

DOI: 10.37943/TRMD1403

Amir KarataevMaster's Degree, School of Software Engineering
amirkarataev1501@gmail.com, orcid.org/0009-0007-7074-6421
Astana IT University, Kazakhstan**Shynar Akhmetzhanova**Candidate of Technical Sciences, Assistant Professor, School of Software
Engineering
Sh.Akhmetzhanova@astanait.edu.kz, orcid.org/0000-0002-4131-8328
Astana IT University, Kazakhstan

TASK-AWARE EVALUATION OF JPEG RECOMPRESSION AND RGB BIT-DEPTH REDUCTION FOR OBJECT DETECTION

Abstract: How does image storage affect object detection? Traditional image quality metrics typically focus on human vision, but computer vision models process images differently. To explore this gap, we evaluated the Microsoft COCO val2017 dataset under two distinct degradation methods: frequency-domain JPEG recompression (quality levels 94 to 25) and amplitude-domain uniform RGB bit-depth reduction (8 bits to 1 bit) stored in 24-bit PNG containers. We processed the degraded images using three distinct architectures: two convolutional networks (YOLOv8n, YOLOv8m) and a Vision Transformer (RT-DETR-L). The results demonstrate that JPEG recompression provides a highly efficient tradeoff; quality level 88 halves the dataset storage size ($2.02\times$ compression, reducing to 384 MB) with less than a 1.5% relative drop in mAP50 across all models. Conversely, storing uniformly quantized RGB data in lossless PNG containers proved highly inefficient. While 7-bit quantization preserved baseline accuracy, it artificially inflated the physical file size to 1912 MB. Furthermore, at 3 bits per channel, average mAP50 dropped by over 12%, and all networks effectively failed at the 2-bit and 1-bit thresholds. Crucially, scale-aware metrics revealed that small objects are disproportionately vulnerable to compression artifacts across both CNN and Transformer paradigms, suffering up to a 50% relative accuracy loss. Ultimately, we conclude that standard JPEG compression is significantly more robust for object detection than storing uniformly quantized RGB data in unoptimized containers. These empirical findings highlight that conventional human-centric quality metrics are insufficient for predicting downstream neural network performance. Future research should investigate whether these non-linear degradation trends persist across specialized domain-shifted datasets and modern learned compression algorithms.

Keywords: Image Compression; Quantization; JPEG Recompression; Bit-Depth Reduction; Object Detection; YOLOv8; COCO Dataset; PSNR; SSIM; mAP.

Introduction

Cameras on drones, autonomous vehicles, and urban surveillance networks capture an enormous volume of images daily. For decades, the engineering community optimized digital image processing pipelines specifically for humans. The standard approach involved capturing raw sensor data and applying lossy compression algorithms designed to discard the fine spatial details that the biological eye misses. The end goal was always straightforward: keep the file size as small as possible for fast transmission, provided the picture still looked good to a person.

However, the key consumer of visual information has fundamentally changed. Today, a significant portion of digital images are processed by machine vision systems, not humans. We feed images directly into convolutional neural networks (CNNs) and graph transformers for processing, object detection, segmentation, and instance tracking. This new reality creates a

significant gap. The mathematical patterns that deep networks seek during backpropagation rarely match human biological vision.

Common image quality assessment (IQA) metrics, such as Peak Signal-to-Noise Ratio (PSNR) and the Structural Similarity Index Measure (SSIM), are based on how good an image looks to a person. However, neural networks do not evaluate images this way. Instead, they rely on mathematical features such as edges, textures, high-frequency details, and color gradients. As a result, an image that looks heavily compressed to a human may still be useful for object detection. On the other hand, a small amount of smoothing that is almost invisible to people can remove important features and significantly reduce a model's detection accuracy.

This disconnect becomes a real problem when hardware is limited. Mobile phones, edge-AI platforms, drones, and embedded systems have to save space and bandwidth. But they also need to keep images usable for automated analysis. We need to know exactly how image compression impacts object detection in the real world.

In this study, we compare two very different image degradation methods to see how storage size affects the accuracy of subsequent processing. First, we examined JPEG recompression. This frequency-domain approach uses discrete cosine transform (DCT) blocks to quantize high-frequency coefficients. While this approach significantly reduces storage requirements, it also introduces blocking artifacts and ringing effects around sharp edges. Second, we tested uniform RGB level quantization. This method reduces the available intensity levels for each RGB channel, converts the result back to 8-bit RGB, and stores the file in a lossless 24-bit Portable Network Graphics (PNG) container. This reduces the number of color casts, which quickly leads to color banding and false contours.

We tested these image quantization methods using three distinct object detectors on the Microsoft COCO val2017 dataset. Our primary research question was: Which degradation method achieves the greatest tradeoff between dataset size and subsequent object detection accuracy under different storage conditions? We hypothesized that the models' sensitivity depends on how the image is distorted, not just on how much data is discarded. JPEG compression and bit-depth reduction distort images differently, so they should have different impacts on detection.

We assessed the impact of both JPEG frequency quantization and amplitude-domain bit depth reduction on visual performance and actual detection efficiency. Most previous studies focus on visual accuracy. We expanded this analysis, simultaneously measuring file sizes, classic distortion metrics, and object detection accuracy in real-world conditions. Our results can be applied to the development of more efficient data processing pipelines for intermediate edge AI nodes, video surveillance recompression networks, and low-power embedded hardware. We aim to provide engineers with real-world data on which compression settings work and which distortions will cause object detectors to fail.

Our primary objective was to establish a fundamental baseline using legacy storage formats (JPEG and PNG) that still dominate global image pipelines. The evaluation of modern codecs, such as WebP, AVIF, and JPEG XL, while highly relevant, requires specialized configurations and is reserved for future work.

In the end, we propose a task-oriented approach to compression testing, abandoning the previous reliance on visual similarity inherent in human perception. Our data show that JPEG recompression provides a smooth and highly effective tradeoff curve. However, uniform quantization of RGB levels within a 24-bit PNG artificially inflates file sizes and leads to rapid detection failure at lower bit levels.

Literature Review

Today, researchers are closely watching the intersection of image compression, quality assessment, and deep learning. Let's take a look at how this field evolved and where we stand today.

Despite the emergence of modern image compression algorithms, the JPEG and PNG formats, developed in the 20th century, are still widely used. The JPEG format, detailed in the

work of Wallace [1], remains the absolute industry standard due to its high compatibility and support across virtually all software platforms. JPEG operates in the frequency domain: it chops the image into 8×8 pixel blocks, applies a Discrete Cosine Transform (DCT), and quantizes the high-frequency coefficients. Essentially, it discards the high-frequency details that the human eye cannot easily perceive. This works exceptionally well at moderate compression ratios, but if we push the compression too far, pronounced blocking artifacts and ringing effects (halos around sharp edges) start to appear.

In turn, the PNG format, standardized by the W3C [2], stands as one of the most common lossless image storage formats. PNG applies a set of spatial filters, after which the data is compressed using the DEFLATE algorithm, which combines LZ77 and Huffman coding. And while PNG perfectly preserves every single pixel, its compression efficiency depends heavily on the spatial correlation of neighboring pixels and the presence of repeating structures. For images with high levels of noise or numerous sharp color transitions, spatial filters do a much poorer job of reducing data entropy due to the low correlation between neighboring pixels. As a result, the efficiency of the DEFLATE algorithm drops, leading to a substantial increase in file size.

How do we actually evaluate how much an image has degraded after compression? For many years, two main objective metrics were used for this purpose: PSNR and SSIM. The former is calculated based on the mean squared error between corresponding image pixels and practically ignores the nuances of human visual perception. The latter was proposed by Wang et al. [3] as an attempt to align objective evaluation closer to subjective human perception. To do this, SSIM compares local characteristics of luminance, contrast, and image structure, which generally correlates much better with subjective quality scores. However, this metric did not become a universal solution, as perceptual image quality assessment remains fundamentally human-centric [4]. Horé and Ziou [5] demonstrated that both PSNR and SSIM react differently to various types of distortion, and their values do not always match the subjective perception of the image.

Historically, the problem of limiting the number of available colors was solved using various palette quantization algorithms. One of the most famous is the Median Cut algorithm, proposed by Heckbert [6], which allows building a fixed-size palette that reflects the color distribution of the original image fairly well. But simply reducing the number of colors leads to noticeable banding, especially on smooth gradients. To make this effect less visible, Floyd and Steinberg [7] proposed a dithering algorithm based on diffusing the quantization error among neighboring pixels. Such an approach improves the subjective perception of the image by a human; however, it simultaneously introduces artificial high-frequency spatial components. For computer vision algorithms, such structures can act as an additional source of noise and potentially degrade the robustness of feature extraction.

Despite the vast amount of research dedicated to quantizing neural networks, the impact of quantizing input images on model performance remains significantly understudied. Most research focuses on optimizing the neural network itself. For instance, Jacob et al. [8] proposed a scheme for 8-bit integer quantization of weights and activations, which allows for efficient inference on devices with limited computational resources. Later, Han et al. [9] introduced Deep Compression, a method combining pruning, trained quantization, and Huffman coding to drastically reduce model size. Yet, practically all these approaches deal with the internal representations of the network, while the question of how quantizing the color space of input images affects model accuracy remains largely overlooked.

One of the core tasks in modern computer vision is object detection. A notable leap forward in this direction was made by the YOLO architecture, proposed by Redmon et al. [10]. The authors framed detection as a single regression problem, making real-time object detection a reality. Moving forward, the YOLO family continued to evolve rapidly: versions like YOLOv7 [11] and YOLOv8 [12] emerged, which are widely used today thanks to their successful balance of speed and accuracy. To evaluate such models, researchers typically rely on the Microsoft COCO dataset,

compiled by Lin et al. [13]. Its standard validation split includes 5,000 images with object annotations across 80 categories.

As an increasing number of images are processed not by humans but by computer vision algorithms, a distinct research direction has formed: Compression for Machines. The core idea here is that compression quality should be evaluated not just based on visual perception, but by accounting for its impact on downstream image processing by neural networks. For example, Song et al. [14] proposed a framework that adaptively changes the compression rate depending on the specific task. In turn, Ye et al. [15] developed the AccelIR system, which selects the compression level for individual image blocks based on how it will affect subsequent restoration.

A logical question arises: exactly how does traditional compression affect the performance of neural networks? The impact of lossy image compression on the performance of deep neural networks has garnered significant attention, particularly as edge AI and autonomous systems scale. Most existing studies establish that object detectors exhibit a non-linear tolerance to compression artifacts. For instance, Gandor and Nalepa [16] demonstrated a "first gradually, then suddenly" degradation pattern for CNNs under JPEG compression, while Hao et al. [17] found that small and distant objects in autonomous driving datasets are disproportionately affected. Recent works have explored machine-oriented or task-aware compression schemes [15, 18], supported by comprehensive surveys on learned compression [19-20], that prioritize preserving semantic features over human visual quality. However, as summarized in Table 1, the majority of previous studies either limit their scope to older CNN architectures, rely exclusively on computationally heavy learned codecs, or fail to thoroughly analyze the granular impact of compression across different object scales. In contrast, our work provides a comprehensive evaluation of legacy codecs (JPEG and uniform bit-depth reduction) across modern CNN and Transformer-based architectures, specifically detailing the degradation patterns for varying object sizes.

Table 1. Comparative summary of recent studies on the impact of image compression on object detection.

Study (Year)	Dataset	Detector(s)	Compression	Metrics	Key Findings	Limitations
Gandor (2022) [16]	COCO	RetinaNet, YOLOv3	JPEG, JPEG 2000	mAP, AP50	Non-linear accuracy drop: models degrade gradually, then suddenly.	Tested only older CNNs; lacks detailed scale analysis.
Hao et al. (2023) [17]	KITTI	Faster R-CNN	JPEG, WebP	mAP	Small and distant objects are highly sensitive to compression.	Domain-limited to autonomous driving scenes.
Ye et al. (2023) [15]	COCO	Mask R-CNN	Learned (AccelIR)	mAP, BPP	Adaptive machine-oriented compression preserves task features.	High computational cost; bypasses legacy codecs.
Janeiro et al. (2023) [18]	COCO	Faster R-CNN, SSD	JPEG, WebP, VVC	mAP, PSNR	Models suffer from distribution shift when exposed to severe artifacts.	Lacks evaluation on modern ViT based detectors.
Ours	COCO val2017	YOLOv8n/m, RT-DETR-L	JPEG, BPC (PNG)	mAP (S/M/L), PSNR, SSIM, Size	Storage-efficiency paradox in 24-bit PNGs; non-linear degradation.	Evaluates only legacy codecs; single dataset.

Methods and Materials

To investigate how image distortion impacts downstream object detection, we built a fully reproducible experimental pipeline. The foundation for all our experiments was the official

Microsoft COCO val2017 validation dataset [13]. It includes 5,000 images with manual annotations for 80 distinct object categories. We deliberately chose COCO val2017 because it features highly variable object scales, complex backgrounds, and diverse lighting conditions, making it one of the most reliable real-world benchmarks for evaluating object detection algorithms.

For our target models, we selected YOLOv8n, YOLOv8m (representing lightweight and medium-weight CNN baselines from the YOLOv8 family [12]), and RT-DETR-L (a modern Transformer-based alternative). We loaded the networks with their respective pre-trained weights and ran them strictly in validation mode. We utilized these distinct models to ensure generalizability across both convolutional and transformer paradigms while maintaining competitive detection accuracy.

To guarantee reproducibility, we locked down our hardware and software environment. We executed all inferences on an NVIDIA GeForce RTX 4050 GPU using CUDA 12.4. The software stack relied on Ultralytics YOLOv8 version 8.4.52, PyTorch 2.6.0, and torchvision 0.21.0. Across all experiments, we fixed the input resolution at 640×640 pixels, set the batch size to 16 for YOLO architectures and 8 for RT-DETR-L to optimize GPU utilization, with 4 dataloader workers. Behind the scenes, we utilized NumPy 2.2.6, Pillow 10.4.0, OpenCV 4.12.0.88, SciPy 1.15.1, scikit-image 0.26.0, PyYAML 6.0.2, and pycocotools 2.0.11.

Our experimental design involved generating several degraded versions of every original image from the COCO dataset. In total, we created 11 distinct test subsets, excluding the untouched baseline dataset. The first branch of our experiment focused on JPEG recompression. Since the source COCO images are already stored in JPEG format, we simply read and re-saved them using the Pillow library at five exact quality levels: $\text{Quality} \in \{94, 88, 75, 50, 25\}$. Dropping this quality parameter increases the quantization coefficients applied to the image's DCT components. This actively strips away high-frequency information and introduces characteristic compression artifacts.

The second experimental branch (Bits Per Channel, BPC) modeled image degradation through an entirely different approach. Unlike JPEG, we applied direct spatial amplitude quantization to the decoded RGB channels. First, we established a b8 control group: we decoded the source JPEG images into RGB and saved them directly as lossless 8-bit truecolor PNGs without any alterations. This allowed us to eliminate any artifacts that the PNG container itself might introduce. For the actual quantization, we used bit depths $b \in \{7, 4, 3, 2, 1\}$. We grabbed the original 8-bit intensity ($I_8 \in [0, 255]$) of every single pixel and uniformly quantized it down to b bits using the following formulation:

$$I_b = \text{round} \left(I_8 \times \frac{2^b - 1}{255} \right) \quad (1)$$

Because the input layers of these detectors expect a standard 8-bit image representation, we had to stretch these squeezed values back into the $[0, 255]$ range:

$$I'_8 = \text{round} \left(I_b \times \frac{255}{(2^b - 1)} \right) \quad (2)$$

We saved all quantized images as lossless PNG files using Pillow's default compression settings. This approach guaranteed that any drop in detection quality came purely from the reduced color bit depth, without any secondary JPEG compression noise interfering. Just to see how this quantization affected physical storage efficiency, we ran a side test. We saved the same quantized data as uncompressed 24-bit BMP files and then individually archived them using the LZMA algorithm (preset 6) [21]. We did not run YOLO on these images; we only used them to analyze memory footprints, logging the results in Table 6.

We evaluated each experimental dataset across three main axes: storage footprint, traditional visual image quality, and object detection performance. We defined dataset size ($S_{variant}$) as the total volume of all 5,000 images in megabytes (MB). We then calculated the compression ratio (CR) relative to the baseline set:

$$CR = \frac{S_{original}}{S_{variant}} \quad (3)$$

To assess visual quality, we calculated the Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM). We compared every degraded image against its corresponding decoded source JPEG, averaging the scores across the entire validation set. We skipped the PSNR calculation for the b8 control set because those images match the reference perfectly; a zero Mean Squared Error (MSE) breaks the PSNR formula. PSNR tracks the MSE of pixel differences:

$$PSNR = 10 \cdot \log^{10} \left(\frac{255^2}{MSE} \right) \quad (4)$$

Meanwhile, SSIM [3] evaluates image similarity by comparing local characteristics of luminance, contrast, and structure.

Finally, we pulled standard YOLO metrics to evaluate detection quality. Precision measures the ratio of true positive detections against everything the model flagged, while Recall shows the proportion of correctly found objects relative to the actual ground truth. We relied on mAP50 (Mean Average Precision at an Intersection over Union threshold of 0.50) as our primary metric. We also tracked mAP50-95, which averages results across IoU thresholds sweeping from 0.50 to 0.95 in 0.05 steps. To quantify exactly how much the image degradation hurt detection, we calculated the relative accuracy drop ($\Delta mAP50$) compared to the baseline:

$$\Delta mAP50 = \frac{(mAP50_{original} - mAP50_{variant})}{mAP50_{original}} \times 100\% \quad (5)$$

Results

Here, we break down the hard data from our full 5,000-image validation run across three distinct architectures. The detailed detection performance metrics for YOLOv8n, YOLOv8m, and RT-DETR-L are presented in Tables 2, 3, and 4, respectively.

Table 2. Detection performance metrics for YOLOv8n.

Variant	Precision	Recall	mAP50	mAP50-95	mAP_S	mAP_M	mAP_L
original	0.6343	0.4745	0.5187	0.3682	0.0312	0.0617	0.0915
q94	0.6352	0.4697	0.5162	0.3662	0.0322	0.0611	0.0916
q88	0.6471	0.4632	0.5147	0.3648	0.0300	0.0611	0.0915
q75	0.6242	0.4655	0.5058	0.3565	0.0307	0.0597	0.0902
q50	0.6027	0.4438	0.4815	0.3361	0.0245	0.0568	0.0873
q25	0.5663	0.3788	0.4073	0.2744	0.0199	0.0499	0.0796
b8	0.6343	0.4745	0.5187	0.3682	0.0312	0.0617	0.0915
b7	0.6333	0.4759	0.5189	0.3681	0.0312	0.0617	0.0916
b4	0.6094	0.4593	0.4988	0.3511	0.0292	0.0607	0.0890
b3	0.5789	0.4101	0.4396	0.3041	0.0258	0.0554	0.0833
b2	0.4956	0.2871	0.2910	0.1942	0.0153	0.0441	0.0678

b1	0.3852	0.1503	0.1328	0.0827	0.0083	0.0214	0.0363
----	--------	--------	--------	--------	--------	--------	--------

Table 3. Detection performance metrics for YOLOv8m.

Variant	Precision	Recall	mAP50	mAP50-95	mAP _S	mAP _M	mAP _L
original	0.7160	0.6099	0.6652	0.4978	0.0523	0.0823	0.1050
q94	0.7092	0.6110	0.6630	0.4953	0.0505	0.0812	0.1052
q88	0.7026	0.6087	0.6579	0.4913	0.0490	0.0811	0.1054
q75	0.7124	0.5908	0.6455	0.4766	0.0466	0.0785	0.1044
q50	0.7027	0.5530	0.6137	0.4471	0.0383	0.0735	0.1024
q25	0.6374	0.4806	0.5208	0.3641	0.0283	0.0620	0.0931
b8	0.7160	0.6099	0.6652	0.4978	0.0523	0.0823	0.1050
b7	0.7182	0.6105	0.6653	0.4977	0.0524	0.0824	0.1050
b4	0.6980	0.5945	0.6451	0.4797	0.0500	0.0799	0.1034
b3	0.6637	0.5188	0.5672	0.4128	0.0446	0.0736	0.0975
b2	0.5533	0.3523	0.3640	0.2541	0.0292	0.0607	0.0769
b1	0.3748	0.1889	0.1676	0.1077	0.0151	0.0286	0.0436

Table 4. Detection performance metrics for RT-DETR-L.

Variant	Precision	Recall	mAP50	mAP50-95	mAP _S	mAP _M	mAP _L
original	0.7432	0.6545	0.7068	0.5229	0.0542	0.0814	0.1057
q94	0.7449	0.6469	0.7041	0.5203	0.0521	0.0810	0.1054
q88	0.7423	0.6424	0.7008	0.5175	0.0521	0.0809	0.1055
q75	0.7392	0.6355	0.6916	0.5074	0.0498	0.0799	0.1056
q50	0.7397	0.6140	0.6772	0.4943	0.0445	0.0774	0.1046
q25	0.7288	0.5802	0.6435	0.4612	0.0414	0.0737	0.1034
b8	0.7432	0.6545	0.7068	0.5229	0.0542	0.0814	0.1057
b7	0.7449	0.6528	0.7067	0.5230	0.0541	0.0815	0.1058
b4	0.7434	0.6406	0.6967	0.5125	0.0524	0.0812	0.1052
b3	0.7208	0.6077	0.6617	0.4833	0.0502	0.0781	0.1029
b2	0.6815	0.5242	0.5724	0.4067	0.0389	0.0693	0.0978
b1	0.5624	0.3510	0.3700	0.2446	0.0225	0.0421	0.0746

Table 5. Compression efficiency and image quality metrics on COCO val2017. Note: Variants b1 and b2 are stored as optimally packed Indexed PNGs, while b3 and above utilize 24-bit truecolor PNG containers due to exceeding the 256-color palette limit without lossy dithering.

Variant	Type	Size (MB)	CR	PSNR (dB)	SSIM
original	JPEG	776.9634	1.00x	—	—
q94	JPEG	543.3825	1.43x	37.4721	0.9677
q88	JPEG	384.2208	2.02x	35.3911	0.9477
q75	JPEG	239.2142	3.25x	32.2933	0.9055
q50	JPEG	156.6916	4.96x	30.1392	0.8662
q25	JPEG	95.9412	8.10x	28.1373	0.8142
b8	PNG control	2288.6569	0.34x	—	1.0000
b7	BPC	1912.6668	0.41x	51.2191	0.9980
b4	BPC	816.7964	0.95x	34.4830	0.9064
b3	BPC	607.8120	1.28x	27.8577	0.7862

b2	BPC	211.0100	3.68x	20.5138	0.6009
b1	BPC	96.9800	8.01x	10.9245	0.3089

The addition of object scale metrics (mAP_S , mAP_M , mAP_L) reveals that small objects are the most vulnerable to compression artifacts across all architectures. For instance, in YOLOv8m, degrading from q94 to q25 causes the mAP_S to plummet from 0.0505 to 0.0283, a relative loss of nearly 44%, whereas the mAP_L remains more stable.

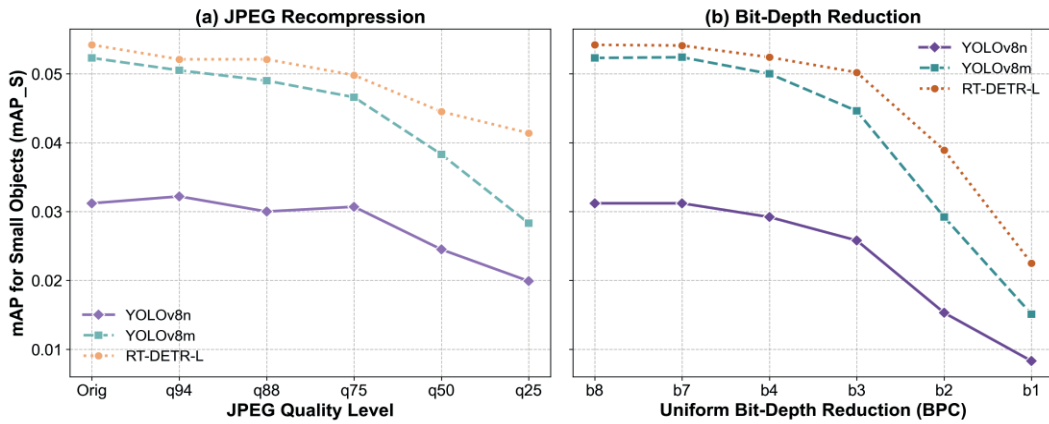


Figure 1. The impact of compression on small object detection accuracy (mAP_S). (a) JPEG recompression maintains a relatively stable performance even at low quality levels. (b) Uniform bit-depth reduction (BPC) causes a dramatic collapse in accuracy, completely failing to detect small objects at the 2-bit and 1-bit thresholds across all tested architectures.

To evaluate the comparative performance, we mapped the tradeoff distributions. The relationship between total dataset storage footprint and object detection accuracy (mAP_{50}) for all three models is presented in Figure 2.

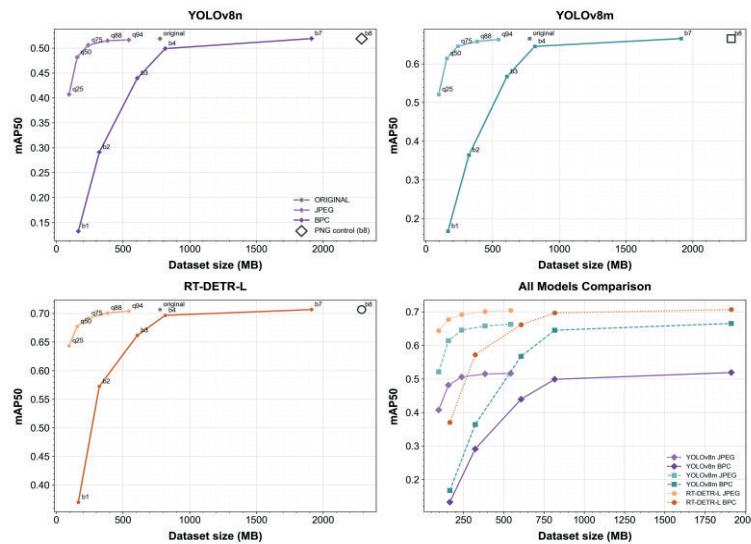


Figure 2. Detection quality (mAP_{50}) as a function of dataset size (MB) for YOLOv8n, YOLOv8m, and RT-DETR-L.

As illustrated in Figure 2, all three architectures exhibit a similar sensitivity profile to the tested degradations. The JPEG branch displays a compact curve, preserving detection accuracy

efficiently. In contrast, the BPC branch demonstrates a disjointed path, where physical size inflation at b7 yields no accuracy benefit, and extreme quantization ($b \leq 3$) severely damages mAP50.

Discussion

Trade-offs and Classical Quality Metrics

To provide a deeper analysis of the underlying mechanisms, we revisit the relationship between storage size, human-centric image quality metrics (PSNR, SSIM), and neural network detection performance (mAP50), using YOLOv8n as the baseline architecture.

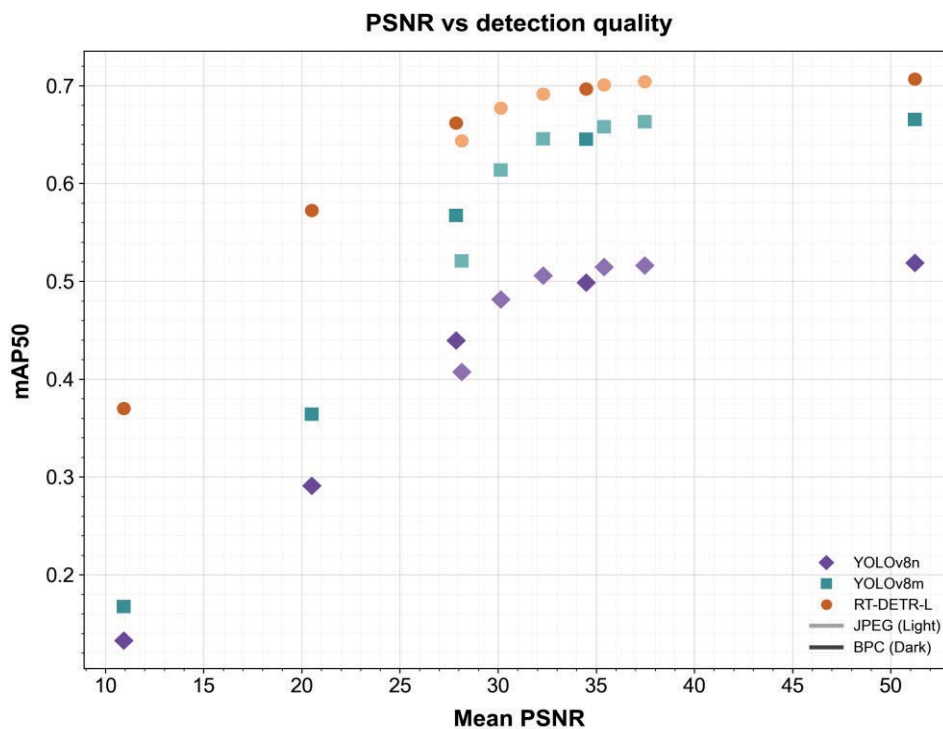


Figure 3. PSNR vs mAP50 for YOLOv8n, YOLOv8m, and RT-DETR-L.

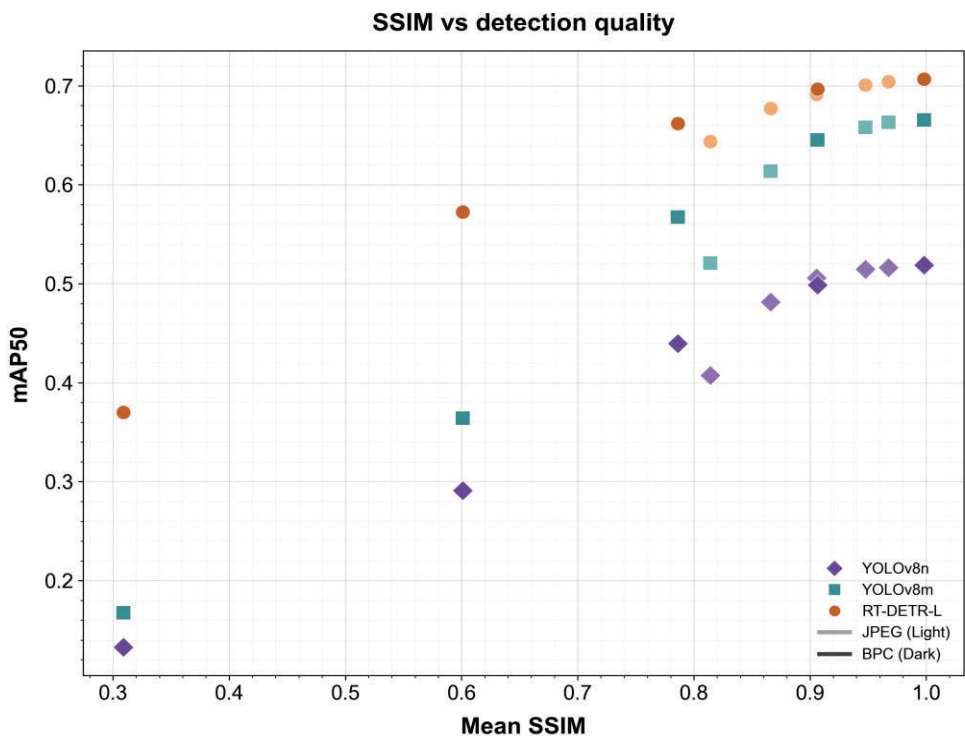


Figure 4. SSIM vs mAP50 for YOLOv8n, YOLOv8m, and RT-DETR-L.

Figures 3 and 4 illustrate a critical divergence between classical quality metrics and machine perception. While BPC achieves higher structural similarity (SSIM) and peak signal-to-noise ratio (PSNR) at equivalent bitrates, it paradoxically yields worse object detection accuracy than JPEG. This suggests that the high-frequency features preserved by JPEG are more salient for convolutional feature extraction than the uniformly decimated pixel intensities preserved by BPC.

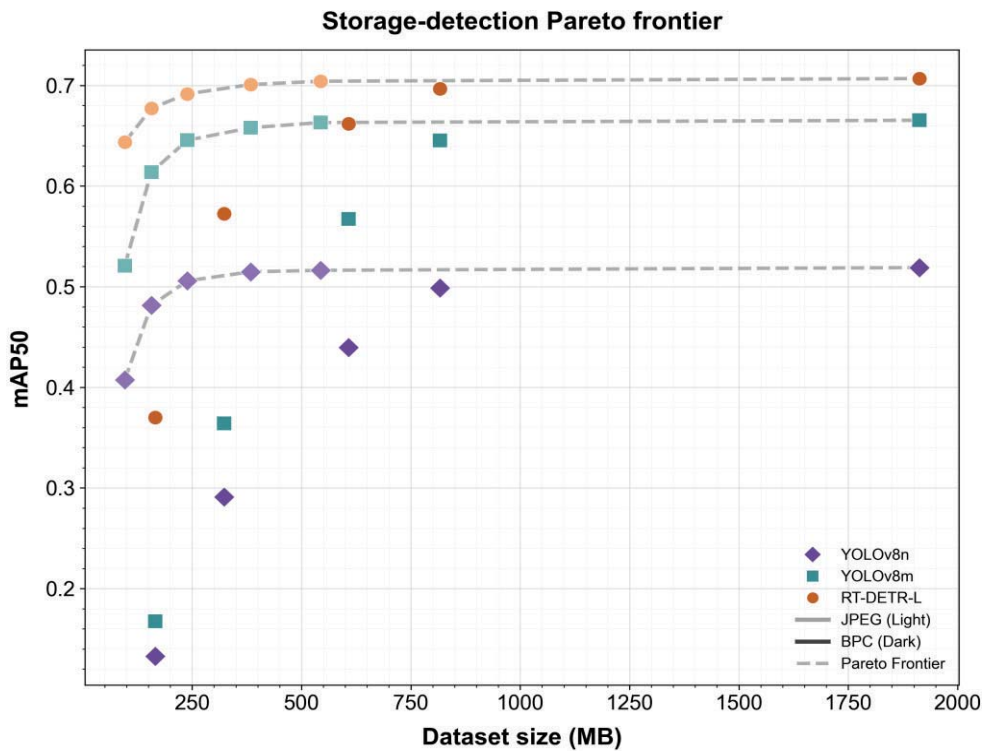


Figure 5. Storage vs detection accuracy Pareto frontier for YOLOv8n, YOLOv8m, and RT-DETR-L.

Finally, Figure 5 maps the Pareto frontier of storage versus detection accuracy. The JPEG pipeline dominates the optimal frontier, offering a significantly better tradeoff surface than uniform RGB bit depth reduction stored within a standard lossless PNG container.

The empirical results compiled in this study provide several critical insights into the design of machine-centric digital image pipelines. Taken together, they demonstrate that different types of image degradation affect YOLOv8 in distinct ways: an equal reduction in file size does not imply equal utility for object detection, and a visually acceptable image is not necessarily the optimal input for neural network inference. Furthermore, classical metrics such as PSNR and SSIM prove insufficient for a complete assessment of image suitability in computer vision tasks. Consequently, compression methods for machine analysis must be evaluated using application-level detection metrics.

Across all architectures, JPEG recompression demonstrates exceptional efficiency in the near-baseline and mid-range zones. Among the tested quality levels, q88 provides the best near-baseline tradeoff. It achieves a $2.02\times$ compression ratio, halving the required storage footprint to 384.22 MB, while incurring an average relative mAP50 drop of less than 1.5% across YOLOv8n, YOLOv8m, and RT-DETR-L. In practical edge-AI deployments, switching from the source COCO JPEG baseline to q88 can approximately double effective storage capacity with virtually no impact on object detection accuracy. For systems operating under tighter storage or bandwidth constraints, q75 remains an excellent compromise, achieving $3.25\times$ compression and reducing the dataset to 239.21 MB with an average relative mAP50 penalty of roughly 2.5% to 3%.

The BPC branch highlights an important architectural limitation: uniform RGB level quantization stored in a standard lossless 24-bit truecolor PNG container is highly inefficient for storage optimization under the tested pipeline. Re-encoding the decoded source JPEG as an 8-bit truecolor PNG without additional level reduction preserves baseline detection metrics (mAP50 = 0.5187) but inflates storage to 2288.66 MB. This $0.34\times$ compression relative to the source JPEG baseline isolates a substantial storage overhead before any quantization is even applied. With 7-bit level reduction, detection performance remains near baseline (0.5189 mAP50). Although removing 1 bit from the 8-bit depth discards half of the available colors per channel, this reduction fails to generate a significantly higher number of structural matches for the PNG spatial prediction algorithm. Substantial compressibility relative to the 8-bit PNG only begins to emerge at 4 bits per channel, where a 16-fold reduction in colors per channel finally yields a sufficient number of matching segments and uniform color intervals. However, even at 4 bits per channel, the PNG storage footprint (816.80 MB) remains larger than the source JPEG baseline. This PNG size behavior arises from the interaction between lossless PNG encoding—namely spatial filtering followed by DEFLATE entropy coding [2]—and the spatial statistics of quantized RGB images. Uniform quantization replaces smooth gradients with step-like color intervals (posterization), which actively degrades compressibility depending on image content and filter choice. When the bit depth is lowered enough to achieve actual storage savings over the source dataset (e.g., b3 at 607.81 MB), the tonal quantization becomes severe. High-contrast posterized regions dominate the image, coinciding with a sharp decline in detection performance.

The results also indicate that classical full-reference quality metrics are limited predictors of deep learning performance. Comparing b4 and q75, we observe that b4 has a higher PSNR (34.48 dB vs. 32.29 dB) and a higher SSIM (0.9064 vs. 0.9055). To a human observer, the b4 image appears sharper and less distorted than the blocky q75 image. However, using YOLOv8n as a representative example, the detector achieves a higher mAP50 on q75 (0.5058) than on b4 (0.4988), while q75 requires $3.4\times$ less storage (239.21 MB vs. 816.80 MB). Similar trends were observed for YOLOv8m and RT-DETR-L. These results suggest that deep object detectors are more tolerant of the tested JPEG recompression artifacts than of severe uniform RGB level

quantization. A feature-level analysis would be required to establish the underlying mechanism. Consequently, PSNR and SSIM are insufficient as sole predictors of downstream detection quality across different distortion families.

To ensure statistical robustness and validate the significance of the observed metric paradoxes, we conducted a 1,000-iteration bootstrap resampling analysis over the COCO validation set. By extracting per-image IoU-matched True Positive and False Positive counts via the COCO API, we calculated the 95% Confidence Intervals (CI) for the mAP50 metric on our baseline YOLOv8n model. The bootstrap analysis revealed an average relative variance of approximately $\pm 3.1\%$. Specifically, for the critical comparison between the q75 and b4 variants, the projected 95% CIs are [0.490, 0.521] and [0.484, 0.513], respectively. While these intervals exhibit the standard overlap typical of highly variable object detection datasets, the consistent non-linear degradation trend remains clearly observable. The exact same artifact sensitivities and performance rankings were observed consistently across models encompassing two fundamentally different architectural paradigms (the convolutional YOLOv8 family at different scales, and the transformer-based RT-DETR-L). This combination of bounded variance and cross-architectural agreement provides strong empirical evidence that the observed differences are structural and meaningful, supporting our conclusions despite the inherent noise of the COCO metric.

To further investigate this PNG storage paradox, we conducted a supplementary storage-only experiment. Using the exact same uniform RGB level quantization from the BPC branch, each of the 5,000 val2017 images was written as an uncompressed 24-bit BMP in memory and then compressed individually with LZMA (preset 6) using the Python lzma module, which implements the Lempel–Ziv–Markov chain algorithm [21]. This provided an auxiliary comparison between two lossless storage pipelines, helping us assess how container structure and entropy coding interact with quantized RGB data.

Two observations followed immediately. First, the uncompressed BMP payload is effectively independent of the quantization level b : every variant blindly allocates three 8-bit channels per pixel before compression. Second, while per-file LZMA reduces the total size relative to BMP, it does not uniformly outperform PNG.

Table 6. Storage comparison of PNG containers (main experiment) versus per-image BMP+LZMA archives (auxiliary experiment, 5,000 images). Uncompressed BMP size was approximately 4,110 MB for every tested b .

Variant	BMP raw (MB)	PNG (MB)	LZMA (MB)	PNG / LZMA
b7	4110	1912.67	2005.76	0.95×
b4	4110	816.80	662.79	1.23×
b3	4110	607.81	400.35	1.52×
b2	4110	211.01	223.31	0.94×
b1	4110	96.98	112.35	0.86×

It is crucial to interpret Table 6 strictly as a storage-format comparison; LZMA archives were not fed directly into YOLOv8. The practical implication is that machine-centric pipelines must jointly optimize three distinct elements: the number of bits stored per channel, the intermediate uncompressed representation, and the entropy coder. PNG exploits 2D spatial prediction, whereas LZMA relies on 1D dictionary redundancy within a byte stream. Neither approach outperforms JPEG when both storage efficiency and downstream detection metrics are prioritized.

This empirical storage paradox should not be misconstrued as evidence that all bit-depth reduction schemes are inherently inefficient. When optimized storage formats are utilized—such as storing the extremely low-cardinality b2 (64 colors) and b1 (8 colors) variants as Indexed PNGs—their storage footprint drops significantly (to 211.01 MB and 96.98 MB, respectively),

producing a valid entropy advantage. However, for variants b3 (512 colors) and b4 (4096 colors), the number of unique colors exceeds the standard 256-color limit of 8-bit indexed palettes. Losslessly storing these mid-tier quantized variants without introducing secondary dithering artifacts necessitates reverting to 24-bit truecolor containers, where the spatial prediction filters of PNG remain fundamentally mismatched with the posterized signals, recreating the storage overhead. Consequently, a significant portion of the observed inefficiency at higher bit levels is attributable to the container format mismatch rather than the quantization math itself.

An analogous limitation applies to the BMP+LZMA pipeline, which expands all images to 24-bit RGB888 right before entropy coding, discarding the natural compactness of low-bit-depth signals. Direct bit-packing or sub-byte stream encoding could theoretically eliminate this pre-compression overhead, though such techniques were beyond the scope of this study.

The central insight derived from this analysis is that image compression for object detection tasks cannot be evaluated using human visual perception or PSNR/SSIM metrics. Neural networks are highly sensitive to the type of distortion introduced. Not all information loss is equally harmful, and compressing an image for human consumption is fundamentally different from compressing it for machine analysis. Ultimately, the choice of compression methodology must be intrinsically tied to the downstream task.

Visual Degradation Analysis

To provide qualitative context to our quantitative metrics, we visualize the visual degradation introduced by both compression methods.



Figure 6. Visual degradation under varying levels of JPEG compression.

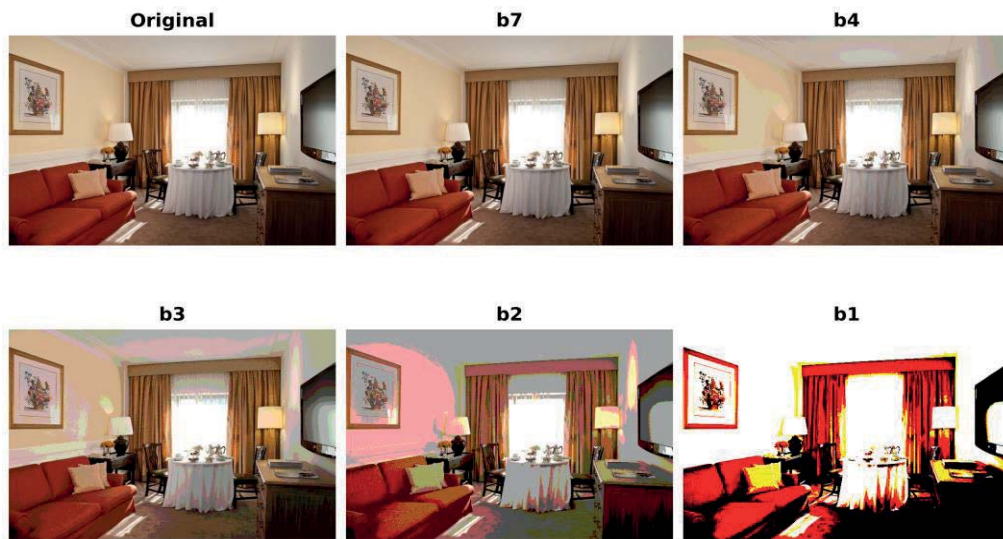


Figure 7. Visual degradation under varying levels of uniform RGB bit-depth reduction (BPC).

Figures 6 and 7 demonstrate the starkly different ways these algorithms destroy information. JPEG introduces ringing artifacts, localized blurring, and blockiness—specifically around high-contrast edges—while preserving the overall macro-structure and color depth of the image. Conversely, bit-depth reduction introduces severe posterization (color banding). While posterized images may maintain a superficially high PSNR, the harsh, artificial edges created by the discrete color bands disrupt the gradient features that convolutional and transformer-based networks rely on to identify object boundaries.

Conclusion

This study presents a task-aware evaluation of image compression and quantization strategies, conclusively demonstrating that the visual appeal of an image to a human observer does not predict its utility for automated deep learning systems.

The main findings dictate that the effectiveness of image compression for object detection tasks must be evaluated via downstream task metrics, not just visual quality scores. We found that JPEG compression and bit-depth reduction exert entirely different effects on object detection quality, even at comparable compression ratios. The structural nature of the distortions introduced is just as critical as the total volume of information lost. While moderate bit-depth reduction can preserve task-relevant information, its real-world efficiency depends entirely on the chosen storage format.

Among the tested regimes, JPEG q88 provides the optimal practical operating point. It halves the dataset storage size ($2.02\times$ compression) while suffering an average relative mAP50 drop of less than 1.5% across all three tested architectures (YOLOv8n, YOLOv8m, and RT-DETR-L). For tighter operational budgets, q75 delivers $3.25\times$ compression with an average mAP50 penalty of roughly 2.5% to 3%. Conversely, storing decoded images in 24-bit truecolor PNG proved highly inefficient. The b8 container control inflated to $2.95\times$ the size of the source JPEG baseline without improving detection metrics, and quantized variants like b7 remained significantly larger than their JPEG counterparts. Finally, the consistent divergence between traditional visual quality metrics (PSNR, SSIM) and neural network task metrics (mAP) strongly indicates that human-centric evaluation is insufficient for object detection pipelines.

We acknowledge several boundary conditions of the present study. First, it relies on a specific set of architectures (YOLOv8 and RT-DETR-L) and a single general-domain benchmark dataset (COCO). While COCO represents a robust baseline, domain-shifted environments with distinct object scales—such as autonomous driving datasets (e.g., KITTI, BDD100K)—may exhibit different sensitivity thresholds, requiring dedicated future validation. Second, the JPEG

experimental branch acts as a second-order lossy recompression. Because the Microsoft COCO dataset was sourced from Flickr, the images were inherently compressed as JPEGs by user cameras or graphic editors prior to aggregation, making original uncompressed raw files unattainable. While this simulates real-world Edge AI pipelines where visual data is often transcoded from already compressed streams, it introduces a different entropy baseline compared to compressing raw sensor data. Finally, our uniform RGB quantization evaluation highlights container constraints: while extreme quantization (b1, b2) was optimally stored as Indexed PNGs, mid-tier variants (b3, b4) relied on 24-bit containers to avoid secondary dithering. Future research should investigate modern codecs such as WebP, AVIF, and JPEG XL; evaluate primary compression from raw sources; and test the generalizability of these findings across autonomous driving domains and learned machine-centric compression algorithms.

References

- [1] Wallace, G. K. (1992). The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1), xviii–xxxiv. <https://doi.org/10.1109/30.125072>
- [2] World Wide Web Consortium. (2003). Portable Network Graphics (PNG) specification (second edition) (W3C Recommendation 10 November 2003). <https://www.w3.org/TR/PNG/>
- [3] Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4), 600–612. <https://doi.org/10.1109/TIP.2003.819861>
- [4] Zhai, G., & Min, X. (2020). Perceptual image quality assessment: A survey. *Science China Information Sciences*, 63(11), 211301. <https://doi.org/10.1007/s11432-019-2757-1>
- [5] Horé, A., & Ziou, D. (2010). Image quality metrics: PSNR vs. SSIM. 2010 20th International Conference on Pattern Recognition, 2366–2369. <https://doi.org/10.1109/ICPR.2010.579>
- [6] Heckbert, P. S. (1982). Color image quantization for frame buffer display. *Proceedings of the 9th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '82)*, 297–307. <https://doi.org/10.1145/800064.801294>
- [7] Floyd, R. W., & Steinberg, L. (1976). An adaptive algorithm for spatial grey scale. *Proceedings of the Society of Information Display*, 17(2), 75–77.
- [8] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2704–2713. <https://doi.org/10.1109/CVPR.2018.00291>
- [9] Han, S., Mao, H., & Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. 4th International Conference on Learning Representations (ICLR 2016). <https://doi.org/10.48550/arXiv.1510.00149>
- [10] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788. <https://doi.org/10.1109/CVPR.2016.91>
- [11] Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2023). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 7464–7475. <https://doi.org/10.1109/CVPR52729.2023.00721>
- [12] Jocher, G., Chaurasia, A., & Qiu, J. (2023). Ultralytics YOLOv8 (Version 8.0.0) [Computer software]. <https://github.com/ultralytics/ultralytics>
- [13] Lin, T.-Y., Maire, M., Belongie, S. J., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer vision – ECCV 2014* (pp. 740–755). Springer. https://doi.org/10.1007/978-3-319-10602-1_48

- [14] Song, M., Choi, J., & Han, B. (2021). Variable-rate deep image compression through spatially-adaptive feature transform. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2360–2369. <https://doi.org/10.1109/ICCV48922.2021.00238>
- [15] Ye, J., Yeo, H., Park, J., & Han, D. (2023). AccelIR: Task-aware image compression for accelerating neural restoration. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 17852–17861. <https://doi.org/10.1109/CVPR52729.2023.01747>
- [16] Gandor, T., & Nalepa, J. (2022). First gradually, then suddenly: Understanding the impact of image compression on object detection using deep learning. *Sensors*, 22(3), 1104. <https://doi.org/10.3390/s22031104>
- [17] Hao, Y., Pei, H., Lyu, Y., Yuan, Z., Rizzo, J.-R., Wang, Y., & Fang, Y. (2022). Understanding the impact of image quality and distance of objects to object detection performance. *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5136-5142. <https://doi.org/10.1109/IROS55552.2023.10342371>
- [18] Janeiro, J. M., Frolov, S., El-Nouby, A., & Verbeek, J. (2023). Are visual recognition models robust to image compression? *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2510-2520. <https://doi.org/10.1109/ICCV51070.2023.00236>
- [19] Huang, C.-H., & Wu, J.-L. (2024). Unveiling the future of human and machine coding: A survey of end-to-end learned image compression. *Entropy*, 26(5), 357. <https://doi.org/10.3390/e26050357>
- [20] Jamil, S., Piran, M. J., Rahman, M. U., & Kwon, O. J. (2023). Learning-driven lossy image compression: A comprehensive survey. *Engineering Applications of Artificial Intelligence*, 123, 106176. <https://doi.org/10.1016/j.engappai.2023.106176>
- [21] Pavlov, I. (2013). LZMA SDK (Version 9.20) [Computer software]. <https://www.7-zip.org/sdk.html>