Additive Manufacturing Defect Detection using Neural Networks

James Ferguson

May 16, 2016

THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

Outline

- Introduction
- Background
- Edge Detection
  - Methods
  - Results
- Porosity Detection
  - Methods
  - Results
- Conclusion / Future Work

Introduction

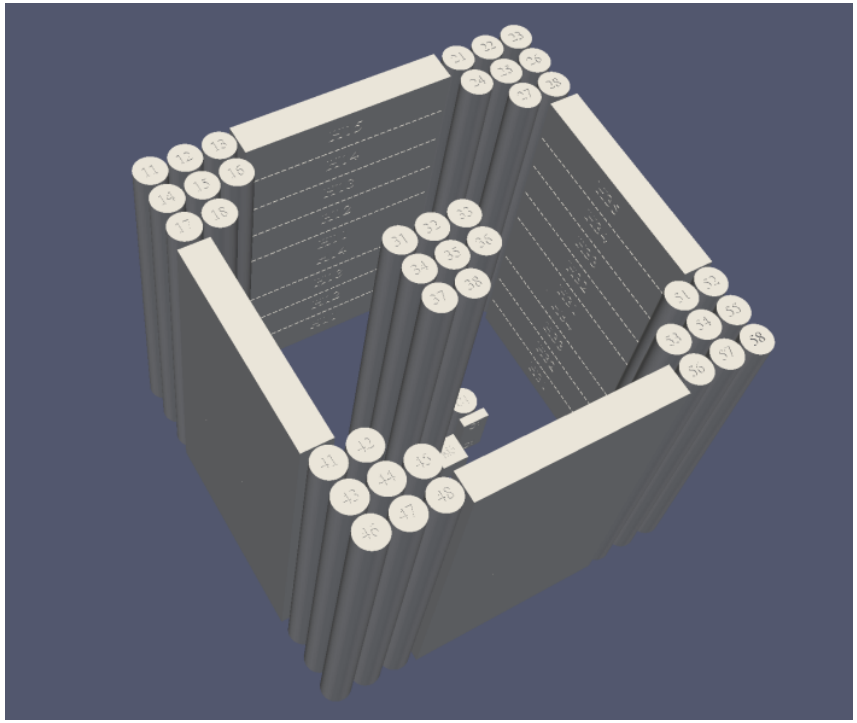
- Defect detection is important for quality control and certification of 3D printed objects

- 2 parts to this project
  - Edge Detection/Analysis
  - Porosity Detection

Background

Printer: Arcam Q10

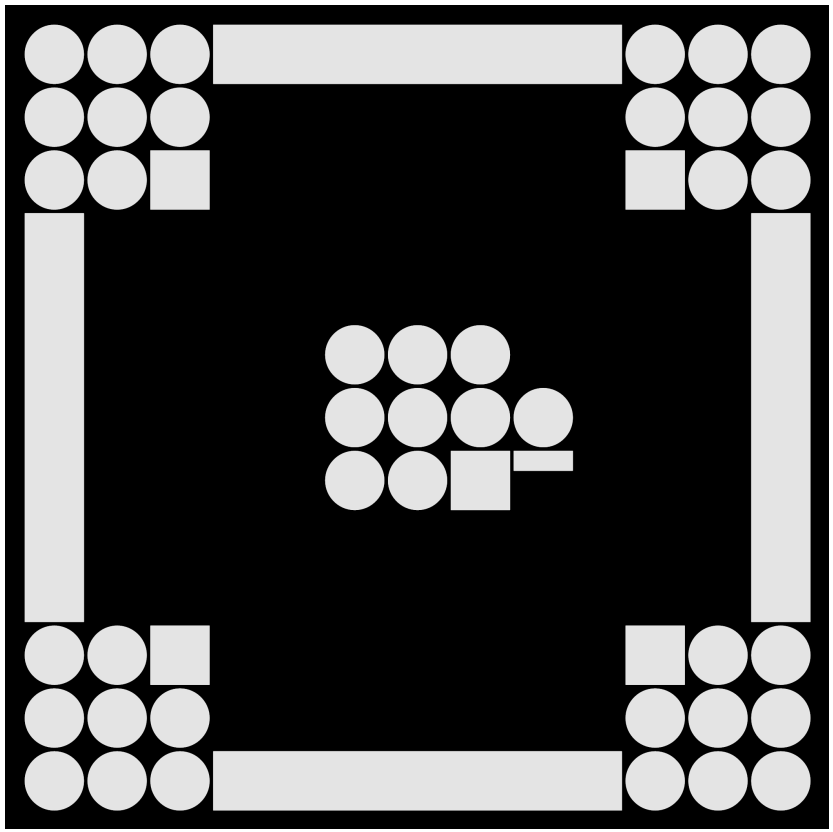
3D CAD model



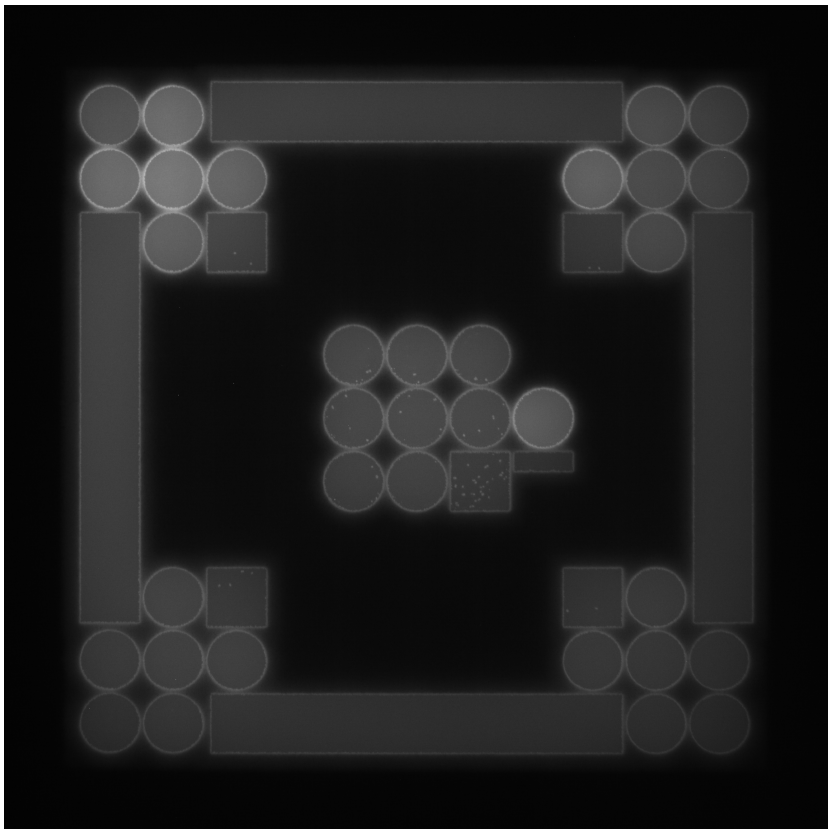
Background

Input Data: Two 2100 x 2100 images

Slice

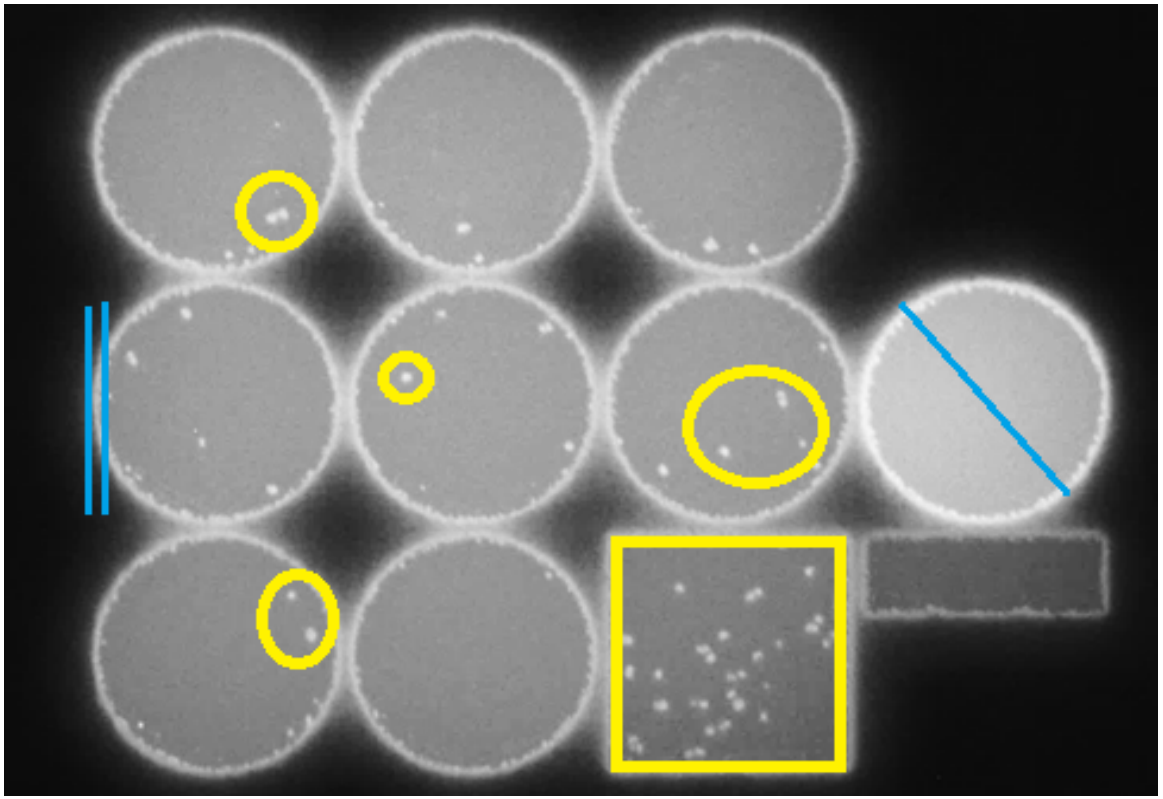


Near-IR image captured each layer



Background

Defects and features



Porosity

Geometric accuracy

Background

Caffe: A deep learning framework.

- **Easy to use**
- **Highly optimized**
- **Open source**
- **Can use CPU or GPU**

Maintained by Berkeley Vision and Learning Center (BVLC)
http://github.com/BVLC/caffe


```
layer {
  name: "mnist"
  type: "Data"
  top: "data"
  top: "label"
  data_param {
    source: "input_leveldb"
    batch_size: 64
  }
}
layer {
  name: "ip"
  type: "InnerProduct"
  bottom: "data"
  top: "ip"
  inner_product_param {
    num_output: 2
  }
}
layer {
  name: "loss"
  type: "SoftmaxWithLoss"
  bottom: "ip"
  bottom: "label"
  top: "loss"
}
```

Background

Caffe: A deep learning framework

- Only Linux and OS X platforms are supported officially.
- First task was to port to Windows platform for compatibility
- Unofficial documentation for porting an older version to Windows on BVLCs github
- Straight forward process to update to a current version to use CUDA 7.0

Edge Analysis

- **Important for two reasons**
  - **Geometric Accuracy**
  - **Porosity Detection**

- **Four step process**
  - **3 preprocessing steps**
  - **Detection step**

Preprocessing Steps

1. Extract Contours
2. Compute local normal
3. Extract Intensity Profile

Detection Step

4a. Downhill Simplex

4b. Neural Network

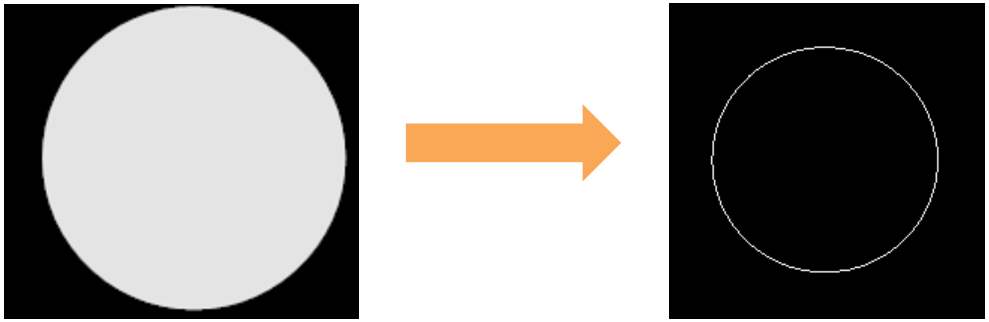
Edge Detection

Extract Contours

- Primary handled by OpenCV library.

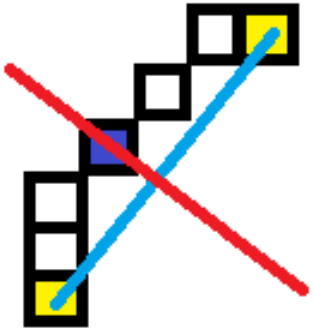
Output

- A list of points along every contour
- A hierarchy of the contours



Edge Detection

Compute Local Normal

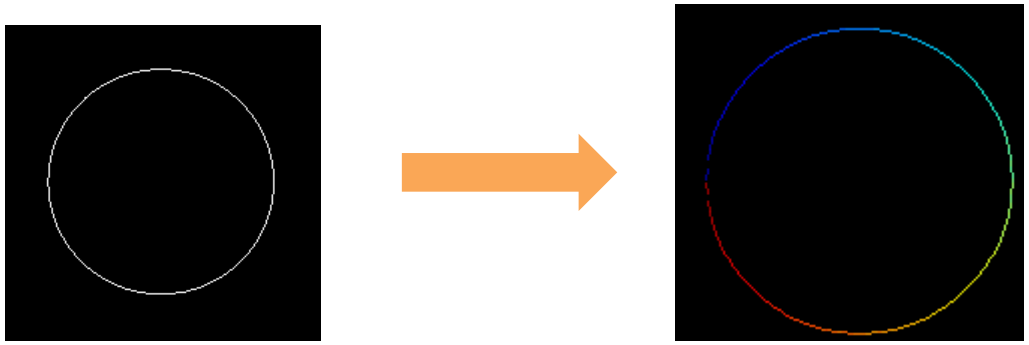


- For every point calculate a line between the points 3 positions to each side.

- Compute a line perpendicular from this line that passes through the main point

Output

- The angle between the normal vector and the X axis



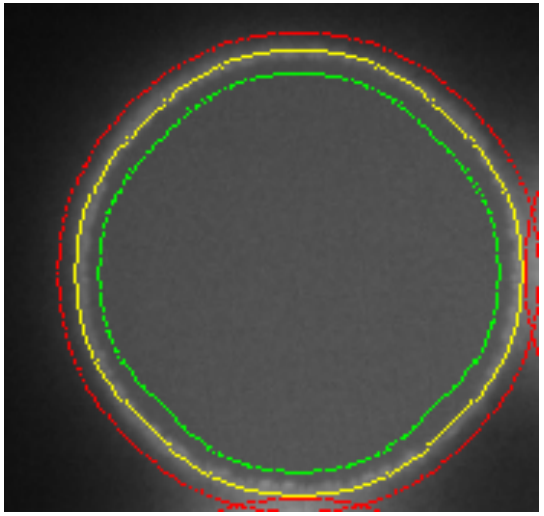
Edge Detection

Extract Intensity Profile

- Get the pixel values along the normal
- Profile length is a fixed length of 15 pixels.

Output

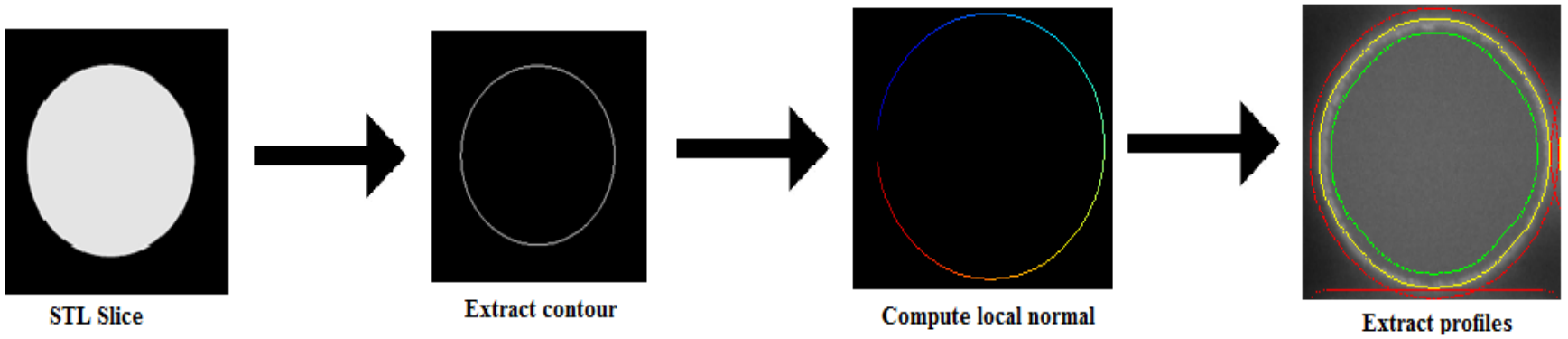
- A list of the profile values



A profile, [0.65, 0.62, 0.62, 0.66, 0.73, 0.79, 0.81, 0.78, 0.73, 0.63, 0.52, 0.51, 0.51, 0.51, 0.51]

Edge Detection

Preprocessing steps



Edge Detection

Downhill Simplex

- Find the edge by fitting a sigmoid to the profile
- Find the index of the sigmoid that crossing a fixed threshold
- Sigmoid is fit to the profile using the Nelder-Mead method.
- Nelder-Mead searchs over multiple dimensions of variables with the goal of minimizing a cost function
- $\text{sigmoid} = tanh(\alpha - (\beta * i))$
- Cost Function $= \sum \uparrow \blacksquare (sigmoid(i) - profile[i]) \uparrow 2$

Edge Detection

Downhill Simplex

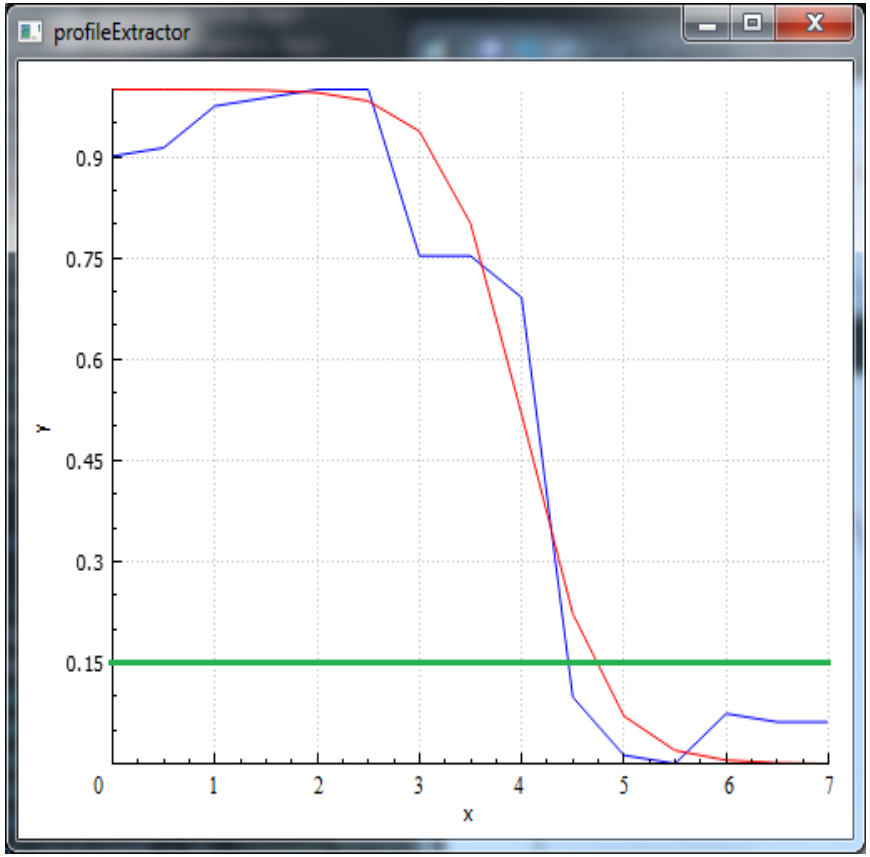
- Algorithm works by moving a simplex through the search space until a local minimum is reached.
- A simplex is a shape with n+1 vertices in n dimensions.
- For this problem the simplex is a triangle.
- Each vertex is evaluated and the worst vertex is removed and a new vertex is added.

Edge Detection

Downhill Simplex

The index which the sigmoid crosses the threshold is the edge.

The threshold can be moved to change the “tightness” of the edge



Edge Detection

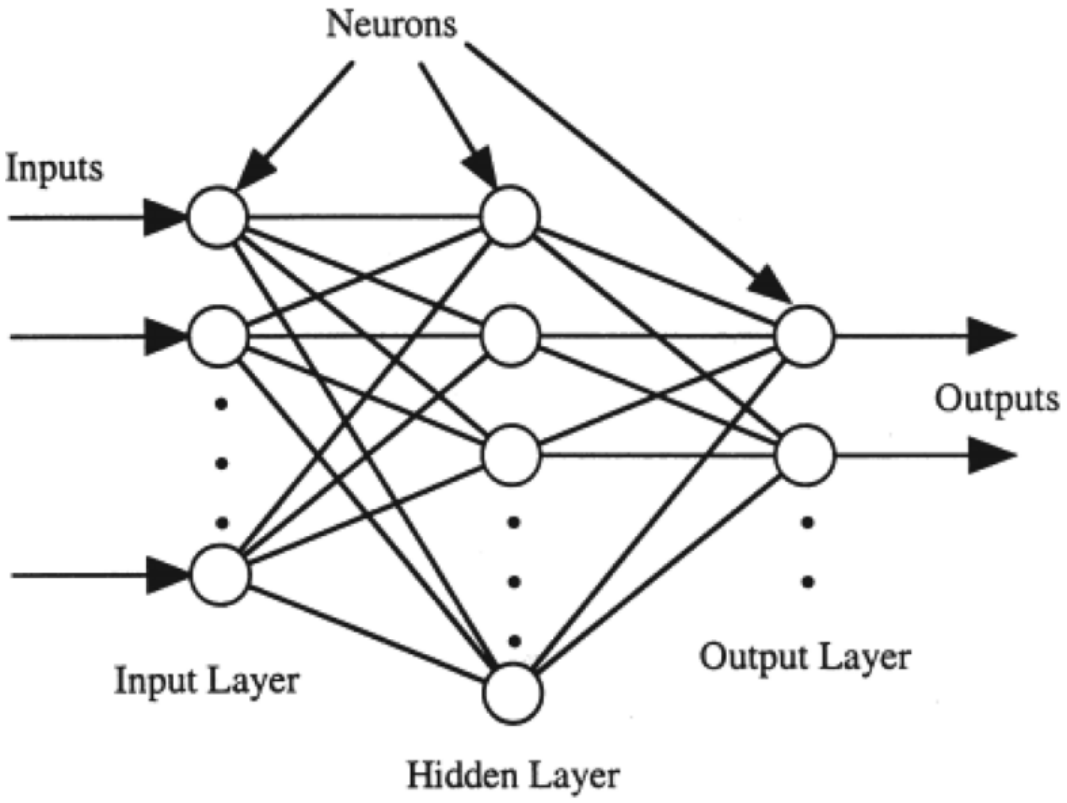
Neural Network

- New method of edge detection using a neural network.
- Easily parallelized on GPU's
- Ease of implementation using Caffe.

Edge Detection

Neural Network - Architecture

- Feed forward fully connected network.
- 15 inputs, the profile
- 1 hidden layer with 50 neurons
- 15 outputs, the index of the edge
- Weights adjusted using gradient descent backpropagation.

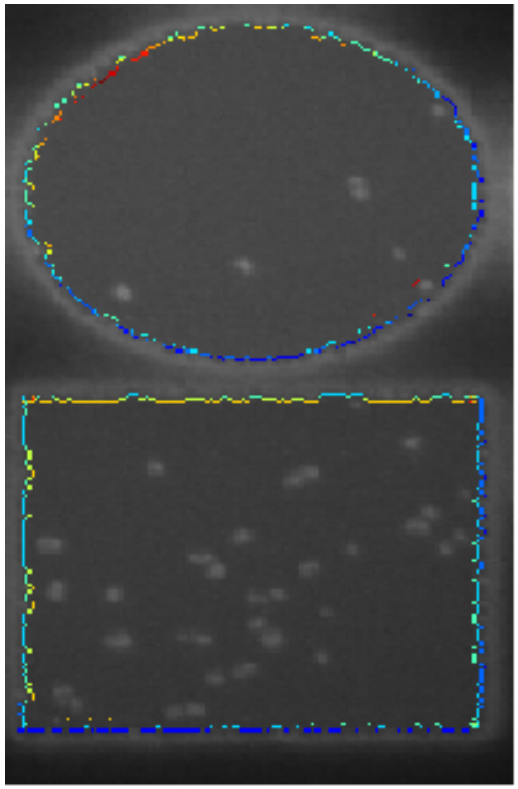


Edge Detection

Neural Network – Training

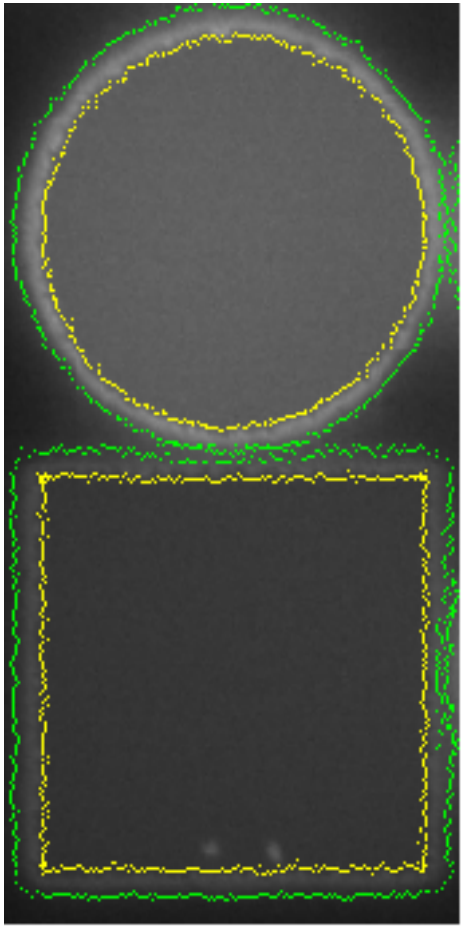
- The results from the downhill simplex are used as ground truth.
- Examples are randomly split into 2 sets
  - Training Set – 20% of the profiles
  - Testing Set – 80% of the profiles
- Caffe trains over the entire training set then processes the training set to get a total error.
- Caffe runs until the error does not decrease for 3 consecutive runs
- Converges after ten epochs or approximately 15 seconds.

Edge Detection - Results

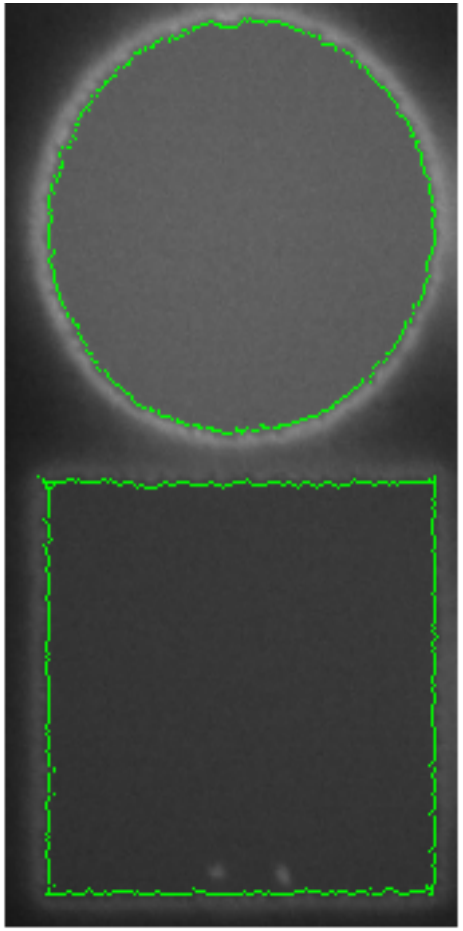


Distance Scale

Min Max



Downhill Simplex Detection



Neural Network Detection

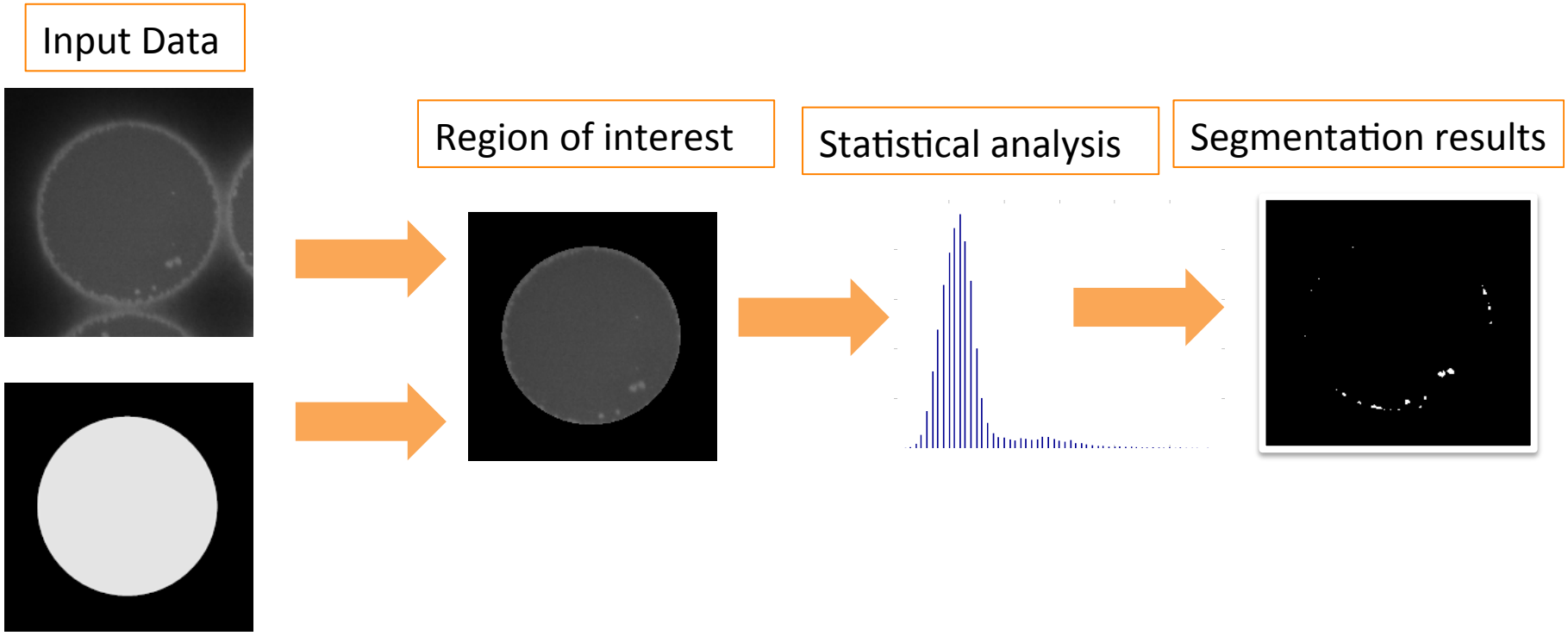
Edge Detection - Results

Method	Number of profiles	Time (seconds)
Downhill Simplex (CPU)	21,325	21.6
Neural Network (GPU)	21,325	0.7

CPU: Intel Xeon ES-1650
GPU: Nvidia Quardo K2200

Porosity Detection

Current Method

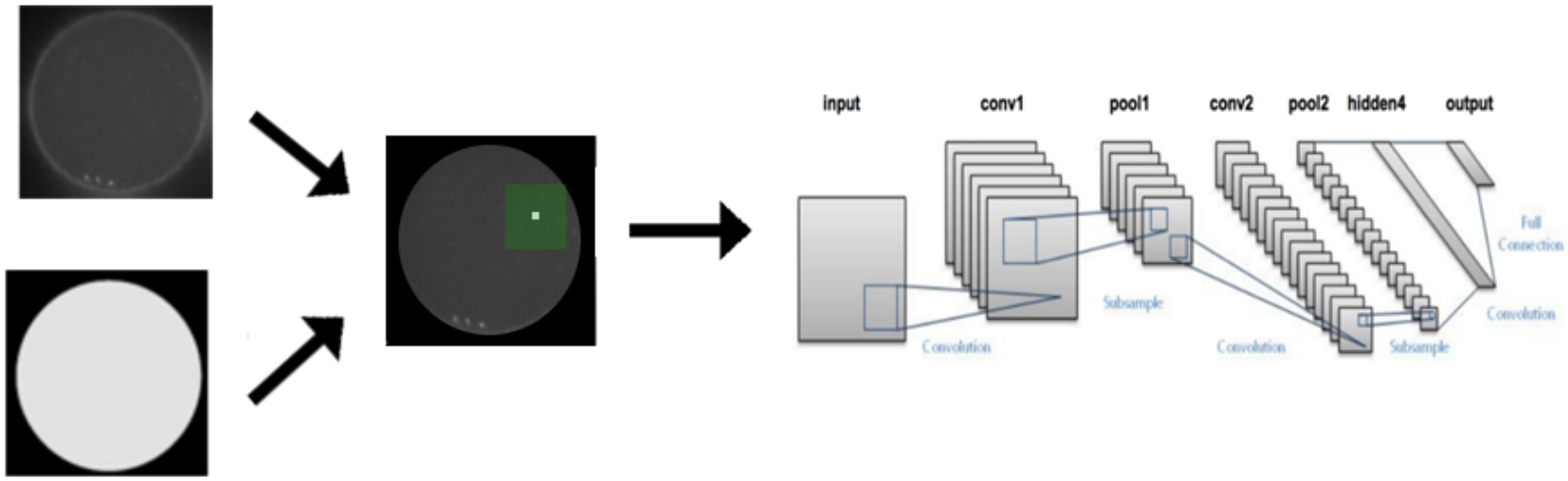


Porosity Detection

Convolutional Neural Network Method

For each pixel *p* in the region of interest a convolutional neural network is used to classify *p* as either a pore or non-pore

Input Data | Region of interest | Convolutional Neural Network

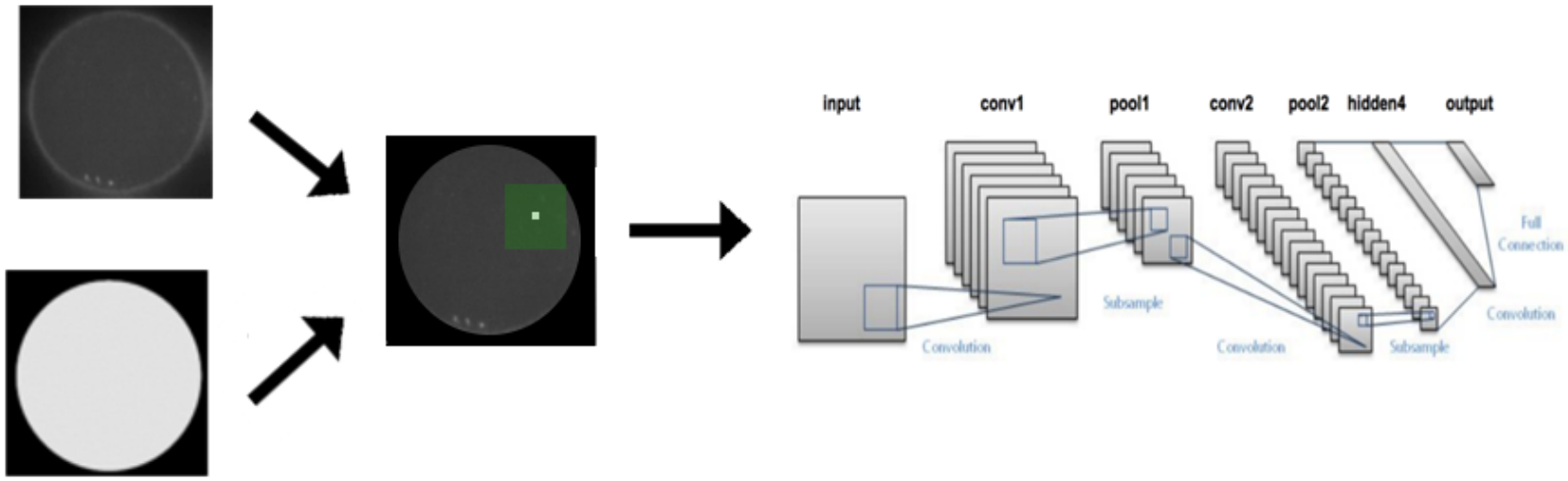


Porosity Detection

Convolutional Neural Network Method

The input to the neural network is a 17x17 window that is centered on *p*.
The output is the probability that *p* is a pore.

Input Data | Region of interest | Convolutional Neural Network



Porosity Detection

Convolutional Neural Network Architecture

- Strategy and architecture was adopted from Ciresan et al. for classifying cell membranes.
- Convolutional layer use kernels that move across the input and generates a 2D activation map
- Max pooling layers reduces the input by only taking the max value within the kernel.

Layer	Type	Maps and Neurons	Kernel Size
0	Input	1 x 17 x 17	
1	Convolutional	16 x 17 x 17	4 x 4
2	Max Pool	16 x 9 x 9	2 x 2
3	Convolutional	16 x 6 x 6	4 x 4
4	Max Pool	16 x 3 x 3	2 x 2
5	Convolutional	16 x 2 x 2	2 x 2
6	Fully Connected	100 neurons	1 x 1
7	Output	2 neurons	

Porosity Detection

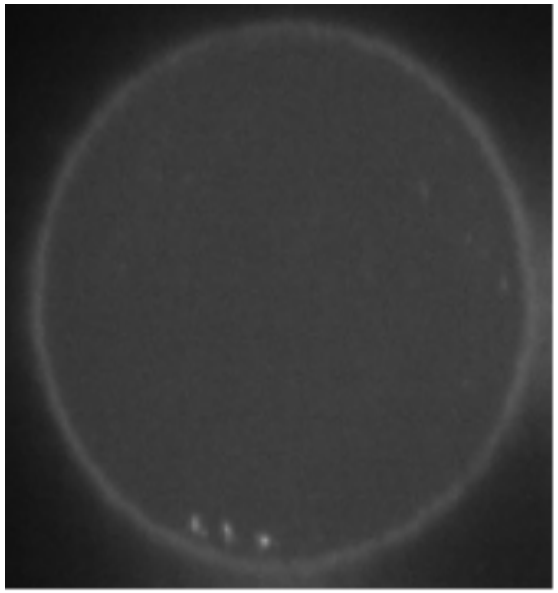
Convolutional Neural Network Results

- Overall the network detected porosity at a comparable level to the current method.

- Several Issues.
  - To large of area around the pores was detected.
  - False positives near the edges

Porosity Detection

Convolutional Neural Network Results



Future Work

- **Edge Detection**
  - Increase speed by processing all the profiles at once instead of by layer
  - Hand label the ground truth data for better accuracy

- **Porosity Detection**
  - Use the edge detection to create a better mask for improved accuracy near edges.
  - Incorporate log data from the printers for better detection.

Acknowledgements

- Committee members Dr. Berry, Dr. Steed, Dr. MacLennan
- Dr. Paquit at ORNL
- CISML for supporting my assistantship at ORNL through GRAMs

References

- Y Jia, E Shelhamer, J Donahue, S Karayev, J Long, R Girshick, S Guadarrama, and T Darrell. Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093, 2014

- J. A. Nelder and R. Mead, A simplex method for function minimization, Computer Journal 7 (1965), 308–313

- Dan Claudiu Ciresan, Alessandro Giusti, Luca Maria Gambardella, and Jurgen Schmidhuber, “Deep neural networks segment neuronal membranes in electron microscopy images,” in Neural Information Processing Systems, 2012

- D. Scherer, A. Muller, and S. Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In International Conference on Artificial Neural Networks, 2010.

Questions?