УДК 622.647.2:004.932

Ван Цзяньфэй, аспирант (КузГТУ, г. Кемерово),
Гордин Сергей Александрович, научный сотрудник, ассистент (КузГТУ, г. Кемерово),
Варнавский Кирилл Александрович, к.т.н., заведующий лабораторией (КузГТУ, г. Кемерово)
Jianfei Wang, PhD student (KuzSTU, Kemerovo),
Sergey A. Gordin, researcher, assistant (KuzSTU, Kemerovo),
Kirill A. Varnavskiy, PhD, head of laboratory (KuzSTU, Kemerovo)

АНАЛИЗ ВЛИЯНИЯ РАЗРЯДНОСТИ ВЫЧИСЛЕНИЙ НА ТОЧНОСТЬ НЕЙРОСЕТЕВОГО ОБНАРУЖЕНИЯ ПОВРЕЖДЕНИЙ КОНВЕЙЕРНОЙ ЛЕНТЫ

ANALYSIS OF THE EFFECT OF NUMERICAL PRECISION ON NEURAL NETWORK-BASED CONVEYOR BELT DAMAGE DETECTION

Аннотация. Сравниваются FP32, FP16 и INT8 при обнаружении повреждений конвейерной ленты моделью YOLO11n на 235 тестовых изображениях. FP16 сохраняет точность; INT8 теряет 11,8 п.п. mAP@50:95 при сжатии модели в 3,3 раза, а наибольшая потеря приходится на классы малых размеров.

Abstract. We compare FP32, FP16 and INT8 inference of a YOLO11n detector for mining conveyor belt damage on 235 test images. FP16 preserves accuracy; INT8 loses 11.8 mAP@50:95 points while shrinking the model 3.3-fold, with the largest degradation on small objects.

1 Introduction

A belt conveyor is one of the main items of mining transport equipment, and its belt is the most heavily loaded element: the surface degrades under impact loading, abrasive wear and foreign objects trapped between the belt and the rollers. A rip that starts as a few-centimetre defect can open into a full-width failure within minutes, so inspection has to be frequent enough to catch damage while it is still local [1, 2]. Camera-based inspection has largely replaced manual rounds, and convolutional detectors now report high accuracy on belt damage [3–6]. Because the camera looks at a moving belt, the detector must run on the equipment itself, on a single-board computer or an industrial PC shared with other monitoring tasks [6].

This deployment constraint, rather than detection accuracy, usually decides whether a model is usable. Reducing numerical precision is the standard way to cut memory traffic and reach the half-precision or integer pipelines of embedded accelerators [7–9]. That work is mostly reported on classification benchmarks or COCO-scale detectors [10–12]. Damage detection differs in a way that matters here: the classes are extremely unequal in the number of instances and in physical size, and the smallest defects occupy a few tens of pix-

els. Whether a precision reduction that is harmless on average is also harmless for those defects is an empirical question that classification results cannot answer.

This paper reports a controlled comparison of FP32, FP16 and INT8 inference for belt damage detection with a YOLO11n detector trained on a four-class mining dataset. All three precisions are evaluated on the same 235 test images through the same preprocessing and post-processing code, so the differences are attributable to the numerical format. We quantify the loss per class and per object-size band, measure latency and storage under matched conditions, and formulate the precision choice as a constrained selection over measured operating points.

2 Materials and Methods

2.1 Dataset. The experiments use a four-class annotated dataset of belt surface conditions recorded at 640 × 640 pixels: block (material blocking the surface), crack (cracks and incipient rips), foreign (objects lying on the belt) and hole (worn-through holes). The supplied splits were used as provided and not re-split, so no test image contributed to training, to hyper-parameter selection or to calibration (Table 1). The annotated images and split definitions are not public and can be obtained from the authors on request.

The distribution is strongly skewed: block accounts for 75.5 % of the training boxes, crack and hole for 3.3 % and 3.1 %. The minority classes also sit at opposite ends of the scale range: on the test split the median box is 427 px² for block, 17 450 px² for crack, 11 195 px² for foreign and 104 px² for hole, that is, side lengths from 10 px to 132 px. This spread, rather than the class label, is what we use in Section 3.

Table 1 – Dataset composition and annotation counts per split

Split	Images	Boxes	block	crack	foreign	hole
train	1876	10950	8271	361	1983	335
validation	234	1557	1219	43	247	48
test	235	1480	1181	44	225	30

2.2 Detector and training. The baseline is YOLO11n, a single-stage detector with 2.59 M parameters and 6.4 GFLOPs at 640 × 640 input [15]. A nano-scale detector is the realistic choice for a belt-side controller and the harder case for quantization: small models carry less redundancy, so a fixed quantization error costs relatively more accuracy [10, 12]. The network was initialised from COCO-pretrained weights and fine-tuned for 100 epochs with batch size 32, AdamW at an initial learning rate of 0.001 with cosine decay, and a fixed random seed. Training used mixed precision with a single-precision master copy of the weights [13]; the final weights are stored in FP32 and the checkpoint with the best validation mAP@50:95 was kept. The test split was evaluated once, after all variants had been produced.

2.3 Precision configurations. FP32 runs the exported graph in single precision and serves as the reference. FP16 stores weights and activations in half

precision, with input and output tensors kept in FP32 so that preprocessing and decoding are unchanged. INT8 applies static post-training quantization in QDQ form, weights per output channel and activations per tensor, both signed 8 bit, following the standard affine mapping [7, 9]:

$$s_p = \frac{r_{max} - r_{min}}{2^b - 1}, \qquad q = round\left(\frac{r}{s_p} + z\right) \qquad r_q = s_p(q - z) \tag{1}$$

Calibration used 256 training images with a fixed seed and the min–max range estimator; the test split was never used for calibration and no configuration was selected by comparing test scores. The output tensor that concatenates box coordinates with class scores was excluded from quantization, because one scale covering both ranges rounds every score to zero. No quantization-aware training was applied, so the INT8 numbers describe what a practitioner obtains without retraining.

2.4 Evaluation protocol. Detection quality is reported as mAP@50, mAP@50:95, precision and recall, with average precision computed over the ten IoU thresholds from 0.50 to 0.95 and over the four classes (Eq. 2), where C is the set of classes and T the set of thresholds.

$$mAP_{50:95} = \frac{1}{|C| \quad |T|} \sum_c \sum_t AP_c^t \tag{2}$$

Every variant is decoded by the same code: letterbox to 640 × 640, class-wise non-maximum suppression at IoU 0.65, max 300 detections per image. The mAP pre-filter is 0.001, so the metric is effectively threshold-free; precision and recall are also reported at a fixed operating point of 0.25. Recall is additionally stratified by ground-truth box area into small (< 32² px), medium (32²–96² px) and large (> 96² px) bands. Latency was measured per image after 30 warm-up iterations over 300 timed iterations (median and 95th percentile), and throughput at batch 8. Two comparisons are kept separate throughout: FP32 against FP16 on the GPU and FP32 against INT8 on the CPU, each within one runtime and one device; cross-device numbers are never used to attribute a difference to precision. The workstation was an Intel Core i5-14600KF (20 threads) with an NVIDIA GeForce RTX 5060 Ti (16 GB); GPU runs used PyTorch 2.10 with CUDA 12.8 and no TF32, CPU runs ONNX Runtime 1.23 with 20 threads.

2.5 Precision selection as a constrained problem. For a precision p, let L(p), M(p) and mAP(p) denote the measured median latency, the model size and the detection quality, with mAP_{32} the reference. The signed change against the reference, the latency gain and the storage gain are

$$\Delta_p = mAP_p^{50:95} - mAP_{32}^{50:95}, \qquad S_p = \frac{L_{32}}{L_p}, \qquad C_p = \frac{M_{32}}{M_p} \tag{3}$$

so a negative Δ is a loss. The composite $\eta = S/(1 - \Delta/mAP_{32})$ discounts the latency gain by the relative accuracy lost and equals 1 for the reference on each device. The admissible precision under an accuracy floor τ is the cheapest operating point that still meets it:

$$p^{*} = \arg\min_{p} L_{p}, \quad s.t. \quad mAP_{p}^{50:95} \geq \tau \tag{4}$$

Two floors are examined, $\tau = mAP_{32} - \delta$ with $\delta = 0.02$ (an illustrative tolerance chosen by the authors and labelled as such) and $\delta = 0.01$. Neither comes from the test set; δ states how much accuracy an operator will trade.

3 Experimental Results

3.1 Overall accuracy. FP16 reproduces the FP32 result (Table 2): mAP@50 changes from 0.749 to 0.745 and mAP@50:95 from 0.446 to 0.445, a difference of 0.11 points, while precision and recall are unchanged at three decimals. The same FP32 weights run through a second runtime (the ONNX export on the CPU) return identical mAP values (0.7489 and 0.4465), so the pipeline adds no offset of its own; the 0.11-point change is attributable to precision and lies an order of magnitude below the tolerances of Section 3.5.

INT8 behaves differently: mAP@50 falls to 0.620 and mAP@50:95 to 0.329, 12.9 and 11.8 points below FP32 (-26.4 % relative), while precision drops from 0.792 to 0.575 and recall from 0.716 to 0.640 — the model both misses defects and adds detections that are not there.

Table 2 – Detection quality on the test split (235 images, 1480 boxes)

Precision	mAP@50	mAP@50:95	P@0.25	R@0.25	Δ, pp	Rel. change
FP32	0.749	0.446	0.792	0.716	–	–
FP16	0.745	0.445	0.792	0.716	-0.11	-0.2 %
INT8	0.620	0.329	0.575	0.640	-11.79	-26.4 %

3.2 Class-level behaviour. Table 3 and Fig. 1 resolve the average into the four classes. Foreign objects are almost untouched (AP@50 0.973 → 0.962). Cracks lose 10.4 points, blocked material 17.5 and holes 22.7. Holes suffer most at the stricter criterion: AP@50:95 falls from 0.221 to 0.054, so the loss is not only in finding the defect but in placing the box. The signed AP@50 change correlates with median object size (Spearman $\rho = +0.8$ over four classes): the change becomes less negative as objects grow, with crack the exception — as large as foreign yet losing ten times more. Four points make the coefficient suggestive only.

Table 3 – Per-class average precision on the test split

Class	Median side, px	Boxes	AP@50 FP32	AP@50 FP16	AP@50 INT8	Δ, pp	AP@50:95 FP32	AP@50:95 INT8
block	20.7	1181	0.719	0.719	0.544	-17.5	0.373	0.234
crack	132.1	44	0.835	0.835	0.732	-10.4	0.463	0.394
foreign	105.8	225	0.973	0.973	0.962	-1.1	0.729	0.632
hole	10.2	30	0.469	0.452	0.242	-22.7	0.221	0.054

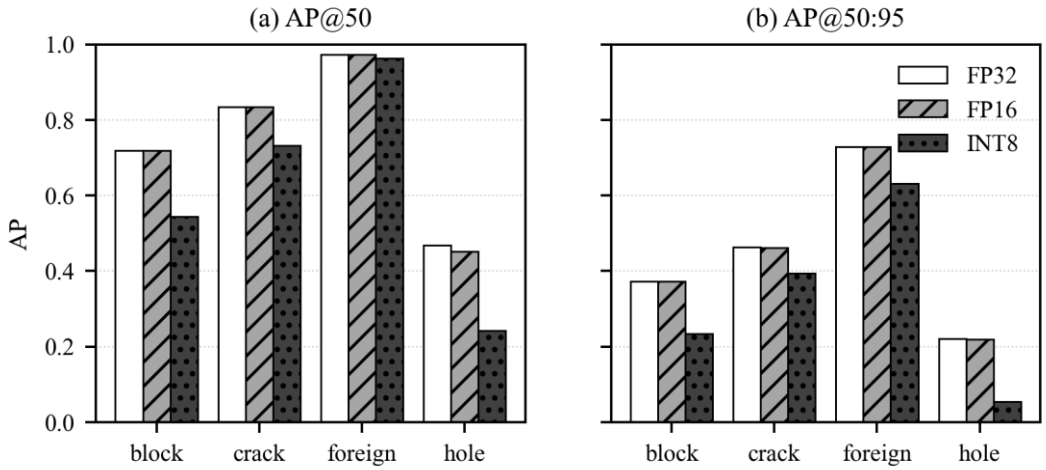

Fig. 1. Per-class average precision: (a) AP@50, (b) AP@50:95

3.3 The loss is a size effect. Stratifying recall by ground-truth box area removes the class labels from the comparison (Fig. 2). For small objects the INT8 recall at IoU 0.5 drops from 0.589 to 0.472, a loss of 11.7 points; for medium objects it is 1.4 points (0.906 → 0.892) and for large objects there is no change (0.969 in both). Small boxes are 924 of the 1480 test objects, so the aggregate loss of Table 2 is largely the small-object loss; FP16 leaves all three bands untouched.

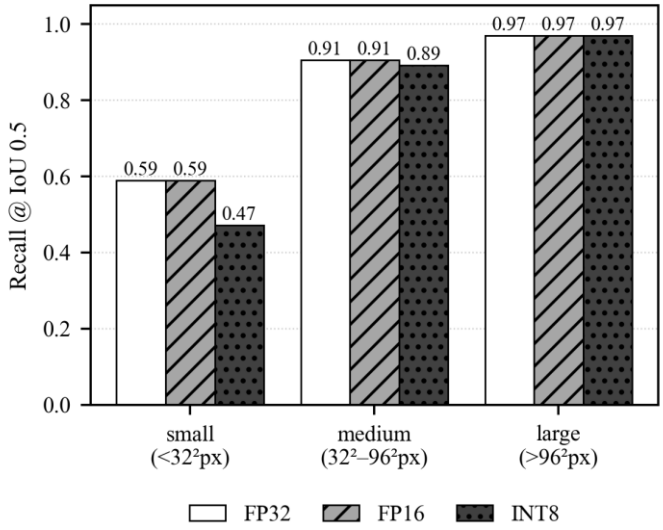

Fig. 2. Recall at IoU 0.5 stratified by object area

3.4 Cost of the three configurations. On the GPU, FP16 halves the model file (10.12 → 5.11 MiB) and is neutral for single-image latency (4.26 vs 4.36 ms, p95 4.81 vs 4.78 ms), but it doubles throughput once the batch is filled (417 → 1040 images/s at batch 8). On the CPU, INT8 compresses the model by 3.32× but is 2.8× slower than FP32 (19.8 → 55.4 ms) — Table 4.

Table 4 – Latency, throughput and storage of the measured operating points (GPU rows: PyTorch/CUDA; CPU rows: ONNX Runtime, 20 threads)

Configu-ration	Latency, ms	p95, ms	Through put, img/s	Size, MiB	Speed-up (batch 1)	Size ratio	η
GPU FP32	4.26	4.81	417	10.12	1.00	1.00	1.00
GPU FP16	4.36	4.78	1040	5.11	0.98	1.98	0.98
CPU FP32	19.84	24.83	–	10.12	1.00	1.00	1.00
CPU INT8	55.37	62.58	–	3.05	0.36	3.32	0.28

The INT8 slowdown is not a file-format artefact but a mapping problem: in the optimized QDQ graph only 10 of the 78 convolutions execute as integer convolutions; the rest keep floating-point kernels with 556 dequantization and 293 quantization nodes on their edges, because the SiLU activation, exported as a sigmoid multiplied by its input, interrupts the dequantize–conv–quantize pattern that the runtime fuses. A second quantization emitting a fully integer graph (88 integer convolutions with integer sigmoid and multiply) was not faster either, at about 116 ms per image, which points to per-tensor requantization around activations rather than to the graph format.

3.5 Precision selection under an accuracy constraint. At the two-point floor ($\tau = 0.4265$) the admissible set on the GPU contains FP32 and FP16 ($0.4454 \geq \tau$); on the CPU it contains FP32 only, because INT8 reaches 0.3286 and misses the threshold by 9.8 points. The stricter one-point floor changes nothing, and within the admissible set the optimum depends on the objective: minimising latency selects FP32 (4.26 ms), minimising storage or maximising batched throughput selects FP16 (5.11 MiB, 2.5×). INT8 becomes admissible only if a loss of roughly twelve mAP points is accepted.

4 Discussion

Half precision costs almost nothing here. FP16 keeps 11 significant bits instead of 24 and narrows the exponent range to roughly $\pm 6 \cdot 10^4$, so both weights and activations are perturbed; what the measurement shows is that at this operating point the perturbation moves mAP@50:95 by 0.11 points and leaves precision and recall unchanged at the operating threshold. No activation statistics were recorded, so we claim no distributional guarantee: the result is empirical and specific to this network and dataset. What FP16 returns is a model file of half the size and, once the pipeline is batched, 2.5 times the throughput.

INT8 post-training quantization is not free, and its cost is concentrated where it hurts. Eleven points of mAP@50:95 is a large loss for a calibration-only procedure, and recall below 32^2 px falls by eleven points while large objects are unaffected. In belt monitoring these are the defects worth catching early, so preserving large rips while losing small holes addresses the wrong end of the risk curve. The pattern agrees with reports that smaller detectors degrade

more under static quantization [12] and with the structure of per-tensor activation quantization, where one scale per feature map is set by the largest response while small objects occupy few anchors with low-magnitude activations; we state that as the plausible mechanism, not as a measured fact.

The third observation concerns efficiency. On the tested stack the configuration with the smallest file was the slowest, by a factor of nearly three. Neither bit width nor compression ratio predicted latency [14]; what mattered was whether the layers could be mapped to integer kernels, which for a SiLU-based detector they mostly could not. The same contingency appears on the GPU, where half precision changed nothing at batch 1 and everything at batch 8, because a single image through a 2.6 M-parameter network is too small to fill the device. Any claim of the form "precision X gives speed-up Y" is a claim about a stack and has to be measured on it. For this task the rule is to deploy in FP16 wherever the device exposes half-precision arithmetic, and to keep FP32 when single-image latency on a CPU-class device is the only criterion. INT8 should not be adopted on the strength of its storage advantage alone; if the requirement is limited to large longitudinal damage, crack AP@50 still reaches 0.73 under INT8, so a system that only has to catch rips could tolerate it — a concession the operator has to make explicitly.

One detector family and one dataset were examined; the nano scale was chosen as the realistic choice for a belt-side controller and the harder case for quantization, but the numbers should not be transferred to larger backbones. Quantization-aware training was not attempted; the INT8 figures are calibration-only post-training quantization. No power measurement was available, so we make no claim about energy: here INT8 latency rose while its footprint fell, and lower latency would not imply lower energy in any case [16]. The test split also holds only 44 crack and 30 hole instances, so those class-level numbers carry a sampling uncertainty the size-stratified analysis does not.

5 Conclusion

For a YOLO11n detector on a four-class mining dataset, evaluated once on 235 held-out images, half precision is safe: it moves mAP@50:95 by 0.11 points, far below the tolerances used for the deployment decision, halves the model file and leaves single-image latency unchanged on the GPU while raising batched throughput by a factor of 2.5. INT8 post-training quantization is not safe at this scale: it costs 11.8 mAP@50:95 points and the loss is concentrated on small objects (11.7 points of recall below 32^2 px against none above 96^2), which are the defects an early-warning system exists to catch. Its 3.3× storage advantage came with a 2.8× latency penalty on the CPU stack tested, because the quantized graph could not be mapped to integer kernels for most convolutions.

The rule that follows is conditional and cheap: state the tolerance, measure the candidates on the target stack, and pick within the admissible set by the binding constraint. For this task FP16 satisfies a two-point tolerance and is rec-

ommended wherever half precision is available; FP32 remains the choice when single-image latency on a CPU is the only criterion; INT8 needs quantization-aware training or a better integer backend, and its acceptance should be judged per size band, since the average hides exactly the loss that matters.

References

[1] Zhang M., Shi H., Zhang Y., Yu Y., Zhou M. Deep learning-based damage detection of mining conveyor belt // Measurement. 2021. Vol. 175. 109130. DOI 10.1016/j.measurement.2021.109130.

[2] Hao X.-L., Liang H. A multi-class support vector machine real-time detection system for surface damage of conveyor belts based on visual saliency // Measurement. 2019. Vol. 146. P. 125–132. DOI 10.1016/j.measurement.2019.06.025.

[3] Chen Y., Zhou M., Hu F., Wen Y. A framework integrating multispectral imaging and deep learning for accurate detection of conveyor belt surface damage // Measurement. 2025. Vol. 256. 118353. DOI 10.1016/j.measurement.2025.118353.

[4] Chen Y., Zhou M., Hu F., Gao L., Wang K. YOLOv8-LDH: a lightweight model for detection of conveyor belt damage based on multispectral imaging // Measurement. 2025. Vol. 245. 116675. DOI 10.1016/j.measurement.2025.116675.

[5] Yang Q., Li F., Tian H., Li H., Xu S., Fei J., Wu Z., Feng Q., Lu C. A new knowledge-distillation-based method for detecting conveyor belt defects // Applied Sciences. 2022. Vol. 12, No. 19. 10051. DOI 10.3390/app121910051.

[6] Klippel E., Oliveira R.A.R., Maslov D., Bianchi A.G.C., Delabrida S.E., Garrocho C.T.B. Embedded edge artificial intelligence for longitudinal rip detection in conveyor belt applied at the industrial mining environment // SN Computer Science. 2022. Vol. 3, No. 4. DOI 10.1007/s42979-022-01169-y.

[7] Jacob B., Kligys S., Chen B., Zhu M., Tang M., Howard A., Adam H., Kalenichenko D. Quantization and training of neural networks for efficient integer-arithmetic-only inference // Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR). 2018. P. 2704–2713. DOI 10.1109/CVPR.2018.00286.

[8] Krishnamoorthi R. Quantizing deep convolutional networks for efficient inference: a whitepaper // arXiv:1806.08342. 2018.

[9] Wu H., Judd P., Zhang X., Isaev M., Micikevicius P. Integer quantization for deep learning inference: principles and empirical evaluation // arXiv:2004.09602. 2020.

[10] Nagel M., Fournarakis M., Amjad R.A., Bondarenko Y., van Baalen M., Blankevoort T. A white paper on neural network quantization // arXiv:2106.08295. 2021.

[11] Gholami A., Kim S., Dong Z., Yao Z., Mahoney M.W., Keutzer K. A survey of quantization methods for efficient neural network inference // arXiv:2103.13630. 2021.

[12] Karimov T., Imani H., Kazakov A. Quantization robustness to input degradations for object detection // 2025 Innovations in Intelligent Systems and Applications Conference (ASYU). IEEE, 2025. P. 1–6. DOI 10.1109/ASYU67174.2025.11208310.

[13] Micikevicius P., Narang S., Alben J., Diamos G., Elsen E., Garcia D., Ginsburg B., Houston M., Kuchaiev O., Venkatesh G., Wu H. Mixed precision training // Proc. ICLR. 2018. arXiv:1710.03740.

[14] Han S., Mao H., Dally W.J. Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding // Proc. ICLR. 2016. arXiv:1510.00149.

[15] Jocher G., Qiu J., Chaurasia A. Ultralytics YOLO11 [software]. 2024. https://github.com/ultralytics/ultralytics.

[16] Horowitz M. Computing's energy problem (and what we can do about it) // Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC). 2014. P. 10–14. DOI 10.1109/ISSCC.2014.6757323.