DATS 6303: Deep Learning
BYU - Locating Bacterial Flagellar Motors 2025
Final Group Project Proposal

Group Number: 2
Pranav Dhawan, Aakash Singh Sivaram, Abhinaysai Kamineni

Problem Statement

- **Objective:**
  Detect and localize bacterial flagellar motors (BFMs) in 3D electron microscopy (EM) images.

- **Significance:**
  BFMs are critical for bacterial motility, and accurate detection plays an important role in microbiological research and diagnostics.

- **Challenge:**
  EM data is high-resolution but suffers from low signal-to-noise ratios and subtle visual cues.

Dataset

- **Source:**
  Kaggle competition: *Locating Bacterial Flagellar Motors (2025)*

- **Contents:**
  - 3D tomographic EM data (as .npy files) representing image volumes.
  - Corresponding motor locations for training volumes in JSON format.

- **Properties:**
  - High volume and detailed labels
  - Adequate for deep learning with augmentation strategies

- **Preprocessing Plan:**
    - Normalize volumes
    - Apply data augmentation (flipping, rotation, scaling)
    - Convert 3D to 2D slices for YOLO-based pipelines (if needed)

Approach

- **Model Architectures to Explore:**
    - **3D Convolutional Neural Networks (CNNs)**
        - Suitable for volumetric data and voxel-wise classification
    - **YOLO (You Only Look Once) with SIAH**
        - For fast object detection in 2D/3D projected slices
    - **MMDetection Framework**
        - Modular object detection pipelines using pre trained backbones
    - **Hybrid/Attention-based Models**
        - Depending on baseline results and complexity needs
- **Frameworks & Tools:**
    - **PyTorch**: Core deep learning framework
    - **YOLOv5/v8**: Real-time detection with bounding boxes
    - **MMDetection**: Modular experimentation with object detectors
    - **Custom Scripts**: For 3D CNN training and preprocessing workflows

---

Evaluation Metrics

Submissions will be evaluated using a combination of the $\mathbf{F}_\beta$-score and Euclidean distance. The goal is to determine whether a tomogram contains a motor and, if it does, to accurately predict its location.

Let the ground truth be y and the predicted location be $\bar{y}$. The Euclidean distance $|y - \bar{y}|_2$ determines classification:

- **True Positive (TP):** If $|y - \bar{y}|_2 \leq \tau$, the prediction is within threshold.

- **False Negative (FN):** If $|y - \bar{y}|_2 > \tau$, the prediction is outside of threshold.

where $\tau = 1000$ Angstroms.

$\mathbf{F}_\beta$-score

The $\mathbf{F}_\beta$-score balances precision and recall, placing greater weight on recall when $\beta > 1$ and on precision when $\beta < 1$ (in our case we use $\beta = 2$, thus we are weighting recall more than precision). It is defined as:

$$\mathbf{F}_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}} = (1 + \beta^2) \cdot \frac{\text{TP}}{(1+\beta^2) \cdot \text{TP} + \beta^2 \cdot \text{FN} + \text{FP}}$$

This metric ensures that both the presence and location accuracy of predicted motors are considered in the final score.

Why this metric matters

This composite metric ensures the model is judged not only on its ability to detect the **presence** of motors, but also on its **spatial accuracy** in predicting motor locations within biological tolerance.

Expected Challenges & Risks

- **Data Challenges:**
    - Low signal-to-noise ratio.
    - Class imbalance between presence/absence of motors.
- **Model Challenges:**
    - Overfitting to small visual cues.

    - High computational cost of 3D CNNs

- **Mitigation:**
    - Data augmentation and noise reduction.
    - Use of pretrained models and fine-tuning.
    - Batch-wise memory optimization for 3D inputs.

References

- Official documentation for PyTorch, MMDetection, and YOLO
- Relevant biomedical imaging research papers on paperswithcode.com
- Open-source GitHub repositories for baseline models.
- Prior Kaggle notebooks and winning solution.

Rough Schedule

Week	Task
Week 1	Understand dataset, preprocessing pipeline setup
Week 2	Implement baseline 3D CNN, integrate with YOLO pipeline. Experiment with MMDetection and SIAH-enhanced detection.
Week 3	Model tuning, cross-validation, and ensemble tests. Results visualisation and error analysis.
Week 4	Final documentation, report, and GitHub repo preparation