Automatic Damage Detection, Segmentation and Characterization using Deep Learning

Sanjeev Kumar, M.Sc.

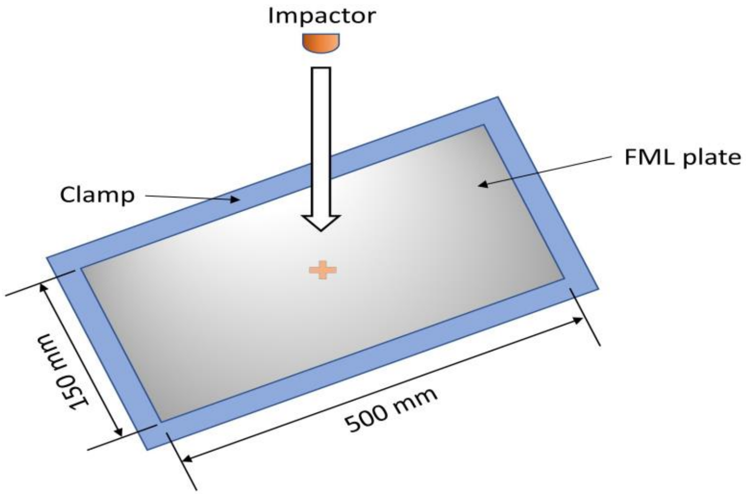
Contact: sanjeev@uni-bremen.de

Introduction

- Developing an automated and accurate process for **detecting, segmenting, and characterizing** impact damages in Fiber Metal Laminate (FML) materials.
- Application: aerospace and automotive industries due to their superior mechanical properties like high fatigue and corrosion resistance, light weight etc.
- GLARE (Glass Laminate Aluminum Reinforced Epoxy) 5-5/4 with 54% Metal Volume Fraction, specimen thickness 4mm and size 150×500 mm.

Introduction

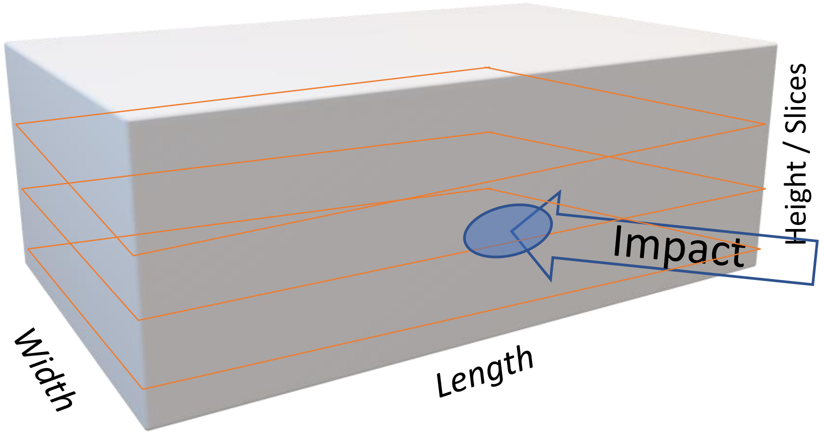
- The impacts were made at varying energy levels of **5J, 7.5J, 10J, and 12.5**J by shooting a projectile from the impact gun.
- The plates were reduced to a size of 50 mm around the impact location via water jet cutting.
- The smaller specimen were then investigated with the X-ray computed tomography (CT) to capture the cross-sectional slices of the damaged plate.



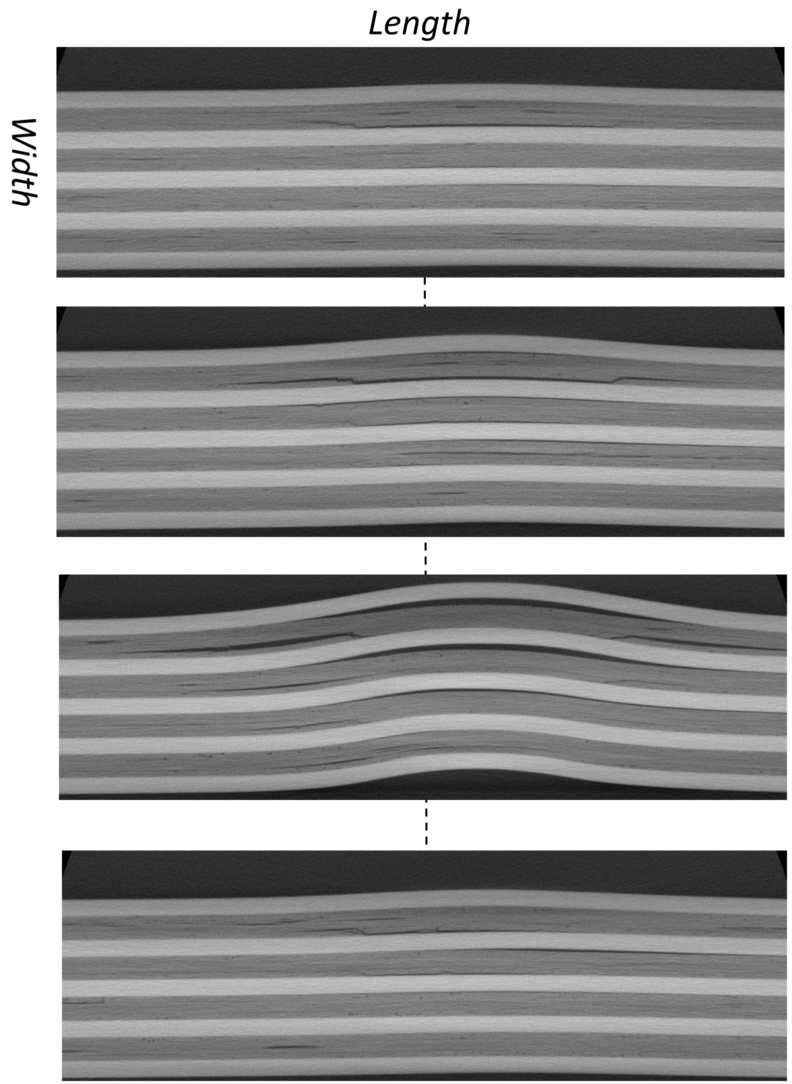
Introduction

- Example CT slices:

FML plate



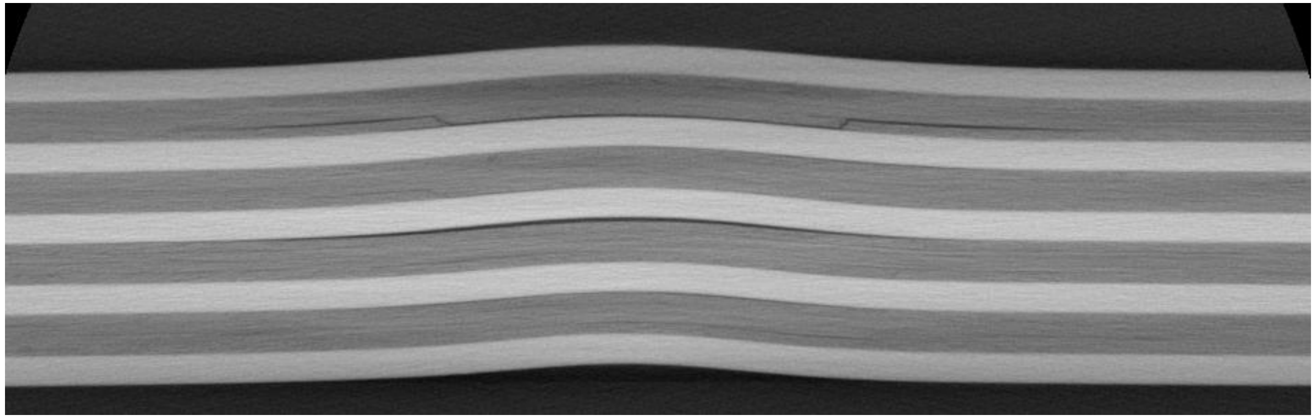
Slices (~1000)



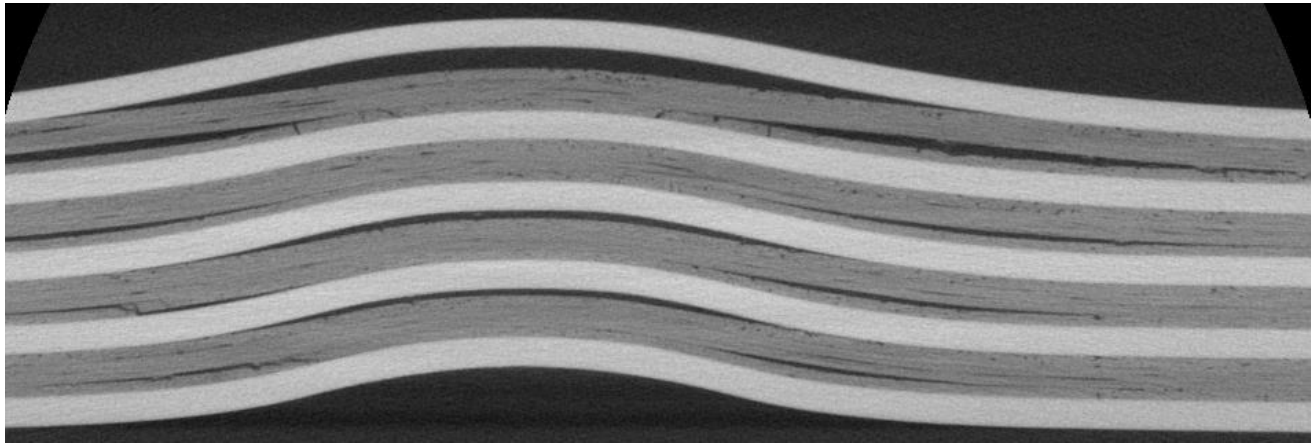
Introduction

- Example CT slice (5 J vs. 12.5 J)

5 Joules



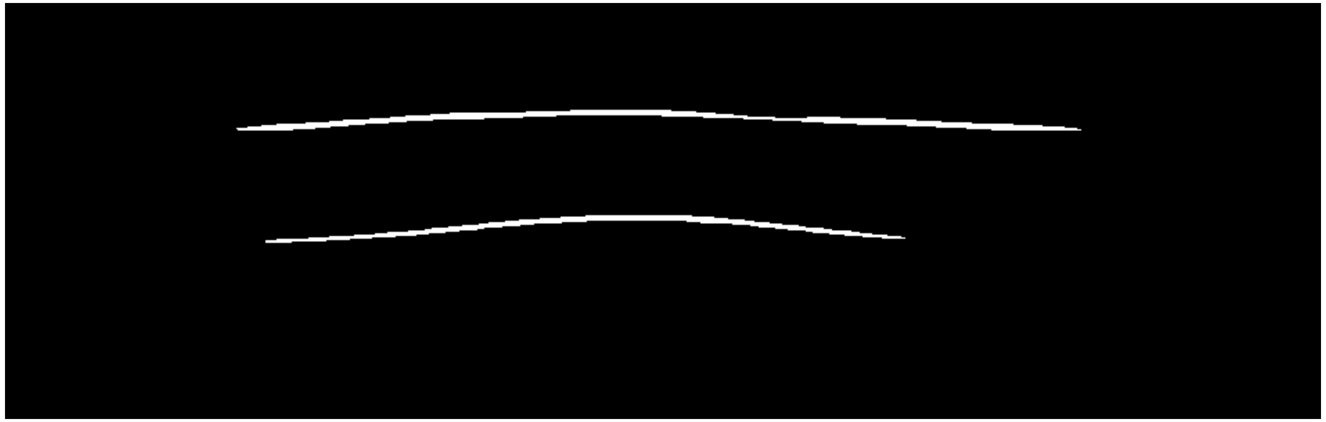
12.5 Joules



Introduction

- Example Masks (5 J vs. 12.5 J)

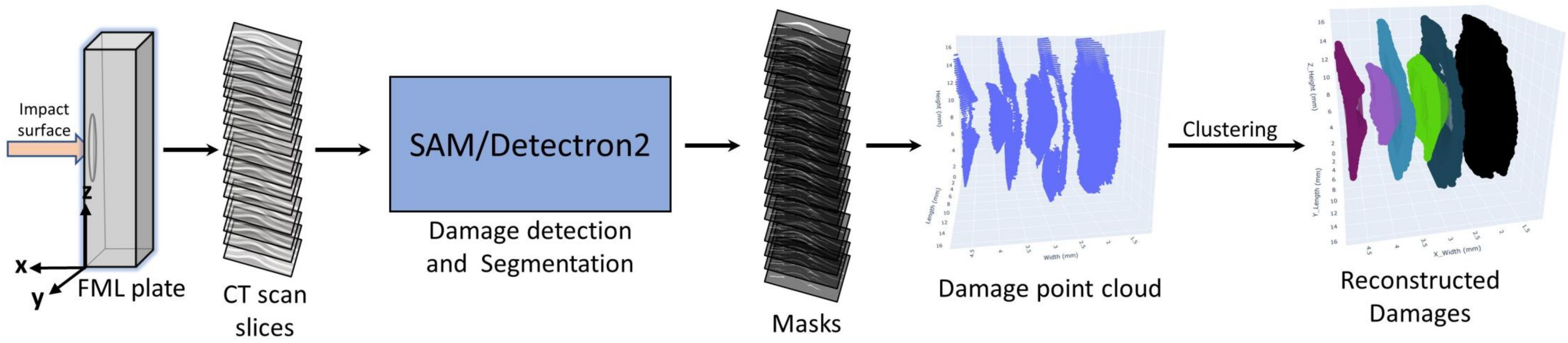
5 Joules



12.5 Joules

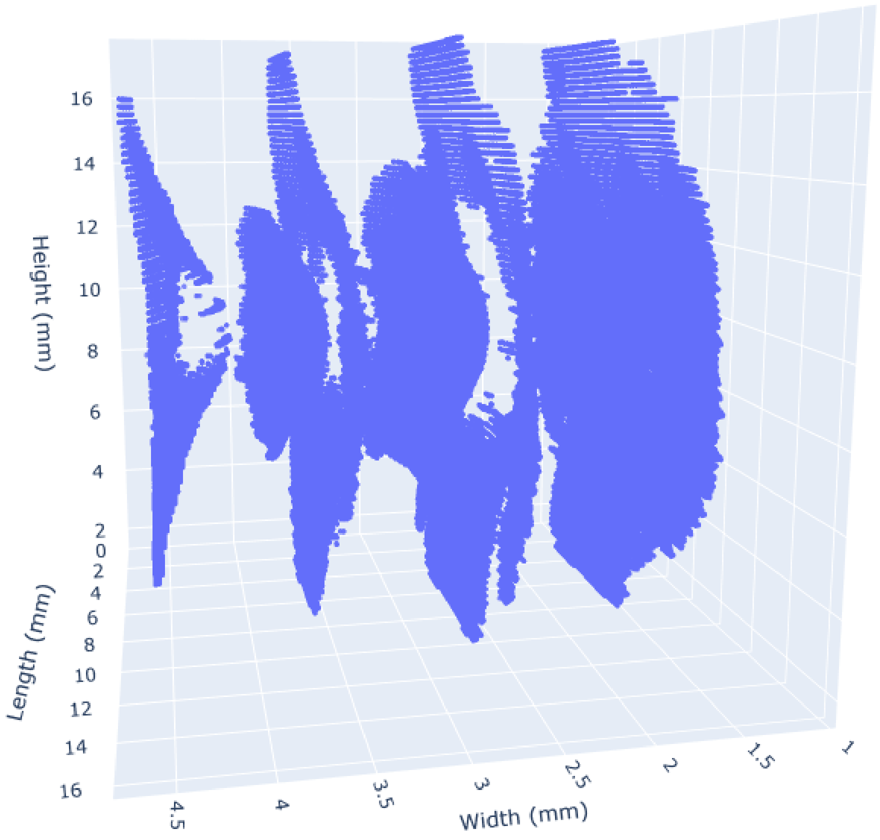
Introduction

- Complete process pipeline.

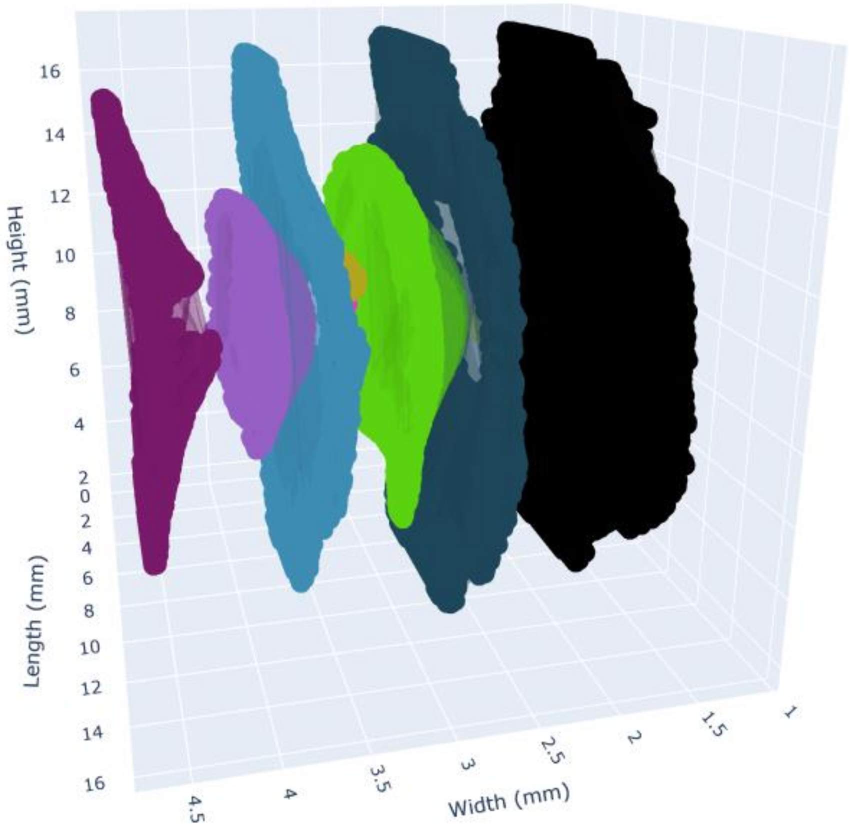


Damage point cloud

- Example (7.5 Joules impact)



Clustering and hull fitting

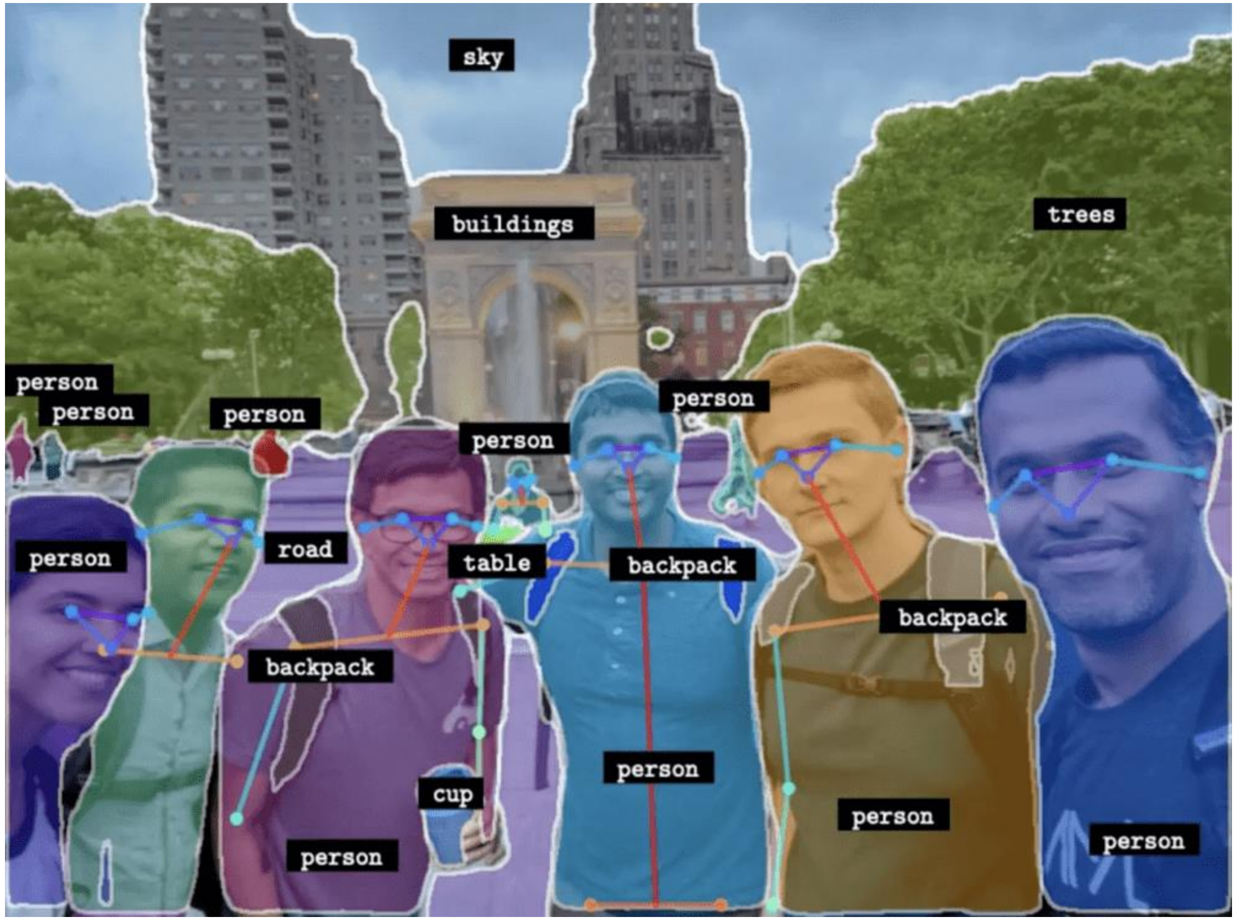


Detectron2

- A powerful, modular computer vision library developed by Facebook AI Research (FAIR) for object detection and segmentation tasks.
- Built on PyTorch framework for deep learning research and applications.
- Key development Features:
  - **Modular Design**: Easy to add new models and tasks
  - **Fast Training**: Supports both single and multi-GPU training
  - **Model Zoo**: Contains pre-trained models for quick deployment
  - **Extensible**: Supports custom datasets and model architectures
- https://github.com/facebookresearch/detectron2

Detectron2

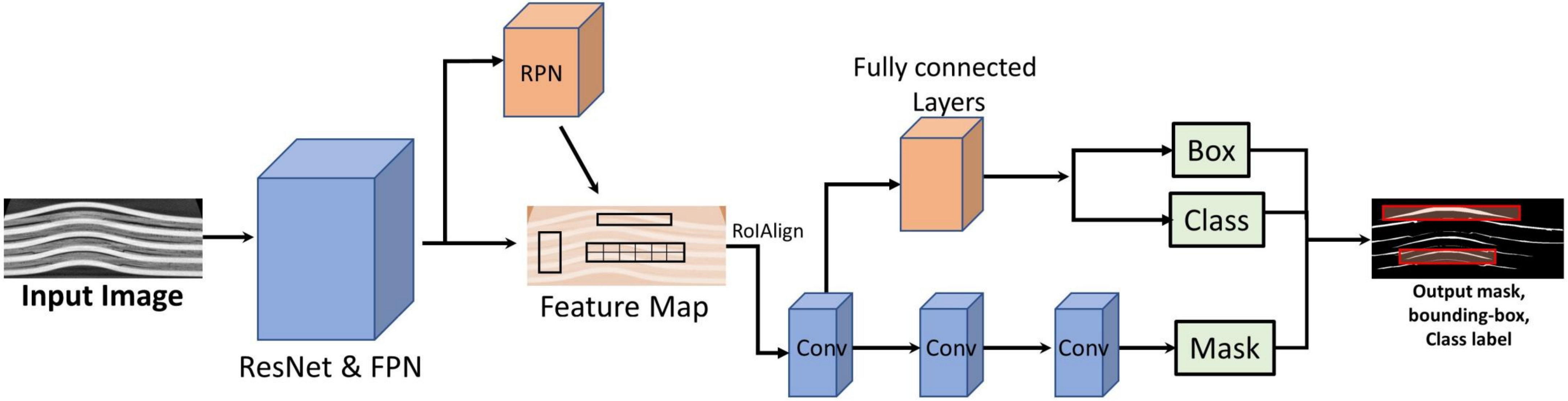
- Supported Vision Tasks:
  - Object Detection
  - Instance Segmentation
  - Semantic Segmentation
  - Key-point Detection
  - Panoptic Segmentation



Mask R-CNN

- Mask Region-Based Convolutional Neural Network.
- An extension of Faster R-CNN that **combines object detection and semantic segmentation**.
- Produces both bounding boxes around objects and pixel-level segmentation masks.
- Key innovation: Adds a parallel mask prediction branch to the existing classification and bounding box regression branches.
- Uses RoIAlign instead of RoIPool for more precise spatial feature extraction.
- Predicts binary masks independently for each class, decoupling classification and segmentation tasks

Mask R-CNN



RPN
Fully connected
Layers
Box
Class
RoIAlign
Input Image
ResNet & FPN
Feature Map
Conv
Conv
Conv
Mask
Output mask,
bounding-box,
Class label

Architecture Components

1. **Backbone Network**
    1. ResNet-50 with Feature Pyramid Network (FPN)
    2. Processes input image to generate convolutional feature maps
    3. FPN creates multi-scale feature pyramid for handling various object sizes
2. **Region Proposal Network (RPN)**
    1. Generates candidate object proposals
    2. Predicts objectness scores and bounding box coordinates
    3. Fully convolutional network design
3. **RoIAlign Layer**
    1. Improves upon RoIPool with precise spatial sampling
    2. Uses bilinear interpolation to avoid quantization errors
    3. Preserves spatial coherence for accurate mask generation
4. **Dual-Branch Head**
    1. Bounding Box Head: Predicts object class and refines box coordinates
    2. Mask Head: Generates binary segmentation mask for each RoI

Residual Network (ResNet)

Challenge:

- Deep networks suffer from vanishing/exploding gradients
- Gradients become extremely small as they propagate backwards
- Training becomes ineffective in very deep networks
- Accuracy degrades despite increased network depth

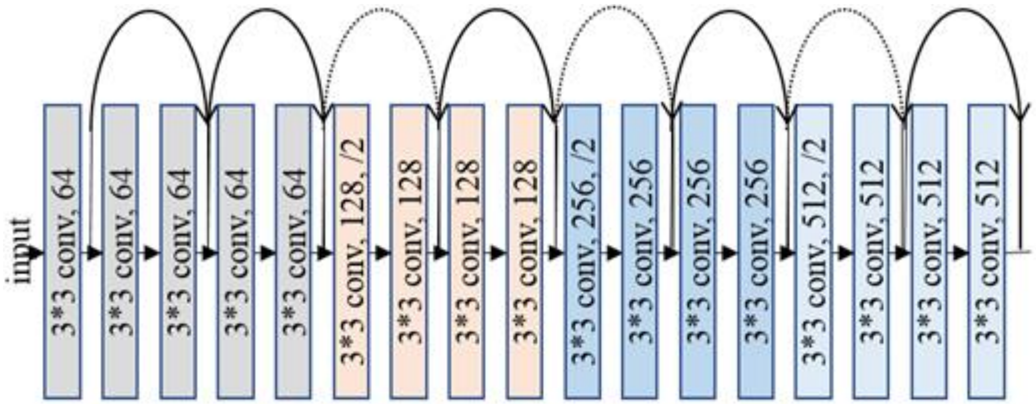
Traditional Solutions

- Normalized initialization
- Intermediate normalization layers
- These help but don't fully solve the problem
- Limited network depth still persists

Residual Network (ResNet)

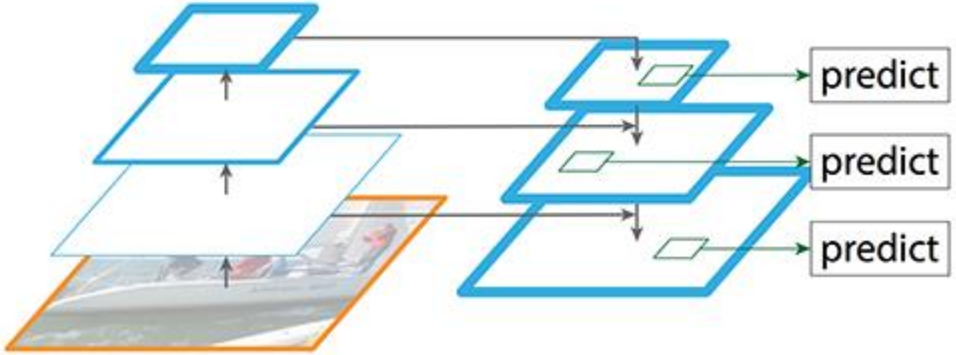
ResNet Solution

- **Skip Connections** (Identity Mappings)
  - Create shortcuts between layers
  - Allow direct flow of gradients
  - Enable better information propagation
  - F(x) + x instead of just F(x)
- **Advantages**
  - Easier optimization
  - No extra parameters or computation
  - Can train very deep networks (50+ layers)
  - Better gradient flow throughout network



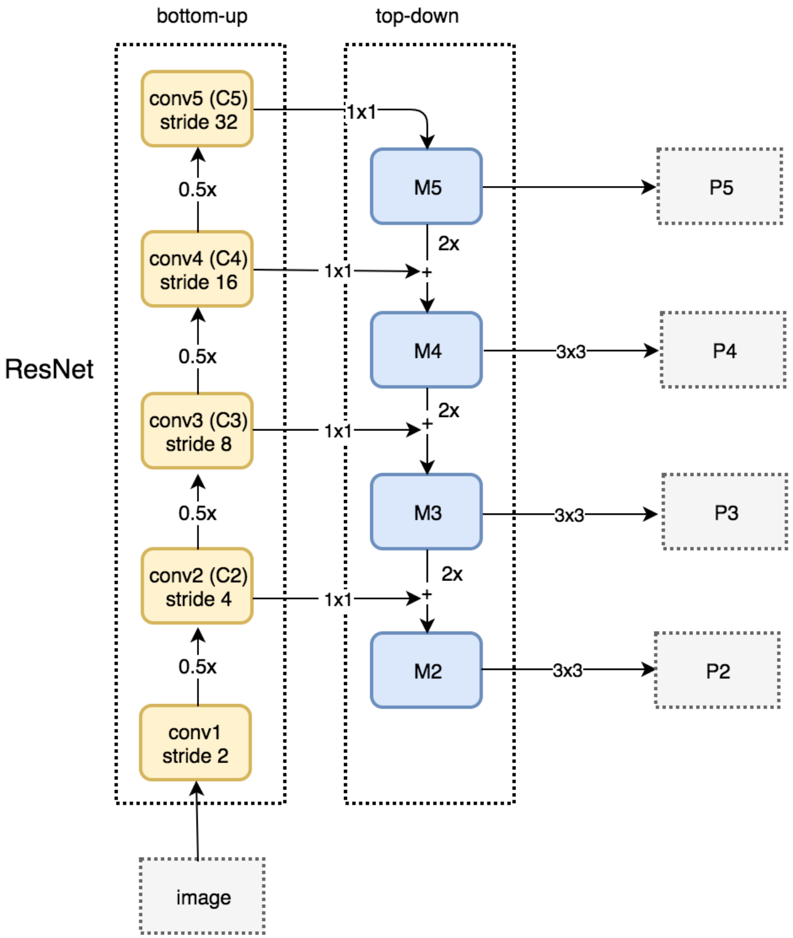
Feature Pyramid Network (FPN)

- Multi-scale feature extractor for object detection and segmentation.
- **Bottom-up Pathway**
  - Traditional convolutional network (e.g., ResNet)
  - Features hierarchically divided into levels
  - Each level has progressively:
    - Larger receptive field
    - Stronger semantic information
    - Lower spatial resolution
- **Top-down Pathway**
  - Upsampling of higher level features
  - 1x1 convolutions for dimension reduction
  - Lateral connections from bottom-up pathway
  - Element-wise addition of features



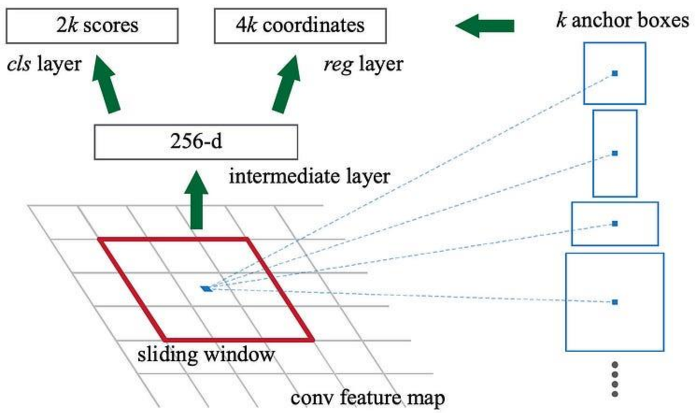
Feature Pyramid Network (FPN)

- 1×1 convolution filter to reduce C5 channel depth to 256-d to create M5.
- In top-down path, upsample by 2 using nearest neighbors upsampling.
- apply a 1×1 convolution to the corresponding feature maps in the bottom-up pathway. Add them element-wise. Apply a 3×3 convolution to all merged layers.
- Reduces the aliasing effect when merged with the upsampled layer.



Region Proposal Network (RPN)

- FPN extracts feature maps then feeds into a detector (RPN).
- A sliding window over the feature maps is applied.
- Predictions on the objectness (has an object or not) and the object boundary box.



Loss Function Components

Total Loss Function:

$$L = L_{\text{cls}} + L_{\text{box}} + L_{\text{mask}}$$

$$L_{\text{cls}} = -\log(p_t)$$

$$L_{\text{box}} = \text{smooth}_{L1}(v - \hat{v}) \qquad \text{smooth}_{L1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

$$L_{\text{mask}} = -\sum_{p,q} \left[m_{pq} \log(\hat{m}_{pq}) + (1 - m_{pq}) \log(1 - \hat{m}_{pq}) \right]$$

1. Classification Loss:

- Standard cross-entropy loss (log loss).
- Measure the error between the predicted and true class.

2. Bounding Box Loss:

- Refines bounding box predictions.
- measure the difference between the predicted and true bounding box coordinates.
- v is the true bounding box coordinates and v cap is the predicted coordinates.

3. Mask Loss

1. Pixel-wise binary cross-entropy
2. Applied only to ground-truth class masks
3. Computed per-pixel for accurate segmentation
4. m is the true binary mask and m cap is the predicted mask, p and q index the pixels in the mask.

Segment Anything Model (SAM)

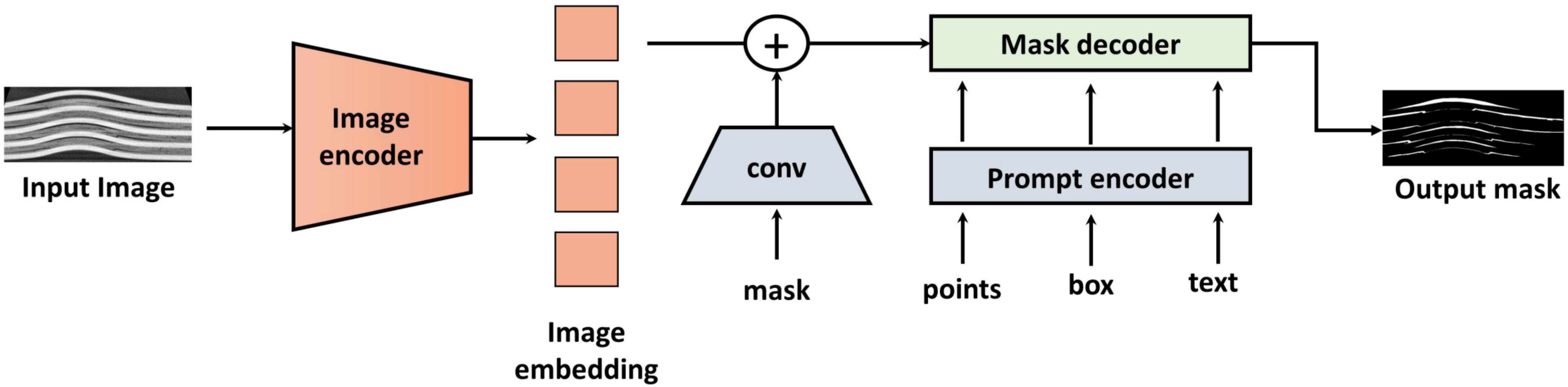
- A foundation model for image segmentation developed by Meta AI
- First universal image segmentation model that works with any segmentation task
- Trained on over 11 million images and 1.1 billion masks
- Open-source model designed for broad accessibility and application

Segment Anything Model (SAM)

- **Promptable Segmentation**
  - Accepts various input prompts: points, boxes, text, and masks
  - Generates high-quality segmentation masks in real-time
  - Flexible interaction modes for different use cases

- **Zero-Shot Performance**
  - Works effectively on unseen objects and scenarios
  - Requires minimal prompt engineering
  - Generalizes well across different domains

SAM Architecture



SAM Architecture

Image Encoder

- Vision Transformer (ViT) backbone
- Processes input images into dense visual features
- Available variants: ViT-H, ViT-L, and ViT-B
- Creates rich image embeddings with spatial and contextual information

Prompt Encoder

- Handles points, boxes, and text inputs
- Uses positional encodings
- Combines with learned embeddings per prompt type

Mask Decoder

- Transforms embeddings into segmentation masks
- Dynamic mask prediction head
- Processes multiple prompts in parallel
- Outputs high-resolution binary masks

Loss Function and Training

$$\mathcal{L} = \alpha\mathcal{L}_{\text{Dice}} + \beta\mathcal{L}_{\text{CE}}$$

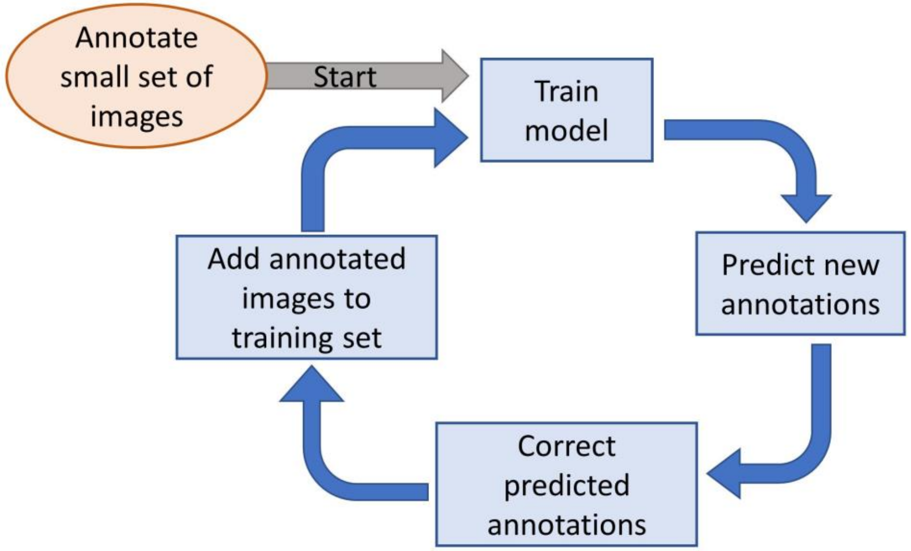
$$\mathcal{L}_{\text{Dice}} = 1 - \frac{2\sum_i p_i g_i}{\sum_i p_i^2 + \sum_i g_i^2}$$

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N}\sum_i [g_i \log(p_i) + (1-g_i)\log(1-p_i)]$$

1. **Dice Loss**
   1. Focuses on overlap between predicted and ground truth masks
   2. Effective for small structure prediction
   3. P_i are the predicted probability and g_i are the ground truth binary label for each pixel i.
2. **Cross-Entropy Loss**
   1. Handles class balance
   2. Pixel-wise binary classification
3. **Balanced Advantages**
   1. CE: Maintains overall class proportions
   2. Dice: Better prediction of small structures
   3. Combined approach handles various segmentation challenges

Training

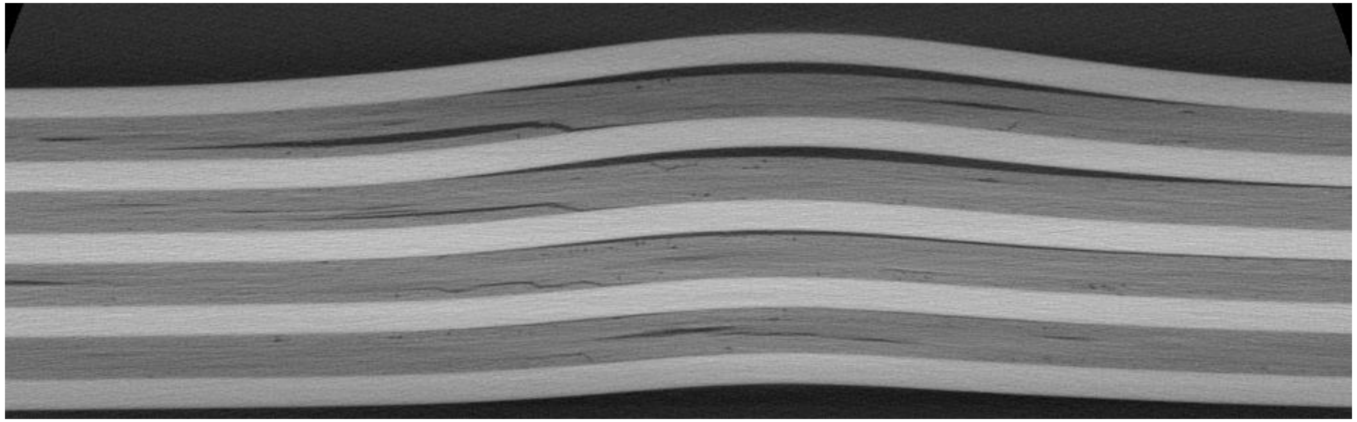
- Trained using 28 images (950x300 , 121 features)
- Evaluated using 8 images (950x300 pixels, 35 features)
- Image pre-processing:
  - Normalisation
  - Gaussian noise removal
  - CLAHE (Contrast Limited Adaptive Histogram Equalization)
- Image annotations:
  - Hand annotate 15-20 images (more the better)
  - Or take help from partially trained model.



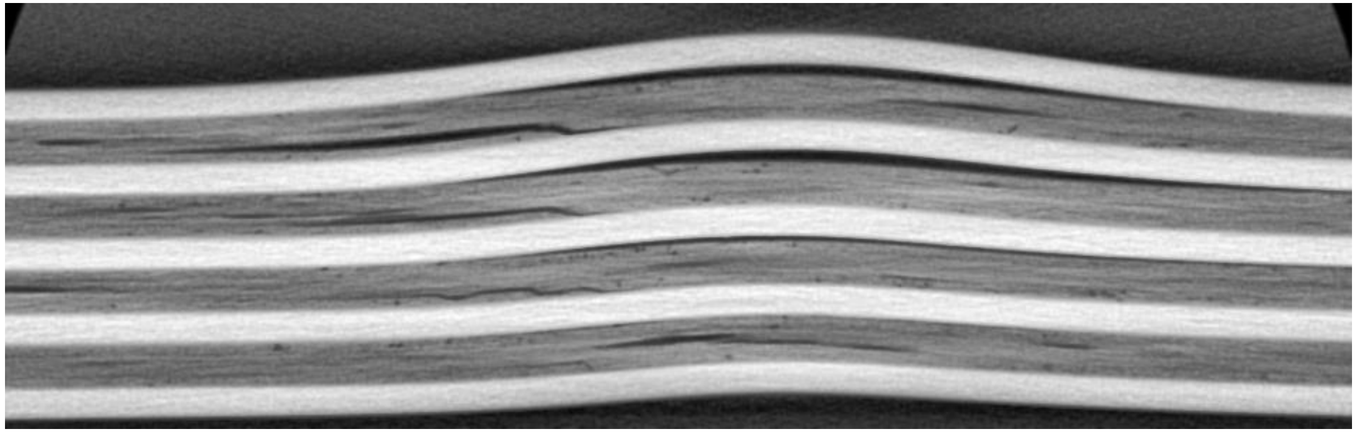
Training

- Example preprocessed training input:

Raw input image



Preprocessed input image

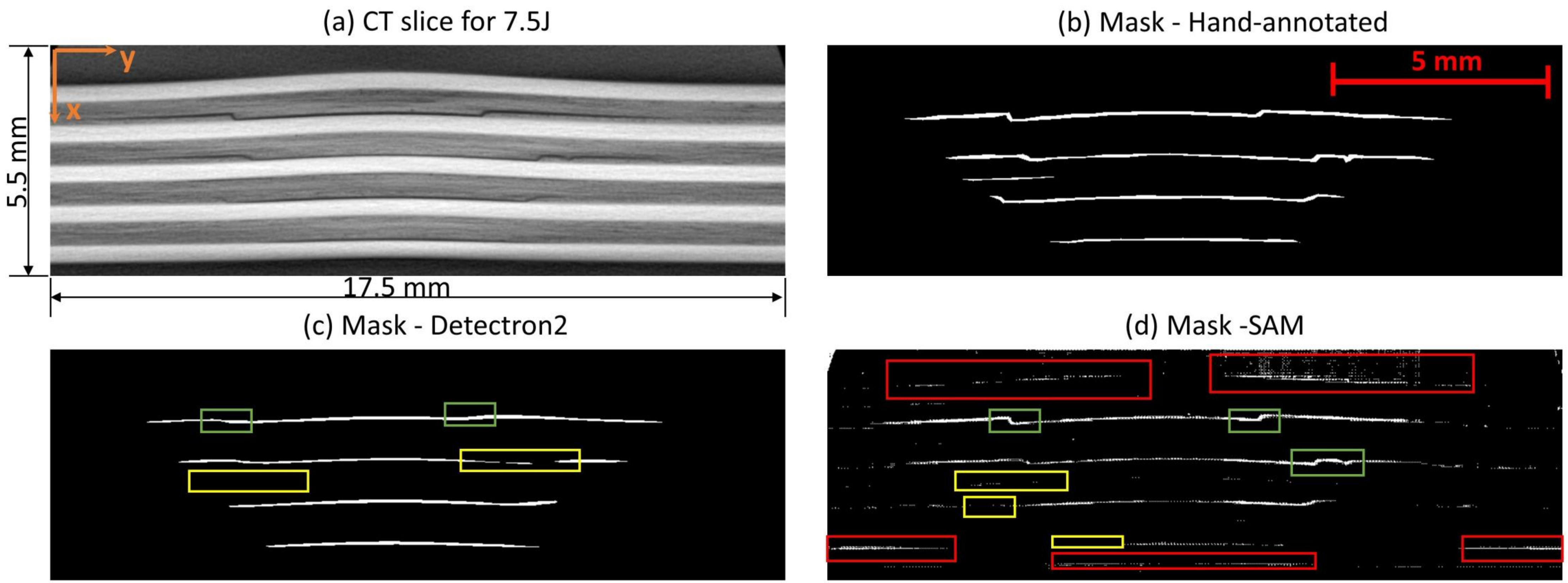


Training

SAM Hyperparameters	**Detectron2 Hyperparameters**
Optimizer: Adam (lr $= 1 \times 10^{-5}$, weight decay $= 0$)	Optimizer: SGD with momentum (lr $= 2.5 \times 10^{-4}$, momentum $= 0.9$)
Batch size: 2, Epochs: 500	Batch size: 2 (ROI batch size per image: 256)
Patch size: 256 pixels, Step size: 256 pixels	Number of workers: 2
Inference threshold: 0.95	Inference threshold: 0.60

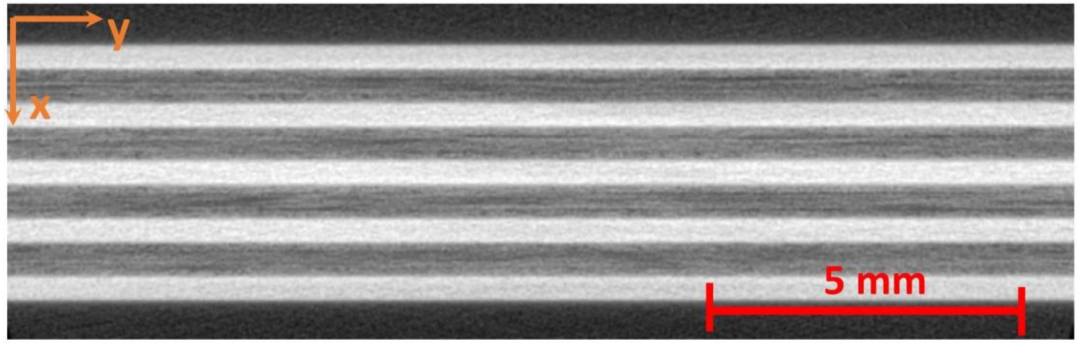
Table II: Key hyperparameters for SAM and Detectron2 training and inference.

Detectron2 vs. SAM



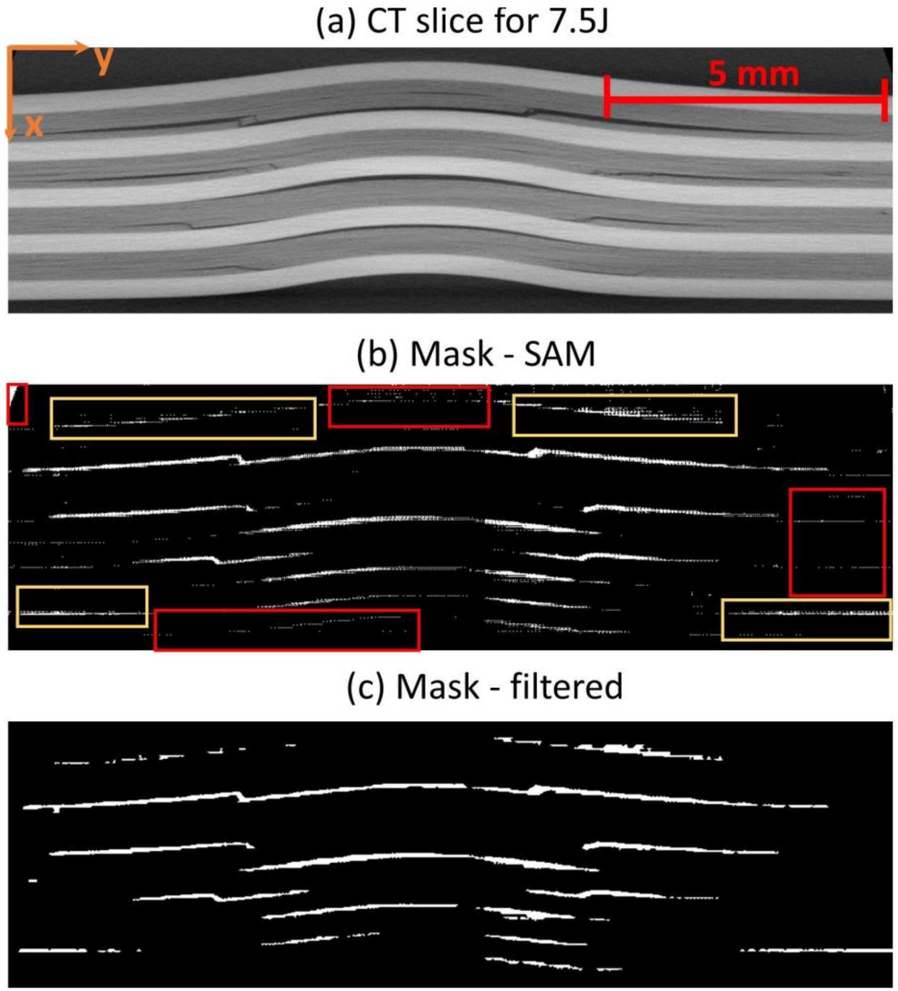
Detectron2 vs. SAM

(a) CT slice for Undamaged plate



(b) Mask -SAM

Detectron2 vs. SAM



Detectron2 vs. SAM

- Detectron2 provides more accurate (approx. 2.12 times better F1-score and 2.65 times better IoU score)
- Approx. 8 times faster training and 80 times faster inference as compared to the SAM.

Metric	Detectron2	SAM	SAM (with filter)
IoU	0.53 ($\pm$ 0.02)	0.19 ($\pm$ 0.01)	0.25 ($\pm$ 0.03)
Precision	0.77 ($\pm$ 0.02)	0.31 ($\pm$ 0.07)	0.39 ($\pm$ 0.14)
Recall	0.64 ($\pm$ 0.02)	0.40 ($\pm$ 0.11)	0.55 ($\pm$ 0.12)
F1	0.70 ($\pm$ 0.03)	0.35 ($\pm$ 0.17)	0.46 ($\pm$ 0.20)

Customized Mask Filter

Algorithm 1: High-level mask filtering algorithm applied on the masks generated by SAM.

Input: Mask image
Output: Filtered mask image
Identify regions of interest in the mask image;
Apply clustering algorithm (DBSCAN):

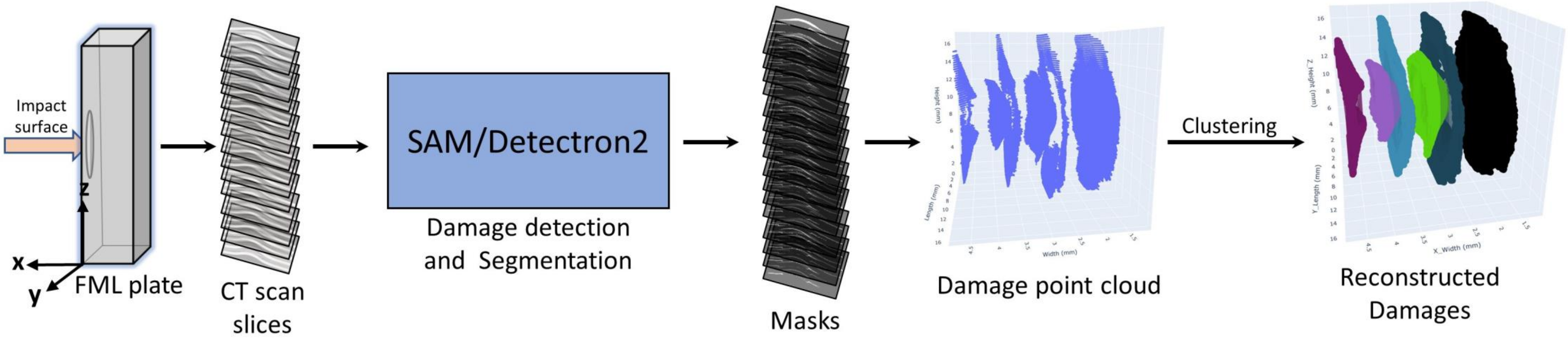
- Group dense regions into clusters
- Identify sparse pixels as noise

Remove noise (sparse pixels) from further consideration;
foreach *cluster* **do**
    Compute the shape of the cluster (fit a concave hull);
    Calculate the area of the cluster;
end
Select top N clusters based on area;
foreach *cluster* **do**
    if *cluster is in top N* **then**
        if *cluster shape meets aspect ratio criteria* **then**
            Mark cluster as accepted;
        else
            Mark cluster as potential false positive;
        end
    else
        Mark cluster as rejected;
    end
end
Generate final filtered mask:

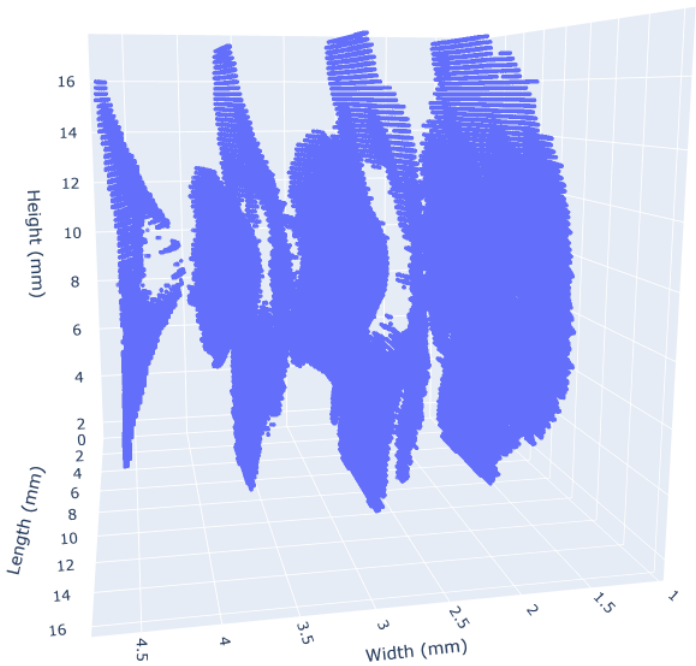
- Retain accepted clusters
- Remove rejected clusters, potential false positives, and noise

return Filtered mask image

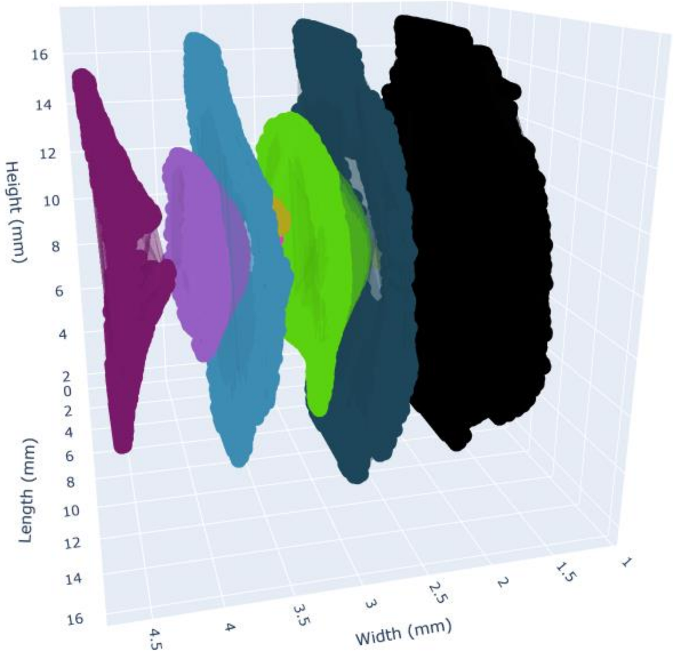
- Complete process pipeline.



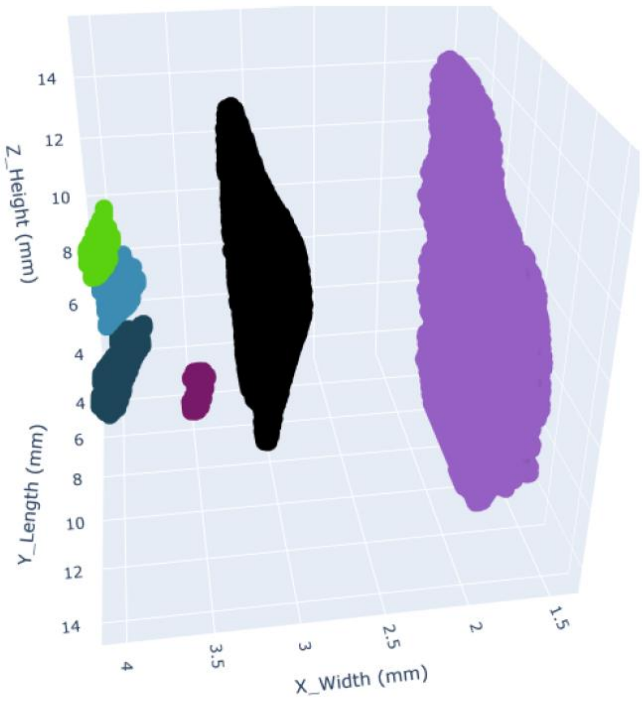
- Clustering of the damage point clouds using DBSCAN (Density-Based Spatial Clustering of Applications with Noise).
- Concave hull fitting.



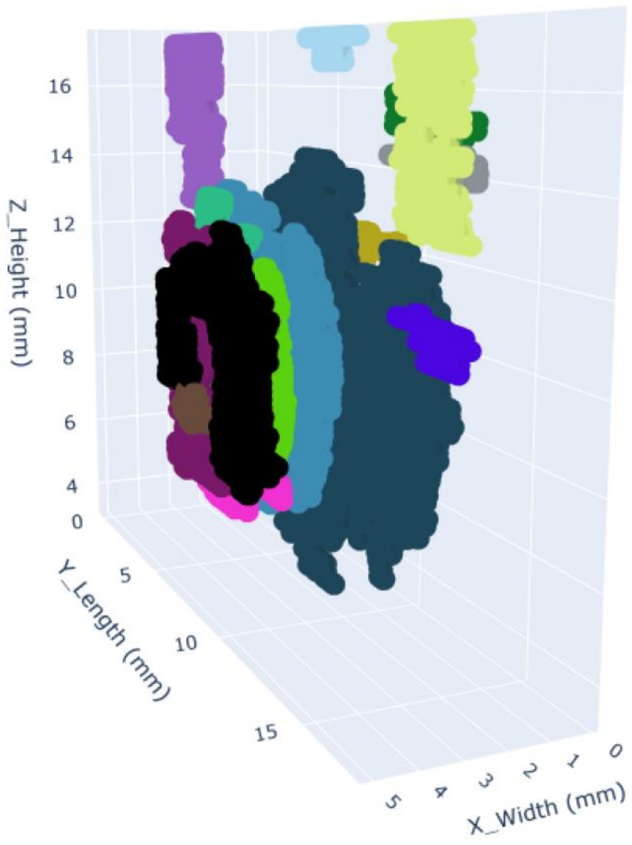
Clustering and hull fitting



Detectron2 vs. SAM

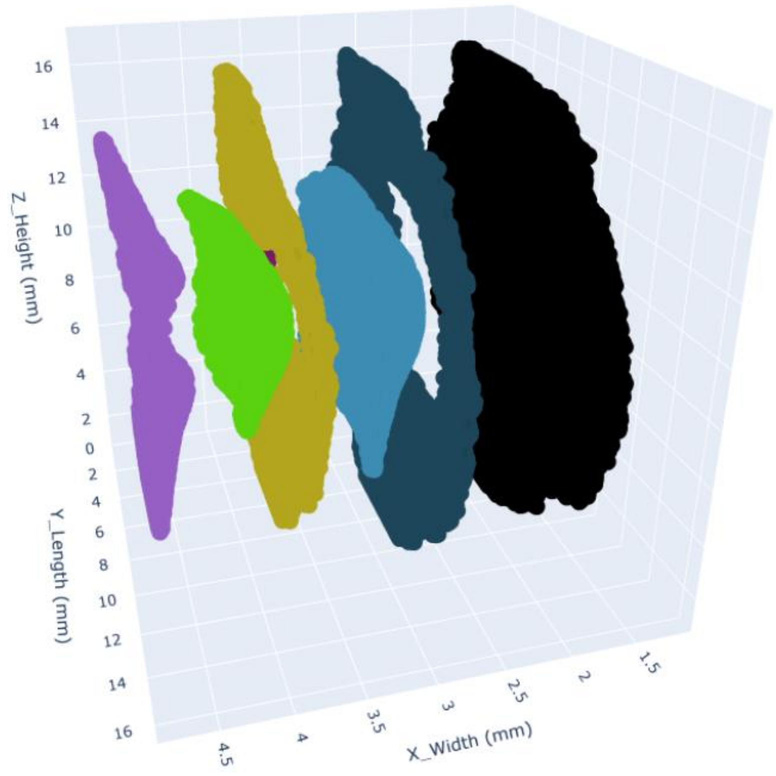


(a) Detectron2 - 5J

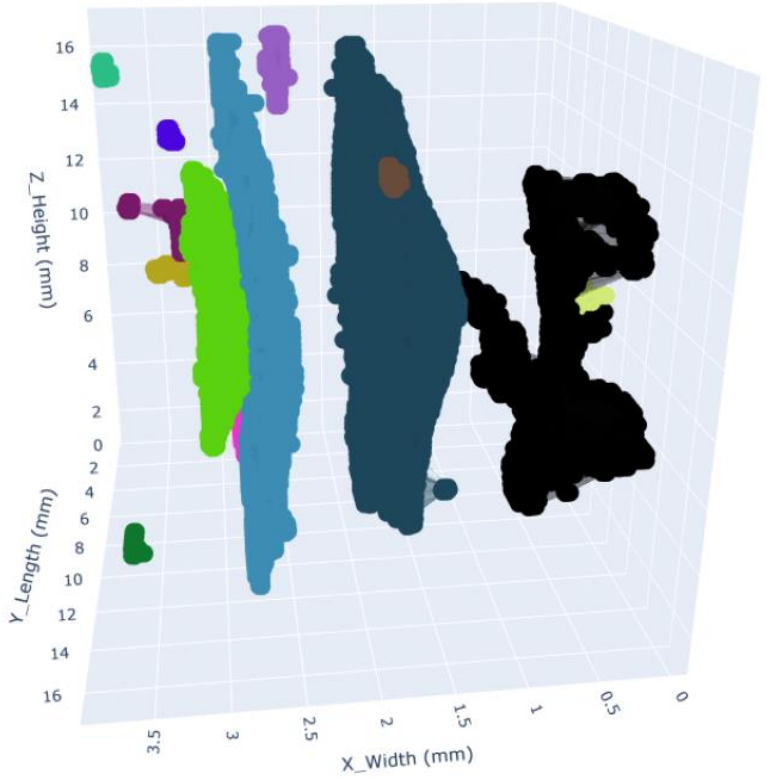


(b) SAM - 5J

Detectron2 vs. SAM

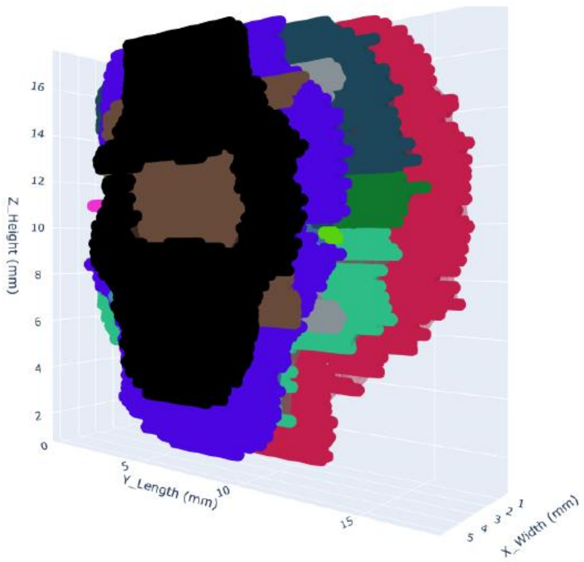


(c) Detectron2 - 7.5J

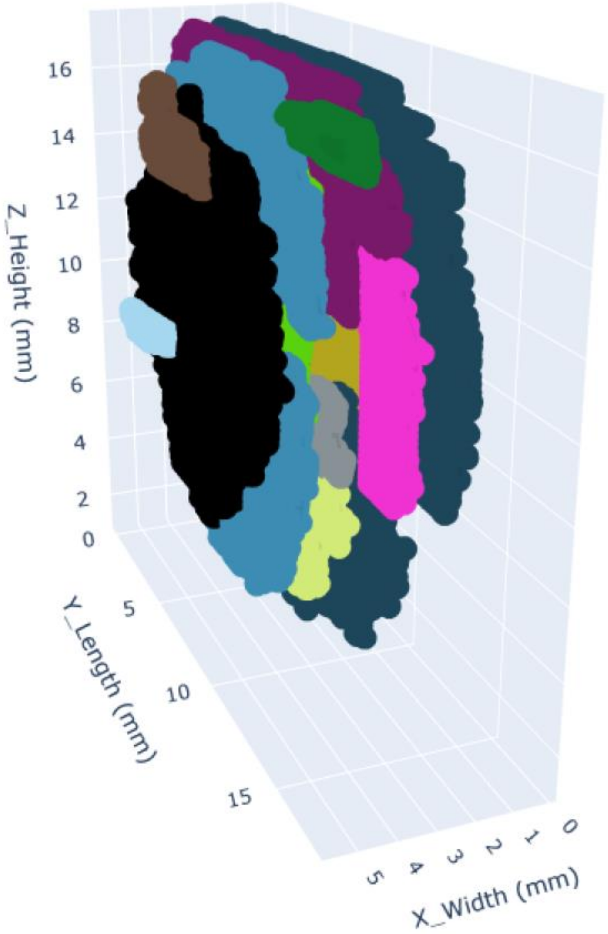


(d) SAM - 7.5J

Detectron2 vs. SAM

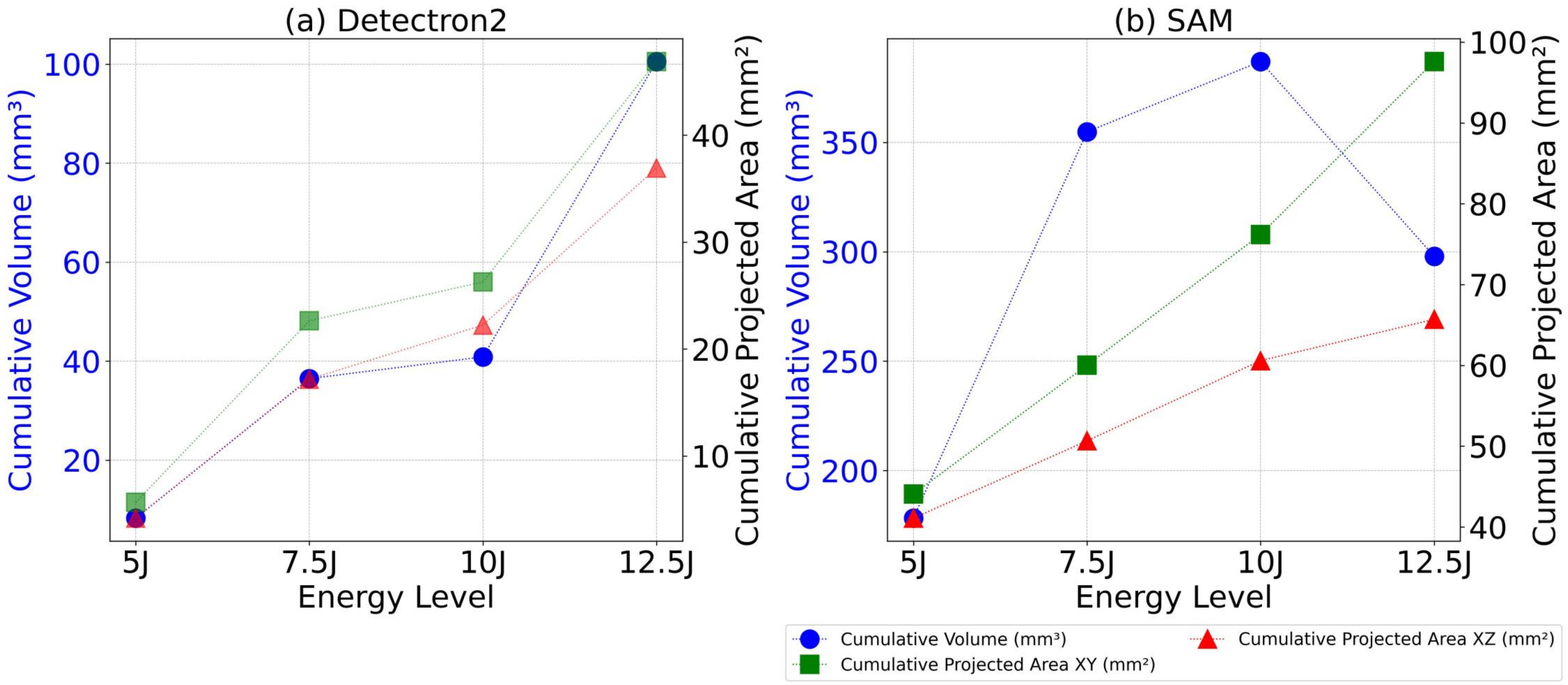


(c) Detectron2 - 12.5J

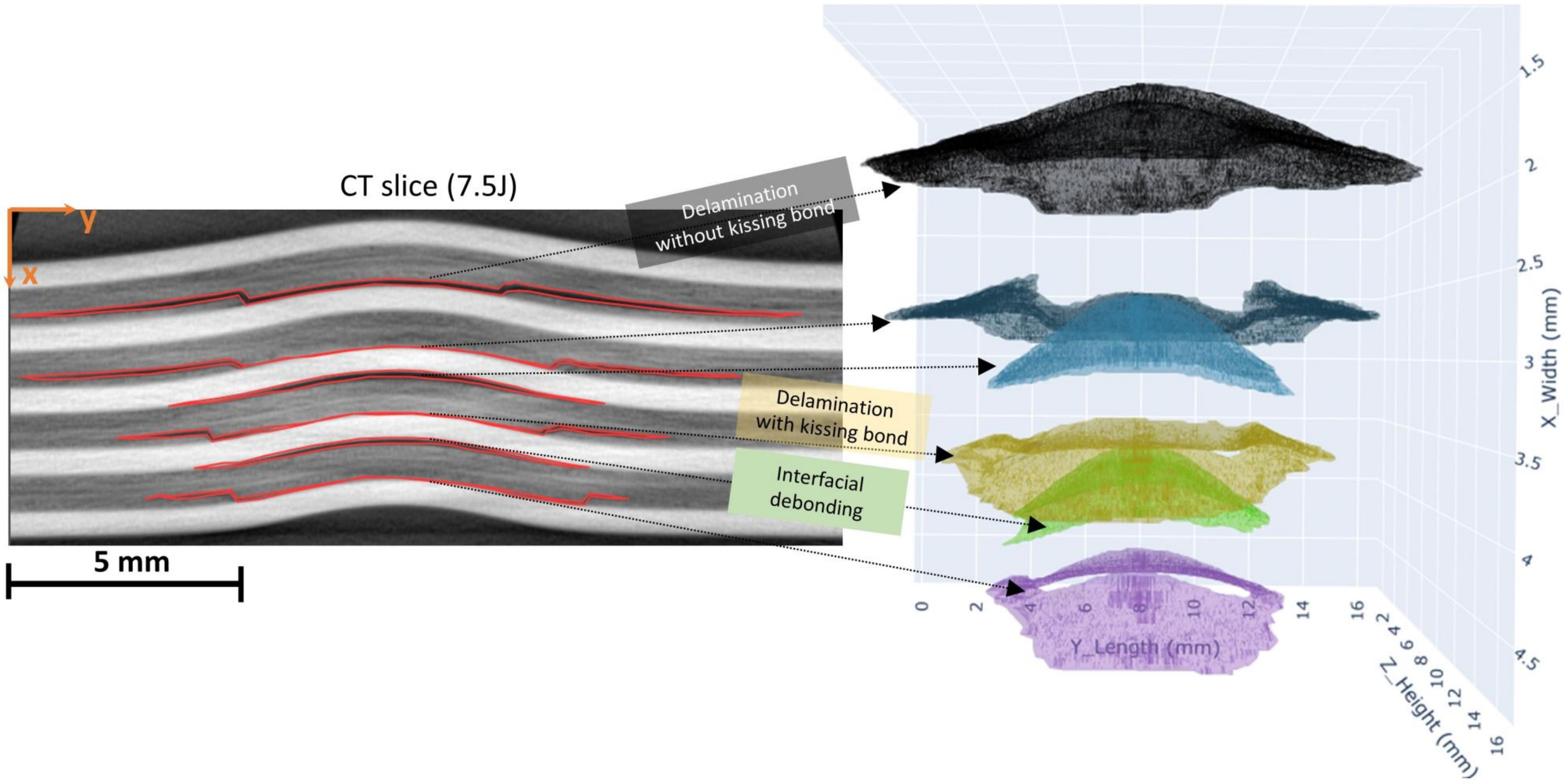


(d) SAM - 12.5J

Detectron2 vs. SAM



Damage types



References

- https://github.com/facebookresearch/detectron2?tab=readme-ov-file
- https://segment-anything.com/
- https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c
- https://insightfultscript.com/collections/programming/neural-network/resnet/
- He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask r-cnn." In Proceedings of the IEEE international conference on computer vision, pp. 2961-2969. 2017.
- Kirillov, Alexander, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao et al. "Segment anything." In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4015-4026. 2023.