


Figure 4: Schematic of CNN architecture with the features learned by a trained network after each ReLu layer.

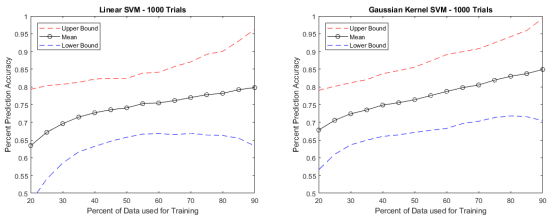
4 Results

4.1 Basic Supervised Learning Models

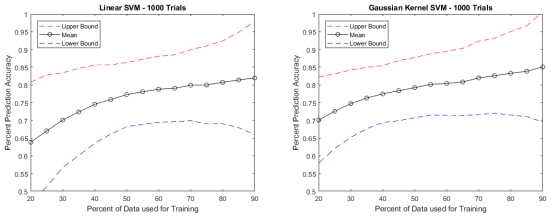
The basic supervised learning algorithms were evaluated based on their prediction accuracies. Because of the limited data set we used a validation technique very similar to cross-validation. Instead of dividing the data into folds we opted to simply assign data to the training and validation sets at random to meet a specific ratio of training to validation examples. To evaluate these models we used training example percentages from 20-90% of the total data with the remaining data used for validation. In this way we can see how sensitive the models are to training set size in this limited dataset.

We generally used 1000 different randomizations to minimize the impact of statistical outliers in the process of assigning examples to the training/validation sets. These outliers are very likely to occur using such a small dataset. To assess the impact of training set size we have created curves that show the relationship between training set size and prediction accuracy, which are shown in Figure 5.

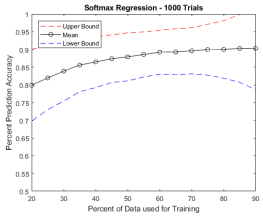
These plots show the mean prediction accuracy and a Gaussian 95% confidence interval. Histograms of individual training percentages generally followed a Gaussian distribution, except for the extremely low and extremely high training percentages. The accuracy distributions of the high and low training percentages were erratic and subject to more outliers, hence the generally wider confidence interval at these extremes.



(a) One vs. One Linear **(b)** One vs. One Gaussian



(c) One vs. All Linear **(d)** One vs. All Gaussian



(e) Softmax

Figure 5: Supervised learning model performances from 1000 training/validation randomizations with 95% confidence intervals.

Considering only the SVM models, not much difference is observed between the two encoding schemes. The one-vs-all scheme consistently performs about 1-3 percentage points better than the one-vs-one scheme. The Gaussian kernel SVM significantly outperforms the linear SVM by 5-7 percentage points. This is not surprising given the fact that that kernelized SVMs tend to be able to handle more complex datasets than linear SVMs.

However, what is surprising is that the softmax regression model classifies these data much more accurately and consistently than any SVM. The mean accuracy is higher for the softmax model at any training

percentage than any SVM and the confidence interval is much tighter. Our interpretation of this is that since the softmax model is inherently a multi-class model it is able to much more robustly handle the complexities of multi-class datasets. Support vector machines, being binary classifiers, were modified ad hoc to handle multi-class data and are therefore less apt to handle such problems than a model built specifically for multi-class problems.

Table 2 shows the average confusion matrices for the SVM and Softmax models, respectively, based on 1000 random realizations of training and validations sets. For the SVM models, the prediction errors seem to be slightly biased to confusing deltas and lakes, but nothing that is too much of a concern. The softmax model is less prone to confusing these two categories and does a better overall job of classifying the different depositional environments. The major interpretation of these three tables is that the classifiers appear to be classifying the three different classes at nearly the same rate.

Table 2: Average confusion matrices for one-vs-all SVMs and softmax with 1000 data randomizations using 75% for training and the other 25% for validation. *Top* linear SVM, *middle* Gaussian kernel, *bottom* softmax.

	Classified as Delta	River	Lake
Delta	15.60	1.27	4.32
River	0.12	18.58	2.09
Lake	3.08	0.85	13.09
Delta	17.22	0.92	3.03
River	0.45	18.94	1.34
Lake	3.78	0.78	12.55
Delta	18.72	1.13	1.21
River	0.88	19.17	0.69
Lake	1.14	0.84	15.22

4.2 Neural Network Models

Figure 6 shows the training curves for the simple neural network with 11 features. Possibly due to the small number of training examples, the training curves in Figure are oscillatory and unstable with a final prediction accuracy of 41%. This is close to the accuracy one may expect from random guessing, 33%. This model was not found to be sensitive to changes in the number of hidden layer neurons, learning rate, or regularization.

When using each pixel in the NDWI image as a feature the same neural network architecture produces better results. This can be seen in the learning curves of Figure 7. This implies that the oscillatory behavior in Figure 6 is not solely due to the small number of

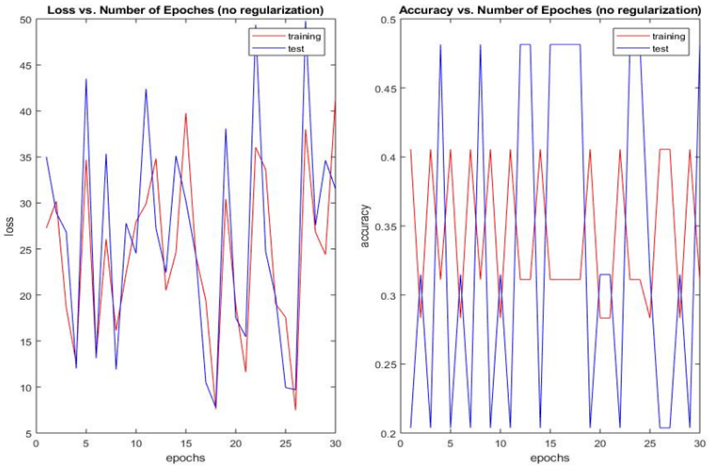


Figure 6: Training curves for unregularized simple neural network using the 11 features from Table 1. Learning rate is 5 and the number of neurons is 300.

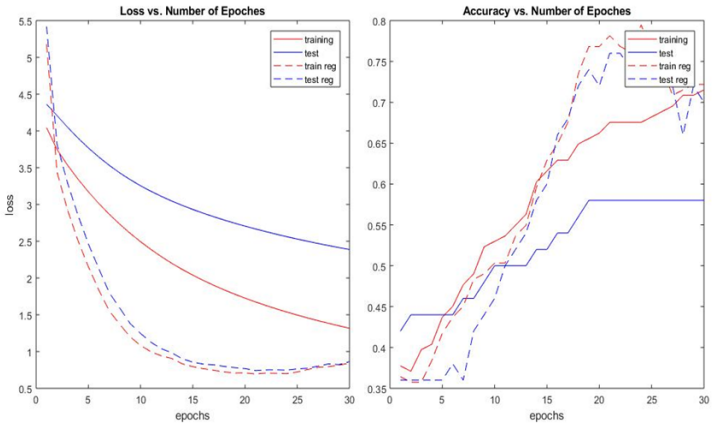


Figure 7: Training curves for the simple neural network using the 250×250 NDWI image pixels as features. Solid curves are unregularized and dashed curves are regularized. Learning rate is 100 and the number of neurons is 300.

training examples, especially considering the other supervised learning models performed much better with the same amount of data.

When considering the simple neural network of with all pixels used as features, there is a gap between the losses and accuracies indicating that the model is overfitting. This can be seen in Figure 7. Applying a regularization term of 0.001 smoothed our model to a good fit, yielding a much better prediction accuracy of 71% on test data.

This model was found to be sensitive to hyperparameters such as the number of hidden layer neurons and the learning rate. When using a learning rate less than one the test and train losses are equal at all epochs. Increasing the learning rate to around 5 results in the training loss to be smaller than the validation loss as expected. The model performs best when using be-

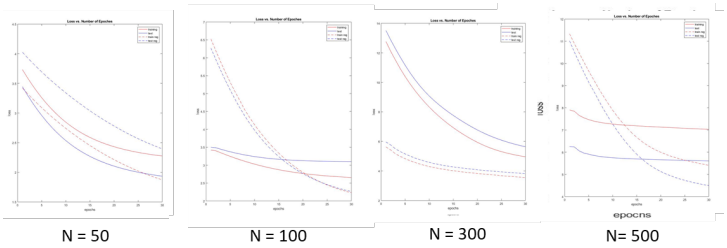


Figure 8: Sensitivity to number of neurons for the simple neural network as measured by the training and validation loss during training, with a learning rate of 5 and an L2-regularization term of 0.001. N is the number of neurons in the hidden layer.

tween 100 and 300 neurons as shown in Figure 8.

The CNN produced better test predictions than the SNN, reaching an accuracy of 76%. Figure 9 shows the training accuracy over 30 epochs of training. An initial learning rate of 0.001, a learning drop factor of 0.05 and L2 regularization of 0.004 were used. More data is expected to increase the accuracy even more. Other architectures produced lower test accuracies. Attempts were made to analyze the features produced after each ReLu layer of the CNN, but no meaningful interpretations were made.

We expect that this prediction accuracy could be significantly higher for the neural network models, especially the CNN, with more architecture fine-tuning, more training examples, and higher-quality data. Some of the training examples are very difficult for even a human to classify due to various filtering problems when extracting the data, such as cloudiness, darkness, and extreme outliers in NDWI calculations.

4.3 Comparison of All Models

Table 3 shows the validation accuracies of all the models using 75% of the data for training and the remaining 25% for validation. The simple supervised learning models generally perform better than the neural networks. Given the fact that neural networks present the most opportunities for future work on this kind of project this is worth examining further.

The basic supervised learning models all used the 11 single-band features. The reason for doing this was to provide some sort of baseline to compare the performance of the neural network models to and it was very easy to extract these features. The neural networks, however, used only the NDWI images. It is likely that there is information in the four bands that is not contained in the NDWI images. Perhaps if we extend the neural network framework to incorporate the other bands then we will get better predictions.

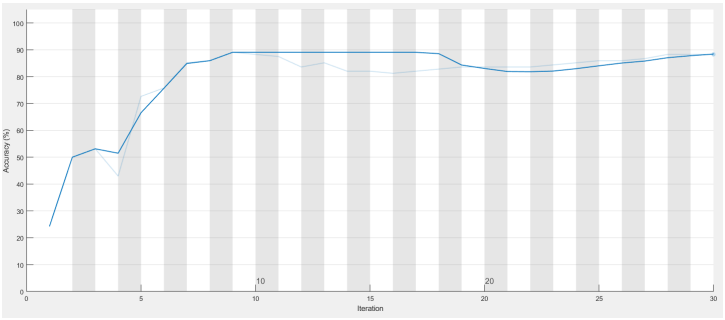


Figure 9: Convolutional neural network training accuracy over 30 epochs.

Table 3: Comparison of model accuracies at 75% training/25% validation. For supervised learning models the value shown is the average of 1000 random realizations of training/validation sets.

Model	Mean Prediction Accuracy (%)
One-vs-one Linear SVM	77
One-vs-one Gaussian SVM	82
One-vs-all Linear SVM	80
One-vs-all Gaussian SVM	83
Softmax	90
SNN (11 features)	41
SNN (images)	71
CNN	76

5 Conclusion

Several machine learning methods (multiclass SVM, softmax, SNN, CNN) were applied to classify satellite images of deltas, rivers and lakes. The softmax model using the 11 averaging features resulted in a test accuracy of 90%, performed the best. This means that our extracted features work well in classifying our target depositional environments. It also probably means our data is limited to implement a proper neural network. Taking this work forward to achieve our goal in developing a geologically driven subsurface interpolation tool, we will have to first be able to locate these depositional environments as regions in a larger satellite image instead of classifying only labeled images. This will result in major complications with the reliability of our averaging features since these features do not capture the spatial structure of such environments. Therefore, we realize that convolutional neural network has to be the way forward with this project along with the need to collect more data and rigorous analysis of the architecture of the CNN. We can also consider developing a generative model to determine the distribution of depositional environment shapes and simulate new realizations of depositional environments based on these distributions.

Contributions

1. Alex

   I wrote the scripts for downloading the images from Earth Engine. The exception of the compression step I did the preprocessing and feature extraction. I figured out how to use Matlab CNN toolbox and wrote a skeleton script that Rayan used and changed to do the CNN analysis. I performed the supervised learning testing and evaluation. Both of us spent equal time on the proposal, milestone, poster, and final report.

2. Rayan

   I was in charge of manually picking the surface features from the Google Earth Engine, and saving them as digitized polygons. I also developed the data loading and compression code in matlab used to automatically take the text files and saving them as compressed 4 band .mat files (250x250x4) using discrete cosine transform method. I also developed most of the deep learning networks. I coded both the single feature and stretched image learning on the simple neural network as well as the convolutional neural network. I also developed the code to view the progression of randomly selected delta, river, and lake throughout the activation layers for interpretation purposes.

References

[1] Bagheri M, Montazer G, & Escalera S. 2012. *Error Correcting Output Codes for multiclass classification: Application to two image vision problems.* The 16th CSI International Symposium on Artificial Intelligence and Signal Processing. http://www.maia.ub.es/~sergio/linked/aisp2012.pdf.

[2] Erickson T. 2016. *Introduction to Earth Engine and TensorFlow in Cloud Datalab.* Github, https://github.com/tylere/ee-jupyter-examples/blob/master/4%20-%20Earth%20Engine%20and%20TensorFlow.ipynb.

[3] MathWorks Documentation. *Create Simple Deep Learning Network for Classification.* https://www.mathworks.com/help/nnet/examples/create-simple-deep-learning-network-for-classificat html.