Automating Myelin Defect Detection

Progress update: May 8th, 2024

Arjun Chandra & Anna Novoseltseva

Creating the Dataset

- 9 fully annotated images (3000x3000x25)
  - Grayscale → RGB
  - ~7000 total annotations
- Large dataset
  - Train: ~4000 images
  - Validation: ~600 images
- Three classes
  - Defect (82%)
  - Swelling (12%)
  - Vesicle (6%)
- Annotation bugs

Annotation Bugs

- Duplicate bounding boxes
- Same bounding box coordinates jumping multiple planes

206	22	206	364	2344	21	12
207	23	207	364	2344	21	12
208	4	208	364	2344	21	12
209	23	209	364	2344	21	12
210	23	210	364	2344	21	12
211	5	211	364	2344	21	12
212	19	212	364	2344	21	12

Annotation Bug

Training Pipeline

- Data cleaning
  - Remove annotations from the initial and final planes (1-7 & 22-25)
  - Remove duplicate bounding boxes
  - Remove bounding boxes which jump multiple planes*
- Crop image around boxing boxes
  - 300x300 .png images
  - Alternative approach to sliding window
- Convert annotation format
  - .mat → .txt YOLO format
- Create Training and Validation Split
  - Reserve images for validation set instead of random splits to avoid overlap

Alternative Cropping Algorithm

Algorithm 1: CropBboxes(G, A, W)

/* Input: G is an annotated 3-D image where $G[z]$ indexes the image in plane z
A is a dictionary where $A[z]$ contains the bounding box annotations in plane z
W is the cropping window size */

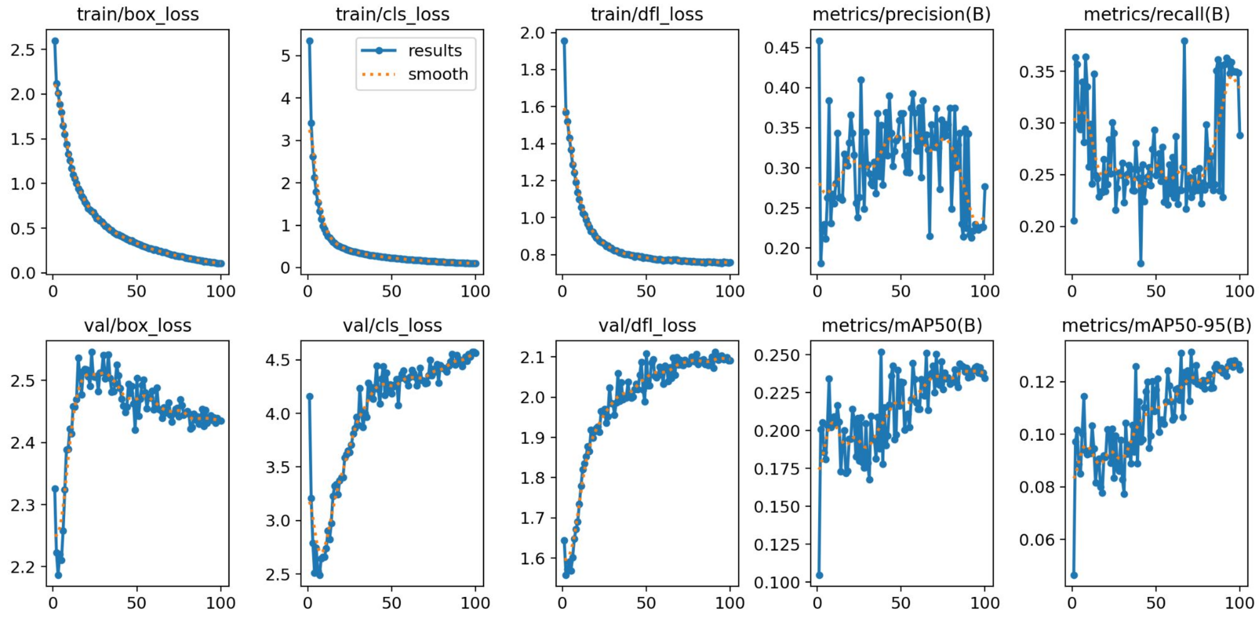
$r = \{\}$; // cropping bounds
$C = None$; // cropped image
$O = \{\}$; // bboxes in current window
for z *in* $1 \ldots z\_plane$ **do**
    for *bbox in* $A[z]$ **do**
        $r \leftarrow$ random cropping bounds around *bbox* with size W;
        $C \leftarrow$ crop $A[z]$ at r;
        $O \leftarrow$ all bounding boxes in C (cropped as needed);
        Write and save C and O;
return $None$

Evaluating the Model: YOLOv8

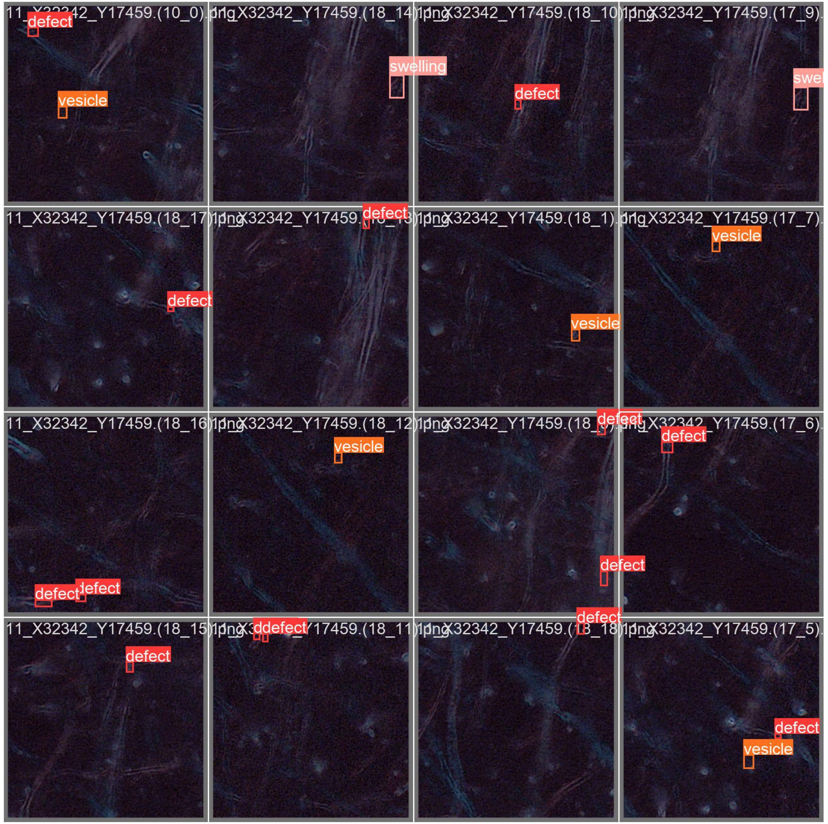
- Loss Functions
  - box (7.5): Bounding box regression
  - cls (0.5): Classification loss
  - dfl (1.5): Distribution focal loss (additional bbox regression)
- Metrics
  - **Precision: TP/(TP+FP)**
    - How many of the model’s detections are correct?
  - Recall: TP/(TP+FN)
    - How many detections did the model miss?
  - mAP50: Average area under PR curve across all classes
    - Standard object detection performance metric
    - Aggregate measure of precision, recall, confidence, IoU

Initial Results (3/30)

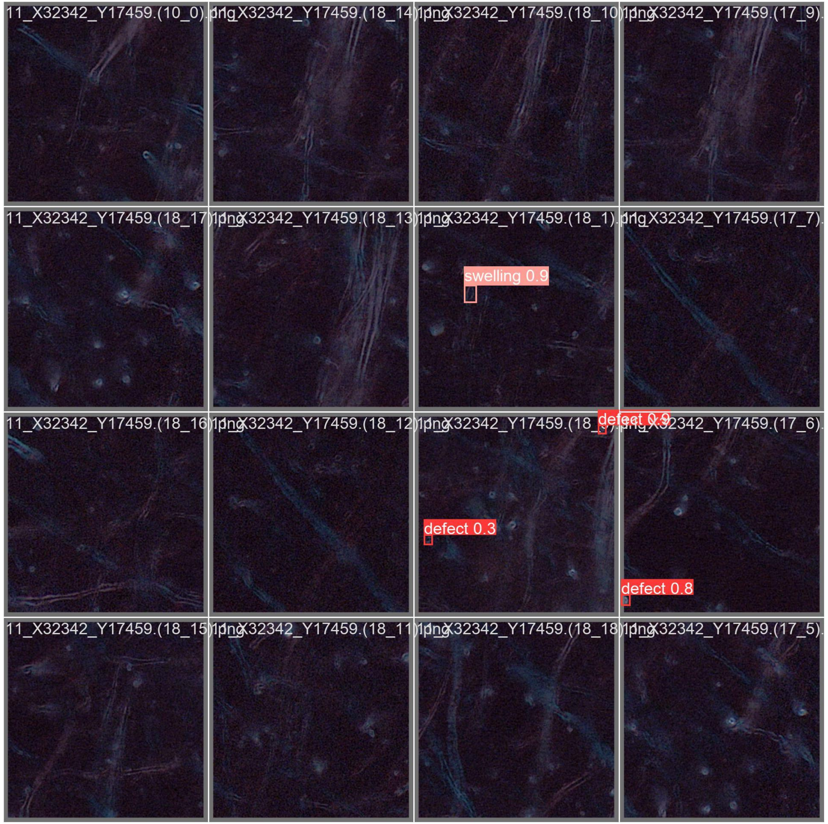
- Classification loss is much higher than other loss metrics
- Overfitting occurs very early



Initial Results (3/30)



Validation Batch Labels

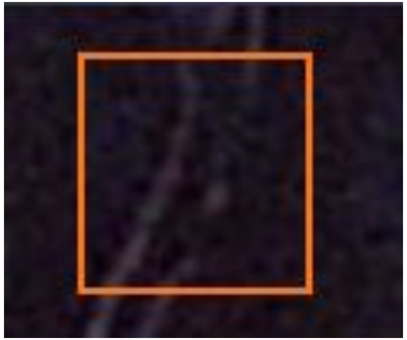


Validation Batch Predicted

Label

Predicted

Distinguishing Classes

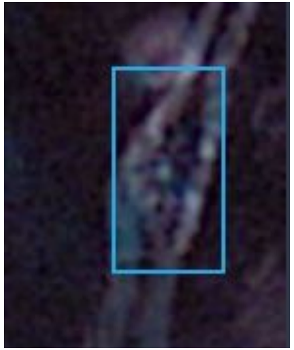


Defect

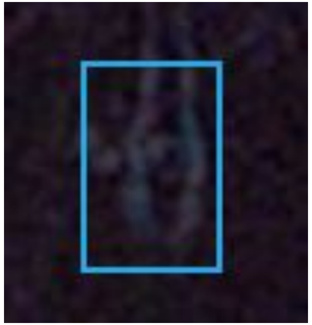
Defect

Defect

Swelling



Swelling



Swelling

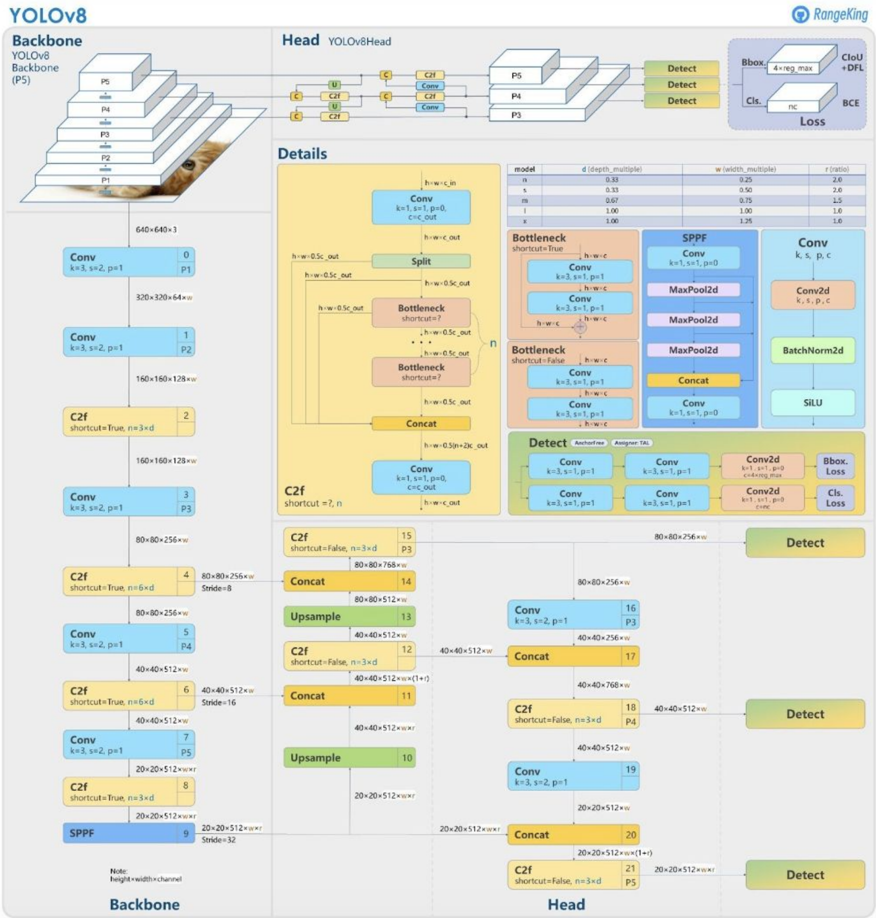
Improving Results

- Combatting overfitting
  - Increasing weight decay (L2 Regularization)
  - Decreasing learning rate
  - No improvement - overfitting is likely related to the dataset and not the model training
- Data Augmentation
  - Custom augmentations added in training pipeline as opposed to default YOLO augmentation
- Increasing image size
  - Generally useful for small object detection
- Transfer Learning
  - Starting from pretrained model weights
  - Freezing layers in the backbone

YOLOv8 Architecture

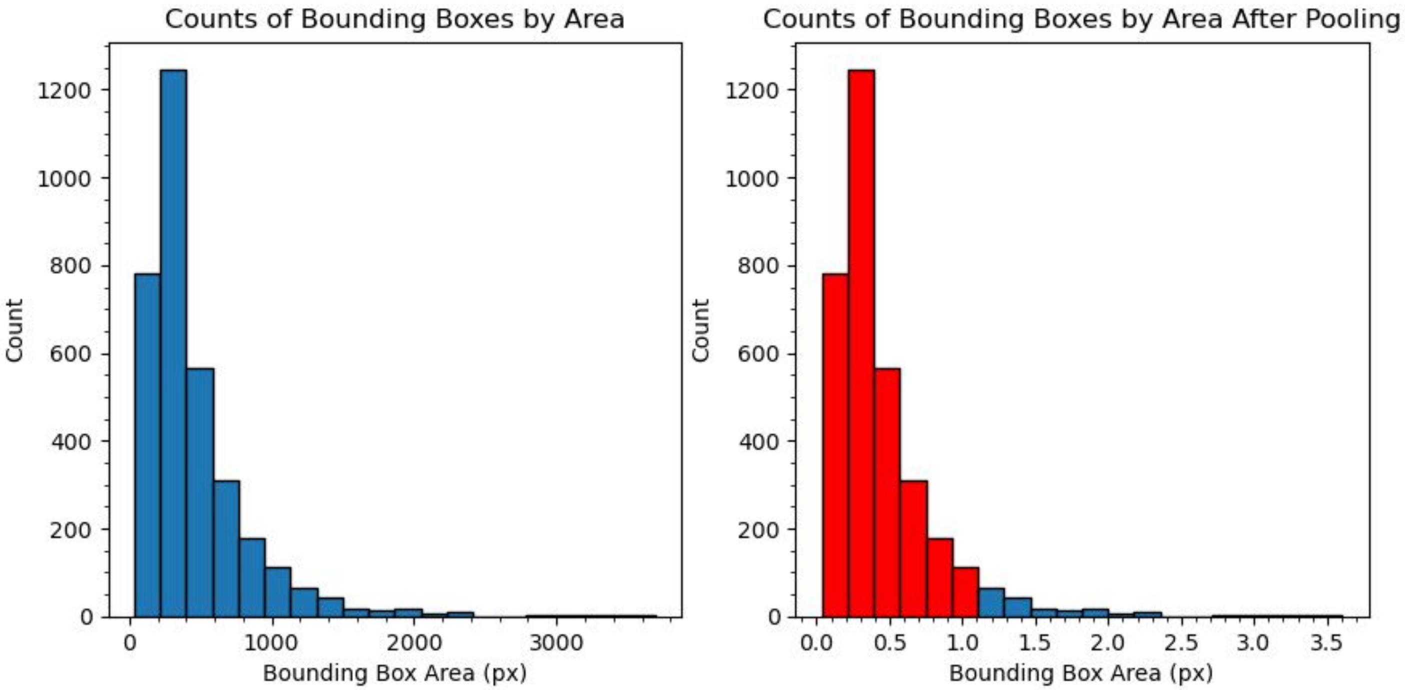
- Backbone is responsible for extracting features from the image
  - Freezing the entire backbone corrupts learning
  - Freezing first layer and first three layers yielded no improvement
  - Starting from random initialization and pretrained weights yielded similar results

- Convolutional layers in the backbone use a stride of 2 to downsample
  - Residual connections along the way accommodate smaller objects



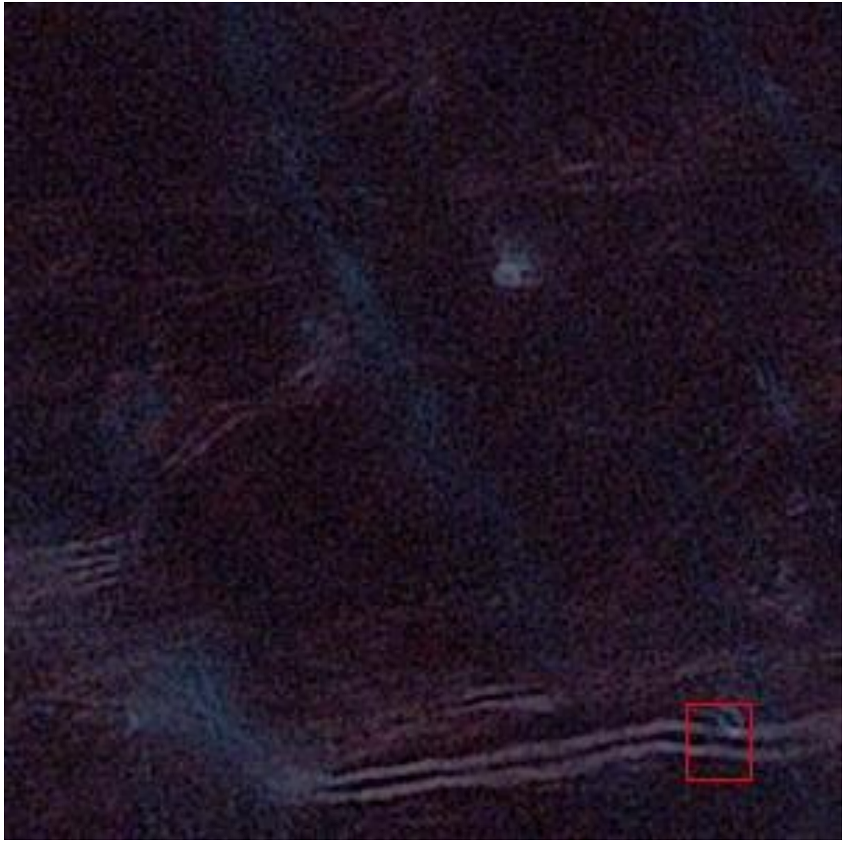
Small Object Detection

- Training with a larger image size (1280x1280) improved results slightly (~0.25 mAP)

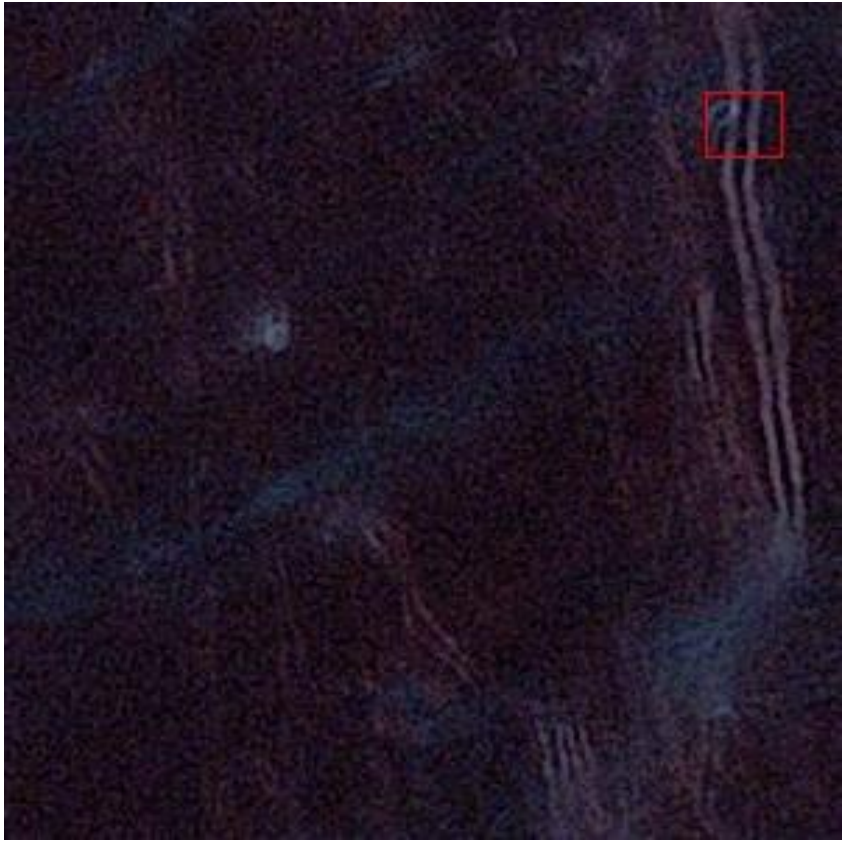


Data Augmentation

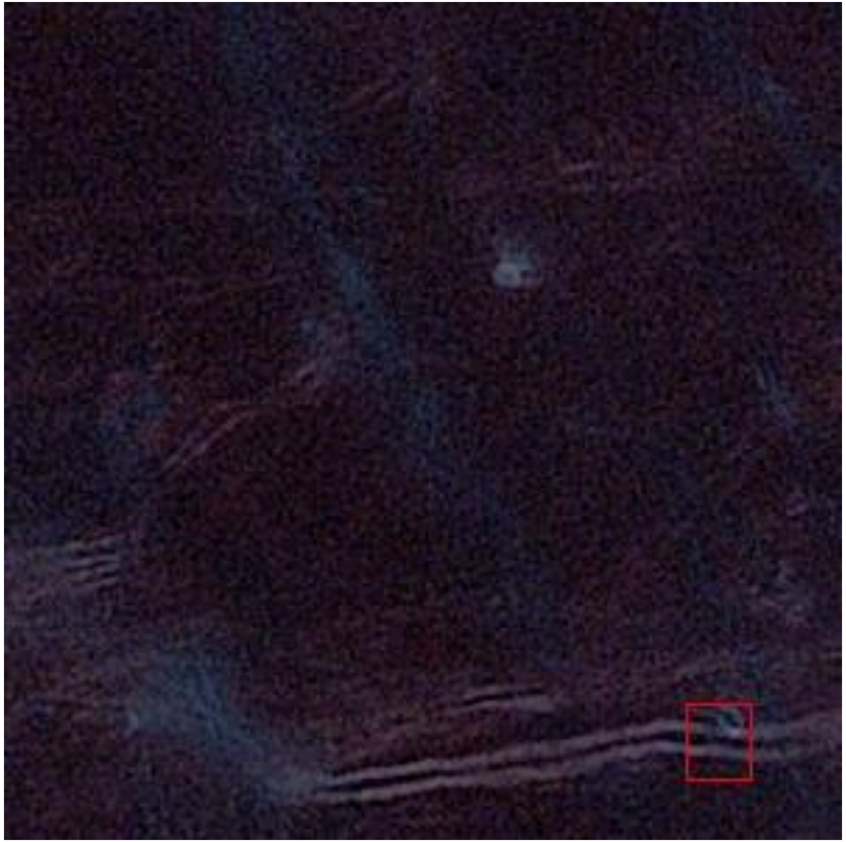
- Default YOLO augmentations are not well suited for medical imaging
  - Simulate different weather conditions, time of day, lighting etc.
- Three custom augmentations
  - Swap red and blue: (R,G,B) → (R-1.4B,G,B)
  - Random cropping
  - 4 orientations
- With 10x random croppings, dataset is scaled 80x



Original Orientation

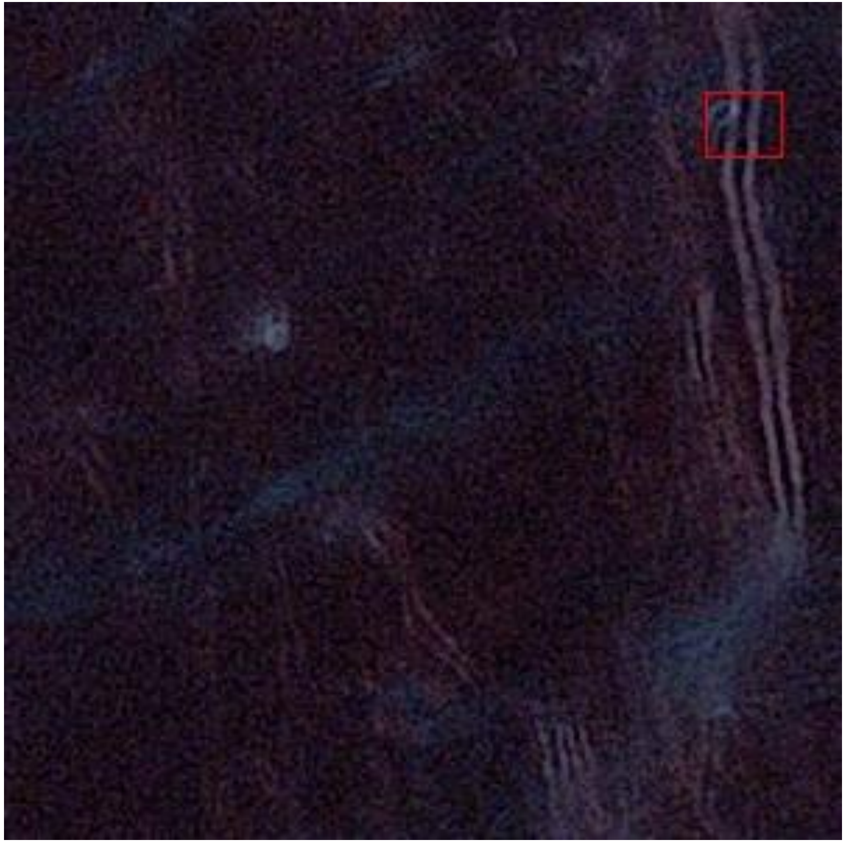


Rotated Orientation (90° CCW)

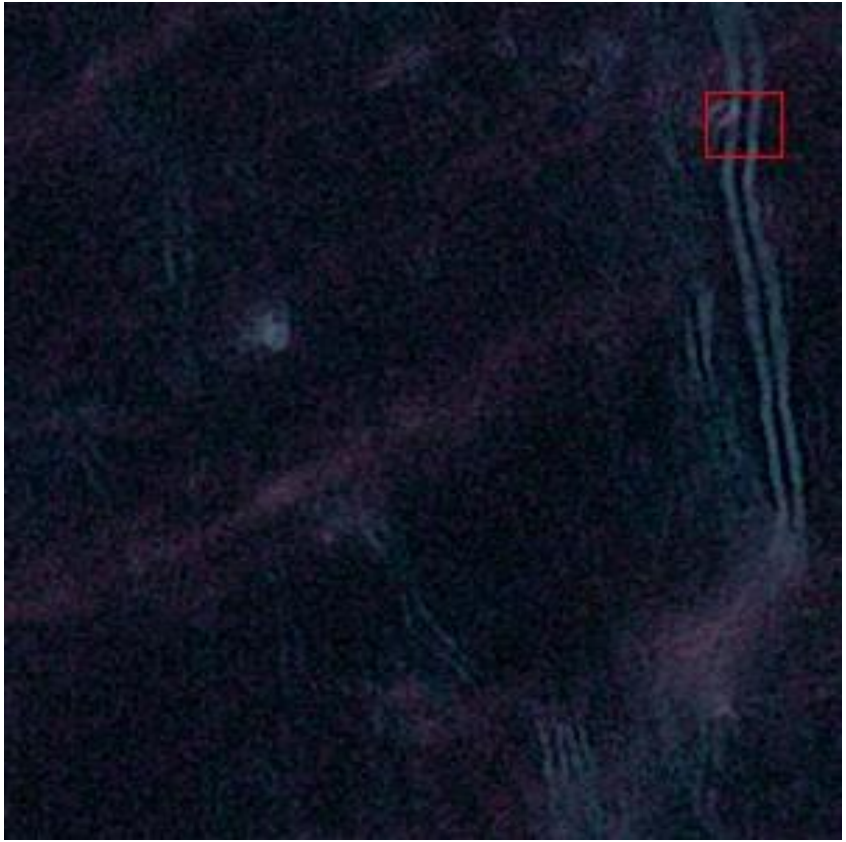


1st Cropping

2nd Cropping



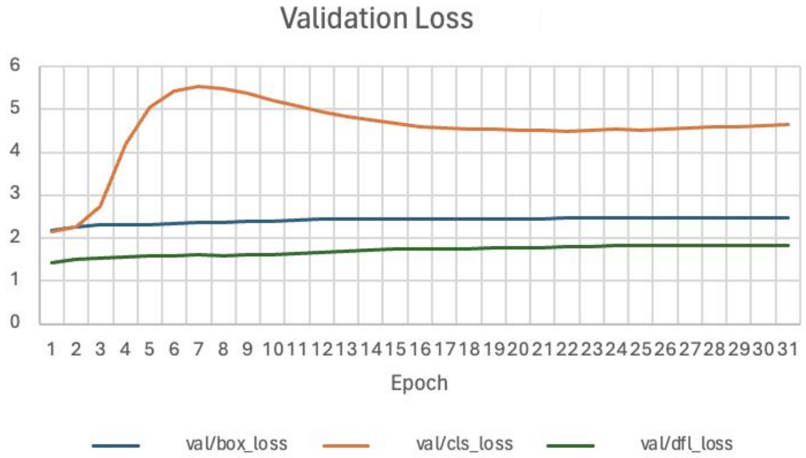
(R,G,B)



(R-1.4B,G,B)

Data Augmentation Results

- Led to even quicker overfitting and similar performance
  - ~10,000 weight updates per epoch as opposed to ~200 without augmentation
  - Augmented images were added directly to dataset rather than being applied at training time



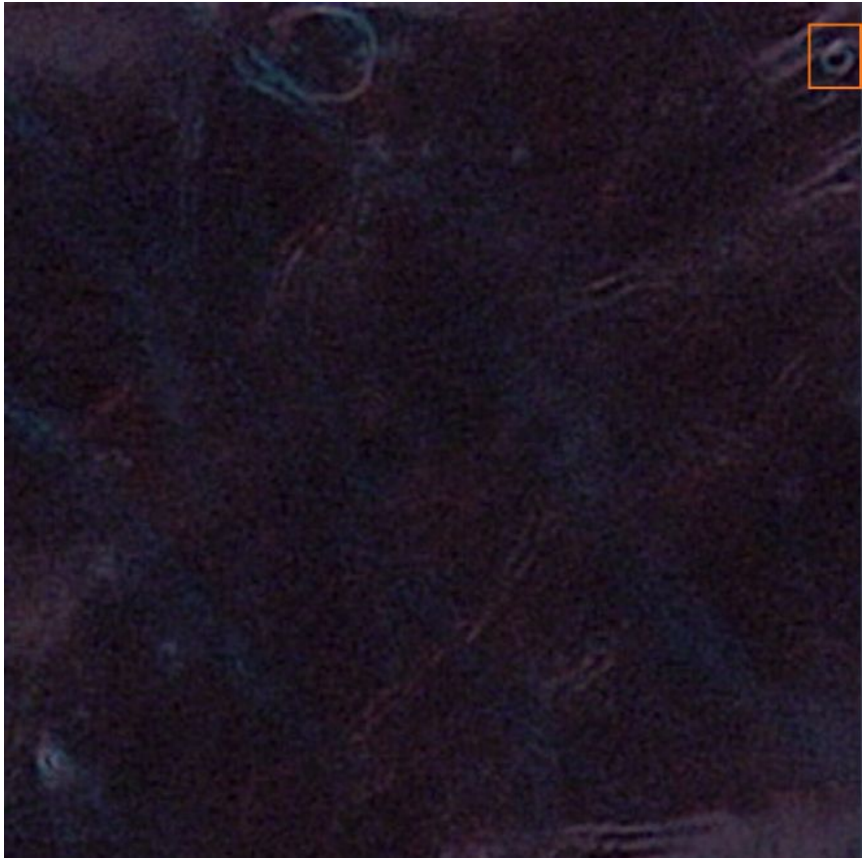
Cropping Algorithm Bug

- Moved dataset to Roboflow to examine it closely
- Discovered bug in the cropping algorithm
  - Bounding boxes after cropping for the first time leading to contradictory annotations

Algorithm 1: CropBboxes(G, A, W)

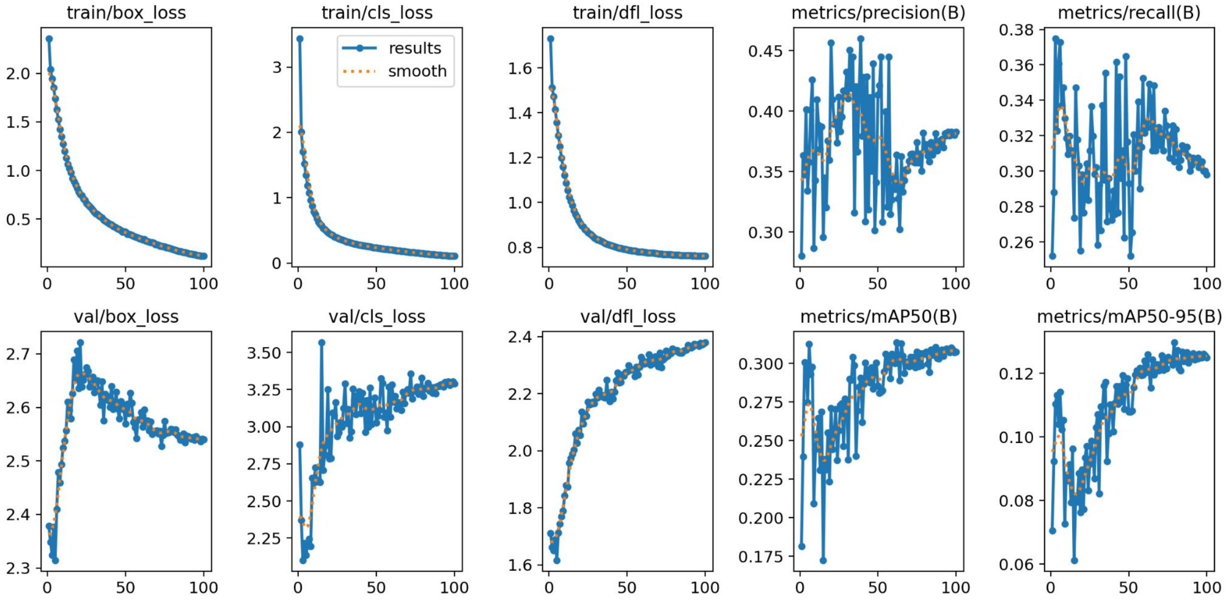
```
/* Input:G is an annotated 3-D image where G[z] indexes the image in plane z
   A is a dictionary where A[z] contains the bounding box annotations in plane z
   W is the cropping window size
                                                                              */
r = {}; // cropping bounds
C = None; // cropped image
O = {}; // bboxes in current window
for z in 1 ... z_plane do
    for bbox in A[z] do
        r ← random cropping bounds around bbox with size W;
        C ← crop A[z] at r;
        O ← all bounding boxes in C (cropped as needed);
        Write and save C and O;
        Delete all bounding boxes in O;
return None
```

Contradictory Annotations

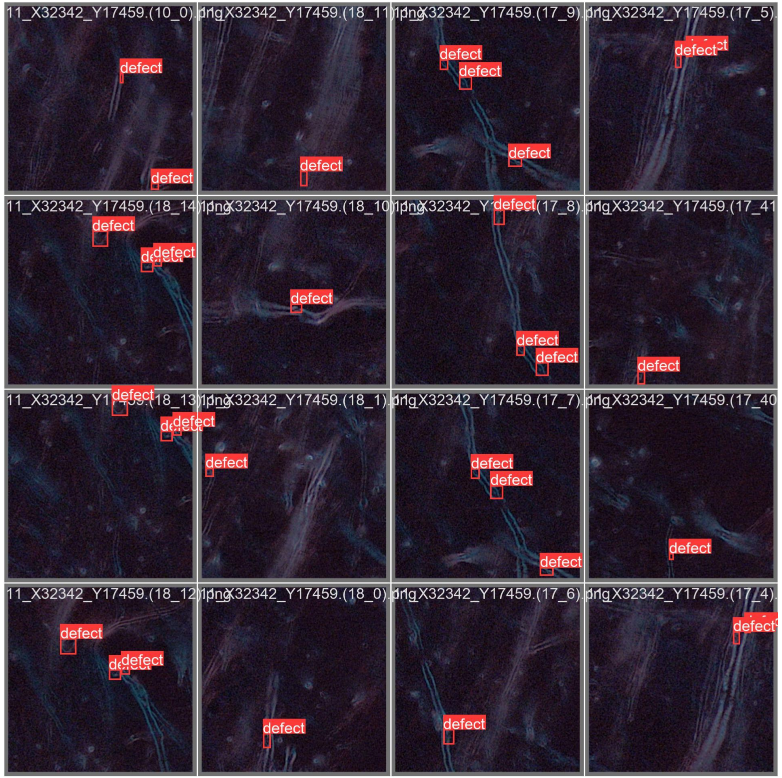


Results (5/3)

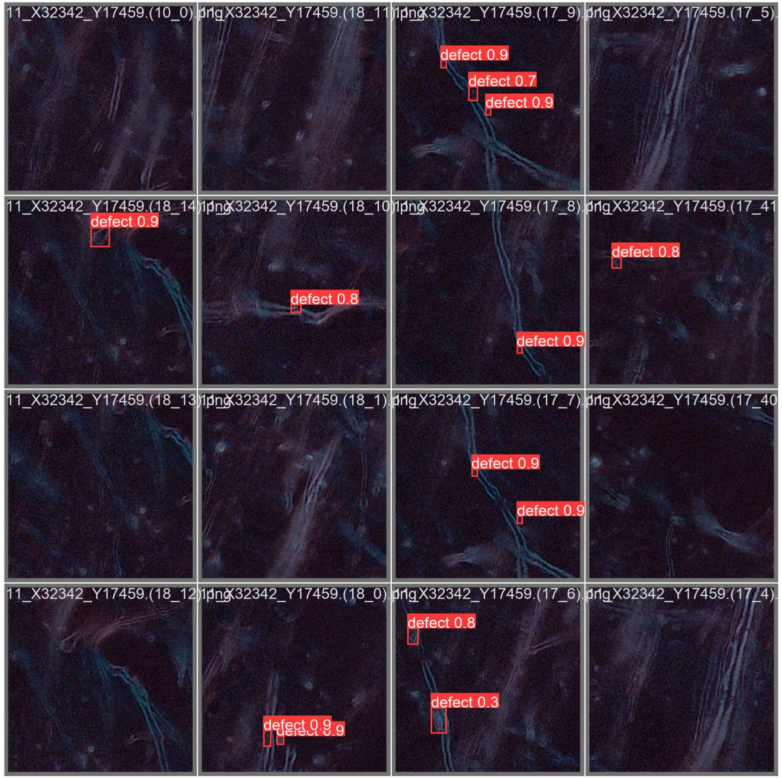
- All annotations combined into a single “Defect” class
- Cropping algorithm bug fix



Results (5/3)

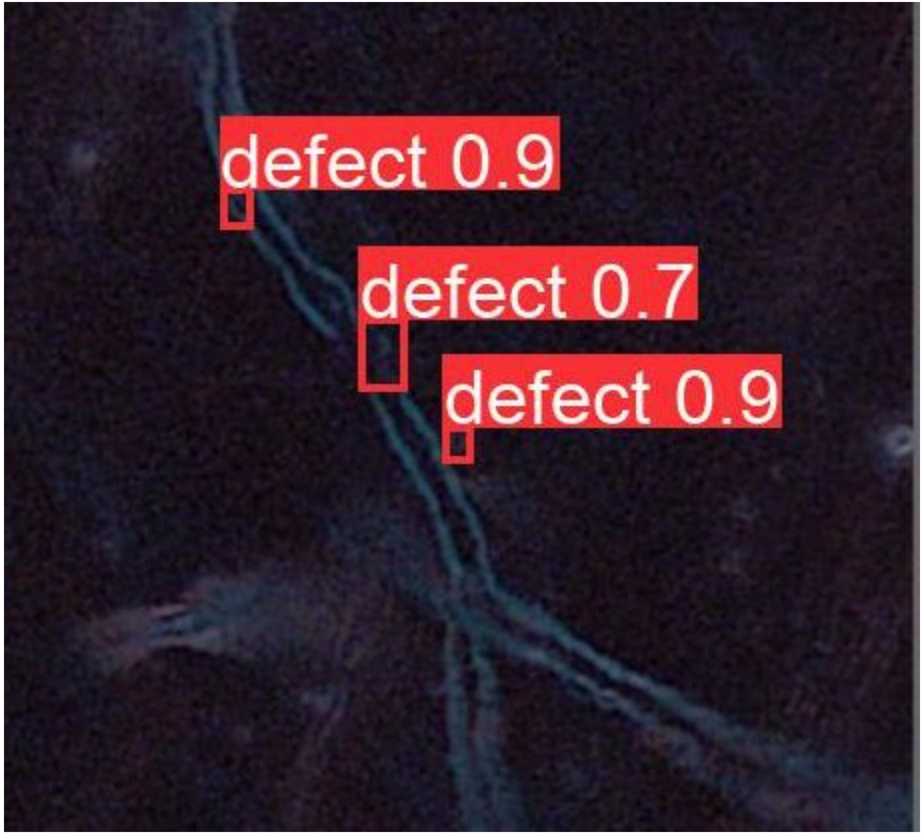


Validation Batch Labels



Validation Batch Predicted

Label



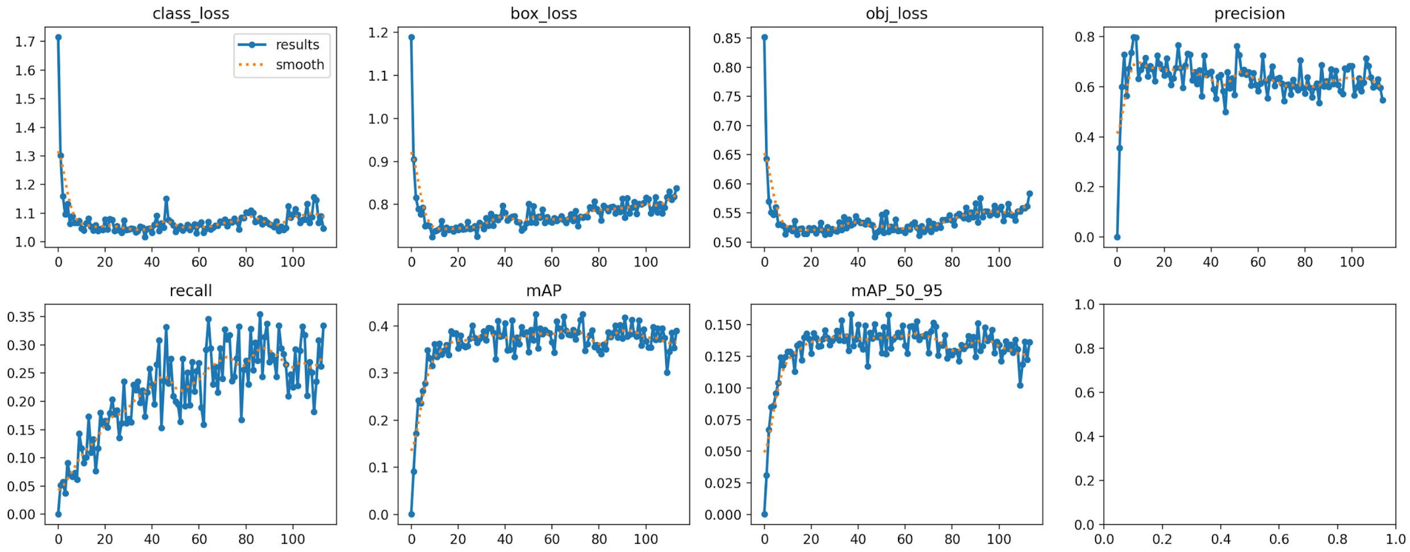
Predicted

Label

Predicted

Results (5/7)

- Training with Roboflow achieves better results
  - Roboflow 3.0 Object Detection: 0.38 mAP50
  - YOLO-NAS Object Detection: 0.425 mAP50



YOLO-NAS

Label

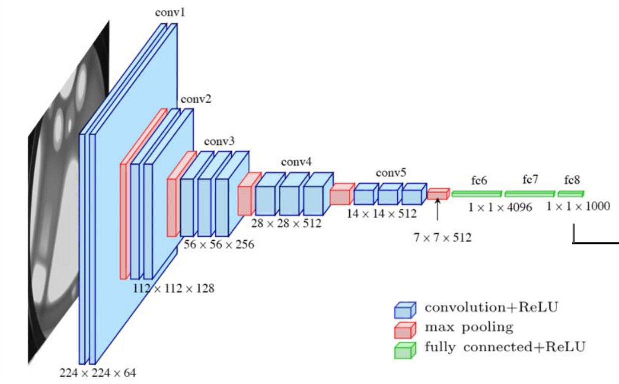
Predicted

Label

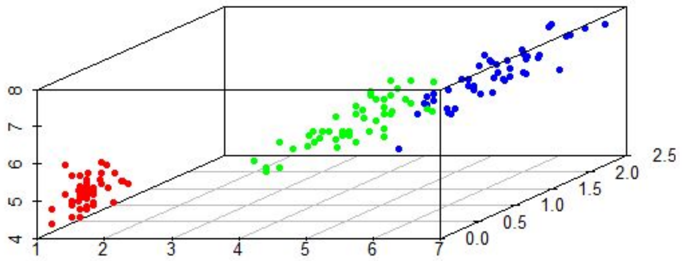
Predicted

Next Steps

- Classifying defects
  - Train CNN/Autoencoder and extract image embeddings
    - Or another suitable algorithm?
  - Clustering algorithm: K-Means



VGG-19 Architecture



KNN Clusters

Next Steps

- Backpropagation
  - Test model on an unannotated image and evaluate performance
    - Split image (3000x3000x25) into subimages → get model predictions on subimages → stitch predictions back together
  - Dataset from Anna’s paper
    - Compare results with previous annotations - expecting linear trend between human annotations and model annotations