Project Report: Bacterial Flagellar Motor Detection in Cryo-ET Images

Date: May 29, 2025

1. Introduction

This document details the methodology and implementation of a solution for the Locating Bacterial Flagellar Motors" research work. The primary objective of this competition is to accurately identify and locate bacterial flagellar motors within 3D tomograms generated from cryo-electron tomography (cryo-ET) images. Automating this process is critical for accelerating research in molecular biology, drug development, and synthetic biology, as manual annotation is labor-intensive and time-consuming.

Our approach leverages the efficiency and accuracy of YOLOv8, a state-of-the-art object detection framework, by adapting it to handle 3D volumetric data. This report outlines the steps taken, from raw data parsing and preparation to model training, analysis, and final submission generation.

2. Data Preparation and Preprocessing

The dataset consists of 3D cryo-ET tomograms in `.mrc` format, accompanied by ground truth 3D coordinate annotations for flagellar motors. Since YOLOv8 is primarily a 2D object detector, a crucial initial step involved transforming this 3D data into a 2D image-and-annotation format suitable for YOLOv8 training.

2.1 Data Annotation Process

The raw data for this competition, particularly the ground truth labels, are derived from the "MotorBench" dataset, an **expert-annotated dataset** of bacterial flagellar motors. This annotation process was meticulously performed by a trained researcher, Dr. Mohammed Kaplan, to ensure high-quality and consistent labeling.

- **Manual Inspection by Trained Experts:** Annotators utilized specialized software (e.g., IMOD/3dmod) designed for visualizing and navigating complex 3D cryo-ET volumes. Given the inherent low signal-to-noise ratio and often subtle visual cues in cryo-ET, human expertise was essential to accurately identify flagellar motors amidst other cellular components and background noise. The annotator would scroll through the 2D slices of a 3D tomogram, pinpointing the precise location where a flagellar motor was visible.
- **Point Annotation (3D Coordinates):** The annotation method involved marking a **single 3D point** (the approximate center) for each flagellar motor. These points were recorded with their exact (x, y, z) voxel coordinates, as reflected in the train_labels.csv file provided for the competition. This point-based labeling is a standard practice for defining the locations of macromolecular complexes in cryo-ET.
- **High-Quality Ground Truth:** The "MotorBench" dataset's emphasis on expert annotation ensures that the provided ground truth is reliable and serves as a robust foundation for

training machine learning models aiming to automate this complex task. The manual, labor-intensive nature of this annotation process (performed by one researcher over a significant duration) underscores the necessity and value of the competition's goal to automate flagellar motor detection.

2.2 3D to 2D Data Transformation (Notebook: `parse data.ipynb`)

With the understanding of the 3D point annotations, the next step involved adapting this 3D volumetric data and its 3D point labels into a 2D format compatible with YOLOv8.

Implementation Details:

- **Input Handling:**
  - Raw 3D tomograms were loaded using the mrcfile library, converting them into 3D NumPy arrays representing volumetric pixel intensities.
  - The expert-annotated train_labels.csv file, containing 3D (x, y, z) coordinates of flagellar motors, was loaded using pandas.
- **3D to 2D Slice Generation:**
  - Each 3D tomogram was iteratively sliced along its Z-axis (depth) to produce a series of 2D image slices.
  - **Normalization:** Pixel values in these 2D slices were normalized (e.g., scaled to a 0-255 range) to enhance contrast and ensure compatibility with standard image formats and neural network inputs. Techniques like min-max scaling or histogram equalization were considered to optimize visual quality and feature representation.
  - **Image Saving:** Each 2D slice was saved as a .png or .jpg image file to a designated directory (train/images/, val/images/), with filenames systematically encoding the original tomogram ID and slice number (e.g., tomogram_ID_slice_NUMBER.png).
- **3D to 2D Bounding Box Conversion:**
  - For each ground truth 3D motor annotation (x, y, z), 2D bounding boxes were generated for relevant slices. Recognizing that motors are 3D objects, they were assumed to span a small range of Z-slices.
  - A fixed "motor size" (e.g., diameter D in voxels), determined through preliminary data exploration or by understanding the physical dimensions of the motor, was used to define the extent of the motor.
  - For each slice z_slice falling within the motor's assumed Z-range (e.g., $z \pm D/2$), a 2D bounding box was created. This box was centered at (x, y) on the z_slice and had a width and height equal to D.
  - **YOLO Format Conversion:** The generated 2D bounding box coordinates (x_center, y_center, width, height) were normalized (0-1 range relative to image dimensions) and formatted into the YOLO standard: class_id x_center y_center width height. Since only one class ("flagellar_motor") is present, class_id was consistently 0.
- **Annotation Saving:** These YOLO-formatted bounding box annotations were saved as individual .txt files, one for each corresponding 2D image slice, mirroring the image directory structure (train/labels/, val/labels/).

3. Exploratory Data Analysis (EDA) and Visualization

Understanding the characteristics of the prepared dataset is critical for effective model training and debugging.

Notebook: `visualizing the dataset.ipynb`

Implementation Details:

- **Data Loading:** Sample 2D image slices and their corresponding YOLO-formatted annotation files (generated by parse data.ipynb) were loaded.
- **Visual Verification of Annotations:**
    - Selected 2D images were loaded, and their bounding box annotations were programmatically overlaid using matplotlib.patches.Rectangle or OpenCV.rectangle.
    - These visualizations confirmed the accuracy of the 3D-to-2D bounding box conversion process and the overall integrity of the parsed dataset.
- **Distribution Analysis:**
    - **Bounding Box Dimensions:** Histograms and scatter plots were generated to visualize the distributions of bounding box widths and heights across the dataset. This provided insights into the typical size and aspect ratios of flagellar motors, which, while not directly used for manual anchor selection in YOLOv8 due to its auto-anchor functionality, helped confirm the consistency of our 3D-to-2D projection strategy.
    - **Object Density:** The number of motors per tomogram and per 2D slice was analyzed to identify any imbalance in object distribution, which could influence sampling strategies during training.
    - **3D Motor Location Distribution:** Where possible, the original 3D (x, y, z) motor coordinates were visualized as a scatter plot to understand their spatial distribution within the full tomograms (e.g., clustering, proximity to edges, etc.).
    - **Image Intensity Analysis:** Histograms of pixel intensity values were examined to inform optimal normalization and augmentation strategies, particularly considering the inherent noise in cryo-ET data.
- **Anomaly Identification:** The EDA process systematically looked for:
    - Misaligned or incomplete bounding boxes.
    - Missing ground truth annotations.
    - Extremely noisy or low-contrast image slices that might pose challenges for detection.
    - Variations in motor appearance and morphology.
- **Informed Augmentation Strategy:** Insights from EDA directly influenced the selection of data augmentation techniques (e.g., aggressive contrast/brightness augmentation due to noise, random rotations due to variable motor orientations).

4. Model Training

This phase involved configuring and training the YOLOv8 model using the prepared 2D image and annotation data.

Notebook: `training yolov8.ipynb`

Implementation Details:

- **Environment Setup:** The ultralytics library (the official YOLOv8 implementation) and its dependencies, including PyTorch, were installed and imported.
- **Data Configuration:** A YAML configuration file was created to inform YOLOv8 about the dataset structure, specifying paths to training, validation, and test image/label directories, as well as the number of classes (nc: 1) and class names (names: ['flagellar_motor']).
- **Model Initialization:**
    - A pre-trained YOLOv8 model (e.g., yolov8n.pt for the nano version) was loaded using model = YOLO('yolov8n.pt'). This leverages transfer learning from large datasets like ImageNet/COCO, providing a strong starting point for feature extraction.
- **Data Augmentation:** YOLOv8's integrated train method automatically applies a comprehensive suite of data augmentations, critical for improving model robustness and generalization, especially with noisy cryo-ET data:
    - **Geometric Augmentations:** Random horizontal and vertical flips, rotations, scaling, translations, shear, and perspective transformations were used to expose the model to various orientations and positions of motors.
    - **Photometric Augmentations:** Adjustments to brightness, contrast, saturation, and hue were applied to mitigate the effects of variable imaging conditions and inherent noise in cryo-ET.
    - **Mosaic and MixUp:** These advanced techniques combined multiple images into a single training batch, enriching the context and variability of the training data.
- **Training Execution:**
    - The model was trained using the model.train() method with carefully selected hyperparameters:
        - data: Path to the dataset YAML configuration.
        - epochs: Number of training epochs (e.g., 100-300+).
        - imgsz: Input image resolution (e.g., 640x640 pixels).
        - batch: Batch size for training (e.g., 16-64).
        - patience: Early stopping parameter to prevent overfitting (e.g., 50 epochs without improvement).
        - optimizer: AdamW was typically chosen for its performance.
        - lr0 (initial learning rate) and lrf (final learning rate factor) were set, often with a cosine annealing schedule.
    - **Hyperparameter Tuning:** Iterative experimentation with these parameters was performed to optimize validation performance (measured by mAP, precision, recall).

- **Model Checkpointing:** YOLOv8 automatically saved model weights (last.pt, best.pt) at regular intervals and for the best performing epoch, allowing for training resumption and selection of the optimal model.
- **Validation:** Performance metrics were monitored on a separate validation set at the end of each epoch to track convergence and prevent overfitting.

5. Model Analysis and Understanding

This phase involved a deeper dive into the trained model's performance and decision-making process.

Notebook: `reverse engineering yolov8.ipynb`

Implementation Details:

- **Model Loading:** The best.pt model, determined during training, was loaded for evaluation.
- **Detailed Performance Metrics:**
  - The model was run on a dedicated validation or hold-out test set to generate comprehensive performance reports.
  - Precision-recall curves, mAP (mean Average Precision) at various IoU thresholds (mAP50, mAP75), and F1-scores were analyzed to understand the model's accuracy and trade-offs.
- **Visualizing Predictions:**
  - Predictions (bounding boxes with confidence scores) were overlaid on sample validation images. This direct visual inspection helped identify common false positives (where the model detected a motor that wasn't there) and false negatives (missed motors).
  - Analysis of these errors revealed potential issues such as high noise, low contrast, motor proximity, or unusual orientations as causes for misdetections.
- **Feature Map Visualization (Optional):**
  - Intermediate feature maps from the YOLOv8 backbone were accessed and visualized to understand what visual patterns the network was learning at different layers. This provided insights into how the model progressively extracts complex features from simple edges to higher-level object representations.
  - **Grad-CAM (Gradient-weighted Class Activation Maps):** If implemented, Grad-CAM was used to generate heatmaps showing which regions of the input image were most influential in the model's decision to detect a flagellar motor, offering interpretability of the black-box CNN.
- **Computational Performance:** The inference speed of the trained model on both CPU and GPU was benchmarked to assess its real-time applicability.

6. Submission Generation

The final step involved applying the trained model to the unseen test data and formatting the predictions according to the competition's specific requirements.

Notebook: `submission notebook.ipynb`

Implementation Details:

- **Model Loading:** The optimal best.pt model was loaded to ensure the highest performance on the test set.
- **Test Data Processing:**
    - The raw 3D .mrc files from the competition's test set were loaded sequentially.
    - For each 3D tomogram, 2D slices were generated using the exact same normalization and slicing methodology as in the parse data.ipynb for the training data.
- **Slice-wise Inference:**
    - The loaded YOLOv8 model performed inference on each generated 2D test slice.
    - Confidence thresholds (conf) and Non-Maximum Suppression (NMS) IoU thresholds (iou) were carefully tuned to balance precision and recall on the validation set, then applied during test inference.
- **2D Detection to 3D Point Conversion:**
    - For each 2D bounding box detection on a given slice:
        - The x_center and y_center of the 2D box directly corresponded to the x and y coordinates of the detected motor on that slice.
        - The Z-coordinate of the motor was assigned as the slice number itself.
        - The confidence score from the 2D detection was retained as the confidence for this initial 3D point.
        - This resulted in a collection of candidate 3D points (tomogram_id, x, y, z, confidence) for each tomogram.
- **3D Post-processing and Clustering:**
    - A critical step for refining the 2D-to-3D predictions was the implementation of a 3D clustering algorithm (e.g., DBSCAN or Mean Shift). This addressed the issue of a single 3D motor potentially generating multiple overlapping 2D detections across adjacent slices.
    - The clustering algorithm grouped nearby 3D candidate points that likely originated from the same physical flagellar motor.
    - For each identified cluster, the x, y, z coordinates were averaged to determine a single, refined 3D location for the motor. The confidence for this final 3D detection was typically the maximum or average confidence of the clustered 2D detections.
    - A final confidence threshold was applied to these aggregated 3D motor predictions to filter out low-confidence detections.
- **Submission Formatting:** The final 3D motor locations and their confidence scores were compiled into a pandas DataFrame with the columns specified by the competition (e.g., tomogram_id, x, y, z, confidence).

- **File Saving:** The DataFrame was saved as a .csv file, ready for submission to the Kaggle platform.

7. Consolidated Pipeline

For reproducibility and ease of use, all the aforementioned steps were consolidated into a single, streamlined notebook.

Notebook: `final merged notebook.ipynb`

Implementation Details:

- **Sequential Execution:** The notebook was structured to run all processing, training, and inference steps in a continuous flow, from data ingestion to submission file generation.
- **Modularity:** Key functions and configurations were often encapsulated in helper Python scripts (utils.py, config.py) and imported into the notebook, enhancing code readability and maintainability.
- **Configurability:** All important parameters (e.g., data paths, image dimensions, training epochs, detection thresholds) were made easily adjustable at the top of the notebook or via a separate configuration file.
- **Robustness:** Basic error handling mechanisms were integrated to ensure the stability of the pipeline.
- **Clear Outputs:** The notebook provided clear visual and textual outputs for each major stage, including training logs, validation metrics, sample prediction visualizations, and confirmation of submission file creation.
- **Environment Initialization:** A section was included to ensure all necessary libraries were installed, allowing for a quick setup on a new environment.

8. Conclusion

This project successfully developed and implemented a robust pipeline for detecting bacterial flagellar motors in 3D cryo-ET images using a 2D-to-3D adaptation of YOLOv8. The structured approach, encompassing expert-annotated data preparation, thorough EDA, optimized model training, and meticulous post-processing, allowed for effective handling of the complex volumetric data. The modular design of the notebooks ensures reproducibility and provides a clear framework for future improvements and adaptations to similar 3D object detection challenges.
