Locating Bacterial Flagellar Motors in 3D Tomograms

50098, 48397, 37386, 39375

Abstract

Accurate detection of subcellular structures such as bacterial flagellar motors in tomographic images are critical for advancing our understanding of microbial machinery. Inspired by the BYU Kaggle competition, we attempt to address the challenge of detecting motors in cryo-electron tomograms using deep learning methods on a curated dataset of labeled tomograms before turning to state-of-the-art object detection models such as YOLO and RT-DETR. This report compares the performance of our attempt against these SOTA models using the mean euclidean distance and compares the performance between YOLOv8n, YOLOV8m, YOLOV11m and RT-DETR using the F-2 score. The results demonstrate RT-DETR's superior performance at locating motors.

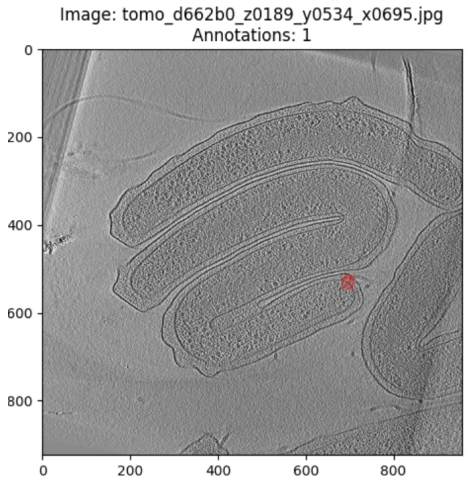

Figure 1. Slice of a Tomogram with a flagellar motor

1. Introduction

The goal of our project is to develop a solution to identify the presence and location of flagellar motors in 3D reconstructions of bacteria. By automating this traditionally manual task, we aim to accelerate the study of macromolecular complexes, which helps answer fundamental questions in molecular biology, improve drug development, and advance synthetic biology.

The flagellar motor is a molecular machine that facilitates the motility of many microorganisms, playing a key role in processes ranging from chemotaxis to pathogenesis. Cryogenic electron tomography (cryo-ET) has enabled scientists to image these nanomachines in near-native conditions. Identifying flagellar motors in these three-dimensional reconstructions (tomograms) is labor intensive. Factors such as a low signal-to-noise ratio, variable motor orientations, and the complexity of crowded intracellular environments complicate automated identification. Cryo-ET studies become limited by the bottle-neck of a human in the loop.

A tomogram is a three-dimensional image that has been reconstructed from a series of 2D projection images. The images are tomograms of bacteria that have been flash-frozen in ice, which preserves the molecular structure of the bacteria for the imaging process.

The first attempt at this project was to focus only on slices with motors to pose a pure regression problem to locate the center of the flagellar motor with x and y coordinates. We then tried using state-of-the-art object detection deep learning models, such as the You Only Look Once (YOLO) and Detection Transformer(DETR) models.

The motivation for this research project comes from the Kaggle Competition.

2. Related Work

Region-based Convolutional Neural Networks (R-CNN) (Girshick et al., 2014) established the foundation for contemporary methods in objection detection. The R-CNN object detection framework consists of three key stages. First, it generates region proposals using selective search, producing a set of candidate object regions per image. These regions are then warped and fed into a CNN to extract high-dimensional feature vectors encoding each proposal's visual information. Finally, the features are classified using support vector machines (SVMs) to identify objects, while a separate bounding box regressor refines their spatial coordinates for precise localization. While effective, this multi-stage pipeline is computationally expensive, which

motivated later improvements like Fast R-CNN (Girshick, 2015) and Faster R-CNN (Ren et al., 2015).

While R-CNN-based models continued to progress, You Only Look Once (YOLO)(Redmon et al., 2016) introduced a fundamentally different approach to object detection by framing it as a single regression problem. YOLO enabled real-time detection by directly predicting bounding boxes and class probabilities from the entire image in one evaluation, offering a significant speed advantage over traditional two-stage methods. Subsequent versions brought notable improvements, focusing on optimizing training techniques, data augmentation, and model scaling strategies. The most recent advancements, seen in YOLOv8 and YOLOv11, further refined accuracy and efficiency, with YOLOv8n offering a lightweight model variant optimized for fast inference while maintaining competitive detection performance. In medical imaging, these advancements are particularly impactful, as evidenced by Ju et al.'s (2023) application of YOLOv8 to pediatric wrist fracture detection, where the model's efficiency and precision address clinical diagnostic challenges.

Aside from the architectural innovations mentioned above, researchers also focused on addressing the unique challenges posed by small object detection. A comprehensive review by Wei et al. (2024) examines deep learning-based methods for small object detection, identifying four major obstacles: (1) the loss of spatial information in deep networks; (2) limited discriminative features in small objects; (3) the scarcity of positive samples due to anchor mismatches; and (4) data scarcity. These obstacles identified future directions, such as improving noise reduction during feature fusion and utilizing diffusion models for feature enhancement, by refining end-to-end object detectors.

The Transformer-based end-to-end object detector, Detection with Transformers (DETR) models (Carion et al. 2020), proposes a solution by removing the post-processing (Non-Maximum Suppression) and, instead, employs bipartite matching to directly predict the one-to-one object set. The limitations of DETR (slow training convergence, high computational costs and hard-to-optimize queries) motivated the architecture for Deformable-DETR (Zhu et al., 2020), DAB-DETR (Zhao et al., 2023), DN-DETR (Li et al., 2022), Group-DETR (Chen et al., 2022), Efficient-DETR (Yao et al.,2021), Sparse-DETR (Roh et al., 2021), Lite DETR (Li et al.,2023), Conditional DETR (Meng et al., 2021) and Anchor DETR (Wang et al., 2022). These predecessors, however, remain computationally expensive and fail to detect in real time. The Real-Time Detection Transformer (RT-DETR) (Zhao et al. 2023), which yields the best results for our task, addresses the high computational cost and further optimizes query initialization.

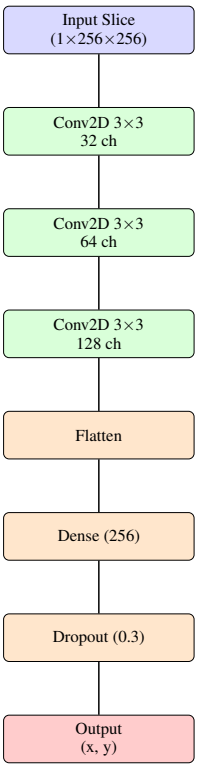

Figure 2. Figure: Our Customized Model Architecture

3. Model Architecture

Customized Model

In our custom model, implemented using PyTorch, we performed hyperparameter tuning to optimize the performance. We experimented with several configurations, such as adjusting the number of convolutional layers, which were each followed by a ReLU activation and a MaxPooling2D operation, as well as adjusting for the size of the dense layer, the dropout rate, learning rate, and choice of optimizer. After testing various combinations, the best-performing model consisted of 3 convolutional layers, a fully connected layer with 256 neurons, and a learning rate of 0.0001 as well as the Adam optimizer.

Basic YOLO Model

YOLO treats object detection as a single regression problem, straight from the image pixels to bounding boxes and class probabilities. Instead of doing region proposals, it just looks at the image once — hence the name.

First, it splits the image into an $S \times S$ *grid*, where each grid cell is in charge of detecting objects whose center falls inside it. For each of those cells, YOLO predicts the following:

1. B bounding boxes — the x and y coordinates for the center of the box and the width and height of the box.

2. A confidence score for each box, calculated as the the probability of an object times the Intersect over Union (IoU) between the predicted box and the ground truth.

3. A set of predicted class probabilities.

The YOLO neural network architecture is built on a backbone, neck, and head. The neck acquires the features extracted from the backbone model and the head predicts the classes and the bounding boxes, which is the final output produced by the model before Non-Maximum Suppression (NMS). NMS is a post-processing step in YOLO that removes redundant bounding boxes by selecting the highest-confidence box and suppressing others that have a high overlap (IoU) with it. While NMS helps reduce duplicate detections and simplifies outputs, it is a fixed and sensitive non-learnable algorithm, which limits its adaptability (Zhao et al. 2023). Inappropriate confidence thresholds can lead to significant false positives or false negatives by the detector.

Alternative YOLO Architectures

We worked with YOLOv8 and also experimented with YOLOv11. Each backbone is built using a combination of different neural network blocks. The four most common blocks are the convolutional block, the concatenate-to-fuse (C2f) block, the bottleneck block, and the spatial pyramid pooling - fast (SPPF) block. The most common of these blocks is the convolutional block, which consists of a 2-d convolutional layer, a 2D batch normalization, and a SiLU activation function[1]. The C2f block splits the input into two paths: one goes through a few convolutional layers (often bottleneck[2] -like), and the other skips them. Then, it concatenates both paths and applies another convolutional block to fuse the features. This fusion helps preserve spatial information while still allowing deep feature extraction.

The SPPF block is a streamlined version of the regular SPP block. The idea is to use fewer parameters to take a feature map and apply multiple max pooling operations with different kernel sizes to capture information at different scales. This allows the model to "see" the object at multiple receptive fields, helping with scale invariance. The pooled outputs are then concatenated and passed through a convolution. See *Figure 3* for a visualization of the YOLOv8 architecture.

The latest YOLOv11 model introduces: (1) C3K2 blocks using parallel 3×3 convolutions in a Cross Stage Partial (CSP) structure for efficient computation, and (2) Cross-Stage Partial Self-Attention (C2PSA) blocks with dual-branch spatial attention for occluded objects. Given the noise within cryo-ET tomograms and the motors being small, the YOLOv11's spatial focus should improve detection.

[1] Sigmoid Linear Unit function

[2] A bottleneck block usually refers to a structure where the input features are first reduced in dimensionality, then processed, and finally expanded again — similar to the structure of ResNet blocks, which often include skip connections and a 1x1 $\rightarrow$ 3x3 $\rightarrow$ 1x1 convolution pattern.

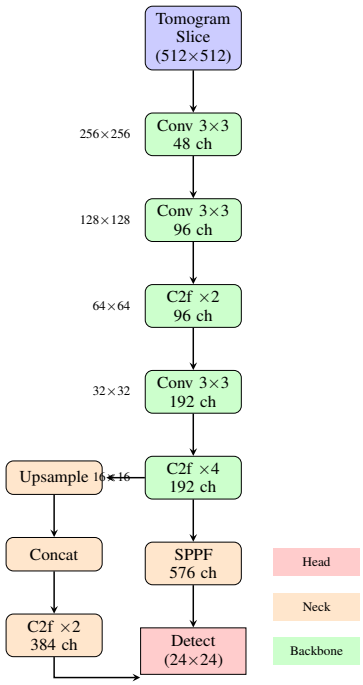

Figure 3. YOLOv8 architecture with optimized connections. The neck pathway features a shorter horizontal connection (arrow from C2f to Upsample) while maintaining the complete processing pipeline from 512×512 input slices to 24×24 motor detections. Resolution markers show spatial downsampling at each stage.

YOLO Model Sizes

For our bacterial motor detection task, we focused on comparing nano (n) and medium (m) models, which differ primarily in scaling factors that adjust their depth and width:

- **Scaling Parameters:**
  - Depth (layer count): YOLOv8m is twice as deep as YOLOv8n (0.67 vs. 0.33 scaling) [3].
  - Width (channel count): YOLOv8m has 3× wider layers (0.75 vs. 0.25 scaling)

The YOLOv8m's larger capacity (25.9M parameters), compared to YOLOv8n (3.2M parameters), enables richer feature extractions through better handling of scale variations and partial occlusions. This is achieved by implementing deeper C2f blocks and SPPF layers in the backbone and

[3] The depth scaling parameter multiplies the depth of the model by the value of the scaling parameter

neck and makes the medium variant outperform it's nano counterpart for our bacterial motor detection task.

DETR

DETR frames object detection as a set prediction problem, which outputs bounding boxes and class labels in parallel. DETR uses a CNN backbone to extract image features, which are flattened and fed into a transformer encoder-decoder. The encoder refines features with self-attention for global context, while the decoder uses learned object queries (positional embeddings) to highlight these features and directly predict bounding boxes and classes. This eliminates the need for anchor boxes and NMS. Predictions are matched to ground truth via the Hungarian algorithm (Kuhn, 1955 & Munkres, 1957), ensuring one-to-one assignments without duplicates. This approach simplifies detection by replacing hand-crafted components with a unified set prediction framework.

DETR's transformer-based attention mechanisms dynamically focus on the most informative parts of the image, independent of scale or predefined anchor shapes. This makes it inherently more flexible and effective for detecting non-uniform or deformable biological features, whose appearance may change across slices or vary between tomograms. Furthermore, the model tends to exhibit stronger generalization in data-limited settings due to their ability to reason over the entire image. Such contextual awareness is particularly beneficial for cryo-ET, where the imaging signal is often sparse and noisy, and the structures of interests can be small, variable, and embedded in cluttered environments.

Despite these advancements, the original DETR model presented significant limitations as defined above in the related work section.

RT-DETR

RT-DETR (Real-Time Detection Transformer) presents a solution to the high computational cost and hard to optimize queries by introducing efficient hybrid encoders and uncertainty-minimal query selection.

The hybrid encoder replaces DETR's pure transformer encoder with a combination of CNN and transformer layers. A CNN backbone first extracts hierarchical features, which are then refined by attention-based intra-scale feature interaction (AIFI) and CNN-based cross-scale feature fusion (CCFF). The CCFF is made up of RepConvs (Ding et al., 2021) which are used for feature fusion. The AIFI approach reduces computational costs by applying the intra-scale feature interaction selectively to only the final layer of the backbone and improves detection by removing the risk of confusing the encoder with lower level features.

The second major improvement is uncertainty-minimal query selection, which replaces DETR's randomly initialized object queries. Instead of learning queries from scratch, RT-DETR selects high-confidence encoder features as queries by combining IoU and classification scores. This ensures the model focuses on reliable regions and dynamically adjusts the number of queries per image. As a result, training converges much faster, and fewer computations are wasted on low-quality predictions.

Together, these enhancements allow RT-DETR to achieve real-time speeds while maintaining DETR's end-to-end advantages. This makes RT-DETR a practical, high-performance alternative to traditional CNN-based detectors like YOLO. See *Figure 4* for a visual of the RT-DETR model architecture.

4. Data Preparation

Our preprocessing pipeline transformed 3D tomographic data into a format suitable for training 2D object detection models. The pipeline consists of the following steps.

Slice Extraction

For each annotated motor location at coordinates (z_c, y_c, x_c), we extract a series of 2D slices along the z-axis:

$$\{z \mid z_c - \tau \leq z \leq z_c + \tau\} \tag{1}$$

where $\tau = 4$ defines the neighborhood size, yielding $2\tau + 1 = 9$ slices per motor. This accounts for potential axial displacement of motors while maintaining sufficient context.

Image Normalization

Each slice undergoes percentile-based intensity normalization:

$$I_{\text{norm}} = 255 \times \frac{\text{clip}(I, p_2, p_{98}) - p_2}{p_{98} - p_2} \tag{2}$$

where p_2 and p_{98} are the 2nd and 98th percentiles of slice intensities, respectively.

Dataset Construction

We organize the data into YOLO format with:

- Train/validation split (80%/20%) at the tomogram level to prevent data leakage
- Square bounding boxes of size 24 pixels centered on each motor
- Normalized annotations ($[0, 1]$ range) following YOLO format: $(x_c/w, y_c/h, \text{box\_w}/w, \text{box\_h}/h)$

The final dataset contains 4,054 slices (3,262 train, 792 validation) extracted from 362 tomograms, covering 451 unique motor instances.

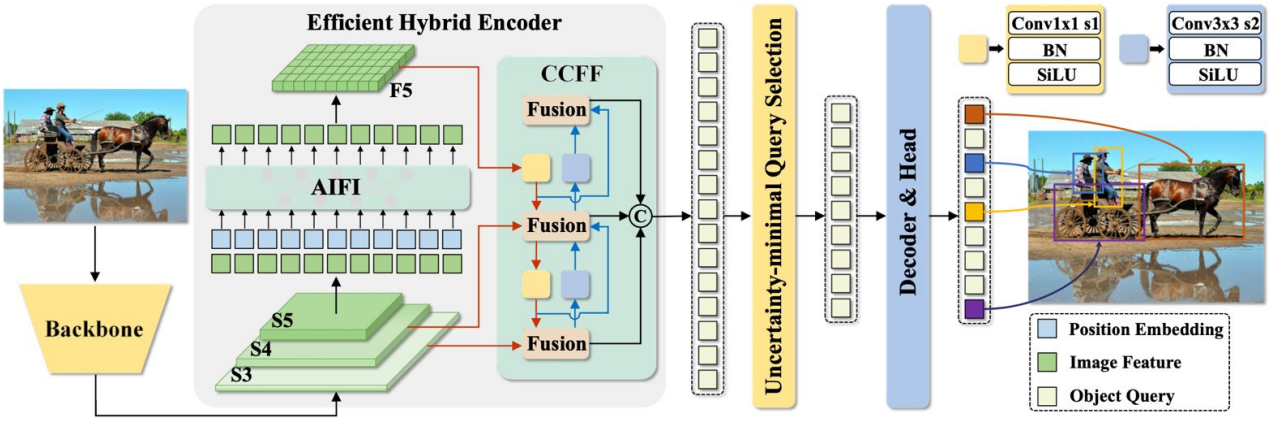

Figure 4. RT-DETR model architecture (Zhao et al., 2023)

Data Augmentation

Data augmentation was employed to enhance the model's ability to detect flagellar motors in bacterial tomograms under diverse conditions. Augmentations included random rotations up to 180 degrees, horizontal and vertical flips (each with a 50% probability), perspective transformations (0.001 scale), image translation (20% of image size), shear transformations (2.0 intensity), and mixup augmentations (25% blend ratio). These transformations simulate realistic variations in cryo-ET data, accounting for different motor orientations, partial occlusions, and spatial distortions that may occur during tomogram acquisition.

Each motor-containing slice underwent these transformations during training, which increased the dataset's diversity. The bounding box annotations were automatically adjusted to remain accurate through all geometric transformations. This comprehensive augmentation approach not only prevented overfitting to the limited training data but also improved the model's robustness to orientation changes, spatial distortions, and other natural variations encountered in cryo-ET datasets.

5. Training

Loss Functions

The YOLO models focus on three different loss functions: box loss, classification loss (CLS) and distributed focal loss (DFL).

The box loss represents how well the model's predicted boxes align with the objects (motors) in our images. The box loss in calculate by applying a smooth L1 loss function to optimise the bounding box regression. The discrepancy between the predicted bounding boxes and the ground truth bounding boxes provides a critical measure of localization accuracy. This discrepancy is evaluated using metrics such as *Intersection over Union* (IoU). IoU is computed by taking the area of overlap between the predicted bounding box and the ground truth bounding box and dividing it by the area of their union. Mathematically, it is defined as:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

An IoU value closer to 1 indicates a higher degree of overlap and thus a more accurate prediction, whereas lower IoU values suggest greater localization errors. By optimizing based on IoU, the model iteratively adjusts its predictions to improve the precision of object localization. The default threshold of 0.7 IoU will indicate a true positive in terms of finding a bounding box on the motor.

RT-DETR uses the generalized IoU, which penalizes non-overlapping boxes by considering how far apart they are spatially. This loss is defined as:

$$\text{GIoU} = \text{IoU} - \frac{|C - A \cup B|}{|C|}$$

where C is the smallest box that encloses A (the predicted box) and B (the true box).

Class (CLS) loss focuses on classification accuracy by measuring how well the model predicts the correct object class for each detected instance. In the context of this project, where the task is to detect the presence or absence of a flagellar motor in tomograms, the CLS loss evaluates the model's ability to correctly assign a class of "motor" to each predicted bounding box. This is specifically important in scenarios with a high class imbalance, as false positives (predicting a motor where there is none) can significantly degrade the model's practical utility. CLS loss uses a

cross-entropy formulation, penalizing incorrect classifications more severely and reinforcing learning from correctly labeled examples. In a single-class detection problem such as this one, the CLS loss helps the model distinguish signal from noise, ensuring that the presence of motor structures are accurately flagged during inference.

The distributed focal loss (DFL) is responsible for refining the model's ability to predict precise bounding box coordinates. Unlike traditional regression losses, which may struggle with discretized predictions. DFL poses the bounding box localization task as a classification problem over discrete bins and uses a soft distribution to improve accuracy.

RT-DETR also uses the L1 loss (mean absolute error). This loss takes the sum of the absolute differences across each of the box metrics (x,y,box width, box height). The loss function takes the form:

$$\mathcal{L}_{\text{L1}} = \sum_{i=1}^{4} |predictions_i - truth_i|$$

Training Parameters

All models were run with the following model parameters:

- **Training Parameters:**
  - Batch Size: 16.
  - Epochs: 100
  - Optimizer: AdamW [4] with a learning rate of 0.002 and momentum of 0.9)

6. Results

Model	Box Loss	CLS Loss	DFL Loss
YOLOv8n	1.683	0.962	0.866
YOLOv8m	1.600	**0.840**	0.851
YOLOv11m	**1.501**	0.854	0.851

Table 1. YOLO model comparisons of loss function metrics

Model	Precision	Recall	F-2	mAP50
YOLOv8n	0.857	0.871	0.868	0.901
YOLOv8m	0.878	0.896	0.892	0.926
YOLOv11m	0.834	0.875	0.867	0.914
RT-DETR	**0.880**	**0.938**	**0.926**	**0.946**

Table 2. Evaluation metrics for YOLO variants and RT-DETR.

[4]AdamW is a variation of the Adam optimization algorithm. AdamW decouples weight decay from the gradient update, leading to a more robust and effective regularization process.

Evaluation Metrics

To measure the loss and accuracy of our customized model - we use the Mean Squared Error (MSE). This metric is calculated for both the predicted x- and y-coordinates of the motor. The formula for MSE is given by:

$$\text{MSE}_x = \frac{1}{n}\sum_{i=1}^{n}(\hat{x}_i - x_i)^2 \quad \text{and} \quad \text{MSE}_y = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

where:

- n is the number of validation samples,
- x_i, y_i are the ground truth coordinates,
- $\hat{x}_i, \hat{y}_i$ are the predicted coordinates.

To compare our customized model to the YOLO and RT-DETR models, we calculate the Euclidean distance in pixels between predicted and ground truth motor coordinates.

$$D = \sqrt{(x_{pred} - x_{true})^2 + (y_{pred} - y_{true})^2} \tag{3}$$

This pixel-wise metric provides a physically interpretable measure of localization accuracy that is consistent across all architectures, enabling direct comparison between our CNN-based regressor and the detection-focused YOLO/RT-DETR approaches.

Model	Euclidean Distance (pixels)
YOLOv8n	27.88
YOLOv8m	28.36
YOLOv11	26.61
RT-DETR	22.54
Customized Model	59.28

Table 3. Euclidean distance comparison between different object detection models

The state-of-the-art models are evaluated using the F_β score, calculated from the precision and recall of each model. Precision is defined as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

where true positives in our work are defined as when a motor was identified within the IoU threshold of 0.7, and the motor actually exists in the given slice.

And Recall is:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

where false negatives are when no motor/bounding box was identified, but a motor actually exists in the given slice.

Since we care more about mitigating false negatives, we use $\beta = 2$, to have the F_β score weigh recall more than precision. The F-2 score is calculated as:

$$F_\beta = \frac{(1+\beta^2) \cdot \text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}}.$$

We also use the Mean Average Precision (mAP) which calculates the average precision[5] values across multiple object classes. In the case of single-class object detection, the mAP50 is the average precision with a 0.5 confidence threshold.

Customized Model

At the end of training, the model did not generalize well to our dataset. This was likely due to the limited complexity of the architecture—the feature extraction layers were not deep or expressive enough to capture the fine details required to accurately locate small objects like bacterial motors. While this architecture might perform better on larger, more prominent objects, it struggles with subtle patterns. We also experimented with integrating attention mechanisms at the end of the architecture to help the model focus on small, critical features. This led to an increase in validation loss, suggesting that the attention layers may have introduced overfitting or were not appropriately placed in the network. See *Table 3* for the Euclidean distance comparisons between the customized model and the SOTA models.

YOLOv8n vs. YOLOv8m vs. YOLOv11m

Based on the loss values reported in *Table 1* and the evaluation metrics in *Table 2* , YOLOv8m consistently outperformed both YOLOv8n and YOLOv11m. This superior performance suggests that the larger parameter count of YOLOv8m enables more expressive feature representations and more stable learning, while avoiding the overfitting tendencies observed in YOLOv11m. The enhanced CSP-Darknet backbone in YOLOv8m contributes to better preservation of spatial information through strategic feature map concatenations, effectively balancing network width and depth.

In contrast, YOLOv11m, despite being a more recent architecture with a substantial number of parameters (20.1 million), appears to be less optimized for the dataset used in this study. The loss metrics indicate signs of overfitting, likely due to the RepVGG-style reparameterization, which can cause the model to converge prematurely to local minima. Although YOLOv11m achieves a lower training loss, its evaluation metrics are inferior to those of YOLOv8m, reflecting a poorer generalization performance.

YOLOV8m outperformed all other YOLO models due to its ability to efficiently balance complexity and accuracy, showing that specialization in feature extraction is crucial for our task. While we didn't have high expectations for YOLOV8n, as it is a lighter model, we quickly realized that its reduced complexity wasn't optimal for this task. Nonetheless, YOLOv8n still offers a computationally efficient way to detect these small objects with comparable results.

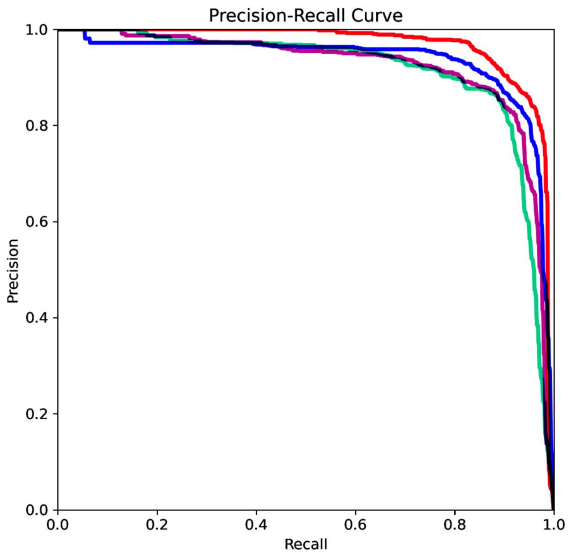

Figure 5. Precision-Recall Curves for all models:RT-DETR, YOLOv8m, YOLOv8n, YOLOv11m

RT-DETR

To complete the model comparison, we implemented an RT-DETR model, which leverages a different set of loss functions for training optimization, including Generalized Intersection over Union (GIoU) loss, classification (CLS) loss, and localization regression (L1) loss. The loss function metrics for this model are outlined in *Table 4*. RT-DETR outperforms the YOLO models in key evaluation metrics, as illustrated in *Table 2*.

As evidenced by our numerical results above, we conclude that the RT-DETR model excelled at small object detection. The RT-DETR was able to effectively utilize the attention mechanisms within the hybrid encoders to help it detect motors better than YOLO. Furthermore, RT-DETR's uncertainty-minimal query selection pushes the model to focus on reliable regions where motors might exist with higher confidence. This explains RT-DETR's higher preci-

[5] Average Precision computes the area under the precision-recall curve, providing a single value that encapsulates the model's precision and recall performance.

sion across all levels of recall, as displayed in *Figure 5*.

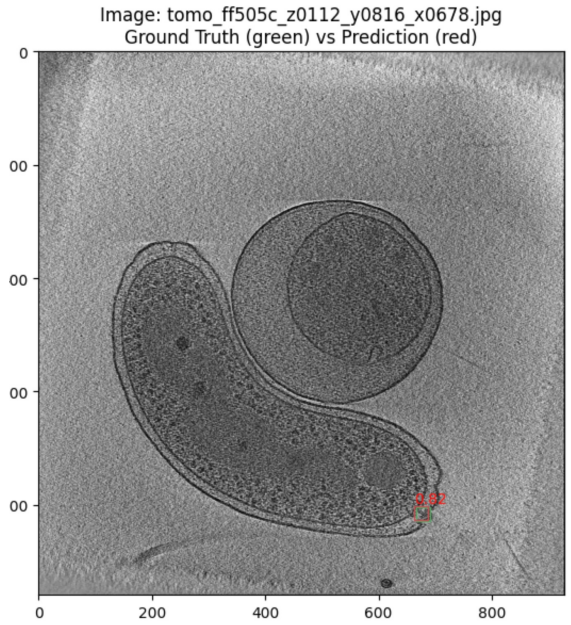

Figure 6. An example of the prediction vs. ground truth boxes from the RT-DETR model

Model	GIoU Loss	CLS Loss	L1 Losss
RT-DETR	0.434	0.447	0.033

Table 4. Euclidean distance comparison between different object detection models

Conclusion

We attempt to pose the task as a pure regression on positive examples only using CNNs feature extractors and a combination of different mechanisms to no avail. The models we designed failed to correctly predict bounding boxes and produced results biased towards the centre of images due to insufficient complexity of feature extraction. We turn to more intricate SOTA models and find that the RT-DETR effectively encodes high-level features and fuses lower-level features to produce high quality queries for the decoder, allowing it to excel in the detection of flagellar motors compared to YOLOv8 and YOLOv11 models. By using 2D slices of 3D objects to pose the problem in a lower-dimension, we leave unexplored the solutions possible with 3D object detectors. Our work supports the rising performance of DETR models, and motivates further research into it's applications in real-time small object detection.

Statement about individual contributions

50098 - Researched end-to-end object detectors and implemented RT-DETR.

37386 - Research and implemented YOLOv8 models.

39375 - Researched and implemented YOLOv11 models.

48397 - Researched and implemented customized model.

Equal contributions for data parsing and report writing.

References

1. Carion N, Massa F, Synnaeve G, Usunier N, Kirillov A, Zagoruyko S. End-to-End Object Detection with Transformers [Online]. *arXiv*; 2020. Available from: https://arxiv.org/abs/2005.12872

2. Chen Q, Chen X, Zeng G, Wang J. Group DETR: fast training convergence with decoupled one-to-many label assignment [Online]. arXiv; 2022. Available from: https://arxiv.org/abs/2207.13085

4. Ding X, Zhang X, Ma N, Han J, Ding G, Sun J. RepVGG: Making VGG-style convnets great again [Online]. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2021. Available from:https://openaccess.thecvf.com/content/CVPR2021/html/Ding_RepVGG_Making_VGG-Style_ConvNets_Great_Again_CVPR_2021_paper.html

5. Girshick R, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv*. 2014. Available from: https://arxiv.org/abs/1311.2524v5

6. Girshick R. Fast R-CNN. *arXiv*. 2015. Available from: https://arxiv.org/pdf/1504.08083

7. Ju Rui-Yang, Cai Weiming. Fracture Detection in Pediatric Wrist Trauma X-ray Images Using YOLOv8 Algorithm. *arXiv*. 2023. Available from: https://arxiv.org/pdf/2304.05071

8. Kuhn HW. The Hungarian method for the assignment problem. Naval Res Logist Quart [Online]. 1955 Available from: https://onlinelibrary.wiley.com/doi/10.1002/nav.3800020109

9. Li F, Zhang H, Liu S, Guo J, Ni LM, Zhang L. DN-DETR: accelerate DETR training by introducing query denoising [Online]. arXiv; 2022. Available from: https://arxiv.org/abs/2203.01305

10. Li F, Zeng A, Liu S, Zhang H, Li H, Zhang L, Ni LM. Lite DETR: an interleaved multi-scale encoder for efficient DETR [Online]. In: Proceedings of the IEEE/CVF

Conference on Computer Vision and Pattern Recognition (CVPR); 2023. Available from: https://arxiv.org/abs/2303.07335

11. Meng D, Chen X, Fan Z, Zeng G, Li H, Yuan Y, Sun L, Wang J. Conditional DETR for fast training convergence [Online]. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV); 2021. Available from: https://arxiv.org/abs/2108.06152

12. Munkres J. Algorithms for the assignment and transportation problems [Online]. J Soc Ind Appl Math. 1957. Available from: https://epubs.siam.org/doi/10.1137/0105003

13. Redmon J, Divvala S, Girshick R, Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. *arXiv*. 2015. Available from: https://arxiv.org/pdf/1506.02640

14. Ren S, He K, Girshick R, Sun J. Faster R-CNN: Towards real-time object detection with region proposal networks. *arXiv*. 2015. Available from: https://arxiv.org/pdf/1506.01497

15. Ren Z, Hu Y, Cheng J, Sadeghi I, Subramaniam S, Liu X. Automatic Detection and Segmentation of Bacteria in Cryo-Electron Tomograms Using Deep Learning. *arXiv*. 2023. Available from: https://arxiv.org/abs/2304.05071

16. Roh B, Shin JW, Shin W, Kim S. Sparse DETR: efficient end-to-end object detection with learnable sparsity [Online]. In: International Conference on Learning Representations (ICLR); 2021. Available from: https://arxiv.org/abs/2111.14330

17. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need [Online]. *arXiv*. 2017. Available from: https://arxiv.org/pdf/1706.03762

18. Wang Y, Zhang X, Yang T, Sun J. Anchor DETR: query design for transformer-based detector [Online]. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI); 2022. Available from: https://ojs.aaai.org/index.php/AAAI/article/view/20163

19. Wei Z, Huang H, Liang X, Lu Y, Wang H. A review of small object detection based on deep learning. *Springer*. 2024. Available from: https://www.springerprofessional.de/en/a-review-of-small-object-detection-based-on-deep-learning/26744794

20. Wong KY. YOLOv9 [Online]. GitHub; 2023. Available from: https://github.com/WongKinYiu/yolov9

21. Yao Z, Ai J, Li B, Zhang C. Efficient DETR: improving end-to-end object detector with dense prior [Online]. arXiv; 2021. Available from: https://arxiv.org/abs/2104.01318

22. Zhao H, Zhang Y, Liu S, Chen Z, Yang J, Liu Y. RT-DETR: real-time detection transformer with efficient hybrid encoder [Online]. arXiv; 2023. Available from: https://arxiv.org/abs/2201.12329

23. Zhu X, Su W, Lu L, Li B, Wang X, Dai J. Deformable DETR: deformable transformers for end-to-end object detection [Online]. In: International Conference on Learning Representations (ICLR); 2020. Available from: https://arxiv.org/abs/2010.04159