Final Summary

SAM 2.1

- **Model**: Based on Meta's SAM2
- **Input**: *57* front-view plastic part images from the provided dataset(left_FLIR_Blackfly-S-BFS-U3-200S6C_23524388_1)
- **Output**:
  - Masked images (either cropped or zero-out background) → DIR: output/masked
  - Mask files → output/masks

Training

- **Dataset location**: data/sam2_data/samples
- Each image must have a **jpg & json** pair
  (The provided COCO JSON+RGB data were preprocessed using "utils/coco_to_sam2.py" to create the sample dataset)
- The jpg and json files must share the same filename.

Inference & Isolation

- Load the fine-tuned SAM2.1 model to generate a clean mask for the target plastic part in images
- Apply the generated masks to produce images with either **cropped** or **zero-out** backgrounds.

PatchCore

- **Model**: Anomalib PatchCore
- **Input**:
  - Preprocessed training & testing datasets → data/patchcore_data
  - Dataset for inference (choose via --masked flag: "zero_out" or "crop")
  - Zero-out background images are used by default.
    (If using "--masked crop", training & testing datasets must also use cropped backgrounds.)
- **Output**:
  - Heatmap
  - Overlay images
  - predictions.csv
  - metrics.json

Data Preprocessing

1. COCO JSON → jpg & json Pair Conversion

- For **SAM2.1 training**, each image must have a jpg & json pair.

2. Data Augmentation

- Since only *57* images (single selected view) are available, weak augmentation was applied to create additional training/testing samples for PatchCore. → In order to use unseed data for testing & training
- Augmentation techniques:
  - Rotation
  - Translation
  - Brightness adjustment
  - Blur
- Implementation: utils/weak_aug.py

Threshold Settings

- **Default** (w/o --threshold flag)
  - The PatchCore model uses Anomalib's F1AdaptiveThreshold to find the threshold that yields the highest F1 score on the Precision-Recall curve.
  - During validation, determine the optimal threshold and normalize **pred_score** so that this threshold corresponds to 0.5
- **Optional** (ROC-based calculation)
  - Uses ROC curve to find the threshold that maximizes the difference between TPR and FPR(Youden's index).
  - Implementation: roc_threshold (custom helper function, based on scikit-learn; inspired by MATLAB's anomalyThreshold)
  - --threshold <float> sets the maximum allowed FPR (0–1).
    → Candidates with an FPR greater than the specified maximum are excluded.
    - e.g.: --threshold 0.8