Automatic Detection and Classification Of Rock Microstructures Through Machine Learning

Stephen Iota, Junyi Liu, Ming Lyu, Bolong Pan, Xiaoyu Wang, Yolanda Gil, Wael AbdAlmageed

University of Southern California

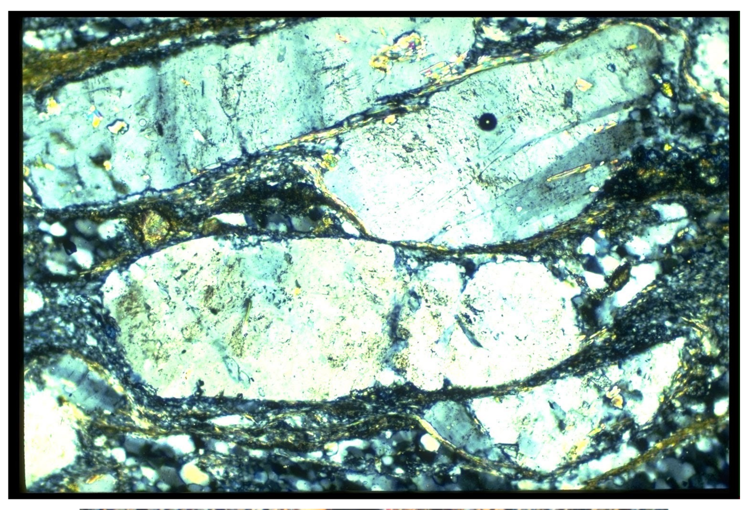
Gurman Gill, Matty Mookerjee

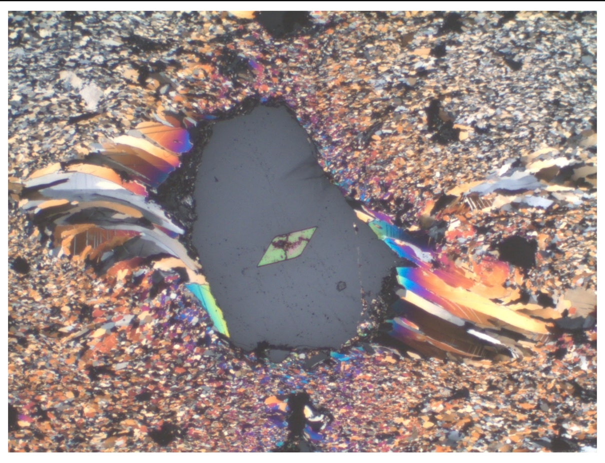
Sonoma State University

Contact: gil@isi.edu

Introduction

Motivation and Dataset





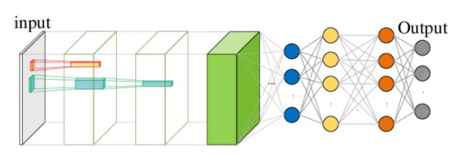
Geologists seek assistance in classifying microscope rock images

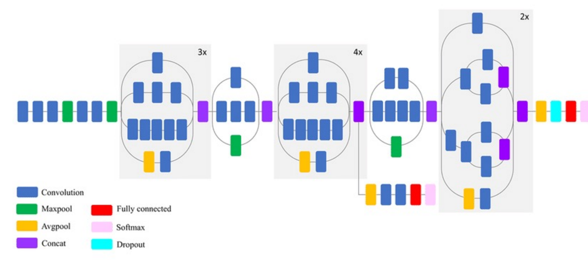
- Sigma clasts are a type of mantled porphyroclasts widely used as kinematic indicators in rock
- Want to automate classification of Sigma clast direction of rotation (CW or CCW)

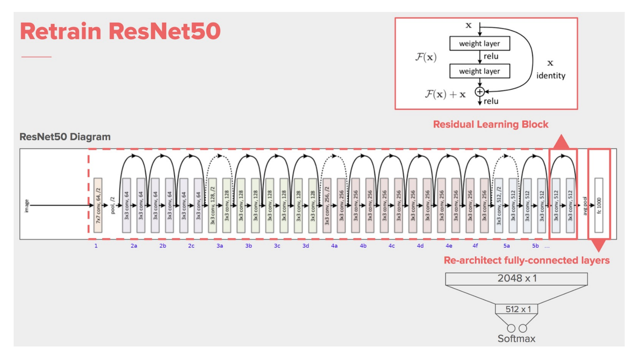
Challenge: Very limited dataset, difficult feature extraction

- Only ~ 100 positive examples
- Sigma clasts are notoriously difficult to classify, even for geologists

Machine Learning Methods







Machine Learning (ML)

- Use positive and negative examples to train models that can be used to predict whether a new image can be classified as a positive example

Convolutional Neural Networks (CNNs)

- CNNs are a type of ML models inspired by the brain.
- Used to extract features such as shape, color and texture to infer image labels.
- Requires thousands of examples!!

Transfer Learning applied to Sigma Clasts Detection Problem

What is Transfer Learning?

- Typically, CNNs require hundreds and thousands of training examples to achieve useful prediction accuracy for a given problem
- Transfer Learning leverages CNN models trained with other data and does additional training with the data at hand
- On top of three widely-used Transfer Learning models, we train additional prediction layers on our sigma clasts data and observe the results.

Comparing Different Transfer Learning Approaches: InceptionV3, ResNet50, VGG19

	model	train_loss	train_acc	val_loss	val_acc	f1_score
0	InceptionV3_epochs132_train_acc-0.9939_val_acc-0.8250_Regularized-False.h5	0.031	0.994	1.806	0.825	[[metric, precision, recall, f1-score, support], [0.556, 0.833, 0.667, 6.0], [0.9, 0.643, 0.75, 14.0], [0.905, 0.95, 0.927, 20.0], [0.825, 0.825, 0.825, 0.825], [0.787, 0.809, 0.781, 40.0], [0.851, 0.825, 0.826, 40.0]]
3	ResNet50_epochs02_train_acc-0.8221_val_acc-0.8250_Regularized-False.h5	0.692	0.853	0.596	0.825	[[metric, precision, recall, f1-score, support], [0.667, 0.333, 0.444, 6.0], [0.733, 0.786, 0.759, 14.0], [0.909, 1.0, 0.952, 20.0], [0.825, 0.825, 0.825, 0.825], [0.77, 0.706, 0.718, 40.0], [0.811, 0.825, 0.808, 40.0]]
1	InceptionV3_epochs18_train_acc-0.9693_val_acc-0.7250_Regularized-True.h5	10.634	0.791	2.451	0.725	[[metric, precision, recall, f1-score, support], [0.333, 0.333, 0.333, 6.0], [0.818, 0.643, 0.72, 14.0], [0.783, 0.9, 0.837, 20.0], [0.725, 0.725, 0.725, 0.725], [0.645, 0.625, 0.63, 40.0], [0.728, 0.725, 0.721, 40.0]]
4	VGG19_epochs14_train_acc-0.9939_val_acc-0.6250_Regularized-False.h5	0.180	0.982	2.622	0.625	[[metric, precision, recall, f1-score, support], [0.3, 1.0, 0.462, 6.0], [1.0, 0.5, 0.667, 14.0], [0.923, 0.6, 0.727, 20.0], [0.625, 0.625, 0.625, 0.625], [0.741, 0.7, 0.618, 40.0], [0.857, 0.625, 0.666, 40.0]]
2	ResNet50_epochs20_train_acc-0.9264_val_acc-0.6250_Regularized-True.h5	2.903	0.779	5.850	0.625	[[metric, precision, recall, f1-score, support], [0.6, 0.5, 0.545, 6.0], [0.667, 0.429, 0.522, 14.0], [0.615, 0.8, 0.696, 20.0], [0.625, 0.625, 0.625, 0.625], [0.627, 0.576, 0.588, 40.0], [0.631, 0.625, 0.612, 40.0]]
5	VGG19_epochs35_train_acc-0.5215_val_acc-0.6000_Regularized-True.h5	1.231	0.521	1.211	0.600	[[metric, precision, recall, f1-score, support], [0.0, 0.0, 0.0, 6.0], [1.0, 0.286, 0.444, 14.0], [0.556, 1.0, 0.714, 20.0], [0.6, 0.6, 0.6, 0.6], [0.519, 0.429, 0.386, 40.0], [0.628, 0.6, 0.513, 40.0]]

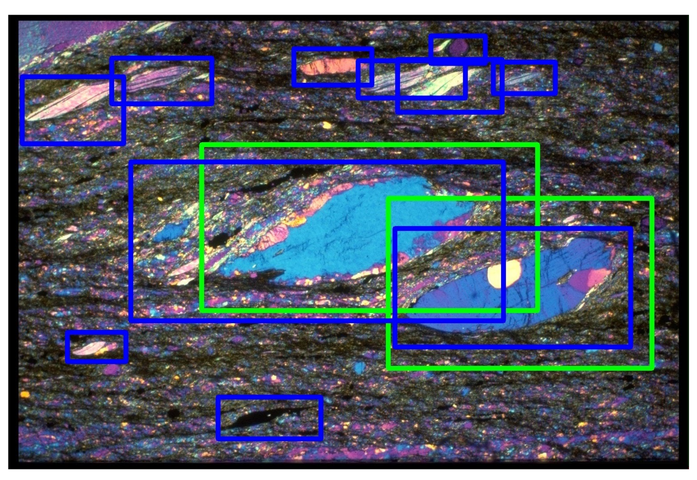
Experimental Setup

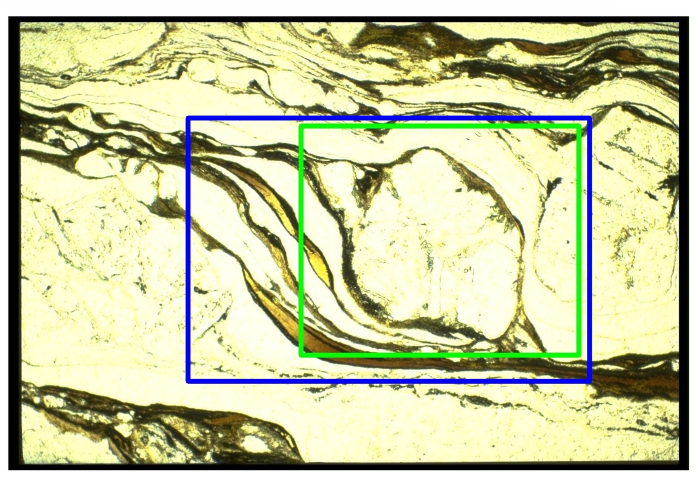
- Train/Validation Split = 0.8 on total of 100 images
- Epochs = 50
- Optimizer = Adam
- Loss = categorical cross entropy
- Iteration step size = 1e-4
- Activation = Relu, softmax for last layer

Through hyperparameter tuning, we are able to optimize prediction accuracy on our data set.

Current Areas of Work

Detecting Multiple Objects in an Image



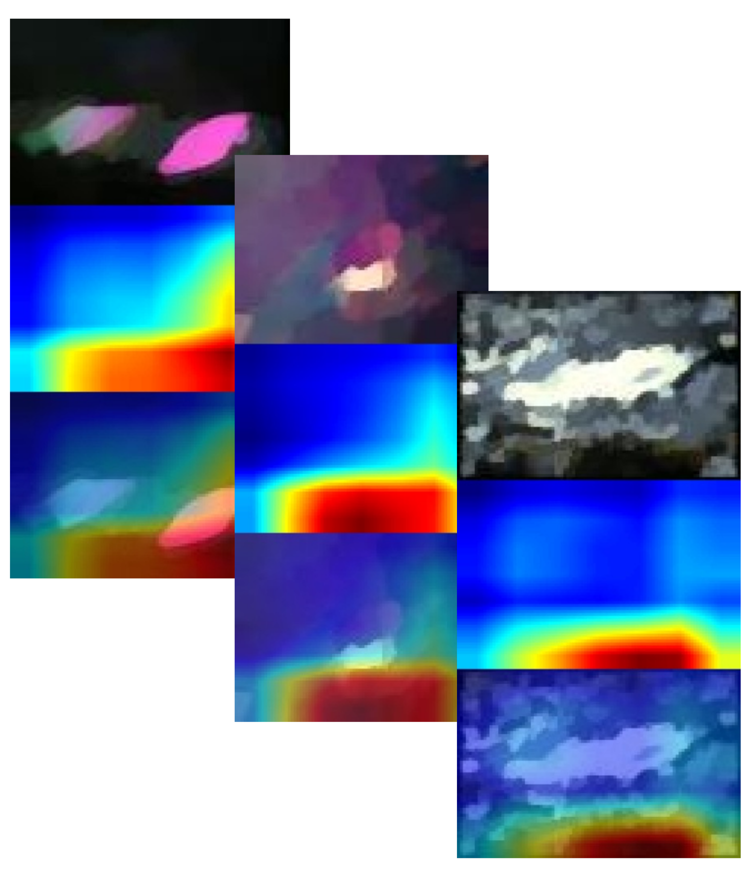


- YOLOV3 is the third iteration of CNN based object detection architecture, able to output real time bounding boxes around Sigma clasts

Blue – ground truth labels
Green – predicted labels

- Initial implementations of YOLO show the ability to distinguish multiple sigma clasts in a single image: not possible through Transfer Learning
- In the future: fine tune this approach using tail detection.

Efficient Exploration of Models through Visualization



Understand Model Prediction Accuracy

- In order to prioritize the exploration of possible new models with different settings, we are developing a computational experimentation environment to visualize different CNN network layers, classification heatmaps, and comparative metrics.
- We propose heatmaps that show where the CNN model is ''looking'' for sigma clasts, to compare and distinguish where some models are underperforming