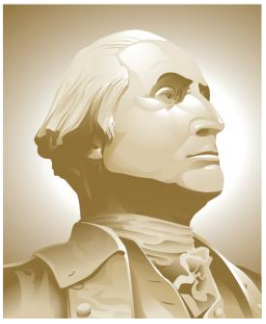

THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

Data Science Program
DATS 6303: Deep Learning
Individual Project Report

BYU - Locating Bacterial Flagellar Motors 2025

Pranav Dhawan

G39786431

Instructor: Dr. Amir Jafari

Date: 30th April 2025

CONTENTS

Table of Figures and Tables

Table of Tables

Table 1: Table of Tables

Table of Figures

Table 2: Table of Figures

1. Introduction

Bacterial flagellar motors (BFMs) are nanoscale rotary engines embedded in bacterial membranes that are responsible for propelling flagella and enabling motility. This motility is not merely a feature of bacterial life but a central mechanism behind key biological processes such as chemotaxis, environmental adaptation, and infection. As such, understanding the structure, function, and spatial distribution of BFMs is essential in microbiology and biomedical research. However, the accurate identification and localization of these motors within biological imaging data, particularly electron microscopy (EM) tomograms, presents a significant technical challenge.

EM tomographic datasets are inherently noisy, high-dimensional, and lack distinct boundaries between cellular structures, making manual annotation both time-consuming and highly error-prone. BFMs themselves appear as subtle features within these images, often indistinguishable without extensive domain expertise. Traditional computer vision methods are insufficient due to their inability to capture the fine-grained and 3D nature of the data, motivating the use of deep learning techniques to automate this process.

In this study, we focus on the BYU - Locating Bacterial Flagellar Motors 2025 Kaggle competition dataset, which provides annotated high-resolution EM tomograms. We aim to develop an automated detection pipeline using three state-of-the-art object detection models—CenterNet, YOLOv10, and Faster R-CNN—each leveraging a different architectural approach. These models are evaluated across a consistent pipeline involving 3D-to-2D data conversion, preprocessing, and augmentation. Through comparative analysis of their detection accuracy and robustness, we explore how each model addresses the unique challenges of biological image analysis.

This project not only benchmarks model performance on a difficult scientific imaging task but also lays the groundwork for scalable, AI-driven microscopy analysis in microbial research. This report details my contributions to the project, including dataset preprocessing, model training and testing and streamlit application development.

2. Description of individual work

2.1 Background and Contributions

In this project, my core responsibilities spanned across model training, dataset integration, application development, and group documentation. I began by experimenting with YOLOv8, a real-time object detection model optimized for detecting small-scale features, which aligned well with the task of locating bacterial flagellar motors (BFMs) in electron microscopy (EM) tomograms. Recognizing its limitations, I transitioned to YOLOv10x, a more advanced version designed for better spatial localization and robustness.

Additionally, I attempted to improve model generalizability by integrating the CryoET dataset, which contains complementary annotated tomographic slices. Beyond model experimentation, I fully developed an interactive Streamlit web application that allows users to upload EM slices, select a detection model, and visualize both bounding boxes and tabular motor coordinates. I also contributed significantly to the final report and the team presentation materials.

2.2 YOLOV8 Training and Testing

YOLOv8 was selected as the initial baseline model due to its well-established performance in detecting small and fine-grained objects. This made it a strong candidate for identifying bacterial flagellar motors in tomographic slices, which are low-contrast and small in size. The training script (yolov8-train-baseline.py) included:

2.2.1 Preprocessing Pipeline

To prepare the dataset for YOLOv8, a similar preprocessing strategy was applied:

- Slice Normalization: Intensity values were normalized using the 2nd and 98th percentiles to enhance visibility of structures.
- Motor Filtering: Only slices with annotated motors and high TRUST values 4 were retained.
- Bounding Boxes: Motor positions were centered in fixed 24×24 bounding boxes and converted to YOLO format [class, x_center, y_center, width, height].
- Image Resizing: Slices were resized to 512×512 pixels, matching the default input size for YOLOv8.
- Train-Test Split: Data was split by tomogram ID to avoid leakage, with an 80:20 train-validation ratio.

2.2.2 Training Configuration

Epoch: 100

Learning Rate = 0.0001

Pretrained weights = yolov8l.pt

2.2.3 Inference and Post Processing

Testing was carried out using the yolov8-test-baseline.py script, which runs inference on unseen test tomograms and saves results in YOLO format. While this model achieved reasonable detection performance, its recall was lower than desired, especially in complex or noisy slices.

2.2.4 Limitations Observed

1. Localization Drift: Bounding boxes were often slightly off-centred, especially in densely populated regions.

2. Missed Detections: Some motors were missed due to insufficient context in the lower resolution 512×512 images.

3. Slower Convergence: Required more epochs to stabilize loss curves.

2.3 CryoET Dataset Integration

To boost the model's generalization capabilities, I attempted to incorporate the CryoET dataset, which contains additional tomograms relevant to bacterial flagellar motors. The processing pipeline (cryoet_preprocess.py) involved:

- Normalization using 2nd and 99th percentile clipping for better contrast
- Resizing slices to 512×512 resolution
- Conversion of filenames and slice indices to match the BYU dataset's naming convention

However, despite careful preprocessing, this integration introduced a domain shift due to visual and structural differences in the imaging modalities. Models trained on the combined dataset performed worse on the original BYU test set, especially in terms of precision, confirming that uniform data distribution is key for high-performing detection models.

2.4 YOLOV10 Training and Testing

I shifted to training YOLOv10x, a recent architecture with superior localization capabilities. Using the script yolov10-train-finetuned.py.

2.4.1 Model Architecture

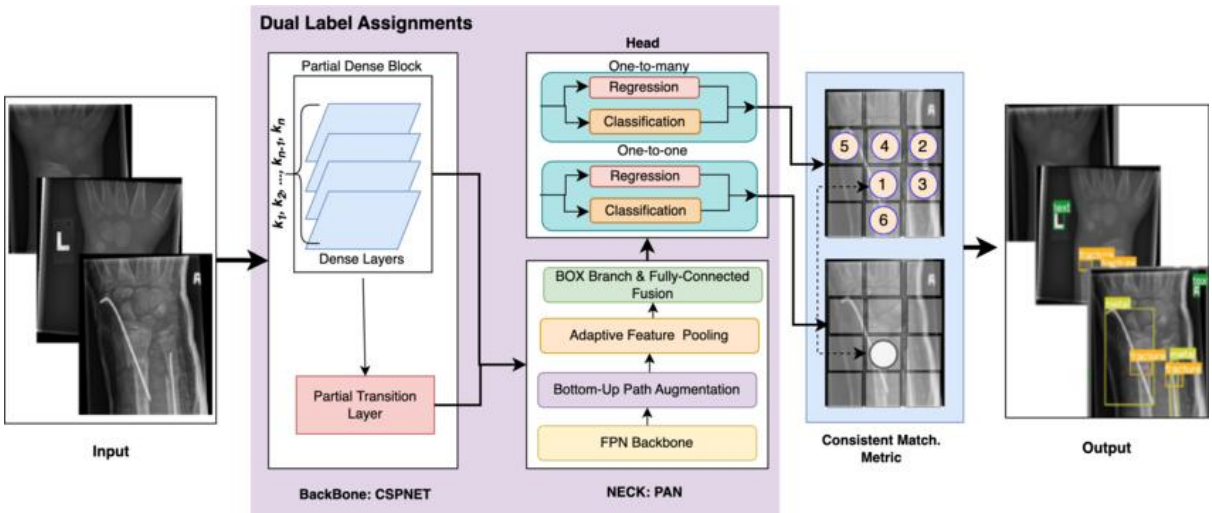

Figure 2.4.1: YOLOv10 Architecture Overview

The YOLOv10 pipeline comprises the following key components:

1. Input:

The model takes input of images that are passed into the model for feature extraction and object detection.

2. Backbone: CSPNet (Cross Stage Partial Network)

The backbone is responsible for feature extraction. CSPNet is used due to:

- High computational efficiency: Reduces FLOPs without compromising accuracy.
- Partial Dense Blocks: Enable deeper feature extraction via dense connections.
- Partial Transition Layer: Connects blocks and forwards critical features, balancing speed and performance.

3. Neck: PAN (Path Aggregation Network)

The neck bridges the backbone and the head, merging features from different scales.

- Feature Pyramid Network (FPN): Enhances multi-scale detection.
- Bottom-Up Path Augmentation: Introduces spatial detail from lower layers to improve localization.
- Adaptive Feature Pooling: Focuses on the most informative regions dynamically.

4. Detection Head: Dual Label Assignments

YOLOv10 employs two parallel prediction heads for enhanced stability and performance:

- One-to-Many Head:
  - Allows one ground truth object to match multiple predictions.
  - Boosts recall and prevents missing detections.
- One-to-One Head:
  - Each ground truth is assigned to a single prediction.
  - Enhances precision and reduces redundant boxes.

Each head performs:

- Regression (bounding box coordinates)
- Classification (object class)

5. Consistent Match Metric

After getting outputs from both heads, YOLOv10 applies a consistency matching strategy. This step evaluates how well the predictions from both heads align. If both heads agree on a prediction, it is more likely to be accurate. It helps in reducing false positives and stabilizes training and inference.
In the diagram, this is visualized with indexed matched boxes between the two prediction streams.

6. Output: Final Bounding Boxes with Labels

The model outputs the detected objects with bounding boxes drawn around them. Each box includes a label (class name) and a confidence score. The boxes represent detected areas.

2.4.2 Dataset Preparation

To train YOLOv10 for detecting bacterial flagellar motors in tomograms, a rigorous preprocessing pipeline was implemented. The following are the key steps taken:

1. Slice Normalization

Each slice was normalized using the 2nd and 98th percentiles to enhance contrast and suppress extreme intensities.

2. Bounding Box Generation

Fixed-size 24×24 pixel bounding boxes were centered around motor coordinates to ensure consistent annotation.

3. TRUST-Based Filtering

Only slices with TRUST scores of 4 or 6 (high confidence) were retained to minimize noisy annotations.

4. Coordinate Transformation to YOLO Format

Boxes were converted into [class, x_center, y_center, width, height] format, normalized to image dimensions.

5. Data Splitting (Preventing Leakage)

Dataset was split by tomogram ID to avoid data leakage between training and validation sets.

6. Image Resizing

Images were resized to 940×940 pixels to match YOLOv10 input specifications.

2.4.3 Model Training

2.4.3.1. Pretrained Model Initialization

The training process uses a pre-trained YOLOv10x model (yolov10x.pt) as the starting point. This allows the model to benefit from prior learning on large datasets and converge faster on the specific medical domain.

2.4.3.2. Dataset Configuration

A dataset.yaml file was created, defining class names, data splits, and paths.

2.4.3.3. Data Augmentation

Several augmentation techniques were applied to enhance model robustness:

Augmentation	Parameter	Description
Mosaic	mosaic=0.5	Combines 4 images into 1 during training. Adds context diversity and forces the model to generalize across spatial configurations.
MixUp	mixup=0.2	Blends two images and their labels, helping with regularization and improving robustness to label noise.
Copy-Paste	copy_paste=0.2	Copies objects (motors) from one image and pastes them into another to simulate occlusion and boost instance diversity.
Flipping	fliplr=0.5, flipud=0.2	Random horizontal and vertical flips simulate imaging variation.
Close Mosaic	close_mosaic=10	Turns off mosaic augmentation after 10 epochs for finer training in the final stages.

Table 2.1: Augmentations used

2.4.4 Post Processing

After the YOLOv10 model completes inference, the raw outputs still require processing to generate final, usable detections. This post-processing stage ensures that only the most accurate and meaningful predictions are kept. The key components are described below:

2.4.4.1 Raw Output Interpretation

YOLOv10 outputs a large number of candidate bounding boxes with:

- Objectness score (confidence that an object exists in the box),
- Class probabilities, and
- Box coordinates in normalized format [x_center, y_center, width, height].

These predictions often overlap, especially in dense regions like biological tomograms, and need to be refined.

2.4.4.2 Confidence Thresholding

- A confidence threshold (0.5) is applied to remove low-confidence predictions.
- This helps eliminate boxes that the model is unsure about, reducing false positives.

2.4.4.3 Non-Maximum Suppression (NMS)

Non-Maximum Suppression is the most critical step in post-processing. When multiple boxes overlap heavily and predict the same object, NMS keeps only the one with the highest confidence score, removing all others.

NMS works by:

- Sorting boxes by confidence score,
- Iteratively selecting the top box,
- Removing all other boxes with IoU (Intersection-over-Union) above a threshold (commonly 0.5).

This ensures only one box per object, preventing duplicate detections.

2.5 Streamlit App Development

I developed the interactive Streamlit dashboard to support user-friendly inference and visualization of the detection pipeline. Key features include:

- Model Selection Dropdown: Users can select from CenterNet, YOLOv10, or Faster R-CNN
- Confidence Threshold Slider: Allows fine-tuning of detection sensitivity
- Drag-and-Drop Image Upload: Users can upload 2D slices in JPEG or PNG format
- Real-time Inference: Inference is triggered immediately after upload
- Processed Output:
    - Image with predicted bounding boxes
    - A table displaying motor coordinates and confidence scores

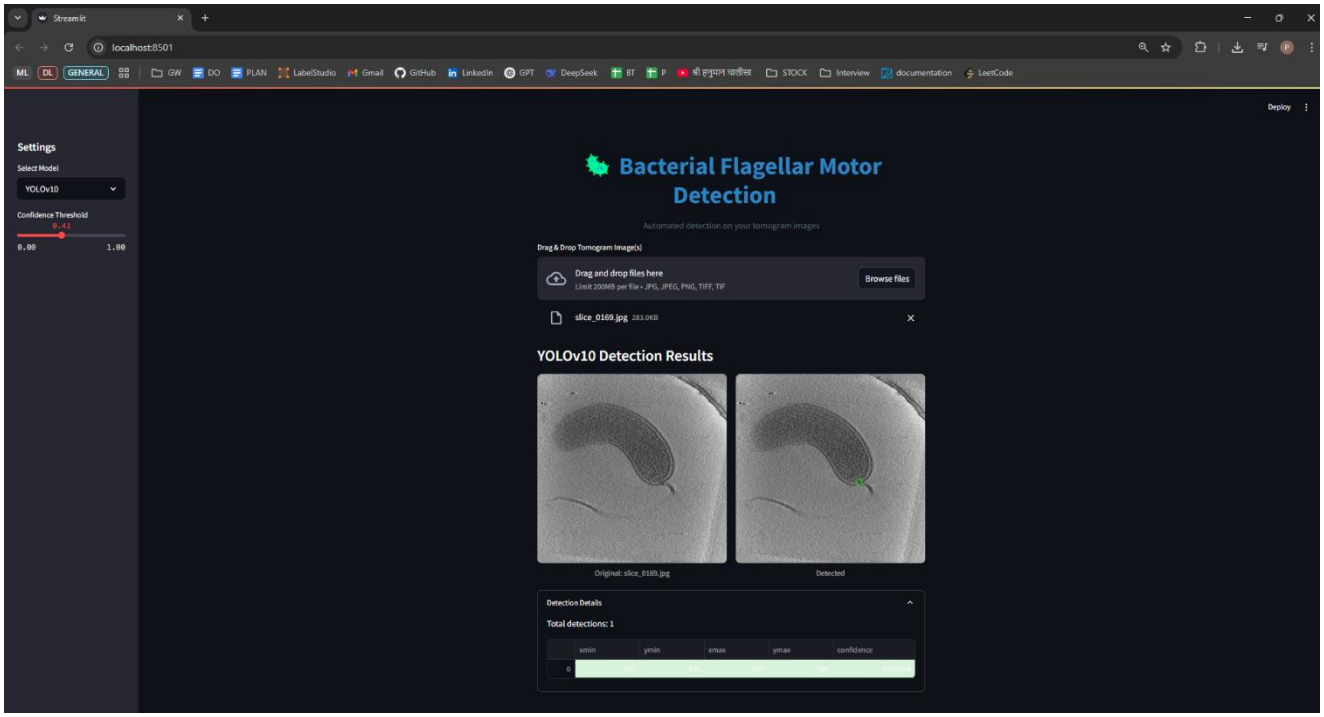

Figure 2.1: Streamlit Demo Application

3. Results

3.1 CryoET Dataset Integration

To improve generalization, I incorporated slices from the CryoET dataset into the training pipeline. The images were carefully normalized and resized to match the BYU dataset using percentile-based contrast normalization and uniform resolution scaling (512×512). However, this integration did not improve performance. In fact, results showed a decline in both precision and recall.

The underlying issue was domain shift—differences in visual quality, noise patterns, and slice composition between CryoET and BYU data made it difficult for the model to generalize. Despite augmentations and harmonization, the mixed-domain training set introduced inconsistencies that the YOLO architecture could not fully reconcile.

This finding emphasizes the importance of domain consistency in biomedical deep learning applications and suggests future use of domain adaptation techniques or separate fine-tuning per dataset.

3.2 YOLOv8 Results

The YOLOv8-L model was used as a baseline, trained on the BYU dataset using 640×640 image slices and standard augmentations. The training metrics, shown below, reflect moderate convergence with acceptable but sub-optimal generalization:

Metric	Value
mAP@50	0.800
mAP@50–95	0.422
Precision	0.676
Recall	0.779

Table 3.1: Performance Metrics of YOLOv8

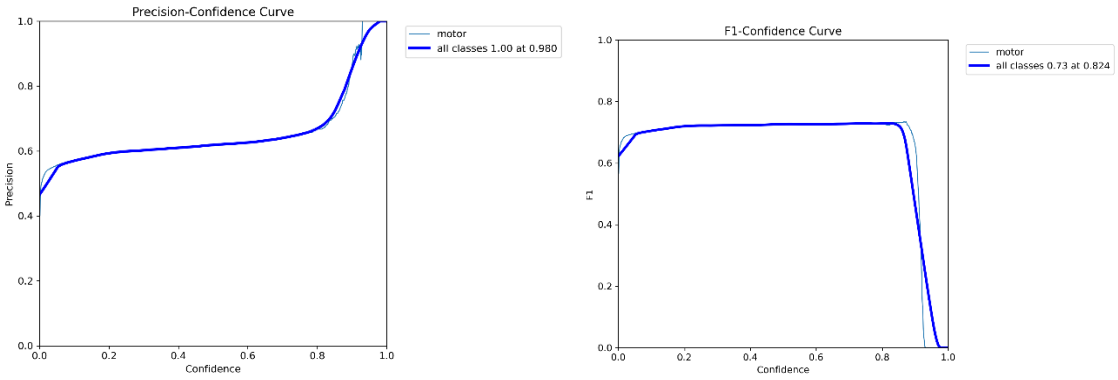

Figure 3.1 and 3.2: Precision Confidence Curve and F1 Confidence Curve

1. The F1 score plateaus at ~0.73 between 0.2–0.8 confidence thresholds, peaking at 0.824. Beyond 0.85, F1 drops quickly indicating a sharp trade-off: increasing precision leads to significantly lower recall.
2. Precision increases sharply after 0.8 confidence and peaks at 1.00 near 0.98, meaning very high confidence thresholds lead to very accurate detections (but fewer of them). Below 0.6 confidence, precision is relatively low, suggesting many false positives at low thresholds.

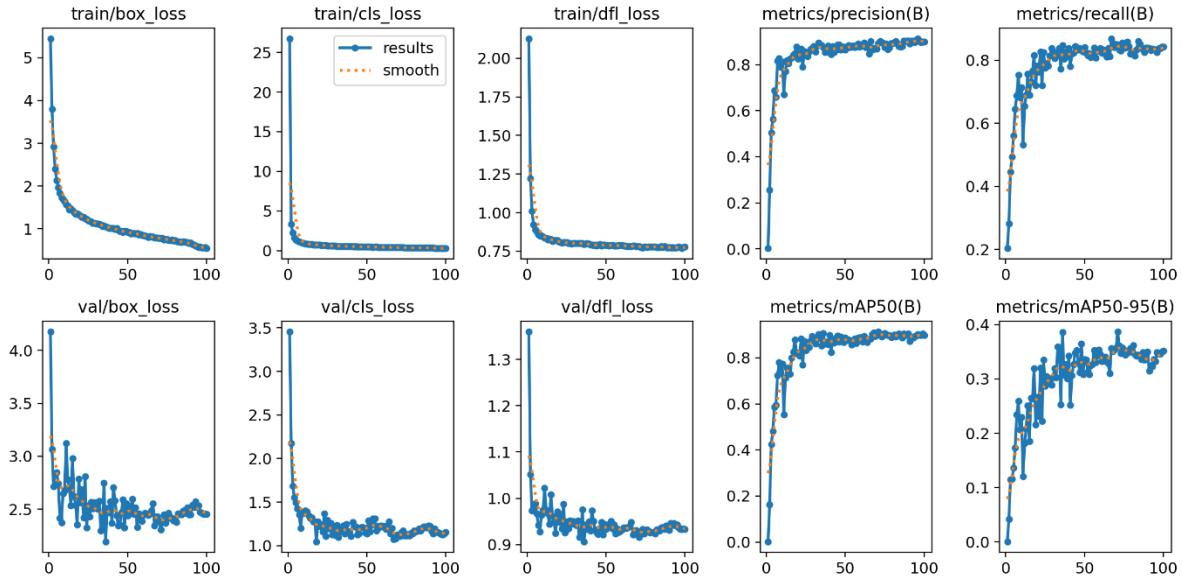

Figure 3.3: Training Metrics for YOLOv8

During training and evaluation, YOLOv8 demonstrated a steady improvement in precision, ultimately reaching a peak value of approximately 0.87. This indicates that the model was highly confident in its predictions and produced relatively few false positives at optimal thresholds. Recall peaked at around 0.78, suggesting that while the model was able to detect a significant number of true positives, it occasionally missed some motor instances, especially in complex or low-contrast regions. The model achieved a mean Average Precision at IoU threshold 0.50 (mAP@50) of around 0.80, reflecting good overall object detection capability. However, its performance on the stricter localization metric, mAP@50-95, plateaued at roughly 0.42, highlighting limitations in accurately aligning predicted bounding boxes with ground truth annotations. This shortfall in precise localization motivated the exploration of more advanced models like YOLOv10.

Key Observations:

1. YOLOv8 provides reasonable performance, especially in early detection.
2. Strong points: High precision at strict confidence, stable loss curves.
3. Limitations: Lower mAP@50-95 and slightly suboptimal recall led to its replacement by YOLOv10 for better localization.

3.3 YOLOv10 Results

The YOLOv10x model was trained using 960×960 images, advanced augmentations (mosaic, mixup, copy-paste), and optimized hyperparameters (DFL loss, higher batch size, and cosine learning rate scheduling). It significantly outperformed YOLOv8:

Metric	Value
mAP@50	0.948
mAP@50–95	0.630
Precision	1.000
Recall	0.960

Table 3.2: Performance Metrics of YOLOv10

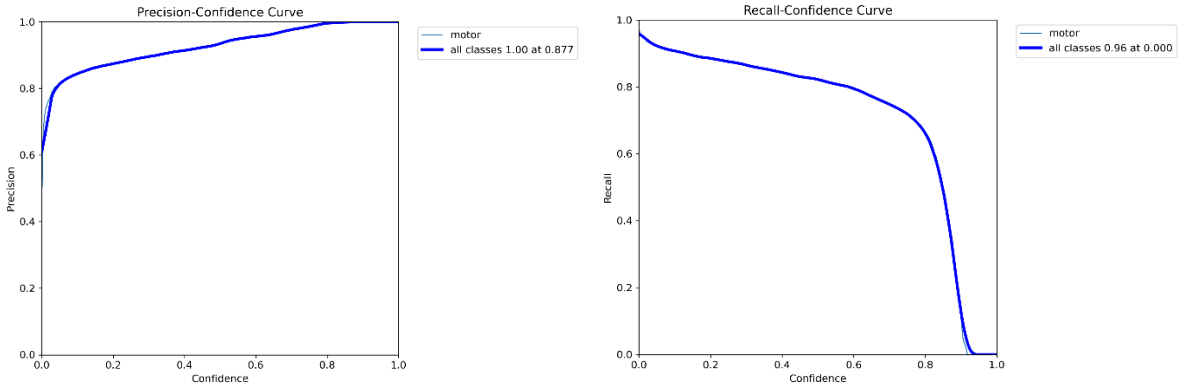

Figure 3.4 and 3.5: Precision Confidence Curve and Recall Confidence Curve

1. The precision remains high across most thresholds and reaches 1.00 near a confidence of 0.877, indicating YOLOv10's capability to deliver highly confident and accurate predictions with minimal false positives.
2. Recall starts high (~0.95) and gradually decreases as confidence increases, consistent with typical model behavior where higher precision sacrifices recall. Peak recall is observed at low confidence thresholds.

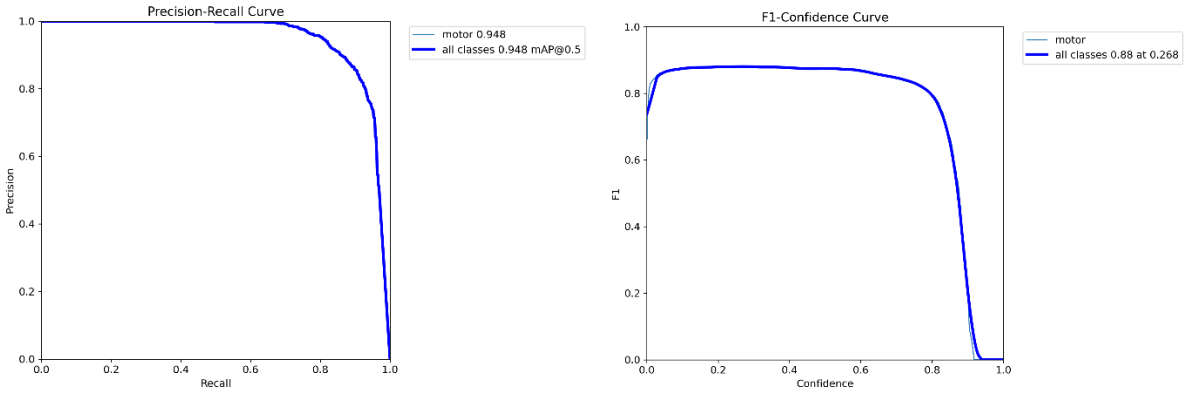

Figure 3.6 and 3.7: Precision Recall Curve and F1 Confidence Curve

1. The F1-Confidence curve peaks around a confidence threshold of 0.268, achieving an F1 score of 0.88. This suggests an optimal balance between precision and recall around this threshold.
2. The PR curve reinforces the model's strong tradeoff between recall and precision, with an area under the curve corresponding to mAP@50 = 0.948. This validates the overall reliability of YOLOv10 across detection scenarios.

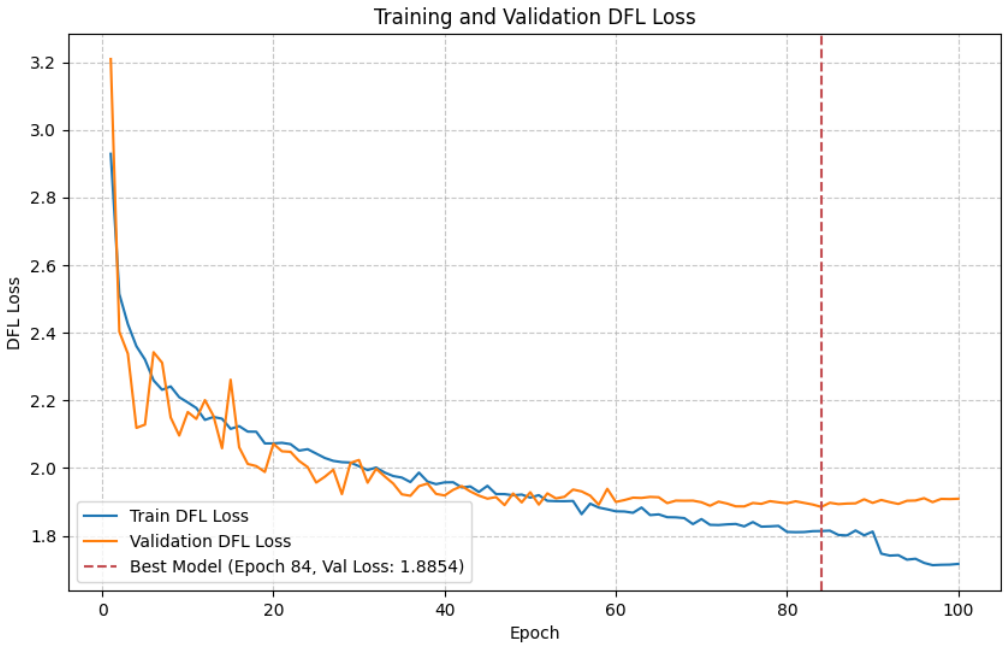

Figure 3.8: DFL Loss

This plot shows a steady decline in Distribution Focal Loss (DFL) during training, with the best model checkpoint observed at Epoch 84 with a validation loss of 1.8854. This indicates strong convergence and no overfitting.

4. Summary and Conclusions

In this project, I contributed to developing and evaluating deep learning models for the detection of bacterial flagellar motors (BFMs) in BYU data. My work focused on training and comparing the YOLOv8 and YOLOv10 models, integrating the CryoET dataset, and building a user-friendly Streamlit application for model inference.

Initially, the YOLOv8-L model served as a baseline and delivered moderate results. While it showed decent precision and recall, it struggled with the subtlety and scale of BFM features in noisy tomographic slices. In an effort to improve model generalization, I incorporated data from the CryoET dataset. However, due to domain differences—such as imaging contrast and motor appearance—the integrated model exhibited reduced accuracy, underscoring the importance of domain consistency in biomedical image analysis.

The transition to the YOLOv10x model yielded the most successful outcome. Leveraging high-resolution input, advanced augmentation techniques, and Distribution Focal Loss (DFL), YOLOv10x achieved near-perfect precision (1.00), high recall (0.96), and the highest mAP@50 of 0.948 across all experiments. Its training curves demonstrated fast convergence and stable learning, while confusion matrix results confirmed its reliability in detecting motors with minimal false positives.

I also developed a full-featured Streamlit web application, enabling users to upload EM slices, select models, adjust detection thresholds, and view prediction outputs visually and numerically.

Through this project, I gained valuable experience in deep learning model training, domain-specific data preprocessing, performance evaluation, and full-stack deployment. In the future, this pipeline could be further improved by:

- Applying domain adaptation techniques for integrating diverse datasets
- Exploring 3D convolutional models to directly handle volumetric data
- Incorporating semi-supervised or few-shot learning to reduce reliance on extensive annotations

Overall, this model namely YOLOv10x performed the best out of all 3 models i.e. CenterNet, FasterRCNN and YOLO.

Metric	YOLOv8-l (Large)	YOLOv10-x (Extra Large)
mAP50	0.800	0.948
mAP50-95	0.422	0.630
Precision	0.676	1.000
Recall	0.779	0.960

Table 4.1: Comparative Results

5. Percentage of Code

In this section, I estimate the proportion of code that was reused from external sources versus what I developed independently.

YOLO Training and Testing Scripts

- YOLOv8 and YOLOv10 scripts were primarily self-developed, including data loading, augmentation, inference routines, and evaluation pipelines.
- I used approximately 10% of the code from the official BYU competition starter kit (mainly for initial data structure and configuration).
- An additional 10% was adapted from the Ultralytics YOLO GitHub repository, mainly for model training calls and some augmentation syntax.

Estimated reuse: 20%
Original code written: 80%

CryoET Preprocessing

- The CryoET preprocessing script was adapted from code available on the dataset's official website/repository.
- I customized the normalization and naming conventions to match the BYU format, but the core image handling loop and percentile normalization logic remained similar.

Estimated reuse: 50%
Original code written: 50%

Streamlit Web Application

- The Streamlit app was designed and implemented almost entirely by me, including model integration, UI controls (dropdown, slider, drag-and-drop), and visualization.
- However, I relied on 20% code snippets and debugging guidance from GenAI tools (like ChatGPT) to resolve certain issues related to file handling and plotting.

Estimated reuse: 20%
Original code written: 80%

Overall Code Reuse Estimate

To compute the overall percentage, let's assume roughly equal weight across the three major components:

Therefore, approximately 30% of the code in my individual contribution was reused or adapted from external sources, and 70% was developed independently

6. References

1. Ultralytics YOLOv8 Documentation
   Ultralytics. *YOLOv8: State-of-the-Art Real-Time Object Detection*.
   https://docs.ultralytics.com

2. BYU Kaggle Competition
   Brigham Young University. *Locating Bacterial Flagellar Motors (2025) - Kaggle Competition*.
   https://www.kaggle.com/competitions/byu-locating-bacterial-flagellar-motors-2025

   https://www.kaggle.com/code/andrewjdarley/train-yolo

3. YOLOv10 Release and Documentation
   Ultralytics. *YOLOv10: Next-Gen Object Detection Architecture*.
   GitHub Repository: https://github.com/ultralytics/ultralytics

4. Streamlit
   Streamlit Inc. *Streamlit: The fastest way to build and share data apps*.
   Available at: https://streamlit.io

5. CryoET Dataset Source
   Shi, J. et al. (2020). *Cryo-Electron Tomography for the Structural Analysis of Bacterial Flagellar Motors*.
   Data Source: https://www.kaggle.com/datasets/brendanartley/cryoet-flagellar-motors-dataset/data

6. Lecture Slides – DATS6303
   The George Washington University. *Deep Learning*.