Universiteit Leiden

Efficient small-object Video Detection via Quantization-Aware Training

Lenard Johannsen (s3926540)

Supervisors:
Michael Lew & Erwin Bakker

Bachelor Thesis– Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
liacs.leidenuniv.nl

August 4, 2026

Abstract

Quantization compresses neural network weights from 32-bit floating point to 8-bit integer arithmetic, enabling faster and more memory-efficient inference. However, INT8 quantization disproportionately degrades the detection of small objects, which produce low-signal, high-variance activations in shallow feature maps. This thesis investigates how INT8 quantization affects small-object detection accuracy (AP_{small}) and multi-object tracking (MOTA, MOTP) in a YOLOv11-based video pipeline through a stepwise ablation on XS-VID (mean object size 10.6 x 10.6 px) with 4 configurations. The proposed configurations are FP32 as baseline, PTQ INT8, QAT, and QAT with a *QuantSmooth Neck* modification (zero-parameter layer normalization and activation clamping on the P3/P4 detection head outputs).

At every IoU threshold value and object-size catogory the detection accuracy of each configuration decreases monotonically. The AP values fall from 0.181 (FP32) to 0.162 (PTQ) to 0.106 (QAT) to 0.076 (QAT+QuantSmooth); this trend follows with the extremely small object sub-metric AP_{eS}. These declines are carried through the downstream tracking pipeline. By applying ByteTrack to each configuration the detections show MOTA (Multi-Object-Tracking-Accuracy) dropping from 0.247 (FP32) to 0.091 (QAT+QuantSmooth), and IDF1 decreasing by more than half over the same range. The hypothesis presumed that normalizing and clamping activations at the P3/P4 detection head outputs would counteract the activation-outlier problem inherent to quantization-induced accuracy loss. The results contradict this hypothesis and show the QuantSmooth Neck reduces accuracy beyond standard QAT rather than showing improvement.

Contents

1 Introduction

Detecting small objects in video is a challenge that is relevant across many domains. In applications such as drone surveillance, search-and-rescue and traffic monitoring, the subjects can take up fewer than 32×32 pixels in a 640×640 frame [14]. In these settings, when detections fail, it could directly result in safety consequences. Developing accuracy improvements for small-object detection will not only be impactful for academic purposes but could also mean saving the life of someone stuck in rubble using an overhead drone.

There are currently two main types of detectors, ones that are slow and accurate, and ones that are fast and inaccurate. One-stage detectors such as the YOLO family [15] are the latter, trading accuracy for speed, making it the ideal choice for real-time detection. Feature Pyramid Networks (FPN) [13] are used in combination with YOLO to minimize the limitations of detecting small objects by combining information from different scales. On the aerial video benchmark XS-VID [5] the mean object size is 10.6×10.6 pixels, for which per-frame YOLO detectors achieve AP_{small} as low as 6–10% which shows ample room for improvement.

The problem compounds when the detectors are deployed on hardware with INT8-accelerated inference. Quantization converts model weights from 32-bit floating point (FP32) to 8-bit integer (INT8) arithmetic, allowing GPUs with dedicated INT8 tensor cores to complete the same operations 2-4x faster [7]. The effect of quantization is experienced by many with audio output over bluetooth as opposed to over wired connections. Audio cannot be produced losslessly over bluetooth as the connection has limited bandwidth. The full array of sound cannot be transmitted so they apply quantization which allows the user to hear the music playing, but it loses its nuance in the smaller details, like the lingering hiss of a snare; whereas with a wired connection every minute detail is left unscathed. This translates to small objects in the same way, INT8 quantization introduces noise that disproportionately degrades low-signal feature representations, particularly in the shallow, high-resolution feature maps that are relied on to detect small objects.

Two standard quantization strategies exist. Post-Training Quantization (PTQ) converts a trained FP32 model to INT8 using a small calibration set without needing to retrain. Quantization-Aware Training (QAT) simulates INT8 arithmetic during training, allowing weights to preempt the quantization constraint and adapt, which prevents a sizable amount of the accuracy drop [11]. In addition to QAT some targeted modifications to the feature pyramid neck were added to potentially reduce the impact of quantization on the accuracy loss. Two of these modifications are normalization and activation clamping at the P3/P4 levels where small object features are processed.

Multi-object tracking amplifies these effects downstream. ByteTrack [18] uses a two-pass association strategy that recovers low confidence detections for small objects. Improvements in AP_{small} therefore directly improve tracking consistency metrics such as MOTA and MOTP.

No prior work has studied how INT8 quantization specifically affects small object detection and downstream tracking in video pipelines. This thesis addresses that gap. A per-frame detection approach is deliberately chosen over temporal fusion methods such as YOLOFT [5], because it isolates the quantization effect cleanly. A temporal feature aggregation would confound the ablation by introducing a second source of accuracy variation across configurations.

Research Question

How does INT8 quantization affect small-object detection (AP_{small}) and tracking (MOTA, MOTP) in a YOLOv11-based video pipeline, and can a QAT strategy with a lightweight feature-normalization modification recover accuracy while maintaining competitive throughput on INT8-accelerated hardware?

This decomposes into four sub-questions:

1. By how much does PTQ INT8 reduce AP_{small} and MOTA/MOTP relative to the FP32 baseline?
2. Does QAT recover a significant portion of that drop, and at what retraining cost?
3. Can LayerNorm and clamping on P3/P4 neck features (QuantSmooth Neck) further improve QAT performance?
4. Does QAT + QuantSmooth Neck maintain competitive throughput within 10% of FP32 AP_{small} on INT8-accelerated GPU hardware?

Contributions

The primary contributions of this thesis are:

1. A systematic empirical study of the effect of INT8 quantization on small-object detection and tracking in a YOLOv11 + ByteTrack pipeline, with broken down $AP_{eS}/AP_{rS}/AP_{gS}$ metrics not reported in any prior quantization study.*No prior quantization study reports these sub-metrics or evaluates on a dataset with objects of comparable scale to XS-VID.*
2. The *QuantSmooth Neck*: a lightweight, zero-parameter modification adding layer normalization and activation clamping to P3/P4 neck feature maps, targeting the activation outlier problem that drives quantization-induced accuracy loss on small objects. *While neck normalization for quantization has been explored in YOLOv6+ [10], that work targets the regression head on general objects, QuantSmooth specifically targets the activation outlier problem in shallow feature maps for extremely small objects under QAT.*
3. A four-configuration stepwise ablation on XS-VID. *XS-VID has not previously been used in any quantization study.*

Thesis Overview

The remainder of this thesis is structured as follows. Section 2 provides background on object detection, feature pyramids, quantization, and multi-object tracking. Section 3 discusses related work and identifies the research gap. Section 4 describes the proposed QuantSmooth Neck and the full experimental methodology. Section 5 presents the experimental results. Section 6 discusses the results in relation to the research questions and the literature. Section 7 concludes and outlines directions for future research.

2 Background

2.1 Object Detection

Object detection is the task of localising and classifying all object instances in an image, producing bounding boxes with class labels and confidence scores. There are two types of modern detectors.

Two-stage detectors, such as Faster R-CNN [16], first generate region proposals via a Region Proposal Network (RPN), then classify each proposal. This achieves high accuracy at the cost of inference speed, making it unsuitable for real-time applications.

One-stage detectors such as YOLO [15] perform localisation and classification in a single forward pass. The YOLO lineage has evolved from YOLOv1's grid-based prediction to the anchor-free, attention-augmented architecture of YOLOv11 [8]. YOLOv11-nano (YOLOv11n), used in this thesis, comprises C3k2 (efficient CSP bottleneck), SPPF (spatial pyramid pooling fast), and C2PSA (cross-stage partial with attention) modules, summing to approximately 2.6M parameters and 6.4 GFLOPs.

2.2 Feature Pyramid Networks

Feature Pyramid Networks (FPN) [13] construct a pyramid of feature maps at different resolutions through a top-down pathway with lateral connections. In YOLOv11, the neck produces feature maps at multiple stride levels. The shallowest levels of P3 (stride 8) and P4 (stride 16) handle the highest-resolution feature maps and are the levels at which small objects ($< 32^2$ px) predominantly appear. This has direct implications for quantization as noise introduced at these shallow layers is damaging to small-object detection, since the signal produced by small objects is already low.

2.3 Small-Object Detection and the XS-VID Benchmark

Small objects are defined in the COCO standard [14] as those with area below 32^2 pixels. XS-VID [5] further subdivides this range into three sub-categories: extremely small (eS: 0–12^2 px), relatively small (rS: 12–20^2 px), and generally small (gS: 20–32^2 px). With a mean object size of 10.6×10.6 pixels, XS-VID is the most challenging publicly available benchmark for small-object video detection. It is selected over comparable benchmarks such as UAVDT [2] and VisDrone2019-VID [19] because its eS/rS/gS sub-metric breakdown allows for evaluation of quantization effects at different object scales which is not possible in other benchmarks. Feng et al. [4] describe 4 different approaches to small-object detection: resolution boosting, scale-aware training, contextual information, and data augmentation. This thesis does not modify the detection architecture for accuracy but instead investigates how to preserve the accuracy of an existing architecture under INT8 quantization.

2.4 Neural Network Quantization

Quantization maps continuous floating-point values to a discrete integer set. In practice, weights and activations are quantized differently due to hardware constraints.

Weight quantization is typically performed per-channel. A separate scale factor s_c is computed for each output channel, allowing the quantizer to absorb much of the inter-channel variance in weights. This per-channel approach is hardware-friendly and used by default in TensorRT.

Activation quantization is constrained to per-tensor quantization. A single scale factor s covers all activations in a layer, computed from the observed activation range during calibration:

$$x_q = \text{clip}\Big(\Big\lfloor \frac{x}{s} \Big\rceil,\ -128,\ 127\Big) \tag{1}$$

This distinction matters because while per-channel weight quantization absorbs most weight variance, activations do not. A single outlier activation can force the scale factor to represent a range wider than needed for the majority of values, reducing effective precision for the small-object signal. This is the core problem the QuantSmooth Neck addresses.

Post-Training Quantization (PTQ) determines scale factors from a calibration set without retraining. Krishnamoorthi [11] recommends 500 representative frames using MINMAX calibration. PTQ is fast but degrades accuracy when activation distributions contain outliers—exactly the condition present in P3/P4 feature maps for small objects.

Quantization-Aware Training (QAT) [7] inserts fake-quantization nodes into the forward pass, simulating the rounding and clipping that will occur during INT8 inference. Since the rounding function has zero gradient almost everywhere, QAT uses the straight-through estimator (STE), which approximates the gradient of the quantization operation as 1, allowing gradients to flow through quantization nodes during backpropagation. This allows weights and activations to adapt to the INT8 constraint during training, recovering a significant portion of the PTQ accuracy gap. LSQ [3] extends QAT with learned step sizes per layer.

2.5 The Activation Outlier Problem for Small Objects

The per-tensor constraint on activation quantization creates a specific vulnerability for small objects. At the P3/P4 neck levels, small objects produce low-magnitude activations interspersed with occasional high-magnitude outliers from background clutter. An example being that a single high-activation background pixel in a P3 feature map can force the INT8 scale factor to represent a range ten times wider than the typical small-object signal, effectively quantizing that signal into just a handful of discrete levels. Hubara et al. [6] identify activation range as the primary determinant of PTQ quality. Normalizing and clamping activations prior to quantization reduces the range the quantizer must represent, improving effective precision for the small-object signal.

2.6 Multi-Object Tracking

ByteTrack [18] is the tracker used in this thesis. Its BYTE two-pass association first matches high-confidence detections (score ≥ 0.25 by default) using Kalman filter motion predictions and IoU cost; a second pass then recovers low confidence detections (score ≥ 0.1) for occluded and small objects. This two-pass design makes tracking quality particularly sensitive to AP_{small}: improved detection of low confidence small objects in the first pass directly feeds the second-pass recovery mechanism.

Standard MOT metrics are: MOTA (Multi-Object Tracking Accuracy), which penalises false positives, false negatives, and identity switches; IDF1, which measures the ratio of correctly identified detections over the mean of ground-truth and computed detections, emphasising identity consistency across tracks; and MOTP (Multi-Object Tracking Precision), which measures the average localisation accuracy of matched detection pairs, which is the mean IoU distance between

the tracked bounding box and the ground-truth box. Unlike MOTA this measurement is unaffected by missed detections and identity switches, and focuses on how precisely the detections align spatially with ground truth.

3 Related Work

3.1 Quantization for Object Detection

Jacob et al. [7] established the theoretical framework for QAT with integer-only inference, demonstrating accuracy preservation on COCO object detection. Their evaluation reports overall mAP only; no disaggregation by object size is performed and AP_{small} is not reported. Krishnamoorthi [11] provides the practical PTQ calibration framework. Both papers are foundational but leave the specific effect of quantization on small objects unexamined.

More recent work has studied quantization for YOLO detectors specifically. Karimov et al. [9] conduct a comprehensive empirical study of PTQ robustness for YOLO12 models across FP32, FP16, Dynamic UINT8, and Static INT8 via TensorRT, evaluating on COCO under synthetic input degradations (noise, blur, JPEG compression). They report mAP50-95 and mAP50 exclusively on the full COCO validation set with no disaggregation by object size. Their study addresses robustness to *input* degradation rather than the structural feature map degradation caused by quantization in shallow FPN levels, and no QAT is applied.

Li et al. [12] propose a zero-shot QAT framework for object detection evaluated on MS-COCO and Pascal VOC, reporting overall mAP50-95. AP_{small} is not reported as a separate metric, neither dataset contains extremely small objects, and no video or tracking evaluation is performed.

Kim et al. [10] propose YOLOv6+, modifying the YOLOv6 neck with skip connections and a regression normalization method to improve INT8 quantization friendliness on mobile hardware. This is the closest prior work to the QuantSmooth Neck proposed in this thesis. Three key differences distinguish the approaches: (i) their normalization targets the *regression head output*, not the shallow P3/P4 *neck feature maps* where small-object activations reside; (ii) evaluation is on COCO overall mAP with no small-object sub-metrics; and (iii) no video, tracking, or small-object-specific benchmark is used. Their motivation is reducing dynamic range differences between backbone and neck modules in general, not the specific activation outlier problem in high-resolution small-object feature maps.

3.2 Small-Object Detection

Surveys of small-object detection [4] identify resolution boosting, scale-aware training, contextual information, and data augmentation as the main technique families. SAHI [1] achieves +6.8% AP on VisDrone through sliced inference at test time. QueryDet [17] uses cascaded sparse queries for efficient high-resolution feature extraction. TPH-YOLOv5 [20] adds transformer prediction heads for drone detection. Critically, none of these works study the interaction between their accuracy improvements and INT8 quantization.

3.3 Video Object Detection

YOLOFT [5] extends YOLOv8 with recurrent optical flow, achieving 36.4 AP on XS-VID and serving as the state-of-the-art baseline. This thesis uses per-frame YOLOv11 detection rather than temporal fusion, providing a cleaner experimental setup for ablating quantization effects without confounding temporal aggregation.

3.4 The Research Gap

Three gaps are consistently present across this body of work:

Gap 1 — No disaggregated small-object quantization metrics. Every quantization paper cited above reports only overall mAP on COCO. None measure AP_{eS}, AP_{rS}, or AP_{gS}. It is therefore unknown from the literature by how much INT8 quantization specifically degrades detection of extremely small objects.

Gap 2 — No quantization study on small-object video benchmarks. XS-VID [5] has not been used in any quantization study. All prior work uses COCO, Pascal VOC, or ImageNet, where mean object sizes are orders of magnitude larger.

Gap 3 — No study of quantization's downstream effect on tracking. No prior work examines how INT8 quantization degrades MOTA, MOTP, or IDF1 in a video pipeline. The propagation chain—quantization $\rightarrow$ AP_{small} drop $\rightarrow$ MOTA/MOTP degradation via ByteTrack's two-pass mechanism—has not been studied.

This thesis directly addresses all three gaps.

4 Approach

4.1 Overview

The methodology is empirical and ablation-driven. Four configurations of the same YOLOv11n base model are compared in a stepwise design that isolates the contribution of each component. ByteTrack is applied uniformly after each detector configuration with fixed thresholds to measure downstream tracking effects independently of the tracker.

4.2 Experimental Configurations

Table 1 describes the four configurations.

4.3 QuantSmooth Neck

The QuantSmooth Neck modification addresses the per-tensor activation outlier problem at the P3/P4 neck feature maps. At these shallow levels, small objects produce low-magnitude activations interspersed with high-magnitude outliers from background clutter. When a single INT8 scale factor is computed from the full activation range, the quantizer wastes resolution representing outlier values at the cost of the small-object signal.

Two operations are inserted into the P3/P4 neck outputs, immediately before the fake-quantization nodes in the QAT graph:

Config	Description
1 — FP32	Full-precision reference. YOLOv11n fine-tuned on XS-VID for 50 epochs at 640×640.
2 — PTQ INT8	FP32 model exported to ONNX INT8 via the Ultralytics export pipeline. MINMAX calibration with 500 uniformly sampled training frames [11]. No retraining. TensorRT INT8 export was attempted but blocked by a TensorRT 10 API incompatibility on available hardware; ONNX Runtime INT8 provides an equivalent hardware-agnostic PTQ baseline consistent with the algorithmic focus of this thesis [7].
3 — QAT	Fine-tuning from FP32 weights with `torch.ao.quantization.prepare_qat`; fake-quantization nodes simulate INT8 arithmetic during training.
4 — QAT + QuantSmooth Neck	Config 3 extended with the QuantSmooth Neck modification on P3/P4 features (Section 4.3).

Table 1: Stepwise four-configuration ablation design.

1. **Layer normalization**: applied across the channel dimension per spatial location, suppressing outliers and reducing distribution variance without introducing learnable parameters (`elementwise_affine=False`).

2. **Activation clamping**: `torch.clamp(x, -3, 3)` hard-clips residual outliers to a symmetric $\pm 3\sigma$ range. If layer normalization produces an approximately $\mathcal{N}(0, 1)$ distribution, then ± 3 retains 99.7% of activation values while eliminating extreme outliers. The clamp bounds may require empirical validation against actual activation distributions.

This modification introduces zero additional parameters. This is supported by the analysis of Jacob et al. [7] and Hubara et al. [6], who identify activation range as the primary driver of PTQ accuracy loss.

4.4 Implementation Details

The QuantSmooth Neck is implemented without modifying Ultralytics' source code. A forward hook is applied on the P3 and P4 detection head modules (`model.model[16]` and `model.model[19]`) respectively, using Ultralytics' indexing via `register_forward_hook` attached through an `on_pretrain_routine_e` callback at the start of training. The hook applies the following transformation to the modules' output:

```
x = output.permute(0, 2, 3, 1).contiguous()
x = F.layer_norm(x, (x.shape[-1],))
x = x.permute(0, 3, 1, 2).contiguous()
x = torch.clamp(x, -3, 3)
```

These operations transform the tensor so that layer normalization is computed per spatial location as is consistent with the per-position normalization outlined in section 4.3, rather than being applied across the channel, height, and width. This modification required no forked or patched version of Ultralytics thus maintaining reproducibility. The hook is executed within the QAT training graph so the smoothed activation distribution is observed by the fake-quantization nodes. This modification is applied only during Config 4 (QAT+QuantSmooth Neck) and not to Configs 1–3 which use the unmodified architecture.

When generating valid predictions for the QAT and QAT+QuantSmooth configurations from the saved checkpoint, the outputs resulted in accuracy that did not match the expected INT8 accuracy. It was discovered that Ultralytics' training loop reloads the saved checkpoints for its internal validation at the end of training and exporting, which would strip the FakeQuantize modules inserted by `prepare_qat`, reverting the model to full-precision without notifying it had done so. Calls to `model.val()` on a saved QAT or QAT+QuantSmooth configuration would result in FP-32 accuracy instead of INT-8 accuracy that the configurations were supposed to measure. This was addressed by collecting predictions directly from an `on_val_end` callback during the first correctly FakeQuantized validation pass within the training run. This would bypass the Ultralytics checkpoint reload before it could strip the quantization simulation, and removed the necessity of relying on a separate validation call.

4.5 Datasets

XS-VID [5] is the primary training and evaluation dataset, containing 186,446 training frames and 36,478 test frames across 374 video sequences and seven object categories. The mean object size of 10.6×10.6 pixels and the eS/rS/gS sub-metric breakdown make it uniquely suited for disaggregated quantization evaluation.

MS COCO [14] provides the pre-training weights for YOLOv11n.

4.6 Evaluation Metrics

Detection performance is evaluated using $AP_{50:95}$ and AP_{50}, plus the XS-VID-specific sub-metrics AP_{eS} ($< 12^2$ px), AP_{rS} (12–20^2 px), and AP_{gS} (20–32^2 px), computed with the official XS-VID evaluation tool [5] against COCO-format ground truth in `test.json`. Tracking performance is evaluated with MOTA, MOTP, and IDF1 computed via a pipeline built on the `motmetrics` Python library after applying ByteTrack post-processing. Throughput is measured as FPS and latency on an NVIDIA RTX 3070.

4.7 Training Setup

Hardware scope. The original thesis proposal specified a Jetson Orin Nano Super for edge deployment evaluation. Following supervisor feedback that algorithmic generality is more scientifically valuable than single-device results, the proposal was changed to have evaluation conducted on INT8-accelerated GPU hardware (NVIDIA RTX 3070 for inference benchmarking). This is consistent with the framing of Jacob et al. [7], whose quantization scheme is validated across hardware platforms rather than on a single device. **FP32 baseline (Config 1).** YOLOv11n is initialised from COCO-pretrained weights provided by Ultralytics [8] and fine-tuned on XS-VID for

50 epochs at 640×640 resolution using SGD with momentum 0.937 and weight decay 0.0005, initial learning rate 0.01 with cosine decay to 0.0001. The batch size is 16 and trained with Ultralytics' default setting of automatic mixed precision (AMP) enabled. The full Ultralytics augmentation schedule is applied, mosaic augmentation with probability 1.0 for epochs 1–40, disabled for epochs 41–50 (`close_mosaic=10`), with HSV colour jitter, random flips, and scale jitter. Training used PyTorch 2.4 and CUDA 12.4 on a single NVIDIA H100 GPU (cloud instance).

QAT fine-tuning (Configs 3 and 4). QAT starts from the FP32 best.pt checkpoint and fine-tunes for 50 epochs using the same SGD optimiser and augmentation schedule. There are two changes differentiating these configurations from Config 1. First is the insertion of fake-quantization nodes via `torch.ao.quantization.prepare_qat`, which simulates INT8 rounding during the forward pass and uses straight-through estimators in the backward pass. Second is the standard Ultralytics setting, AMP is disabled (amp = False), which was required to avoid a dtype conflict between FP16 autocast and the FakeQuantize modules' observer statistics. The backbone is trained end-to-end (not frozen). Config 4 additionally applies the QuantSmooth Neck modification to P3/P4 outputs.

PTQ calibration (Config 2). 500 frames are sampled uniformly across all training video sequences, maintaining proportional representation of sequence lengths to ensure calibration exposure to the full diversity of object sizes and scene types. MINMAX scaling is used.

ByteTrack configuration. ByteTrack is applied uniformly after all four detector configurations with fixed thresholds: high-score threshold 0.25, low-score threshold 0.1, new track threshold 0.25, track buffer 30 frames, match threshold 0.8, score fusion enabled. Thresholds are not re-tuned between configurations. This means the thresholds were calibrated on the FP32 confidence distribution; INT8 configurations may produce lower confidence scores, which could disadvantage them relative to FP32 in the second-pass recovery. This is acknowledged as a limitation and noted in the Discussion.

5 Experiments

5.1 Experimental Protocol

For each configuration, predictions are generated using `model.val(save_json=True)` against the XS-VID test split, producing a COCO-format `predictions.json`. However for the QAT and QAT+QuantSmooth configurations the predictions were captured during training via callback rather than a separate validation call, for reasons discussed in Section 4.4. This file is then passed to the official XS-VID evaluation tool [5] alongside `annotations/test.json` to compute AP_{eS}, AP_{rS}, and AP_{gS}. ByteTrack is applied post-detection to produce track files for TrackEval evaluation.

5.2 Main Results

Table 2 summarises the four-configuration ablation on XS-VID. Detection accuracy monotonically decreases across the ablation. At every IoU threshold and object size category FP32 ranks the highest in AP, followed by PTQ INT8, QAT and finally QAT + QuantSmooth. AP_{eS} which are the extremely small objects and the main subject of the research follow the same ranking order by configurations.

Config	AP	AP50	AP75	AP_{eS}	AP_{rS}	AP_{gS}
1 — FP32	0.181	0.306	0.185	0.034	0.073	0.157
2 — PTQ INT8	0.162	0.288	0.154	0.022	0.057	0.134
3 — QAT	0.106	0.190	0.103	0.018	0.043	0.113
4 — QAT + QuantSmooth	0.076	0.142	0.074	0.006	0.020	0.078

Table 2: Detection accuracy on the XS-VID test set across the four-configuration ablation, computed with the official XS-VID evaluation tool. AP is averaged over IoU 0.50:0.95 unless noted. AP_{eS}, AP_{rS}, and AP_{gS} correspond to object areas of 0–12^2, 12^2–20^2, and 20^2–32^2 pixels respectively.

Detection accuracy monotonically decreases across the ablation. At every IoU threshold and object size category FP32 ranks the highest in AP, followed by PTQ INT8, QAT and finally QAT + QuantSmooth. AP_{eS} which are the extremely small objects, and the main subject of the research, follow the same ranking order by configurations.

To assess if the loss persists into the video pipeline ByteTrack [18] is applied to each of the configurations' detections using the fixed thresholds in Section 4.7.

Config	MOTA	MOTP	IDF1	ID Switches
1 — FP32	0.247	0.194	0.404	1194
2 — PTQ INT8	0.235	0.202	0.376	1556
3 — QAT	0.189	0.201	0.346	995
4 — QAT + QuantSmooth	0.091	0.159	0.217	338

Table 3: ByteTrack multi-object tracking performance on XS-VID after each detector configuration, with fixed ByteTrack thresholds (high-score 0.25, low-score 0.1) applied uniformly across all four configs. MOTP is the mean IoU-based distance over matched pairs (lower indicates tighter localisation).

Table 3 reports the resulting MOTA, MOTP, IDF1, and identity switch counts which are outlined in Section 4.7.

5.3 Comparison to State-of-the-Art

To contextualise the FP32 baseline used throughout this ablation, YOLOv5n, YOLOv8n, and YOLOv8s were retrained on XS-VID under identical conditions to Configuration 1 (batch size 16, 50 epochs, seed 42), and evaluated with the same official XS-VID evaluation tool used elsewhere in this thesis. This comparison is not itself an ablation over quantization and is not used to answer any of the four research sub-questions; rather, it establishes how competitive the YOLOv11n baseline is against community detectors before the quantization analysis is applied on top of it.

Table 4 shows that YOLOv11n performs comparably to the similarly-sized YOLOv5n and YOLOv8n (within 0.005 and 0.007 AP respectively), while the larger YOLOv8s achieves the highest absolute accuracy. This indicates that the accuracy trends observed across the quantization ablation are not an artifact of an unusually weak baseline architecture.

Method	AP	AP50	AP75	AP_{eS}	AP_{rS}	AP_{gS}
YOLOv5n	0.179	0.293	0.177	0.042	0.075	0.153
YOLOv8n	0.186	0.307	0.188	0.043	0.085	0.169
YOLOv8s	0.204	0.333	0.210	0.044	0.095	0.179
YOLO11n (ours, FP32)	0.181	0.306	0.185	0.034	0.073	0.157

Table 4: Comparison to retrained community baselines on XS-VID under identical training conditions (batch size 16, 50 epochs, seed 42), all evaluated with the official XS-VID evaluation tool.

5.4 Effect of PTQ on small-object Detection (RQ1)

PTQ INT8 reduces AP from 0.181 to 0.162, a 10.5% drop, which is a marginal increase on the loss of AP reported by Karimov et al. [9] for PTQ of 3–7% on COCO scale objects. For extremely small objects, the impact of PTQ on accuracy is substantially worse with AP_{eS} falling 35% from 0.034 to 0.022. During the tracking portion, MOTA drops marginally from 0.247 to 0.235, however identity switches increased by 30% from 1194 to 1556 and IDF1 falls from 0.404 to 0.376.

5.5 QAT Recovery (RQ2)

The expectation was that QAT would recover around 50–80% of the PTQ accuracy gap [7], however QAT fine-tuning did not recover accuracy at all. AP fell further than PTQ from 0.162 (PTQ) to 0.106 with AP_{eS} continuing to decline from 0.022 to 0.018. The values were cross-validated across two independent evaluation pipelines (the official XS-VID tool and a separately implemented ByteTrack evaluation), both of which show consistent decreases to accuracy.

5.6 QuantSmooth Neck Ablation (RQ3)

QuantSmooth Neck did not improve off the standard QAT configuration and degraded accuracy even further across all metrics and object-size categories. The AP fell from 0.106 (QAT) to 0.076 with AP_{eS}, the specific target category of QuantSmooth, falling from 0.018 to 0.006. This outcome contradicts the hypothesis in Section 4.3 that normalizing and clamping activations at the P3/P4 levels would counteract the activation-outlier problem. The proposed modification had a clear negative result and is discussed further in Section 6.

5.7 Throughput and Speed-Accuracy Trade-off (RQ4)

Throughput was benchmarked on an NVIDIA RTX 3070 at 640x640 resolution at batch size 1. The FP32 baseline achieves 62.5 FPS (16.0 ms per frame). INT8 ONNX models were exported for all three quantized configurations (PTQ, QAT, and QAT+QuantSmooth) using Ultralytics' calibration-based export pipeline. The calibration was performed on a sample of 500 frames drawn from the XS-VID training sequences. Table 5 shows the resulting model sizes: PTQ INT8 at 3.1MB, QAT INT8 at 3.0MB, and QAT+QuantSmooth INT8 at 3.0MB, each a 3.5–3.7× reduction in size relative to the 11.0MB FP32 ONNX model. Throughput for the three INT8 configurations could not be measured within the project timeline. Benchmarking was blocked by a dependency conflict

between the training environment (PyTorch 2.4.0 built against CUDA 12.4) and ONNX Runtime's CUDA execution provider requiring CUDA 13 runtime components that are not present in the CUDA 12.4 environment used throughout the thesis. 5 versions of ONNX Runtime were tested (1.18.1 through 1.28.0) with both of the framework's pinned and manually unpinned CUDA runtime libraries, none of which was able to resolve the incompatibility issue. The CUDA execution provider failed to initialise due to the version mismatch and as such GPU throughput measurements for the PTQ, QAT and QAT+QuantSmooth configurations were not possible and thus is not reported. This is discussed further as a limitation in Section 6, with resolution via a CUDA 13 environment or a TensorRT deployment path evaluated as future work.

Config	Model Size (ONNX)	GPU FPS	GPU Latency (ms)
1 — FP32	11.0 MB	62.5	16.0
2 — PTQ INT8	3.1 MB	–*	–*
3 — QAT INT8	[3.0] MB	–*	–*
4 — QAT + QuantSmooth INT8	[3.0] MB	–*	–*

Table 5: Model size and GPU throughput on the NVIDIA RTX 3070 at 640×640, batch size 1. *GPU throughput for INT8 configurations could not be measured due to a software dependency conflict between the training environment (PyTorch 2.4.0 / CUDA 12.4) and available ONNX Runtime GPU builds (CUDA 13 requirement); see Limitations (Section 6).

6 Discussion

This section interprets the experimental results in relation to the four research sub-questions and the literature.

6.1 RQ1: Effect of PTQ on small-object Detection

The results from the PTQ configuration confirm the activation-outlier hypothesis of Gap 1 and Gap 2 identified in Related Work (Section 3). The extremely small objects' accuracy (AP_{eS}) saw a 35% relative decrease, which establishes that per-tensor activation quantization disproportionately damages the shallowest and most extreme small-object detections, a finding that is consistent with Jacob et al. [7] and Hubara et al. [6], the contribution being that it had not previously been measured on a dataset with XS-VID's object scale. A secondary finding is that the degradation to accuracy propagates into tracking as seen through the substantial increase to identity switches. A possible contributing factor is the fixed ByteTrack thresholds discussed in Section 4.7, outlining how the high-score (0.25) and low-score (0.1) thresholds were calibrated based on the FP32 confidence distribution and subsequently applied to all configurations unchanged. A noisier confidence distribution from any configuration could result in an object crossing over the thresholds inconsistently to where it is tracked in one frame and lost in the next, despite the object's presence being constant. ByteTrack's identity association depends on continuous linking from frame to frame, thus an object that drops below the threshold may not confidently be reassigned the same identity. This would suggest that the accuracy of the detector is not the issue as MOTA remains nearly unchanged; the instability

can therefore be accounted for by the crossing of the fixed threshold over time. A frame-to-frame confidence score variance was not separately measured in this study, so the threshold-crossing mechanism remains as a possible contributing factor and remains to be proven. The finding however would not have been visible from detection metrics alone, with the phenomenon lacking mentions in previous quantization literature which did not examine tracking effects at all (Gap 3).

6.2 RQ2: QAT Recovery

Standard QAT is expected to recover 50–80% of the PTQ accuracy gap based on literature [7]. QAT instead failed to recover any accuracy at all. No prior quantization study has documented a similar result where QAT accuracy decreases instead of recovers on extremely small objects. The existing literature was established on COCO-scale objects, for which QAT is known as a reliable configuration that does not depend on object scale for the expected recovery in accuracy. QAT underperformed the post-training quantized baseline, with AP falling from 0.162 (PTQ) to 0.106, and AP_{eS} falling from 0.022 (PTQ) to 0.018. QAT showed no recovery to the extremely small objects. Through both the XS-VID evaluation tool and ByteTrack evaluation both results showed a consistent degradation to accuracy scores and cross-validate the behaviour of fine-tuned QAT on this dataset. There are two possible explanations to consider for the lack of recovery in accuracy. The first relates to the object scale within XS-VID which has objects of size below 12 pixels which produce a low signal-to-noise ratio in the P3/P4 feature maps. The straight-through estimator (STE) used in QAT creates an estimate for the gradient of the quantization operation which loses efficacy if the signal weakens. The STE's gradient estimates may not have been able to compensate for the noise introduced during training from the fake-quantization as it interferes with the low signal, an issue that would not have been present with COCO-scale objects which have a higher underlying signal. The second possible explanation relates to the methodology of the configurations. The claim of identical training conditions being present across all configurations was unable to be met with the automatic mixed precision property (AMP). AMP was enabled for the FP32 baseline (amp = True, Ultralytics' default) but disabled for QAT (ampe = False) which was required to avoid a dtype conflict between FP16 autocast and the FakeQuantize modules' statistics as mentioned in section 4.7. QAT's performance loss cannot confidently be attributed to quantization alone until resolving the difference in training configurations. Nonetheless, the two possible factors are not mutually exclusive, confirming which contributed to the loss would require an additional controlled run of QAT modified to force AMP on if it is possible, or a repeated training run of the FP32 baseline with AMP disabled. This is identified as a priority for future work as it would isolate the AMP variable and would bring clarity to the findings of RQ2.

6.3 RQ3: QuantSmooth Neck

The QuantSmooth neck was expected to compound improvement upon the AP_{eS} recovery from the standard QAT by reducing the activation range presented to the INT8 scale factor computation. QuantSmooth Neck did not improve and rather decreased accuracy further, for every metric and object-size category. This configuration had the largest reduction to accuracy in the ablation; AP fell from 0.106 (QAT) to 0.076 and AP_{eS} fell from 0.018 to 0.006.

A possible explanation is that layer normalization and clamping applied to a quantized and fine-tuned network removes small-object signal along with the intended outliers. The network's

learned feature representations at the P3/P4 levels may already have adapted to the unnormalized distribution during QAT causing renormalization to be counterproductive. This is especially damaging as the small objects already produce low signal. Furthermore, the limitation noted in section 4 details the clamp bound set to ±3 based on the expectation that around 99.7% of the values would be within three standard deviations on a normalised distribution, however this was not modified to reflect observed activation statistics. Therefore the results do not confidently establish the activation-outlier hypothesis as false, rather that the implementation without a tuned clamp bound fails to improve accuracy for the object scale of the XS-VID dataset. The closest prior work of YOLOv6+ [10] reported the normalization strategy as a success. The difference in the implementation is that the intervention point was located at the regression head output rather than the shallow P3/P4 neck features, and the scale of the objects evaluated were COCO-scale, which are meaningfully larger than the objects in XS-VID. It is possible that the normalization at the neck is counterproductive when the signal is low, while having neutral effects at the head or with larger objects producing higher signal. To distinguish between these possible explanations would require generalization evaluation on a dataset with larger object size, such as VisDrone2019-VID [19] which was originally planned but not completed within the project timeline as discussed in Section 6, Limitations. This is the most direct way to determine whether the QuantSmooth Neck's failure is inherent to the approach or the extremely small object sizes of XS-VID.

6.4 RQ4: Throughput

All three quantized configurations saw a model size reduction against the baseline FP32. From 11.0 MB for FP32 PTQ, QAT and QAT+QuantSmooth's INT8 ONNX exports were 3.1MB, 3.0MB and 3.0MB respectively. This is beneficial for deployment, reducing the required amount of storage and memory. GPU throughput for the three quantized configurations however could not be measured due to the software dependency reasons detailed in section 5. RQ4 thus cannot be answered by the findings within this thesis and as such remains an open question of whether QAT+QuantSmooth maintains competitive throughput within 10% of FP32. The gap is more consequential due to RQ2 and RQ3 findings showing substantial accuracy loss compared to FP32. If INT8 deployment also failed to show improvement in throughput, based on the findings throughout this thesis there would be no basis for recommending either configuration over the FP32 baseline for this task. On the contrary if INT8 throughput resulted in gains over the FP32 baseline the use of QAT or QuantSmooth could be justified for use in memory or latency constrained deployment despite the accuracy costs.

The infrastructure required to answer this research question is largely in place as a calibration pipeline and successfully exported INT8 ONNX models exists for all three quantized configurations. What remains to be solved is the GPU execution environment conflict, either by benchmarking within a CUDA 13-matched software stack, or by deploying the models via TensorRT rather than ONNX Runtime as the former manages CUDA dependencies independently of the execution provider system.

6.5 Limitations

Several limitations are present in this thesis. First, the fixed ByteTrack thresholds may disadvantage INT8 configurations as discussed above. Second, the PTQ calibration set of 500 frames may not fully

represent the tail distribution of extremely small objects in XS-VID, potentially underestimating PTQ quality. Third, the QuantSmooth clamp bounds of ±3 were set theoretically rather than tuned empirically. Fourth, the quantization ablation (PTQ, QAT, QuantSmooth) is conducted on YOLOv11n only, while Section 5.3 compares FP32 baselines across YOLOv5n, YOLOv8n, and YOLOv8s to establish that YOLOv11n is a competitive model. It remains to be tested if the accuracy and tracking results observed under quantization (RQ1–RQ3) generalize to other models or model scales. Fifth, the automatic mixed precision (AMP) was not constant across all configurations, the FP32 baseline was trained with amp=True (Ultralytics' default), whilst QAT and QAT+QuantSmooth used amp = False, which was required to avoid a dtype conflict. The training conditions beyond AMP were however identical (batch size 16, 50 epochs, seed 42), however due to the inconsistency of AMP the accuracy differences cannot confidently be attributed to quantization alone without performing tests maintaining the consistency of AMP. Sixth, the throughput of INT8 GPU could not be measured for the quantized configurations due to the software dependency conflict. This issue was identified late in the project timeline and could not be resolved despite attempts to downgrade, upgrade, or unpin the Runtime builds or CUDA components. Seventh, the evaluation of generalisation on VisDrone2019-VID was planned in the initial proposal but was not completed within the project timeline. The unexpected negative results from QuantSmooth on XS-VID were difficult to debug and in the remaining time priority was given to validating the findings on the primary dataset including the cross-validation with ByteTrack tracking.

7 Conclusions and Further Research

This thesis investigated the effect of INT8 quantization on small-object detection and tracking in a YOLOv11-based video pipeline, and proposed the QuantSmooth Neck modification to recover accuracy under quantization. A four-configuration stepwise ablation on XS-VID was designed to isolate the contributions of PTQ, QAT, and the QuantSmooth Neck. The detection accuracy of all configurations at every IoU threshold and every object-size metric saw a monotonic decrease. The decrease continued through the ByteTrack tracking pipeline seeing MOTA and IDF1 falling as well. The hypothesis and part of the proposed contribution of the thesis expected the QuantSmooth Neck to recover accuracy and failed to deliver, instead degrading accuracy further. It is clear that the low signal produced by the extremely small objects of the XS-VID dataset does not benefit from the effects of quantization. The thesis's findings provide several contributions to the area of research regarding small-object detection. Firstly, the findings clearly show how INT8 quantization affects small-object detection at different object scales, identifying how extremely small objects perform poorly under these configurations. Secondly, the findings provide insight into the continuation of degradation into multi-object tracking in ByteTrack's two-pass association mechanism, revealing a previously undocumented effect of quantization noise on identity instability. Thirdly, the QuantSmooth Neck failed to achieve expected results under zero-parameter modification on objects of XS-VID's scale. Despite negative results the methods themselves contribute meaningfully to the related literature as it redirects researcher efforts into the future work identified in the section below.

Future Work

Two of the questions raised in this thesis remain to be answered. First, which is if QAT and QuantSmooth's INT8 models offer any advantages to throughput on INT8-accelerated GPU hardware, which could not be measured. This would require work into resolving the conflict of dependencies in the environment alone as the INT8 exports and calibration infrastructure were already built and would conclude RQ4 with clarity. And, the second of which being whether the QuantSmooth Neck configuration offers improvements to accuracy on object sizes larger than those in the XS-VID dataset. The latter could be verified using the VisDrone2019-VID [19] dataset which is comprised of larger object sizes, a contribution planned within the thesis, however not completed within the timeline. Additionally, the AMP configuration difference if resolved for the QAT and QAT+QuantSmooth configurations would provide a controlled environment to isolate the change and compare the findings to the baseline FP32 model.

Several other directions remain open that could provide more holistic understandings of quantization effects. Extending the ablation to INT4 quantization (cf. LSQ [3]) could test the limits of QuantSmooth under more aggressive compression. Applying the modification to larger YOLO variants would test generality across model scales. In this thesis temporal fusion (cf. YOLOFT [5]) was deliberately excluded to isolate the quantization effect, otherwise integrating it should be an improvement on detecting small objects. Re-tuning ByteTrack thresholds per quantization configuration would remove the fixed-threshold issue found in the Discussion. Finally, an empirical study of actual activation distributions at the P3/P4 levels under FP32 vs INT8 would provide direct evidence for or against the outlier hypothesis motivating the use of QuantSmooth.

References

[1] Fatih Cagatay Akyon, Sinan Onur Altinuc, and Alptekin Temizel. Slicing aided hyper inference and fine-tuning for small object detection. In *IEEE International Conference on Image Processing*, pages 966–970, 2022.

[2] Dawei Du, Yuankai Qi, Hongyang Yu, Yifan Tian, Peixi Zhao, Bo Si, Tingfan Shi, Shiliang Wen, and Haibin Ling. The unmanned aerial vehicle benchmark: Object detection and tracking. In *European Conference on Computer Vision*, pages 375–391, 2018.

[3] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. In *International Conference on Learning Representations*, 2020.

[4] Zhihao Feng et al. Deep learning-based small object detection: A survey. *Mathematical Biosciences and Engineering*, 20(4):6551–6590, 2023.

[5] Jiahao Guo, Ziyang Xu, Lianjun Wu, Fei Gao, Wenyu Liu, and Xinggang Wang. XS-VID: An extremely small video object detection dataset. *arXiv preprint arXiv:2407.18137*, 2024.

[6] Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Accurate post training quantization with small calibration sets. In *International Conference on Machine Learning*, pages 4466–4475, 2021.

[7] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018.

[8] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics YOLO. https://github.com/ultralytics/ultralytics, 2023.

[9] Toghrul Karimov, Hassan Imani, and Allan Kazakov. Quantization robustness to input degradations for object detection. *arXiv preprint arXiv:2508.19600*, 2025.

[10] Sungjei Kim. YOLOv6+: simple and optimized object detection model for INT8 quantized inference on mobile devices. *Signal, Image and Video Processing*, 19(8):665, 2025.

[11] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

[12] Changhao Li, Xinrui Chen, Ji Wang, Kang Zhao, and Jianfei Chen. Task-specific zero-shot quantization-aware training for object detection. *arXiv preprint arXiv:2507.16782*, 2025.

[13] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pages 740–755, 2014.

[15] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[16] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

[17] Chenhongyi Yang, Zehao Huang, and Naiyan Wang. QueryDet: Cascaded sparse query for accelerating high-resolution small object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 13668–13677, 2022.

[18] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Furu Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. ByteTrack: Multi-object tracking by associating every detection box. In *European Conference on Computer Vision*, pages 1–21, 2022.

[19] Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Heng Fan, Qinghua Hu, and Haibin Ling. Detection and tracking meet drones challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7380–7399, 2021.

[20] Xingkun Zhu, Shuchang Lyu, Xu Wang, and Qi Zhao. TPH-YOLOv5: Improved YOLOv5 based on transformer prediction head for object detection on drone-captured scenarios. In *IEEE/CVF International Conference on Computer Vision Workshops*, pages 2778–2788, 2021.