Probe Detection with Deep Learning

Sviatoslav Voloshyn

September 19, 2025

1 Introduction

This project implements and evaluates deep learning models for detecting ultrasonic thickness probes mounted on the Elios3 drone. The task is to predict a bounding box for the probe (if present) or report no detection. Robustness to challenging backgrounds, illumination, and object size variation is emphasized due to the industrial inspection setting.

2 Dataset Analysis

The provided dataset contains 308 images, each annotated with a single bounding box corresponding to an ultrasonic probe mounted on the Elios3 drone.

Structure and Naming

- Filenames follow the format: `[MISSION]_[ORBIT]_[PASS]_1flight_[TIME]_[VARIANT].jpg`.
- 5 distinct missions were collected on different days and locations.
- Each mission contains 2–9 orbit-pass segments. The `TIME` field increments in 300, 600, or 900 s steps, meaning little to no overlap between consecutive frames.
- The final field, `VARIANT`, encodes probe mounting:
  - 0: probe mounted at bottom (96.6% of boxes in lower half).
  - 2: probe mounted at top (100% of boxes in upper half).

Geometric and Visual Properties

- Probes always touch at least one image border (due to the arm connected to the drone).
- Average relative size $\approx 8\%$ of the image area (range $\approx$ 3–25%).
- Aspect ratio ≈ 1.5 (taller than wide); wider-than-tall probes are rare.
- High blur is common; motion blur especially problematic in some missions.
- Lighting varies strongly, with some missions dominated by glare or low illumination.
- Backgrounds differ substantially, from uniform surfaces to dense pipe and tube clutter.

Labeling and Priors

- Bounding boxes consistently include the full probe and connecting arm, explaining the border-touching property.
- The probe's vertical position is highly correlated with the variant field (top vs. bottom), creating a strong positional prior.
- The arm configuration is stable within an orbit when probe is not attached; changes in probe appearance are mostly due to lighting and some frames when the probe is in contact with the surface and then it can change position and orientation.

Implications for Modeling

- **Edge-touching prior:** must be preserved during augmentation (no random translations that remove edge contact).
- **Small-object detection:** high resolution is beneficial; anchor tuning may be required for 8% area targets.
- **Blur/lighting robustness:** augmentations such as motion blur and photometric adjustments are essential.
- **Variant bias:** random vertical relocation of the probe would break the positional prior and harm performance.
- **Label noise:** IoU metrics may penalize predictions that focus on probe tip while ignoring arm; evaluation should account for this.

3 Evaluation Strategy

A central question in this project is how to define and measure "success". Because the dataset consists of distinct missions, each with its own background and visual conditions, the evaluation must emphasize generalization rather than memorization. For this reason, I adopted a **leave-one-mission-out cross-validation** (LOMO-CV) scheme: at each fold, one mission is held out entirely for validation while the model is trained on the others. This ensures that validation performance reflects true robustness to unseen conditions rather than leakage of mission-specific artifacts.

Mission-level variability. The five missions differ markedly, which makes per-mission evaluation critical:

- **E300PREMP00002 (Fold 1):** large probe on a uniform background. Easy case overall, though illumination varies between frames.
- **E300SA22440034 (Fold 2):** slightly smaller probe with non-uniform surface colors, introducing moderate texture variation.
- **E300SA22440035 (Fold 3):** cluttered backgrounds and frequent truncation (often only $\sim$20% of the probe visible at the image edge).
- **E300SA23130257 (Fold 4):** highly complex backgrounds with dense tubes and rods in a top-down view, making detection challenging despite the probe being in sight.
- **EL300858804493 (Fold 5):** a heterogeneous mix. Some samples are straightforward, while others show probes spanning nearly the full image height, complicating bounding box regression.

This variability means that a strong model must generalize not only to new backgrounds but also to different probe scales, truncations, and lighting.

Primary metric. At the end of each epoch, the model is evaluated by sweeping confidence thresholds (0.005–0.90). For each threshold, precision, recall, and F1 are computed. The **F1 score at the best threshold** is used as the primary selection metric:

- *Early stopping:* training halts if F1@best fails to improve beyond a patience window.
- *Best model selection:* the epoch with the highest F1@best is retained as the final checkpoint.

As a small note, I only consider models with a false positive rate (FPR) under 5%, regardless of anything else. Anything beyond would be difficult to deploy in industrial settings, regardless of F1.

Secondary metrics. Because F1 alone may not capture all aspects of performance, several complementary metrics are monitored:

- **mAP@0.5 and mAP@0.5-0.95**: standard object detection metrics for ranking detectors across IoU thresholds.

- **Mean IoU of true positives**: assesses the quality of bounding boxes independently of classification.
- **Precision and recall curves**: highlight trade-offs at different operating points.
- **Confidence stability**: the variability of best thresholds across epochs/folds is examined; highly unstable thresholds suggest overfitting or poor calibration.

Decision hierarchy. Model comparison follows a structured order:

1. F1@best (primary driver).
2. Mean IoU of true positives (tie-breaker for bounding box quality).
3. mAP@0.5:0.95 (secondary confirmation, especially for broader comparison across architectures).
4. Stability of best confidence thresholds (practical robustness).

This evaluation framework ensures that model selection is not based on a single number but on a consistent and transparent set of criteria, interpreted both per mission and in aggregate across folds. In the following Results section, per-mission plots are shown to illustrate how different models handle easy versus difficult conditions.

4 Methodology and Results

Our experimental strategy was to clearly separate **task-dependent hyperparameters** (e.g., augmentations, image preprocessing) from **model-dependent ones** (e.g., architecture, optimizer). The first three steps were performed using a YOLOv8-Nano baseline with a stable learning rate, sufficient for effective learning, though not fully optimized. This provided a consistent reference point for stepwise comparisons.

During inference, we always retained only the top-1 predicted box, since at most one probe was expected per image. We also tested discarding all boxes that did not touch an image edge, reflecting the dataset property that 100% of probes were edge-touching. However, this yielded only marginal gains and introduced risk: in practice, we may want to detect exactly when a probe is falling off the frame. For this reason, the constraint was disabled in the final setup.

Because the dataset contained only positive samples, we generated negatives by cropping out the largest region of positive images without a probe, preserving the original aspect ratio and resizing to the target resolution. These synthetic negatives were used to fill 50% of the validation set, enabling computation of standard object detection metrics. For a positive detection we use an IoU threshold of 0.8. Early stopping monitored the best F1 score, with patience set to 15 epochs and a minimum delta of 0.001.

Additional results supporting these findings are available in the project repository (note that some require training a model first):

- TensorBoard logs per run, including training and validation losses, mean IoU of positive samples, mAP@50, mAP@50–95, best confidence threshold, best F1 score, false positive rate, recall and effective learning rate. All are per epoch.
- Plots of mean IoU of positive samples and mAP@50–95 in the `results/` folder.
- Visual outputs for all validation samples, including worst-case detections.

4.1 Step 1: Photometric Augmentation Strength

Each dataset copy was expanded by duplicating every image: one unaltered and one augmented. Three regimes were tested:

- **No Aug**: duplicate with no changes.
- **Light Aug**: mild adjustments (brightness/contrast, CLAHE, blur, noise). No vignette.

- **Heavy Aug**: stronger versions of the same. Still no vignette, since we're indoors.

Result: Figure 1 shows that light augmentations provided the most balanced performance. Heavy augmentations showed diminishing returns.

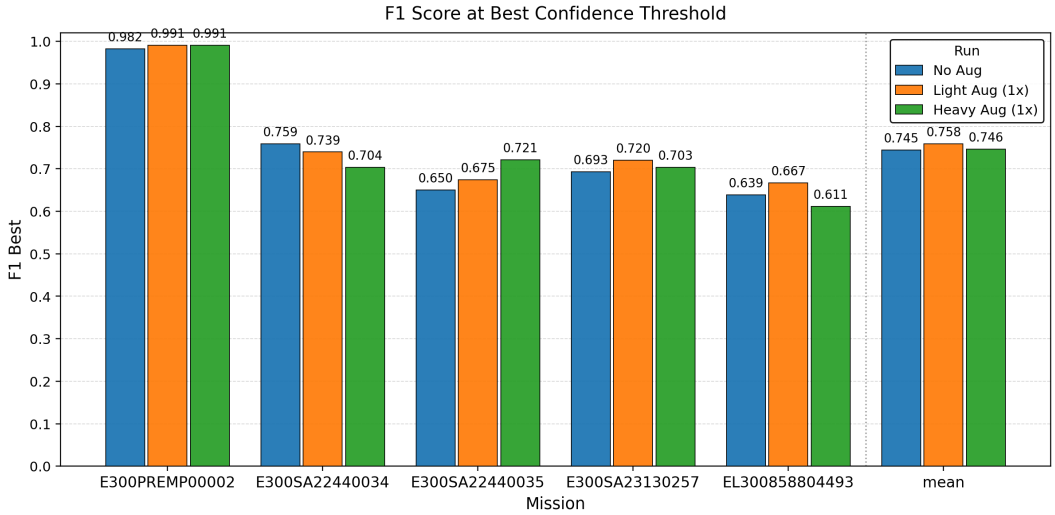


Figure 1: Comparison of augmentation strength regimes (no aug, light aug, heavy aug).

4.2 Step 2: Augmentation Multiplicity

Beyond strength, I tested how many augmented variants per image should be added. Settings included +1, +2, +3, and +7 augmented copies. **Result:** Figure 2 shows that adding 2 augmented samples for each image yielded best performance.

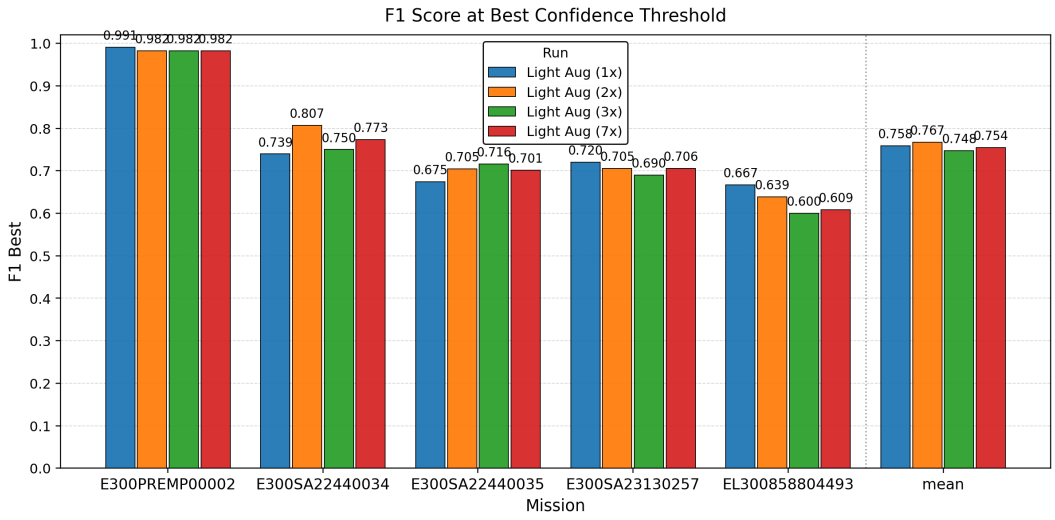


Figure 2: Comparison of augmentation multiplicity regimes.

4.3 Step 3: Geometric Augmentations

Geometric transformations were evaluated to reduce overfitting to fixed probe locations. This is run on top of light photometric augmentations (2x), found best. Settings included:

- **None**: no geometric augmentations.

- **Flip-only**: horizontal + vertical flips, each with 50% probability. These preserve the edge-touching prior while reflecting plausible drone perspectives.
- **All-but-translate**: scale, shear, rotate, flips (but no translation).
- **All**: full set including translation.

Result: While the performance of all augmentations was quite similar in terms of F1, per Figure 3, I went with flip only augmentations, since they were more robust according to secondary metrics.

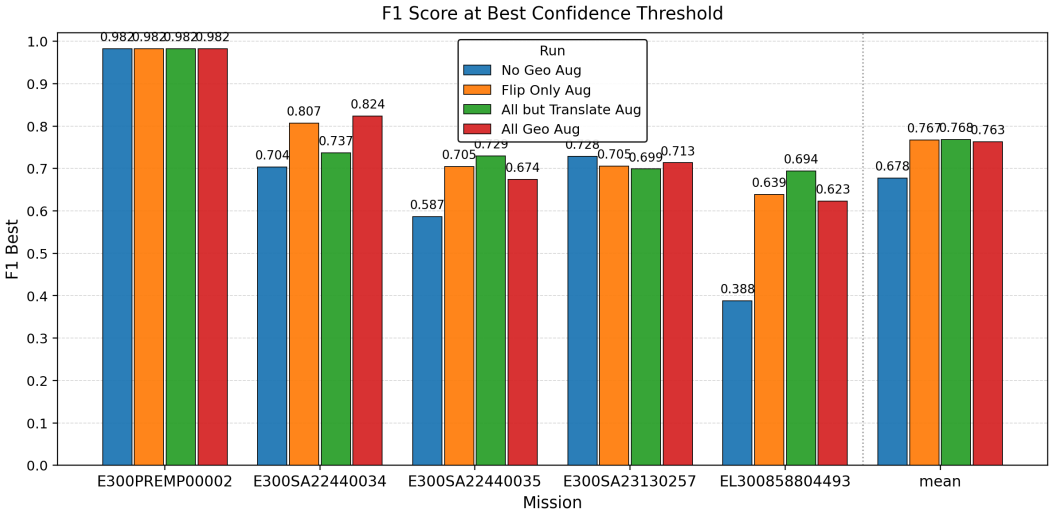


Figure 3: Comparison of geometric augmentations.

4.4 Step 4: Model Selection

To evaluate both efficiency and accuracy trade-offs, I compared lightweight one-stage detectors against a heavier two-stage baseline. The motivation was twofold: (1) establish whether a classical two-stage detector (Faster R-CNN) would provide any advantage on this small-object detection task, and (2) test whether recent lightweight YOLO variants (Nano series) can deliver comparable accuracy while being suitable for deployment on embedded hardware (Jetson).

The four tested families were:

- **YOLOv8-Nano:** established lightweight baseline.
- **YOLOv8 (base):** stronger reference model within the same family.
- **YOLOv12-Nano:** newer generation, advertised as improved efficiency and accuracy.
- **Faster R-CNN:** classical two-stage detector, strong on small objects but computationally heavy.

Result. Faster R-CNN performed poorly, likely due to limited tuning and slow convergence, making it impractical for this dataset and hardware setup. The issue was most evident in the final validation fold, where the probe spans the full image height in some samples; in these cases, the predefined anchors failed to capture it, resulting in severely degraded performance. YOLOv12-Nano consistently outperformed YOLOv8-Nano and even YOLOv8-base, while retaining the efficiency of a Nano model. This made YOLOv12-Nano the preferred model family for subsequent fine-tuning, as per Figure 4.

4.5 Step 5: Input Dimensionality

The original images have a resolution of 640×400. Up to this point, we preserved the native resolution, but in this step we evaluated whether resizing the input images would affect performance. Specifically, we tested both larger and smaller resolutions to see if upscaling could improve F1 score, or if downscaling could reduce runtime without a substantial loss in accuracy.

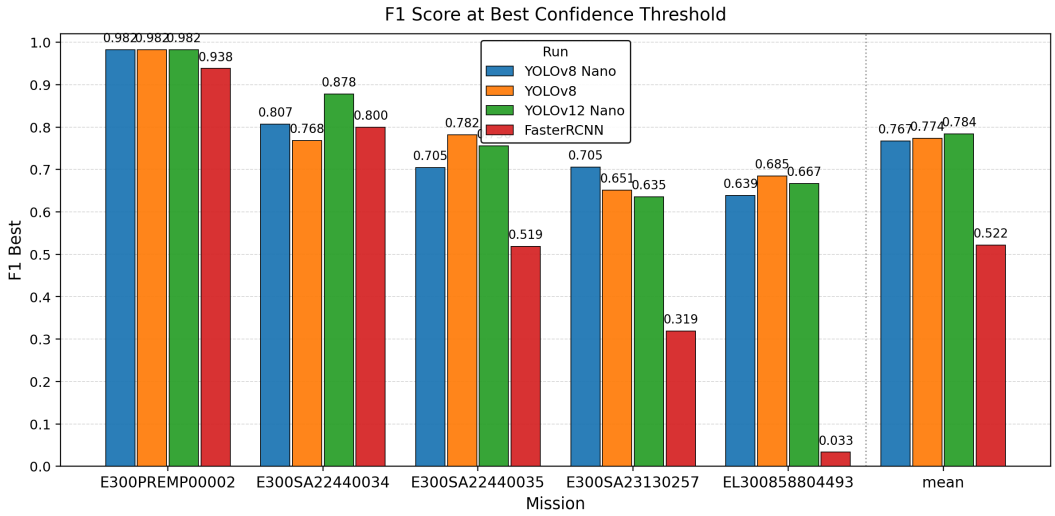


Figure 4: Model comparison.

Result. As shown in Figure 5, reducing the resolution led to a modest drop in performance, while surprisingly, increasing the resolution also degraded results. Based on these findings, we retained the original 640×400 resolution for all subsequent experiments.

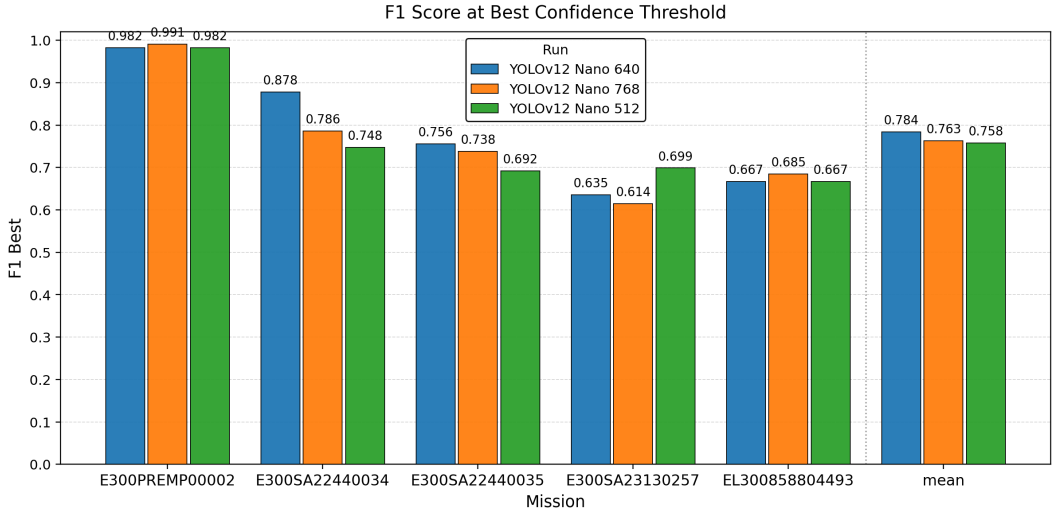


Figure 5: Effect of input image resolution on F1 score across folds. Both upscaling and downscaling degraded performance relative to the original 640×400 resolution.

4.6 Step 5: Learning Rate and Optimizer

Pilot experiments indicated that the dataset required a learning rate lower than the default. The objective here was to test whether small variations around this adjusted value could yield further improvements. Results showed that the baseline choice was already close to optimal. After considering Figure 6 together with per-epoch training curves, $\text{mAP}_{50!:!95}$, and mean IoU for positive detections, the configuration with higher momentum and light weight decay was selected as the final model, described in the next section.

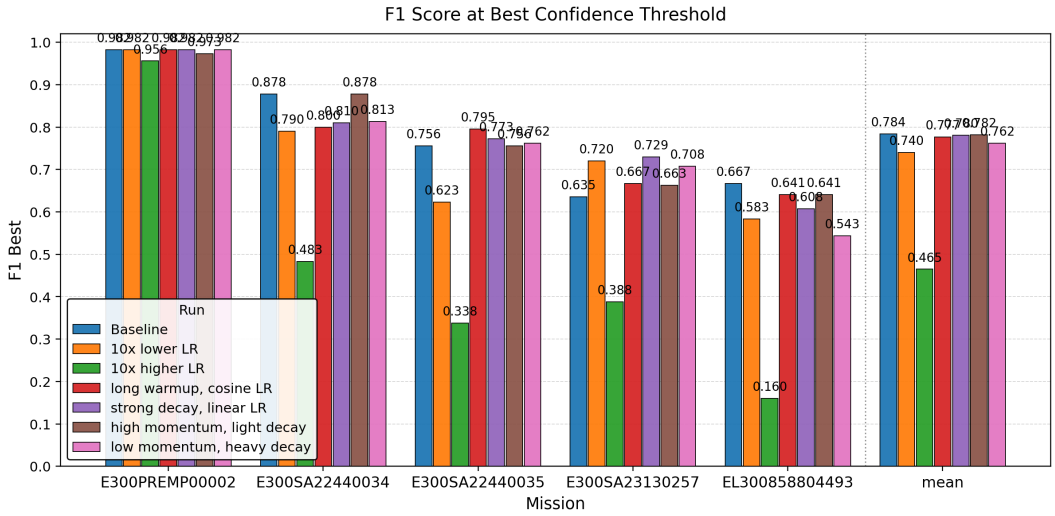


Figure 6: Training hyperparameter search.

4.7 Final Model Configuration

The final selected model was based on `YOLOv12-Nano`. Training was performed for 30 epochs with a batch size of 48 on the probe detection dataset. The optimizer was `AdamW`, with a cosine learning rate schedule and the following key hyperparameters:

- Initial learning rate: 0.00020, cosine decay with $lrf = 0.15$
- Warm-up: 6 epochs, warm-up momentum 0.95, warm-up bias LR 0.0008
- Momentum: 0.95
- Weight decay: 0.0003
- Exponential Moving Average (EMA) of weights enabled
- Confidence threshold for detection: 5%

Data augmentation. Horizontal and vertical flips ($p = 0.5$) were combined with mild photometric changes: brightness/contrast adjustments ($\pm 10\%$, $p = 0.25$), gamma jitter ($\pm 5\%$, $p = 0.2$), and low-probability effects such as motion blur, Gaussian noise, CLAHE, and vignette ($p = 0.05$ each). Geometric augmentations (scaling, translation, shear) were disabled to preserve object geometry.

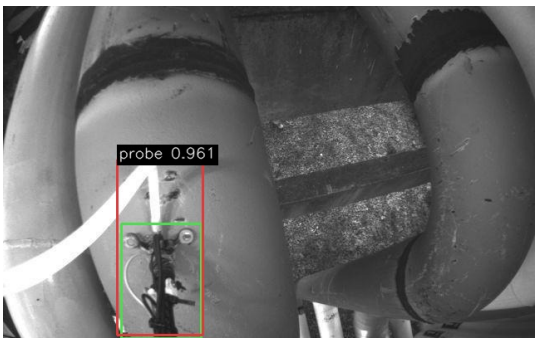
5 Failure Cases

Figure 7 shows representative failure cases from the K-fold validation runs. In this setup, an entire inspection mission is excluded from training in each fold. Therefore, performance here is expected to be lower than that of the final model trained on the full dataset. Nevertheless, these examples illustrate typical challenges for probe detection.

The failure cases fall into three categories:

- **Background merging.** The probe visually blends into the background due to similar intensity or texture, leading to missed detections (Image c).
- **Unseen viewpoints.** View angles that were underrepresented in training cause poor generalization (Images a and d).
- **Annotation inconsistency.** There is ambiguity in whether the connected white tubing should be labeled as part of the probe, resulting in inconsistent learning signals (Images a vs. b).

(a) Tubing included in GT



(b) Tubing excluded in GT

(c) Missed detection, merged with background

(d) Missed detection, unseen viewpoint

Figure 7: Failure cases from K-fold validation runs. Green: gt, red: prediction. Examples illustrate background merging (c), unseen viewpoints (a,d), and annotation inconsistency (a vs. b).

6 Runtime Analysis

We benchmarked inference on an RTX A5000 with batch size 1 at resolution 640×400. The performance frontier is shown in Figure 8.

Two findings stand out. First, `YOLOv12-Nano` was consistently slower than `YOLOv8-Nano`, despite having fewer parameters and FLOPs on paper. This highlights that runtime efficiency is not fully captured by simple complexity metrics. Second, a downscaled variant (512-dim input) ran slower than the same model at native resolution, underlining the non-trivial trade-offs between resolution and kernel efficiency.

Table 1: Runtime comparison on RTX A5000 (batch=1, 640×400). FLOPs are computed with THOP in MAC-counting mode (1 MAC = 1).

Model	Params (M)	MACs (G)	Peak GPU Mem (MB)	FPS
YOLOv8-Nano	3.006	4.043	69.141	143
YOLOv12-Nano	2.557	3.160	70.567	78

7 Embedded Deployment Estimates

We next estimate throughput on Jetson modules commonly used in drones and field robots. To stay consistent with THOP outputs, we treat FLOPs as MACs (1 MAC = 1) and convert device peak performance accordingly. The RTX A5000 has a theoretical peak of 27.8 TFLOPs FP16, equivalent to 13.9 TMAC/s under this convention.

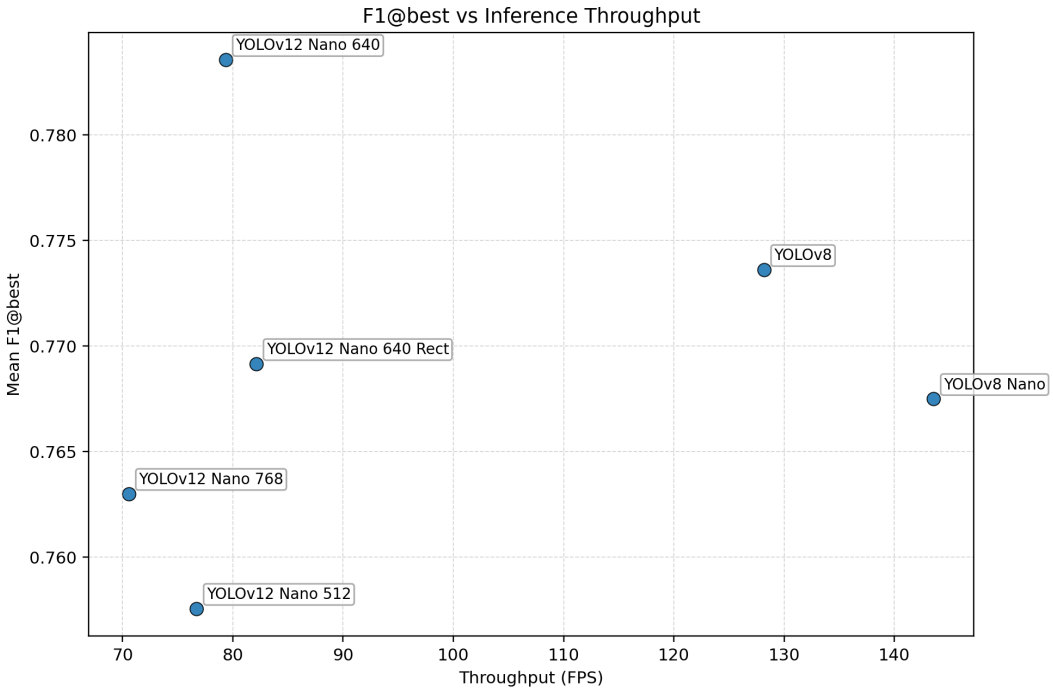


Figure 8: Measured inference speed versus accuracy for candidate models on RTX A5000 (batch=1).

Methodology

Effective utilization on A5000 is computed as:

$$u = \frac{\text{MACs per image} \times \text{FPS}}{\text{Peak TMAC/s}},$$

yielding $u \approx 2.1\%$ for YOLOv8-Nano and $u \approx 0.9\%$ for YOLOv12-Nano. We then scale by each Jetson's published FP16 capability [1, 2, 3, 4] and assume 45–60% of theoretical peak is practically usable. The idea is that relative GPU utilization will be roughly maintained when we move to Jetson.

Estimated Jetson Performance

Table 2: Estimated FPS on Jetson modules (batch=1, 640×400). Ranges reflect 45–60% effective utilization of FP16 throughput.

Device	FP16 TFLOPs	Power (W)	YOLOv8-Nano (FPS)	YOLOv12-Nano (FPS)
Jetson Orin Nano	10–17	7–15	46–78	25–42
Jetson Xavier NX	∼6	10–15	∼28	∼15
Jetson Orin NX	13–19	10–25	60–88	32–47

These estimates indicate that probe detection alone could run at ∼15 FPS even on the legacy Xavier NX, and substantially higher on Orin-class devices. In practice, probe detection would share compute with SLAM, control, and telemetry workloads. However, probe detections are not required at video framerate, making these ranges acceptable.

Model size is not a constraint: with only 2–3 M parameters (< 15 MB in FP32, smaller once quantized), both YOLOv8-Nano and YOLOv12-Nano fit easily into Jetson memory. Power consumption is also within the 7–25 W envelopes of Orin modules. Should throughput still become a bottleneck, INT8 quantization can be considered. No matter what Jetson module Elios3 is equipped with, YOLO Nano models should be easily deployable for probe detection tasks within a reasonable frame rate.

8 Future Improvements

- **Hard negative mining.** Current negatives are mostly trivial background crops. This is by far the largest limitation. Future work should include more realistic distractors, such as drone wiring without probes, other payloads or background structures resembling the probe. I also thought about diffusion-based inpainting to synthesize probe-free versions of positive images, but I thought it was an overkill.
- **Augmentation refinements.** Edge-aware augmentations that preserve the border-touching prior could improve performance on this dataset. However, in real-world deployment this prior may be harmful. For instance, if a probe detaches and appears in the middle of the frame, a model overfitted to edge-touching probes might fail. Future work should carefully evaluate whether such priors are desirable. In addition, domain-specific photometric effects (e.g., simulating glare, rust, or underwater scattering) could further increase robustness to inspection environments not represented in the current dataset.
- **Model compression.** Quantization (INT8) and pruning should be tested for Jetson deployment. Both approaches could reduce latency and power consumption, while maintaining accuracy.
- **Data collection strategy.** New inspection missions could be recorded specifically to cover underrepresented probe viewpoints and lighting conditions, reducing failure modes identified in validation. We should be careful to have high quality labels and be consistent in whether we want to include the tubing in the ground truth bounding box. Since the dataset is so small, we could even consider manually re-labelling the few that there are.

References

[1] NVIDIA. *NVIDIA RTX A5000 Graphics Card Datasheet.* https://www.pny.com/File%20Library/Company/Support/Product%20Brochures/NVIDIA%20Quadro/English/nvidia-rtx-a5000-datasheet.pdf.

[2] NVIDIA. *Jetson Xavier NX Module Data Sheet (DA-09366-001).* https://developer.nvidia.com/embedded/jetson-xavier-nx.

[3] NVIDIA. *Jetson Orin NX Series Module Datasheet.* https://developer.nvidia.com/downloads/jetson-orin-nx-series-data-sheet.

[4] NVIDIA. *Jetson Orin Nano Series Module Datasheet (DS-11105-001 v1.1).* https://www.mouser.com/pdfDocs/Jetson_Orin_Nano_Series_DS-11105-001_v11.pdf.