A. Related Works

This section provides a brief overview of the studies relevant to our work, focusing on data-driven quantization and zero-shot quantization.

Data-driven Quantization Post-training quantization (PTQ) and quantization-aware training (QAT) [27, 44] are the most commonly employed quantization methods. PTQ methods typically utilize a small calibration set, often a subset of the training data, to optimize or fine-tune quantized networks [17, 18]. For instance, AdaRound [42] introduced a layer-wise adaptive rounding strategy, challenging the quantizers of rounding to the nearest value. Additionally, BRECQ [30] implemented block-wise and stage-wise reconstruction techniques, striking a balance between layer-wise and network-wise approaches. QDrop [53] innovatively proposed randomly dropping activation quantization during block construction to achieve more uniformly optimized weights. Despite their simplicity and minimal data requirements, PTQ methods often face challenges related to local optima due to the limited calibration set available for fine-tuning. On the other hand, most QAT approaches leverage the entire training dataset to quantize networks during the training process [25]. PACT [8] introduced a parameterized clipping activation technique to optimize the activation clipping parameter dynamically during training, thereby determining the appropriate quantization scale. LSQ [15] proposed estimating the loss gradient of the quantizer's step size and learning the scale parameters alongside other network parameters. LSQ+ [2], an extension of the LSQ method, introduced a versatile asymmetric quantization scheme with trainable scale and offset parameters capable of adapting to negative activations. Both QAT and PTQ methods rely on training data for quantization, rendering them impractical when faced with privacy or confidentiality constraints on the training data.

Zero-shot Quantization Zero-Shot Quantization (ZSQ) is a valuable approach that eliminates access to real training data during the quantization process. Presently, most ZSQ research is confined to classification tasks. Data-free quantization (DFQ) represents a subset of ZSQ methods that enable quantization without relying on any data. For instance, DFQ [41] introduced a scale-equivariance property of activation functions to normalize the weight ranges across the network. SQuant [19] developed an efficient data-free quantization algorithm that does not involve back-propagation, utilizing diagonal Hessian approximation. However, due to the absence of data, DFQ methods may not be suitable for low-bit-width configurations. For example, in the case of 4-bit MobileNet-V1 on ImageNet, SQuant achieved only 10.32% top-1 accuracy. Another branch of ZSQ methods leverages synthetic data [5, 29] generated by the full-precision network. GDFQ [50] introduced a knowledge-matching generator to synthesize label-oriented data using cross-entropy loss and batch normalization statistics (BNS) alignment. TexQ [6] emphasized the detailed texture feature distribution in real samples and devised texture calibration for data generation. Recently, the latest works extended ZSQ to more downstream tasks including object detection. PSAQ-ViT V2 [32] introduce an adaptive teacher–student strategy to the patch similarity metric in PSAQ-ViT and generate task-agnostic images to fine-tune the quantized model with knowledge distillation for segmentation and object detection. Similarly, MimiQ [10] proposed inter-shead attention similarity and apply head-wise structural attention distillation to align the attention maps of the quantized network to those of the full-precision teacher across downstream tasks. CLAMP-ViT [48] employed a two-stage approach, cyclically adapting between data generation and model quantization for classification and object detection tasks. However, the synthetic images they use in downstream tasks are task-agnostic, without containing the specific information required for the corresponding task. Our work validates task-specific image lifting performance of quantized model, yielding state-of-the-art results in ZSQ for object detection.

B. Implementation Details

We report mAP and mAP50 on the validation sets of MS-COCO 2017 [33] and Pascal VOC [16] for the object detection task. Our replication experiments not only contain one-stage YOLO networks such as classic YOLOv5 and recent YOLO11 networks, but also include two-stage Mask R-CNN networks with both ResNet and Transformer-based backbones. All experiments are conducted using a pre-trained model as the teacher. The YOLOv5/YOLO11 experiments are executed on two NVIDIA GeForce RTX 4090 GPUs, while the Mask R-CNN/ViT experiments are conducted on 4 and 8 GPUs, respectively.

B.1. Adaptive Label Sampling

While theoretically merging labels and image updates into a single stage seems feasible, our experiments in Section C.1 indicate that a continuously evolving target can negatively impact the quality of the final generated images. To address this issue, we first conduct a rapid adaptive label sampling at a low resolution (160) and then use the fixed labels to generate high-resolution images (640). We also provide details of the initial label random generation in adaptive label sampling in Table 6, and an outline of the overall algorithm in Algorithm 1. Fig. 5 provides a visual representation of the algorithm's process.

Table 6. Bounding box sampling details: we start by sampling one object **Y** for each image, where **C** represents the number of categories. We assume that the relative width and height of the image are both 1. W_{min} and H_{min} are set to 0.2, while W_{max} and H_{max} are set to 0.8. $\mathcal{U}$ denotes uniform distribution.

Variable	Sampling Distribution	Description
Y[i,0]	-	Batch index
Y[i,1]	$\mathcal{U}(0, C)$	Category
Y[i,2]	$\mathcal{U}(W/2, 1 - W/2)$	Bounding box x-center
Y[i,3]	$\mathcal{U}(H/2, 1 - H/2)$	Bounding box y-center
Y[i,4]	$\mathcal{U}(W_{\text{min}}, W_{\text{max}})$	Bounding box width
Y[i,5]	$\mathcal{U}(H_{\text{min}}, H_{\text{max}})$	Bounding box height

Algorithm 1 Adaptive Label Sampling Algorithm

Input: current stage image and labels $\{img, tgts\}$, pre-trained detection network $teacher$, filtering threshold: confidence $conf\_thresh$, iou iou_thresh

1. $new\_tgts = teacher(img).predictions[conf > conf\_thresh]$
2. $ious = \text{IOU}(new\_tgts, tgts)$

Add labels that do not overlap with the existing labels.

3. $add\_tgts = new\_tgts[(\max(ious, dim = 1) < iou\_thresh)]$

Remove labels from the existing list that are not detected by teacher.

4. $minus\_tgts = (\max(ious, dim = 0) < iou\_thresh).bool()$
5. $tgts = tgts[\sim minus\_tgts]$
6. $tgts = \text{cat}([tgts, add\_tgts], dim = 0)$

B.2. Calibration Set Generation

We apply Eq. 6 and set the optimal trade-off parameters for $\{\alpha_{detect}, \alpha_{BN}, \alpha_{TV}, \alpha_{l_2}\}$ as $\{0.5, 0.01, 0, 5\text{e-}4\}$ for the YOLOv5 series model, $\{1\text{e-}3, 1\text{e-}3, 0, 5\text{e-}5\}$ for the YOLO11 series model, $\{5.0, 2\text{e-}3, 0, 1.5\text{e-}5\}$ for CNN-backbone Mask R-CNN model and {10.0, 1.0, 0, 1e-3} for Transformer-backbone Mask R-CNN model. We generate X_{inv} by optimizing the cost function for 2500/3000/8000/4000 iterations for YOLOv5 series model, YOLO11 series model, CNN-backbone Mask R-CNN model and Transformer-backbone Mask R-CNN model respectively. We use Adam as the optimizer with an initial learning rate of 1e-2, adjusted by cosine annealing [36]. We also use cutout [12] as a data augmentation method to enhance the diversity of the synthetic calibration set.

B.3. Quantization Aware Training

Since the original LSQ is only evaluated in classification tasks on ImageNet, we extended it to object detection tasks. For each of our networks, LSQ is attached to all internal layers except the first and last layers following [15]. Our training data are from the synthesized calibration set aforementioned. During QAT, we use per-tensor symmetric quantization for both activations and weights and learn the quantization scaling/bias factor via backpropagation with the $Adam$ optimizer. The learning rate for YOLOv5, YOLO11, CNN-backbone Mask R-CNN and Transformer-backbone Mask R-CNN are 1e-4, 1e-5, 1e-4 and 1e-6 respectively. Other experimental hyper-parameters follow official implementations. We use Eq. 9 as our loss function, with optimized hyper-parameters for $\{\beta_{detect}, \beta_{KL}, \beta_{feat}\}$ being $\{0.04, 0.1, 1.0\}$ for the YOLOv5 series model, { 0.01, 0.1, 1.0 } for the YOLO11 series model, $\{0.04, 0.2, 0.1\}$ for CNN-backbone Mask R-CNN model, and $\{1.0, 1.0, 1.0\}$ for Transformer-backbone Mask R-CNN model. After training, we report mAP/mAP50 as our results.

C. Ablation Study

C.1. Adaptive Label Sampling Stage

We conduct ablations on the impact of the sampling stage number for the YOLOv5-s model, results are shown in Table 7. Overall, the two-stage sampling strategy outperforms the one-stage strategy, which we attribute to the continuous variation of targets causing fluctuation in the regression targets of the image, thus hindering stable convergence. It also matches the performance of the three-stage approach. Ultimately, we opt for the two-stage strategy to strike a balance between performance and cost.

Table 7. Ablations on Adaptive Label Sampling stages number. One stage: update images and labels simultaneously in one process. Two stages: Sample out labels first, then synthesize images with fixed labels. Three stages: Generate images with one random label first, then sample out labels with fixed images, and finally synthesize images with fixed labels

Stages Num	Precision	mAP	mAP50
One	W6A6	30.6	48.8
	W5A5	25.2	41.1
	W4A4	**16.0**	**27.9**
Two	W6A6	**32.1**	**50.1**
	W5A5	**26.3**	**42.3**
	W4A4	15.8	28.1
Three	W6A6	31.7	49.3
	W5A5	26.1	42.5
	W4A4	15.7	27.8

C.2. Calibration Set Size

After hyper-parameters are fixed, the calibration set size S is searched for its optimal trade-off between computation cost and effectiveness with grid search by quantizing YOLOv5-s to 4-8 bits, as displayed in Table 8. When S reaches 2k, the performance of the quantized network approaches convergence. Further increasing the size will lead to increased data generation time and computational costs.

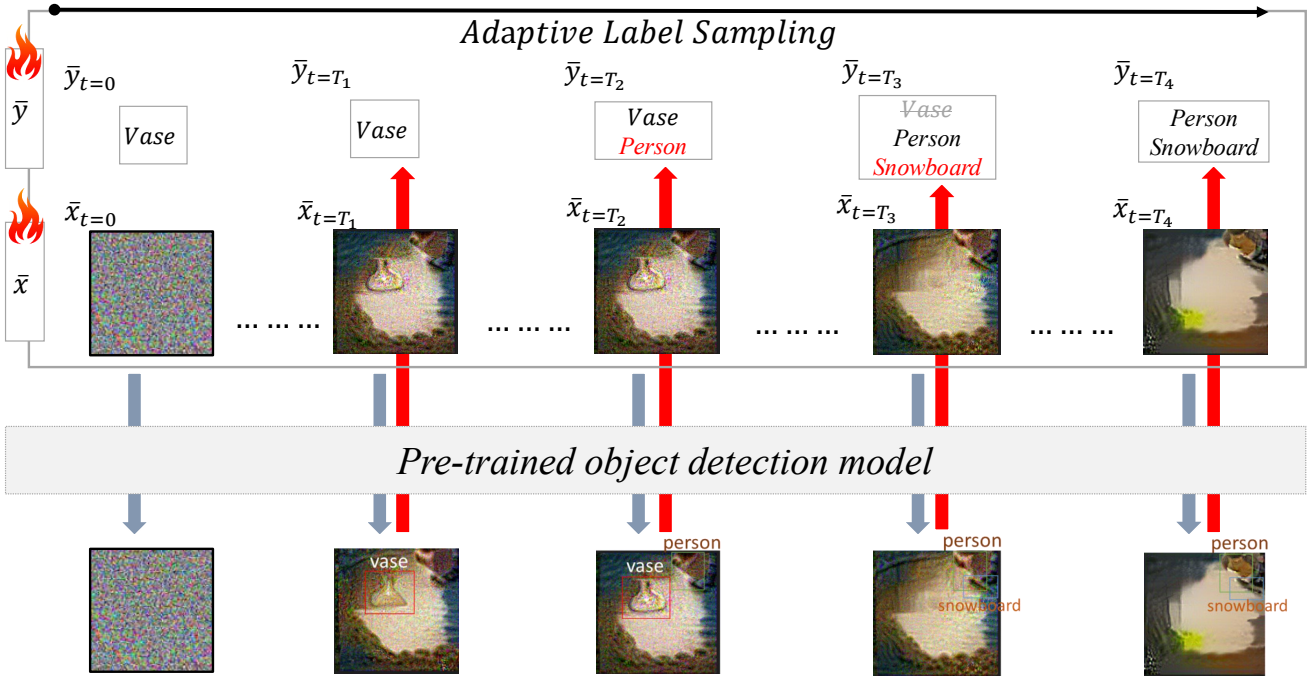


Figure 5. An overview of **Adaptive Label Sampling** process. We randomly initialize the label $\bar{y}$ and initialize the input image $\bar{x}$ using Gaussian noise. For Every fixed interval, we use a pre-trained object detection model to re-detect objects in $\bar{x}$ and update the target $\bar{y}$. In the subsequent iterations, $\bar{x}$ is optimized toward the updated target $\bar{y}$. We observe that, over iterations, $\bar{x}$ and $\bar{y}$ become increasingly aligned with each other.

To avoid complex searches, the same S is used for all experiments. While this may not be optimal for all networks, it is sufficient to demonstrate the superiority of our approach.

C.3. Modules

Ablation on key modules of the QAT stage including $\mathcal{L}_{KD}$ (Kullback-Leibler Divergence), $\mathcal{L}_{detect}$, and $\mathcal{L}_{feat}$ is conducted. As presented in Table 9, dropping one or two of them results in a mAP loss. The largest mAP loss (7.2%) occurs when both $\mathcal{L}_{KD}$ and $\mathcal{L}_{feat}$ are removed, indicating their cooperative relationship: $\mathcal{L}_{feat}$ constrains features of network layers, facilitating $\mathcal{L}_{KD}$ to align the network's predictions with the targets.

C.4. Full Comparison with Data-Free Methods

In this section, we present a comparison of our adaptive label sampling method with other baseline methods under the data-free scenario across multiple precision levels. From the results in Table 10, we observe that despite being restricted from accessing specific real label information or distribution details, our method consistently outperforms other data-free approaches. Moreover, it achieves results comparable to those obtained using real labels across different precision levels.

C.5. Comparison with YOLOv5 at Lower Precision

In Table 11, we provide additional performance results of the YOLOv5 series models on the MS-COCO dataset at lower precision levels. Notably, even at ultra-low 4-bit precision, our method still outperforms LSQ trained with full data in most cases. For example, on YOLOv5-l, our approach surpasses the best baseline by 1.7%, with the gap further increasing to 6.3% at 5-bit precision.

C.6. Comparison with Other ZSQ methods

In this section, we compare our approach with two widely-used zero-shot quantization (ZSQ) methods, Genie [23] and ZeroQ [3], to highlight the impact of incorporating task-specific information. Notably, both baselines are **task-agnostic** and therefore **lack essential knowledge** specific to object detection. As shown in Table 12, task-specific information plays a critical role in quantization-aware training (QAT). While all methods perform comparably under W4A8 post-training quantization (PTQ), our method achieves a significant **3.3% improvement in mAP after QAT**, outperforming all task-agnostic baselines.

Table 8. A detailed analysis of calibration set size S across different bit widths using YOLOv5-s on MS-COCO validation set.

Method	Real Data	S	mAP				
			W4A4	W5A5	W6A6	W7A7	W8A8
LSQ	✓	120k (Full)	23.3	26.9	31.5	33.4	35.7
Ours	×	5k	19.1	**28.0**	32.6	34.9	35.7
	×	4k	18.9	27.9	**32.8**	34.7	35.8
	×	3k	**19.2**	27.9	32.7	**35.0**	**36.0**
	×	2k	19.0	27.4	32.7	34.7	35.4
	×	1k	18.3	27.8	32.6	34.8	35.6

Table 9. Ablations on modules. We use 2k calibration set and report mAP/mAP50 of 4-bit YOLOv5-s on MS-COCO validation set.

$\mathcal{L}_{feat}$	$\mathcal{L}_{KD}$	$\mathcal{L}_{detect}$	mAP	mAP50
✓	✓	✓	**19.0**	**33.4**
	✓	✓	16.8	30.1
		✓	11.8	21.5

Table 10. Comparison with Data free Methods on MS-COCO validation set across multiple precision levels. All methods utilize 2k synthetic images for QAT on YOLOv5-s.

Prec.	Method	Real label	Data distri.	mAP	mAP50
W5A5	Real Label	✓	✓	28.0	45.8
	Gaussian noise	×	×	-	-
	Tile(Out-of-distri.)	×	×	16.1	27.9
	Tile(In-distri.)	×	✓	17.7	31.0
	MultiSample(Out-of-distri.)	×	×	21.9	37.3
	MultiSample(In-distri.)	×	✓	22.5	37.4
	Ours(Adaptive Label Sampling)	×	×	**26.1**	**42.3**
W4A4	Real Label	✓	✓	19.0	33.4
	Gaussian noise	×	×	-	-
	Tile(Out-of-distri.)	×	×	5.4	11.1
	Tile(In-distri.)	×	✓	6.8	13.4
	MultiSample(Out-of-distri.)	×	×	11.9	22.3
	MultiSample(In-distri.)	×	✓	13.1	23.3
	Ours(Adaptive Label Sampling)	×	×	**15.0**	**27.0**

Table 11. Comparison with real data QATs on YOLOv5 on MS-COCO validation set.

Method	Real Data	Num Data	Prec.	mAP / mAP50		
				YOLOv5-s	YOLOv5-m	YOLOv5-l
Pre-trained	✓	120k(full)	FP	37.4/56.8	45.4/64.1	49.0/67.3
LSQ	✓	120k(full)		26.9/44.9	32.9/50.6	35.2/53.0
LSQ+	✓	120k(full)		27.0/44.9	33.1/51.0	35.2/53.4
LSQ	✓	2k	W5A5	24.7/42.2	31.2/49.3	35.2/53.1
LSQ+	✓	2k		25.0/42.9	31.2/49.2	34.8/52.7
Ours	×	2k		**28.0/45.8**	**37.1/55.7**	**41.5/59.7**
LSQ	✓	120k(full)		23.3/40.0	27.9/45.4	33.1/50.3
LSQ+	✓	120k(full)		**23.3/40.2**	27.7/44.6	33.3/50.9
LSQ	✓	2k	W4A4	17.2/32.2	25.5/42.3	28.9/45.7
LSQ+	✓	2k		17.3/32.1	26.1/42.6	28.6/45.8
Ours	×	2k		19.0/33.4	**29.5/47.1**	**35.0/52.6**

Table 12. **The validity of task-specific information. (Ours)** mAP/mAP50 of YOLOv11s model on the MS-COCO validation set is reported. Experiments are conducted under the same PTQ or QAT settings, with 512 images from different ZSQ methods.

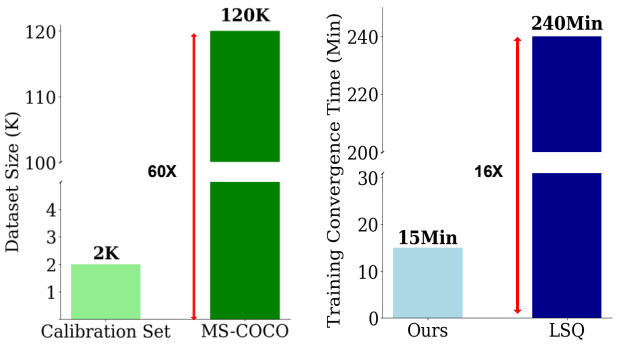
Precision	Quantizer setting	Genie [23]	ZeroQ [3]	**Ours**
W8A8	PTQ	45.8/62.3	46.0/62.5	**46.0/62.7**
	QAT	39.8/54.6	43.9/60.4	**45.9/62.2**
W6A6	PTQ	39.7/54.9	40.1/55.5	**40.3/55.8**
	QAT	36.9/51.9	39.5/55.4	**42.8/59.2**
W4A8	PTQ	11.2/18.0	**11.3/18.3**	11.2/18.2
	QAT	34.6/49.1	37.9/53.9	**41.2/57.1**

D. Sample Efficiency

We also demonstrate that by employing **Adaptive Label Sampling**, we achieved comparable or even superior results on QAT using a synthetic calibration set that is only $\mathbf{1/60}$ the size of the original training dataset. Additionally, by integrating self-distillation into the fine-tuning process of the quantized object detection network, we enabled a more efficient knowledge transfer. In the initial stage, utilizing 8 RTX 4090 GPUs for image generation allow us to produce 256 images every 20 minutes, resulting in a total of 160 minutes to generate 2,000 images. It is important to note that the calibration set we generate captures the overall characteristics of the original training set, allowing it to be reused multiple times during the quantization-aware training process. As the number of training iterations increases, our method progressively enhances the training convergence speed, achieving up to $\mathbf{16}\times$ faster convergence compared to the LSQ method trained on the entire real dataset. The corresponding results are visually illustrated in Fig. 6.

E. Additional Qualitative Results

Visualization for different object detection models In this section, we present visualizations of images generated by all the models discussed in this paper, including YOLOv5, YOLO11, as well as CNN and Transformer-



(a) Number of images involved in different training sets.

(b) Quantization-aware training (QAT) convergence time on 5-bit YOLOv5-m.

Figure 6. (a) Our synthetic condensed calibration set is 1/60 the size of the MS-COCO training set. (b) The training convergence speed can be improved by up to 16× compared to LSQ.

backbone Mask R-CNN, as shown in Fig. 7.

As observed, despite initializing the images with Gaussian noise and without referencing any real image data, our generated images can still accurately restore the real-world positions and sizes of objects. For instance, in the images generated by YOLOv5, a ballet dancer can be seen gracefully performing. In the images produced by YOLO11, there is a person sitting on a bench waiting for someone, as well as a table with several pizzas on it. In the CNN-backbone Mask R-CNN generated images, a herdsman is riding a horse while a giraffe roams aimlessly. Meanwhile, in the Transformer-backbone Mask R-CNN generated images, a television is displaying a program. By leveraging task-specific image generation, we can create more realistic images across different network architectures without relying on any external support.

Qualitative results for synthetic data In this section, we visualize the advantage of our Adaptive Label Sampling method over random sampling for multiple labels. As shown in Fig. 8, the left side illustrates our Adaptive Label Sampling method, which initially starts with single-label random sampling as presented in Table 6. After Adaptive Label Sampling, the model leverages the information stored during pre-training and add objects it considers highly confident, ultimately producing high-quality images. For instance, you can observe a person riding a horse, three boats gently floating on the shimmering water, and someone about to sit and rest next to a couch, among other realistic scenes.

Next, we use the obtained labels to perform multi-label random sampling by generating the corresponding object sizes and locations based on the sampling distribution in Table 6. The resulting images are shown on the right side of Fig. 8. In this scenario, the image quality deteriorates significantly, and the visual features fail to accurately reflect the generated objects. Consequently, compared to multi-label sampling, our Adaptive Label Sampling method captures the model's internal information more effectively, producing higher-quality and more coherent images.

Qualitative results for object detection performance In this section, we present visualizations illustrating the object detection capabilities of various neural networks. Specifically, we randomly selected four images from the MS-COCO validation set and used the detection results of a full-precision YOLOv5-s network as the reference. The visual comparisons in Fig. 9 display the detection results of neural networks trained with our adaptive label sampling method under 4-bit quantization-aware training (QAT).

The visualizations demonstrate that our quantized network can effectively detect objects. For example, when multiple teddy bears are present in an image, it accurately identifies each one. Similarly, when there is only a single object, such as a bear, it correctly recognizes it with confidence levels comparable to those of the full-precision network.

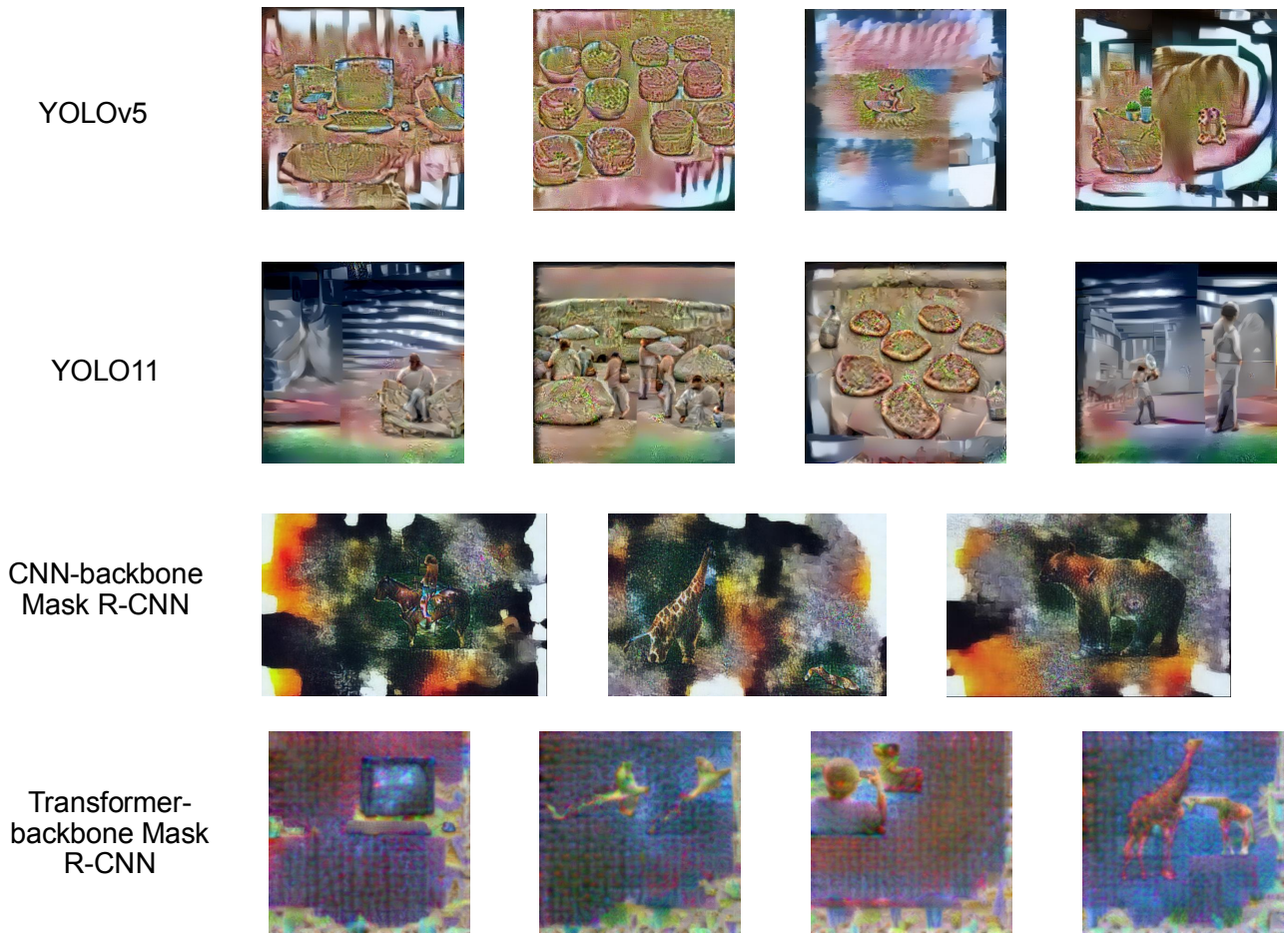


Figure 7. Visualization of images composed by different architecture-based object recognition networks.

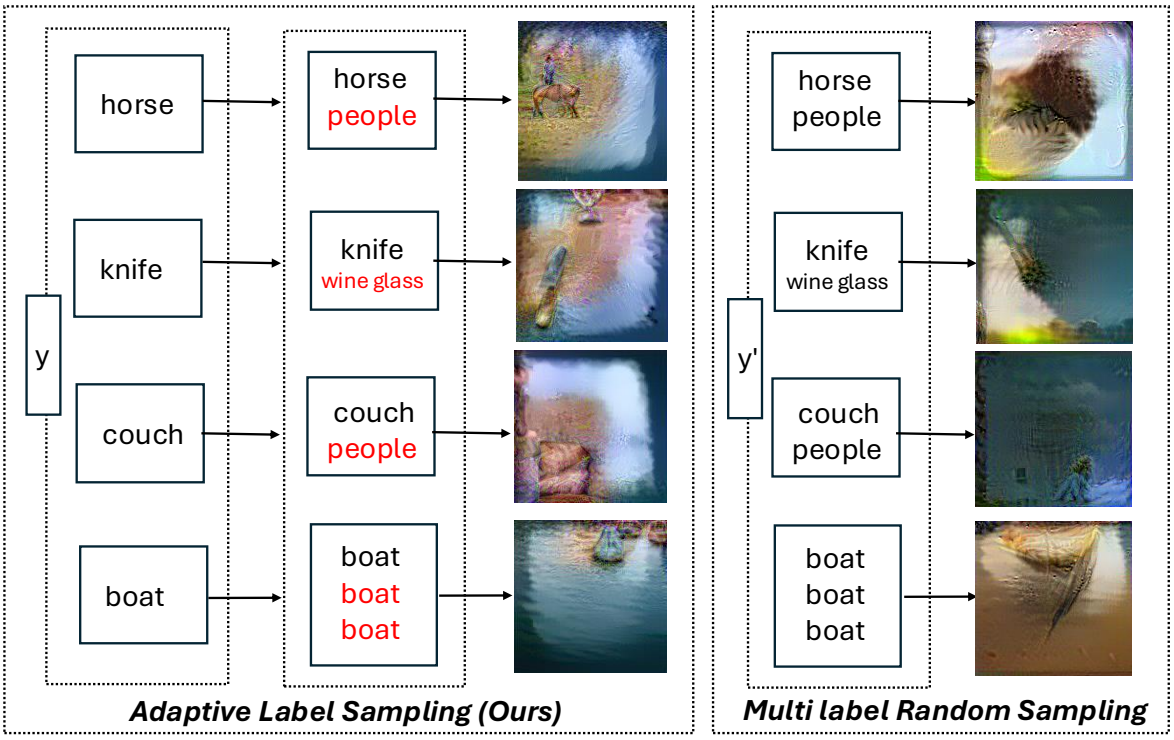


Figure 8. A comparison of the image quality generated by various sampling methods.

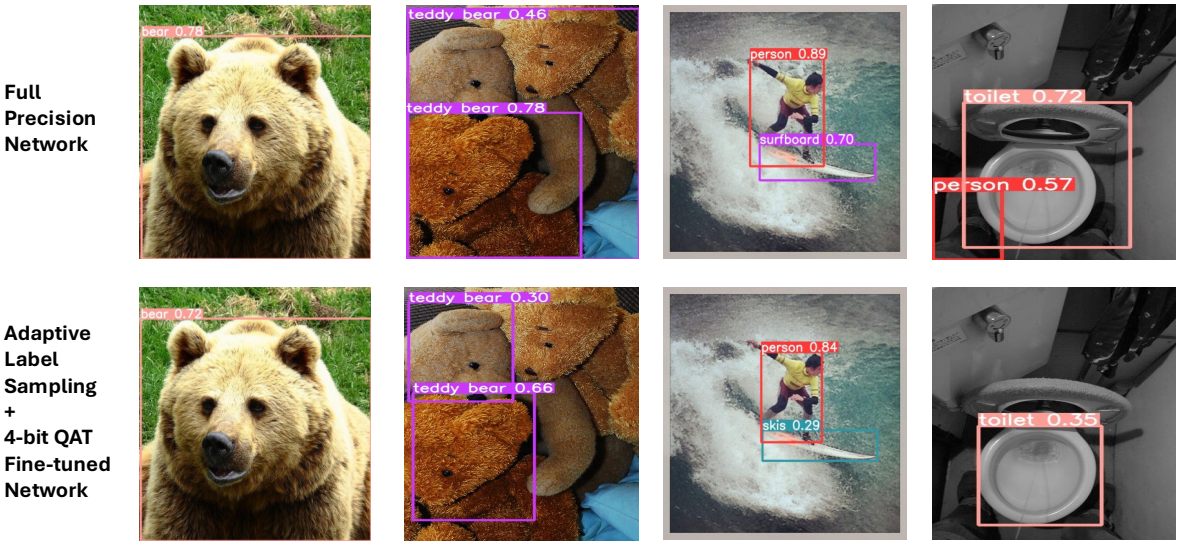


Figure 9. Qualitative analysis of object detection performance across different neural networks