Automating Measurements of Carbon Nanotube Transmission Electron Micrographs

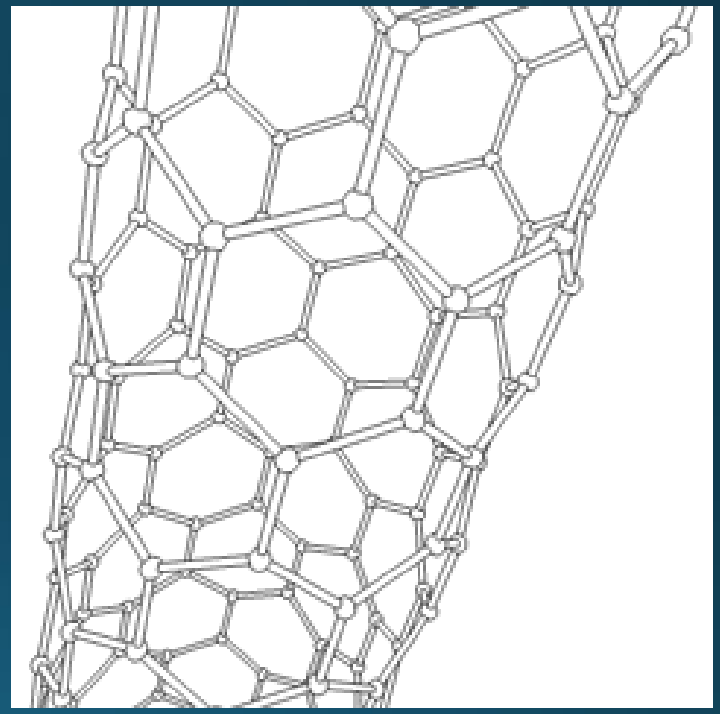
Max Wiedmann

General Assembly DSI Capstone Presentation

9/8/20

Introduction to Carbon Nanotubes

- Carbon nanotubes (CNTs) are cylindrical tubes made of carbon.
- CNTs can be single-walled or multi-walled.
- Their diameters are on the scale of a few nm.
- They possess useful mechanical and electrical properties.
- Researchers are interested in measuring the distribution of diameters in the samples they grow.

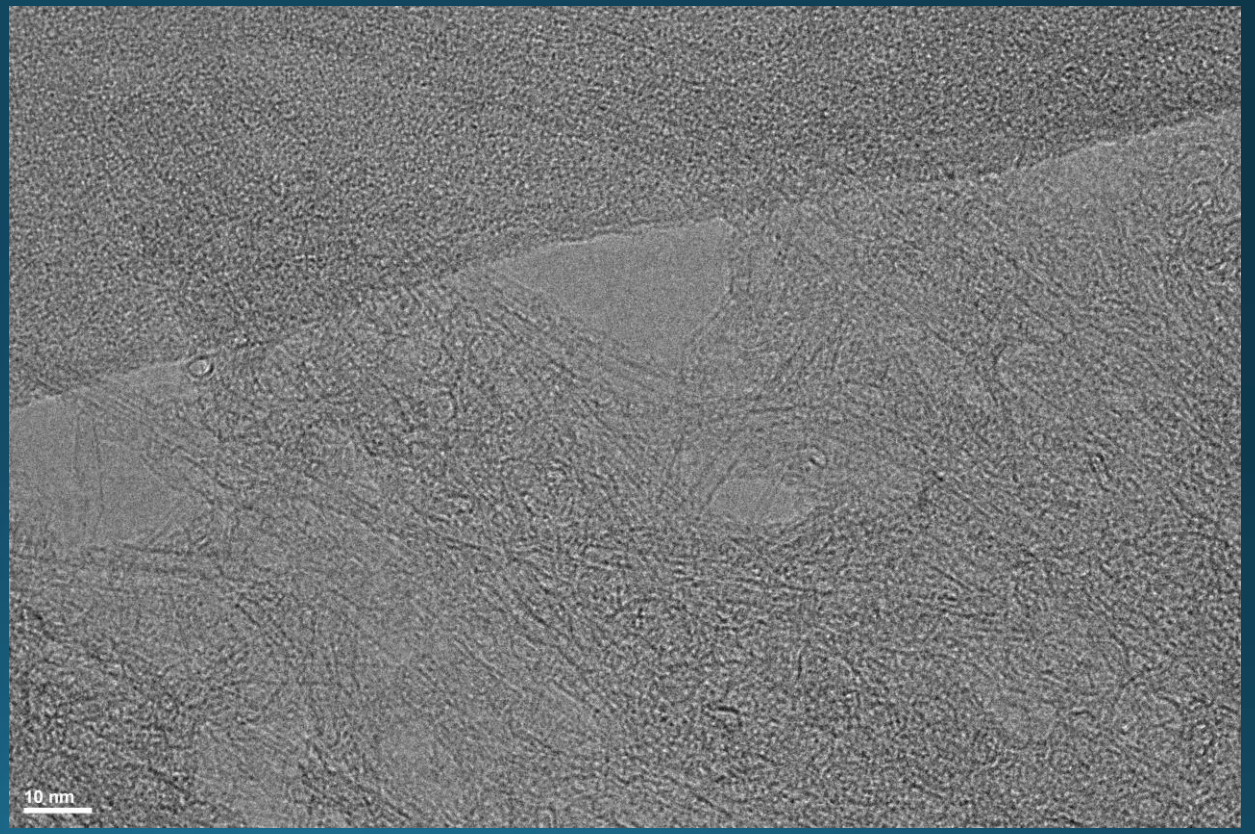


Rotating single-walled CNT

Image credit: https://en.wikipedia.org/wiki/File:Kohlenstoffnanoroehre_Animation.gif

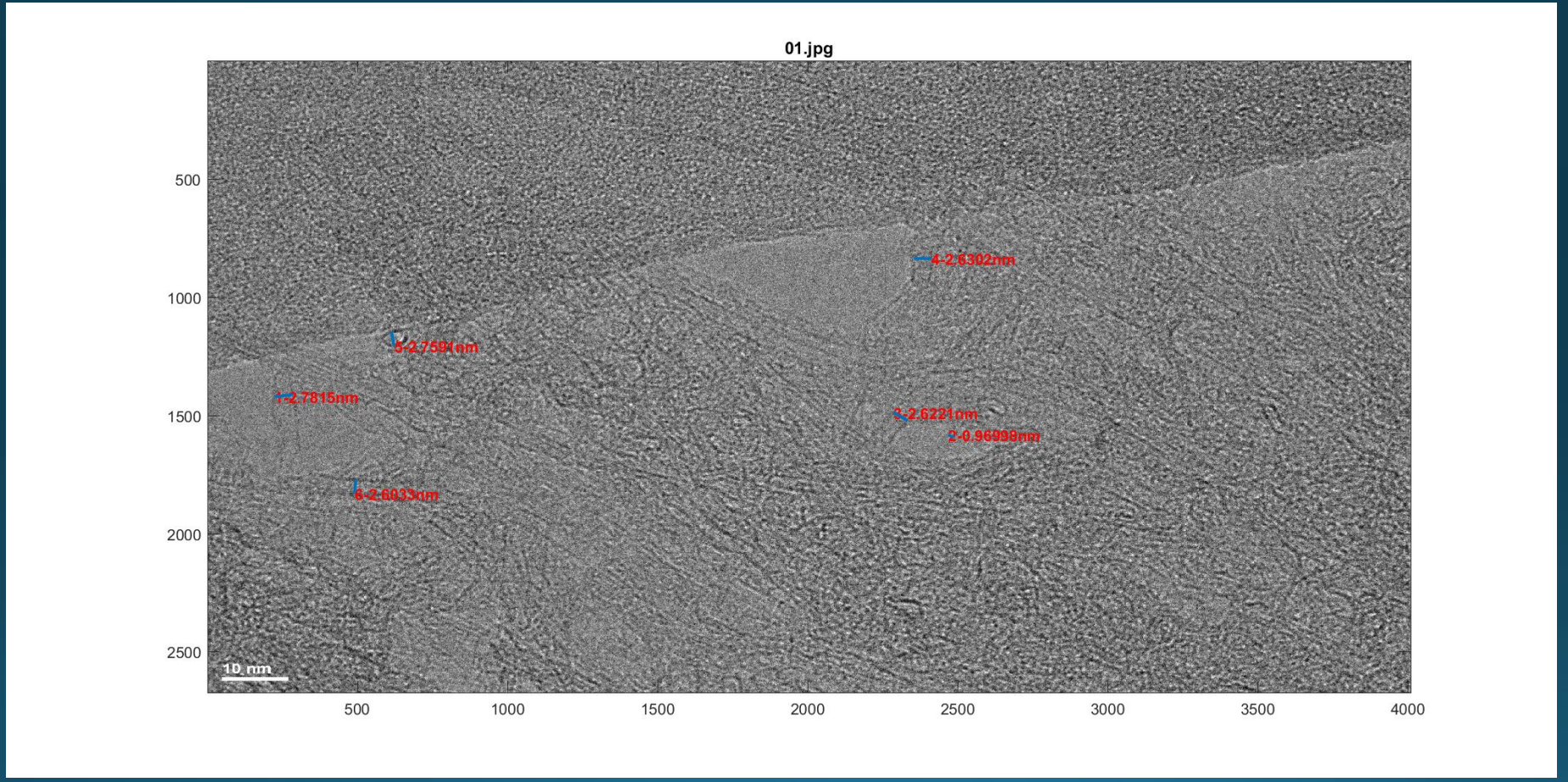
Problem Statement: Manual CNT Diameter Measurements

- Currently, diameter measurements are taken manually.
- For this particular end-user, a researcher uses MATLAB script draw to a line across a CNT.
- The script automatically records the length of that line and produces a labelled image.
- Each dataset, consisting of around 30 images, takes about 2 hours to process.
- Datasets are processed year-round at a rate of about 1 per week.



Unlabelled CNT TEM micrograph

Manually Measured Micrograph with Six Measured CNTs



Output image from MATLAB script

Automating the Process

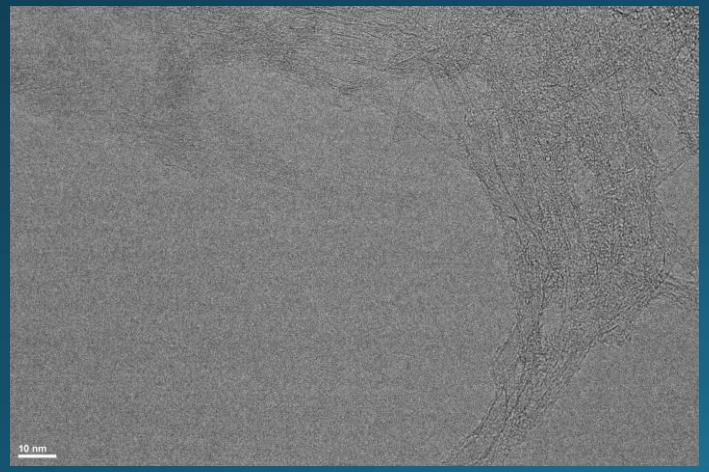
Automatically measuring the diameters of CNTs can be broken into two steps:

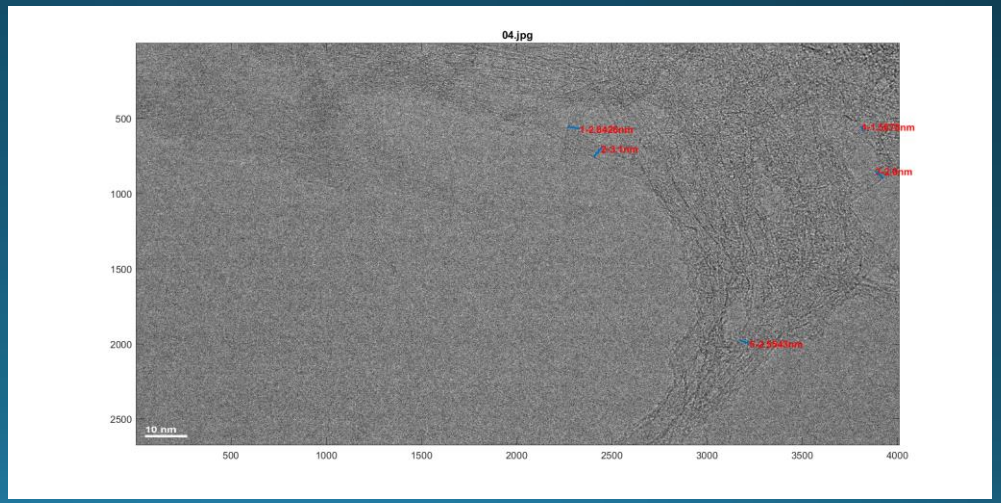
1. Identifying which CNTs are suitable for measurement
2. Measuring the diameters of identified CNTs

Step 1. is now mostly complete

What was provided

- Multiple datasets each containing:
  - Raw CNT TEM images
  - Labelled Images
  - A text file with all measured diameters
    - Each measurement was not named, however



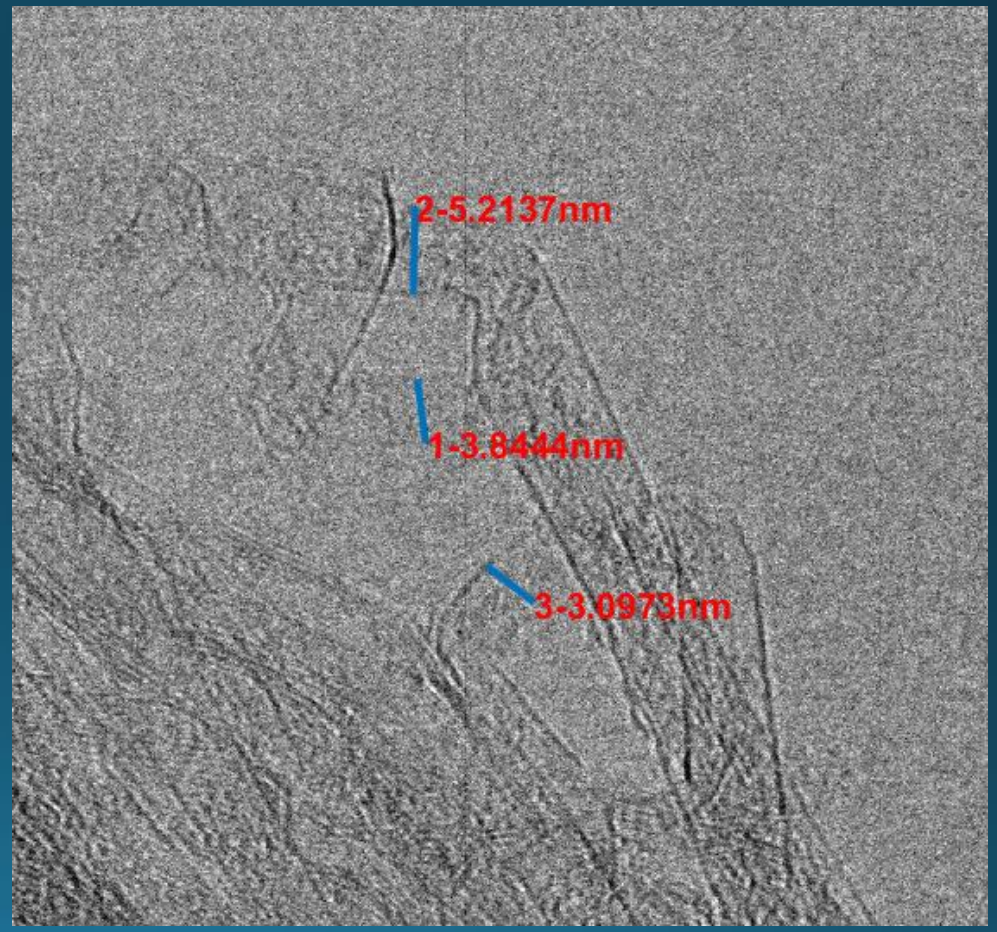


Starter Model: CNT Classifier

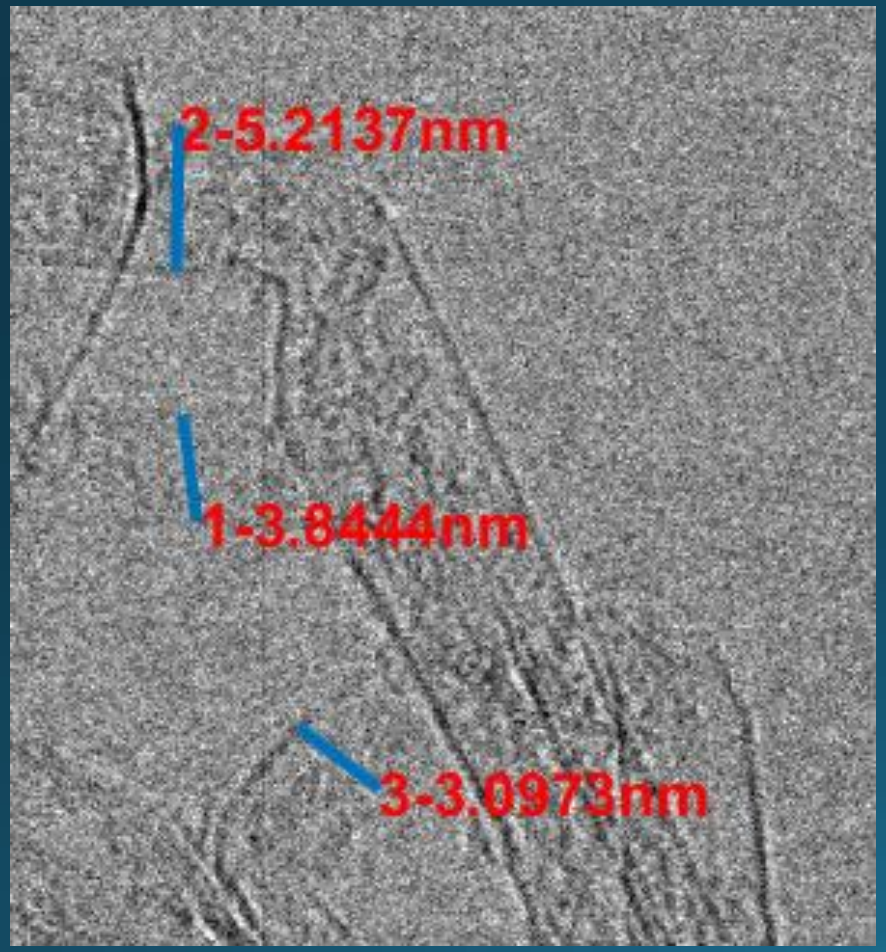
- To start, create a model that can classify if an image has a measurable CNT.
- Extract regions of the raw images known to have good (isolated, measurable) CNTs present, using the locations of each measurement from the labelled images.
- Extra regions with no good CNTs (negative)
- Train a convolutional neural network to classify good CNTs
- Later, segment whole images into grids and run the classifier model to find CNTs for measurement.

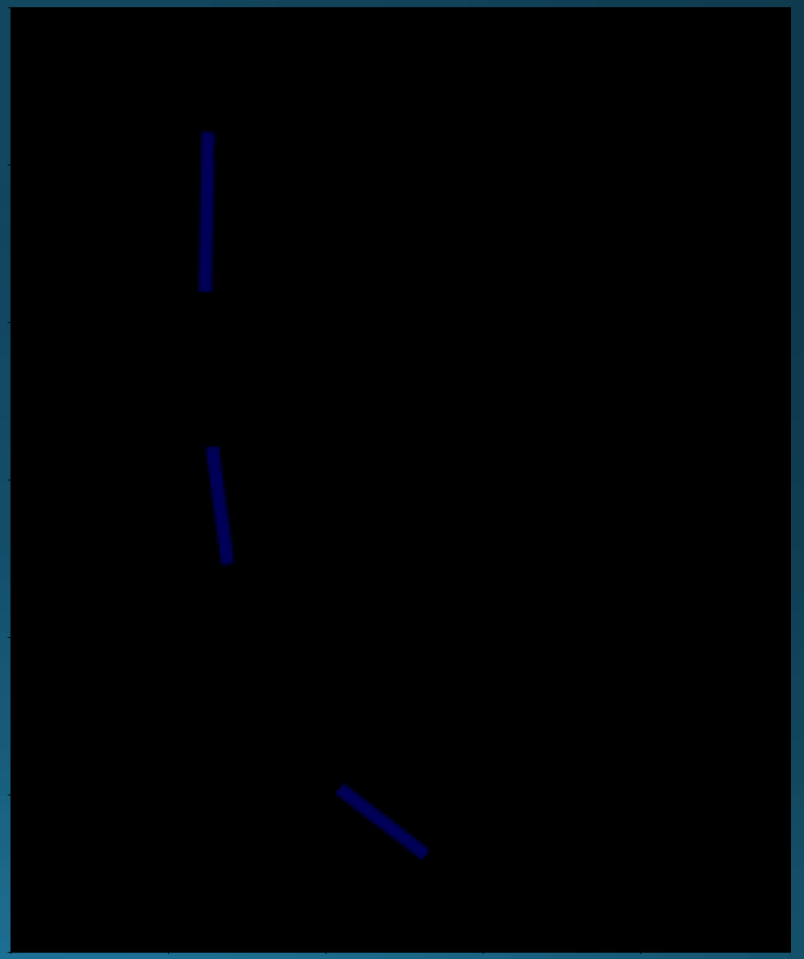
Extracting Good CNTs

1. Use color filtering to extra blue lines from labelled image.
2. Get coordinates of each blue line.
3. Find line center point and measure length for later diameter measurements.
4. Use center point to extract images about each measurement point.

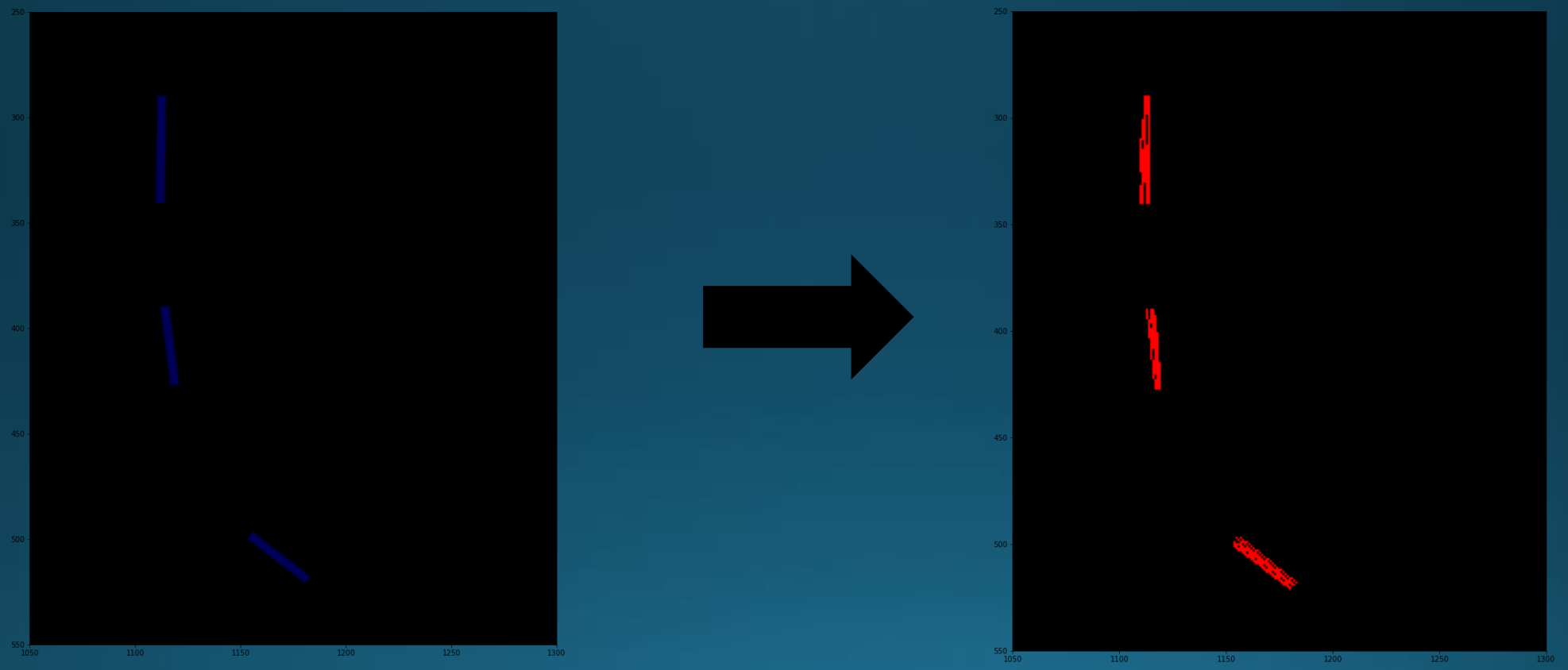


Color Filtering





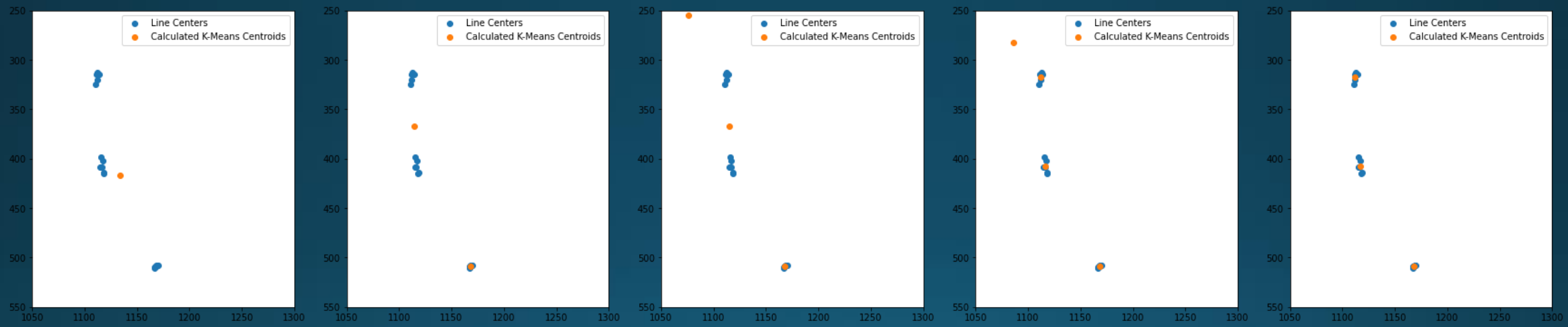
Apply Hough Transform to get Line Segment Coordiantes



OpenCV’s HoughLinesP function returns line segments (x1,y1,x2,y2) from an image with sharp edges.

Line Segment Clustering

- Use K-means to cluster line segements



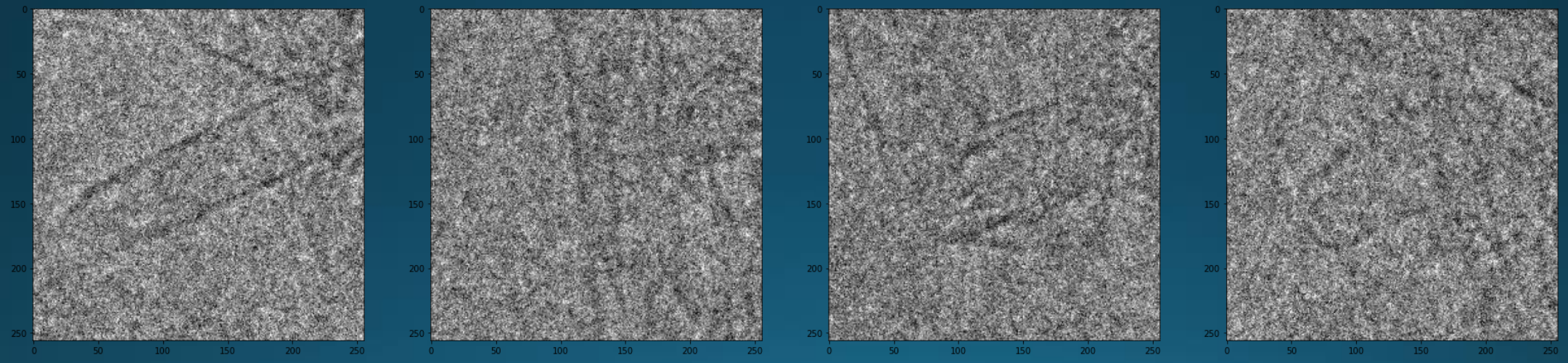
- How do we select k when we don’t know the number of clusters?
- Silhouette score!

But wait, just use the MATLAB Script .fig Files!

- After speaking with the future end user of this product… on Sunday, it was revealed that the MATLAB script that was used to generate each labelled image also output a .fig files.
- Scipy supports reading .fig files, somewhat easily.
- Each .fig file contains the raw image data along with the line and text labels used to create labelled images.
- **Talk to your end user!**

Extracting Good CNT Images from .fig Files

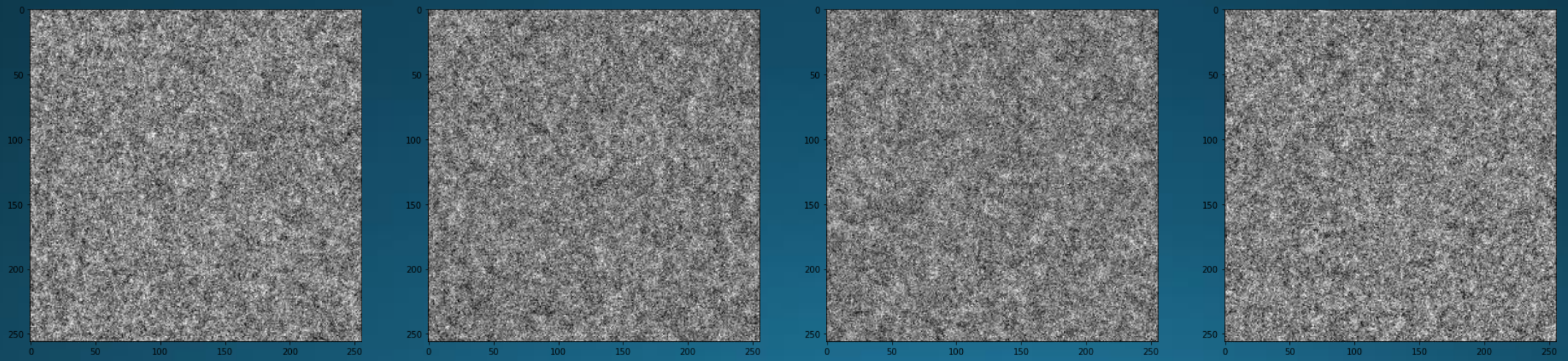
- Use stored user-drawn measurement lines to extract images from measurement locations.



128 px x 128 px images of the four measured CNTs from a single TEM micrograph.

Negative Images

- Images that did not contain measurement points were samples randomly for each image.
- The location of each randomly sampled image was checked against the locations of each measurement to ensure there was no overlap between good CNT images and negative images.

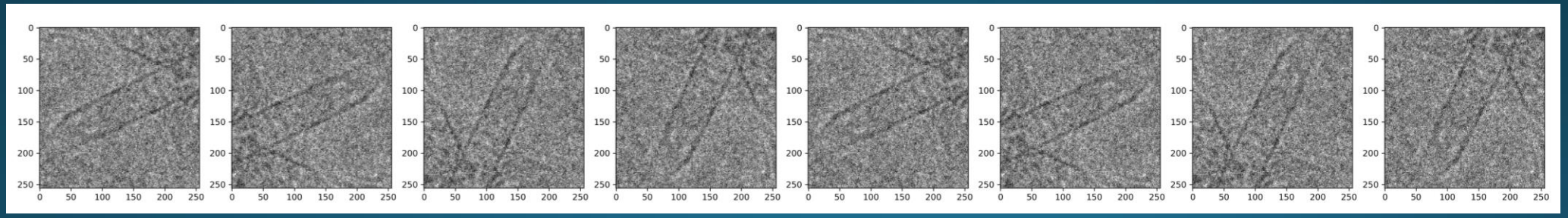


Four negative images that do not contain measurement points.

Data Augmentation

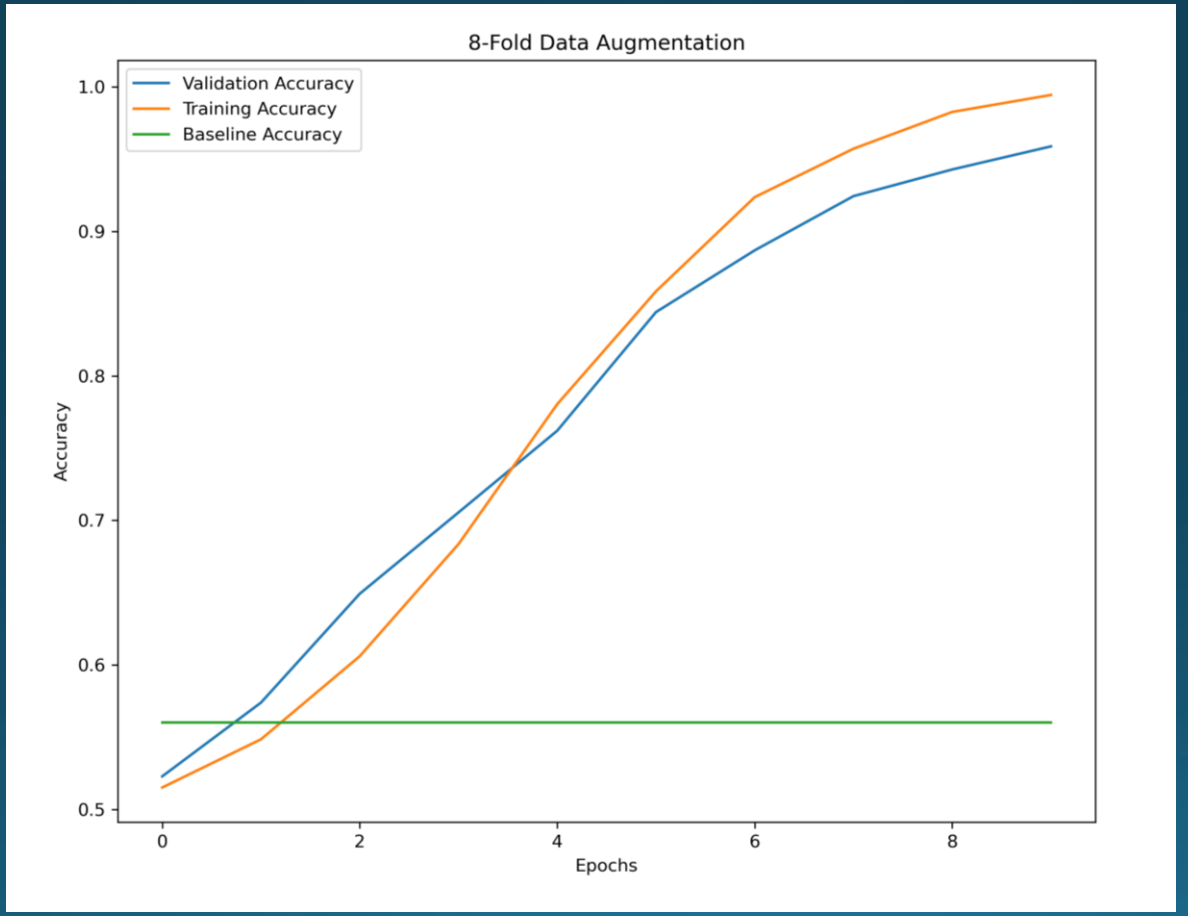
- Total datasets used: 3
- Images per dataset ~ 25
- Average number of measurements per image ~ 5
- Total measurements ~ 375

- The number of good CNT images were augmented by rotating each image 90, 180, and 270 degrees. This quadrupled the number of images. Then each image was flipped, doubling the number of images.

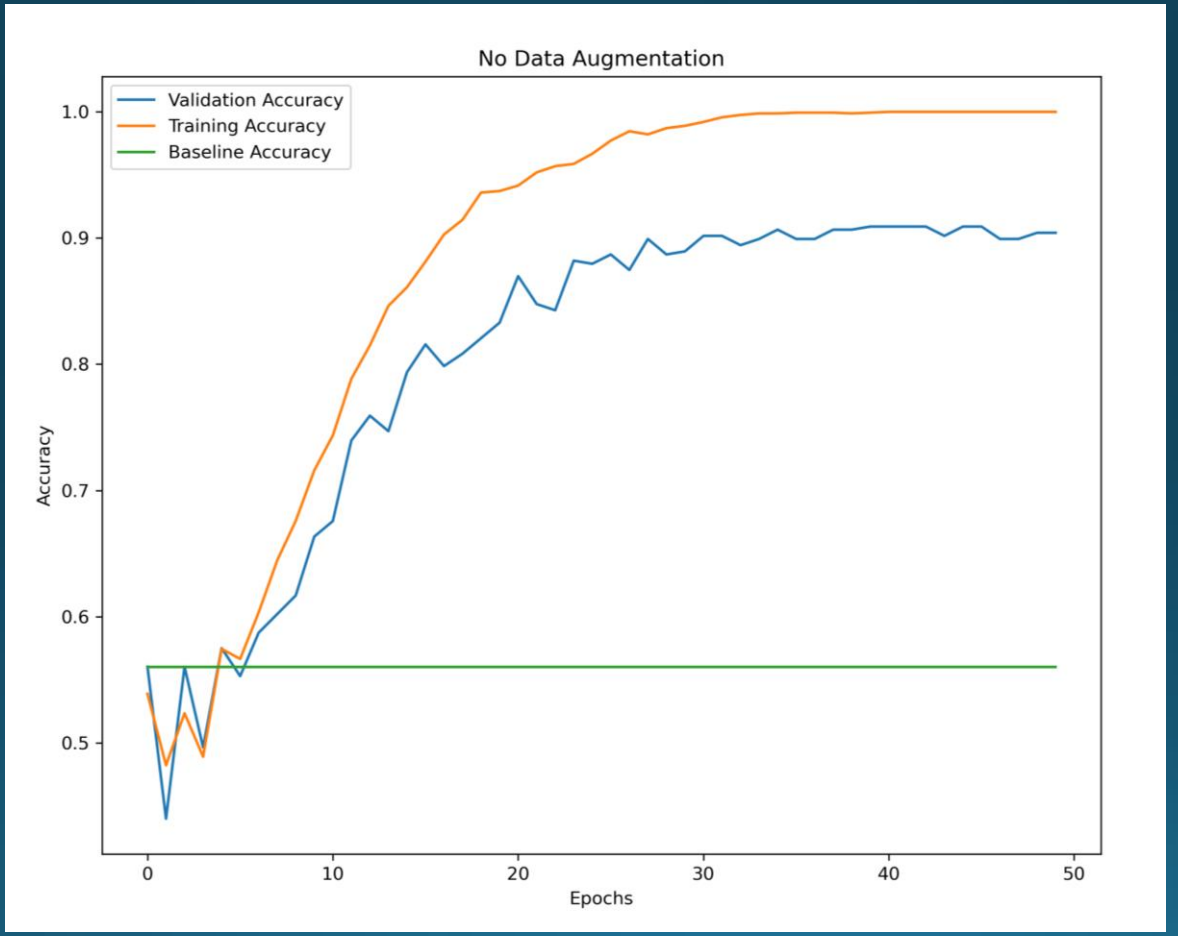


Eight images created from rotations and flips of a single image.

Train and Test CNN



Train and Test CNN: No Augmentation



Conclusion and Next Steps

- Data was organized and used to train and CNN classifier to recognize good CNT measurement locations. Validation accuracy just below 90% was achieved.
- Finding good CNT measurement locations within an image comes next.
- The code developed is extensible to Step 2, measuring diameters.
- More datasets may increase validation accuracy.
- Explore generating segmentation labels.
- Explore deployment options (Flask).

Thank you!

- Thanks for DSI's instructors for their help on this project.
- Thanks to my DSI cohort for all your help and suggestions.
- Thanks to Steven Buchsbaum at Lawrence Livermore National Laboratory for the CNT data.