

Article

ISER: Instance-Specific Early Stopping with Dynamic Low-Rank Adaptation for Learned Image Compression [†]

Unki Park ^{1,2} , Seongmoon Jeong ³ , Sangmin Kim ⁴, Jeungsub Lee ⁴, Gyeong-Moon Park ^{5,*} and Jong Hwan Ko ^{6,*}

¹ Department of Semiconductor and Display Engineering, Sungkyunkwan University, Suwon 16419, Republic of Korea; unkipark@g.skku.edu

² Samsung Institute of Technology, Hwaseong 18448, Republic of Korea

³ Department of Artificial Intelligence, Sungkyunkwan University, Suwon 16419, Republic of Korea; giftjsm@skku.edu

⁴ Hanwha Systems, Seongnam 13524, Republic of Korea; smkim0153@hanwha.com (S.K.); js.lee82@hanwha.com (J.L.)

⁵ Department of Artificial Intelligence, Korea University, Seoul 02841, Republic of Korea

⁶ College of Information and Communication Engineering, Sungkyunkwan University, Suwon 16419, Republic of Korea

* Correspondence: gm-park@korea.ac.kr (G.-M.P.); jhko@skku.edu (J.H.K.)

[†] This article is a revised and expanded version of a paper entitled ‘Test-Time Fine-Tuning of Image Compression Models for Multi-Task Adaptability’, which was presented at the 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 11–15 June 2025.

Abstract

Image compression has evolved from human-centric perceptual coding toward support for diverse machine vision applications, requiring modern codecs to serve both human viewing and downstream tasks in closed-set settings (where target tasks are incorporated during training) and open-set settings (where previously unseen tasks arise at test time). While recent learned compression methods jointly optimize perceptual quality and closed-set task performance, they often fail to generalize to unseen open-set tasks due to fixed training assumptions and objectives. Our prior work, LoRA-comp (Low-Rank Adaptation Compression), effectively addresses open-set challenges via instance-specific test-time fine-tuning (TTFT) without requiring task-specific pre-training. Nevertheless, its fixed LoRA architecture, which assigns a uniform rank across all layers, often leads to suboptimal instance-level performance. Moreover, allocating the same number of training epochs to every instance introduces unnecessary encoding-time overhead. To address these challenges, we propose Instance-Specific Early Stopping with Dynamic Rank Adaptation (ISER), which extends LoRA-comp. Building upon the LoRA-comp-based instance-specific adaptation framework, ISER introduces (i) instance-specific early stopping (ISES) combined with a multi-scale training strategy (MSTS) to reduce TTFT overhead and (ii) instance-specific dynamic rank adaptation (ISRA) to tailor the LoRA architecture per instance. Experiments demonstrate that ISER consistently outperforms competing methods. Compared to LoRA-comp, ISER achieves up to a 7% BD-Rate improvement and up to a 44% reduction in encoding time. Moreover, ISER achieves up to a 98% relative improvement in BD-Rate gain and up to a 19.2% reduction in decoding time over TransTIC.

Keywords: learned image compression; test-time fine-tuning; instance-specific optimization; low-rank adaptation; SVD-LoRA; multi-task adaptability



Academic Editors: Miin-shen Yang, Cocianu Catalina-Lucia and Cristian Răzvan Uscatu

Received: 14 July 2026

Revised: 11 August 2026

Accepted: 18 August 2026

Published: 25 August 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Image compression has played a crucial role in building visual systems by reducing storage requirements and transmission bandwidth. In recent years, advances in deep learning have driven a paradigm shift toward learning-based image compression, which is now widely regarded as a promising and effective approach [1,2]. While learned image compression (LIC) is widely recognized for outperforming traditional rule-based methods in compression efficiency, its more fundamental contribution lies in the flexibility to redefine optimization objectives beyond compression efficiency alone. Conventional image compression techniques have primarily focused on optimizing human perceptual quality. In contrast, LIC enables direct optimization not only for human perception but also for downstream objectives of visual systems. This paradigm shift is explicitly reflected in recent learning-based compression standards, such as JPEG AI [3] and Video Coding for Machines (VCM) [4,5]. However, models trained under the independent and identically distributed (IID) assumption suffer from inherent generalization limitations, making it difficult to accommodate previously unseen open-set tasks at inference time. Even representative LIC models such as TIC [6] and Cheng2020 [7], while achieving high perceptual image quality, exhibit limited generalization performance on downstream tasks including object detection and instance segmentation [8–11]. Therefore, as a core component of visual system design, LIC must effectively address this limitation to ensure practical applicability. In response to the misalignment between perceptual optimization and downstream task performance, a growing body of work has reformulated image compression as a task-oriented problem, commonly referred to as Coding for Machines (CfM) [12–14] or Image Coding for Machines and Humans (ICMH) [10,15].

At a higher level, the limitations of LIC can be viewed as manifestations of the broader OOD (out-of-distribution) problem in data-driven neural networks. Because these models are trained under the IID assumption, they frequently struggle when confronted with unseen tasks. To alleviate this issue, many prior studies have adopted general-purpose OOD robustness techniques [8–11,13,15–17]. Broadly, these approaches can be categorized into two paradigms: generalization and adaptation. In the context of LIC, these paradigms align with multi-task bitstream methods [8,9,16,18–21] and single-task bitstream methods [10,11,17,22–24], respectively.

The multi-task bitstream approach aims to support multiple downstream vision tasks within a unified bitstream, emphasizing generalization through general-purpose design. This method offers flexibility in responding to a range of machine vision tasks, but it introduces challenges. For example, scalable coding [8,16,18,19] increases the processing complexity and bitstream overhead for individual downstream tasks, and unified image coding [9,20,21] struggles with optimization of each subtask individually.

In contrast, the single-task bitstream approach avoids the limitations of multi-task methods by focusing on dynamically generating bitstreams tailored to a specific downstream task, thereby enabling task-optimized compression and performance. Representative methods are TransTIC, Adapt-ICMH and LoRA-Comp (Low-Rank Adaptation Compression). TransTIC [17] utilizes prompt tuning for adaptation of the human-target backbone codec to generate bitstreams optimized for the other tasks specified by the prompt. While this technique enables switching of pre-defined (closed-set) tasks without modifying the backbone codec, it requires task-specific pre-training of the prompt generator. Therefore, it leads to increased training demands as the number of tasks grows and faces challenges in handling new, untrained (open-set) tasks. Adapt-ICMH [10] is another task-specific framework that inserts a Spatial-Frequency Modulation Adapter (SFMA) between layers of a human-centric backbone codec, as illustrated in Figure 1a. This design suppresses spatial redundancies and enhances task-relevant frequency components for subtask-wise compres-

sion optimization. However, this approach also requires task-specific pre-training of the SFMA whenever the target subtask changes. Furthermore, SFMA tends to overfit strongly to individual subtasks, leading to substantial performance degradation under open-set conditions. In contrast, LoRA-comp [11] (our previous work) applies instance-specific LoRA fine-tuning to a human-centric backbone codec, as illustrated in Figure 1c. This approach effectively addresses open-set challenges without requiring task-specific pre-training. This enables the model to adapt to unseen tasks during test time. Nevertheless, its fixed LoRA architecture assigns a uniform rank across all layers. Since different instances may require fine-grained adaptation at different feature scales to improve target-task performance, this fixed design often leads to suboptimal instance-level results. In addition, its fixed epoch scheduling allocates the same number of training epochs to every instance. This can incur unnecessary encoding-time overhead, as early-converging instances still consume the full training budget despite requiring fewer optimization steps.

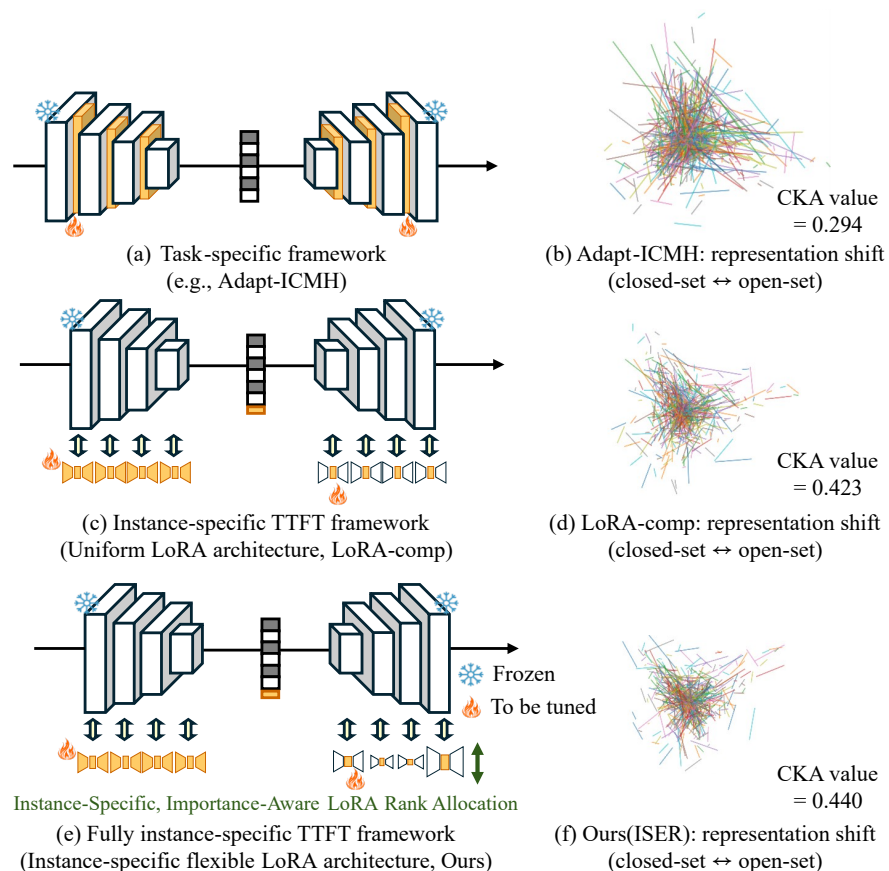


Figure 1. Comparison of LIC adaptation frameworks and their ability to handle open-set tasks. From top to bottom, the figure shows Adapt-ICMH [10], LoRA-comp [11], and the proposed method (ISER). The left panels show how each method adapts a frozen backbone, highlighting differences in parameter update scope and rank allocation. The right panels visualize closed-set–open-set representation shifts via CKA, computed at the final layer of the Faster R-CNN backbone over the COCO-2017 validation set, where a higher value indicates closer similarity to the closed-set condition, which serves as a reference for the best achievable adaptation given its task-specific pre-training. Compared to Adapt-ICMH and LoRA-comp, ISER exhibits more compact and stable shifts, indicating stronger open-set adaptability.

To address these limitations, we propose the Instance-Specific Early Stopping with Dynamic Rank Adaptation (ISER) framework, as illustrated in Figure 1e. Building upon the instance-specific adaptation framework of LoRA-comp, which is effective under open-set conditions, ISER introduces two key mechanisms. First, to improve instance-level opti-

mality, we propose instance-specific dynamic rank adaptation (ISRA), which dynamically designs the LoRA architecture for each instance during fine-tuning. Second, to reduce the computational burden of test-time fine-tuning (TTFT), we combine a multi-scale training strategy (MSTS) with an instance-aware epoch-scheduling mechanism, termed instance-specific early stopping (ISES). These mechanisms are jointly applied within a single TTFT process. Specifically, in each instance-specific TTFT process, the relative importance of individual LoRA layers is quantified by monitoring the temporal variation of their gradient magnitudes during adaptation. Based on the resulting layer-wise importance scores, we determine both the instance-specific LoRA architecture and the number of training epochs allocated to adaptation. In addition, we employ multi-resolution inputs across different TTFT phases to promote rapid convergence and stable optimization while improving performance by enhancing global contextual modeling.

Figure 1b,d,f compares open-set adaptation capability among single-task bitstream methods. For each input image, we generate reconstructions using codecs trained under closed-set and open-set conditions and analyze the resulting representation shifts through layer-wise activations of downstream task models, quantified using Centered Kernel Alignment (CKA) [25]. Because the closed-set condition is pre-trained on the target task itself, it serves as a reference for the best achievable adaptation. A high CKA similarity to this reference indicates that open-set adaptation approaches the task-informed optimum without task-specific pre-training, whereas a low similarity implies convergence to a degraded representation. The line segments depict these closed-set–open-set representation shifts, extracted from the final layer of the Faster R-CNN [26] backbone, immediately preceding the FPN and detection-specific heads, and visualized in a 2D principal component analysis (PCA) space [27]. This comparison shows that instance-specific adaptation (LoRA-comp and ISER) yields stronger open-set adaptability than the task-specific Adapt-ICMH.

For performance evaluation, we adopt two complementary settings. First, we compare directly with LoRA-comp under its original experimental conditions, where ISER achieves up to a 7% improvement in BD-Rate and up to a 44% reduction in average per-instance encoding time. Second, we evaluate against competing methods under open-set scenarios, quantifying representational distances among tasks using CKA and additionally including depth estimation, which exhibits a large representational gap from the basis task (image classification). Here, ISER consistently outperforms competing approaches across individual open-set tasks, with a particularly large gain of 98% over TransTIC on depth estimation.

This paper extends our prior work, LoRA-comp (CVPR 2025) [11], and makes the following contributions:

- Fully instance-specific TTFT: This work extends LoRA-comp with an instance-specific LoRA architecture design and adaptive early stopping to mitigate instance-level suboptimality and reduce TTFT overhead. The resulting framework preserves LoRA-comp's strong multi-task adaptability while enabling more efficient instance-wise adaptation.
- Quantification of representational distances among open-set tasks: By quantifying representational distances using CKA, we can systematically analyze open-set conditions. This analysis provides useful insights for the design of open-set task configurations that more accurately reflect the generalization challenges encountered in real-world multi-task vision systems.
- Hardware-efficient and plug-and-play design: With negligible computational/memory overhead and plug-and-play compatibility with both CNN- and Transformer-based architectures, our method provides scalable gains in multi-task adaptability as backbone LIC models evolve.

2. Related Work

LIC for Human Perception and Machine Vision Tasks. LIC has been extensively studied as an end-to-end neural alternative to conventional image codecs [1,2,6,7,28–31]. Ballé et al. [1] proposed one of the first CNN-based end-to-end LIC frameworks, demonstrating clear gains over traditional codecs such as JPEG [32] and JPEG 2000 [33]. Building upon this line of work, Cheng2020 [7] incorporates attention modules into a CNN-based architecture, achieving notable improvements over earlier CNN-based models. More recently, TIC [6] adopted a Transformer-based autoencoder architecture, further improving compression efficiency while maintaining high perceptual quality and serving as a representative Transformer-based LIC approach.

With the rise of deep learning, LIC models have evolved to support multiple targets, including human perception and diverse machine vision tasks. Aligned with this direction, recent compression standards, including VCM [4,5] and JPEG-AI [3], explicitly target both human perceptual quality and machine vision utility. Concurrently, a growing body of research has investigated unified compression frameworks capable of accommodating multiple downstream tasks, which can be broadly classified into multi-task bitstream and single-task bitstream approaches.

Compression Standards for Human and Multiple Machine Vision. The MPEG and JPEG standardization groups are also actively focusing on establishing compression standards that enable a single model to simultaneously support a diverse range of vision tasks. VCM [4,5] focuses on a compression paradigm that explicitly prioritizes support for diverse downstream machine vision tasks over strict preservation of human perceptual quality. The standard is currently under active exploration, with formal standardization targeted for July 2026. JPEG-AI [3] aims to provide a compact, single-stream compressed-domain representation that simultaneously supports both human visualization and machine-oriented tasks. The standardization effort is being conducted in two distinct versions. Version 1, which standardizes the core neural network-based encoder and decoder architecture, was published in March 2025. The flexible extension toward a broader range of image processing and vision tasks is addressed in Version 2, with standardization targeted for 2027. However, the JPEG-AI framework under consideration requires task-specific decoders to support diverse downstream tasks, which introduces a limitation in terms of scalability and deployment efficiency.

Generalization-Oriented Approaches: Multi-task Bitstream LIC for Multiple Targets. One line of research in multi-target LIC adopts a generalization-oriented approach [8,9,16,18–21]. It aims to enhance the generalization capability of a single shared LIC model so that multiple tasks can be supported through a unified bitstream. However, this approach increases the complexity of both the encoder and decoder [8,9]. It also leads to significant bitstream overhead from the perspective of individual downstream tasks [8]. Additionally, it faces the challenge of Pareto optimality, which makes it difficult to optimize each subtask individually [9].

Adaptation-Oriented Approaches: Single-task Bitstream LIC for Multiple Targets. Another line of research in multi-target learned image compression follows an adaptation-oriented approach [10,11,17,22–24]. This paradigm generates a task-specific bitstream for one target task at a time by explicitly adapting the compression model to the given task. By tailoring the compression process to each well-defined subtask, such approaches enable optimal performance under task-specific operating conditions.

Representative studies in this field involve applying task-adaptive importance modulation or fine-tuning techniques to the backbone LIC. Song et al. [24] proposed a spatially adaptive image compression method using Spatial Feature Transform (SFT) guided by a real-valued quality map that enables pixel-wise optimization for machine vision tasks.

Liu et al. [23] introduced a task-adaptive image compression method that uses a gating module to selectively activate channel-wise latents from a pre-trained codec. J.P. Klopp et al. [34] introduced a learned pre-filter that suppresses task-irrelevant data prior to compression by distinguishing between important and unimportant information. TransTIC [17] employs Visual Prompt Tuning (VPT) [35,36] to adapt the large-scale pretrained model TIC, originally trained for human perception, to suit various target tasks. This approach has demonstrated adaptability to various machine vision tasks, such as object detection, instance segmentation, and classification. However, VPT itself faces limitations such as non-monotonic performance changes depending on the size of the applied prompt, as well as increased computational cost due to the additional prompts during the inference stage [37]. More importantly, TransTIC adjusts the prompt to guide the output of a static pretrained model, but the generator responsible for creating this prompt also needs to be pretrained. Also, it requires prompts equivalent to one-fourth of the input feature size of each layer to perform task switching. Therefore, for application of fine-tuning to individual test samples at the decoder stage, these prompts must be incorporated into the bitstream, making TransTIC unsuitable for instance-specific decoder fine-tuning. Consequently, pre-training is essential to adapt to new tasks, and it is difficult to adapt to untrained open-set tasks. Adapt-ICMH [10] introduces the first frequency-domain adaptation approach in the LIC field, demonstrating that frequency-domain adaptation can achieve strong task-specific optimization for individual target tasks. However, this method requires task-specific pre-training of the SFMA for each target, which substantially limits its applicability in open-set scenarios. Moreover, such frequency-domain adaptation tends to induce pronounced task specialization, leading to significant performance degradation when evaluated on unseen tasks. LoRA-Comp [11], on the other hand, addresses open-set scenarios by leveraging TTFT to adapt a backbone LIC model to diverse tasks without task-specific pre-training. However, its reliance on a fixed LoRA configuration with uniform rank allocation across layers limits instance-specific adaptability. Moreover, using a fixed number of fine-tuning epochs for all instances leads to avoidable encoding-time overhead. To address the limitations of LoRA-comp, we propose the ISER framework, which extends LoRA-comp by more explicitly incorporating instance-specific characteristics. Unlike LoRA-comp, ISER does not employ a fixed LoRA architecture across all instances. Instead, during the TTFT process, it applies dynamic rank allocation by assigning higher ranks to layers with greater temporal variation of importance scores. This strategy enables performance optimization tailored to each individual instance. In addition, to reduce encoding-time overhead, we dynamically allocate the number of training epochs for each instance based on an assessment of training stability.

3. LoRA-Comp: Instance-Specific TTFT

3.1. Framework Overview of LoRA-Comp

LoRA-comp aims to adapt the given backbone codec to various downstream tasks, including open-set tasks that are unseen during training. To achieve this, LoRA-comp adapts the backbone LIC to target downstream tasks by applying TTFT [38,39] to both the encoder and decoder. Specifically, LoRA-comp uses LoRA [40], a widely used approach in test-time scenarios. By tuning the LoRA weights to be optimized for each target task and image instance, the backbone LIC can be adapted to better suit the target task. However, transmitting LoRA weights to the decoder for fully instance-specific tuning incurs a non-negligible bit-rate overhead. To minimize the bit-rate issue, LoRA-comp employs a reformulated version of LoRA, referred to as SVD-LoRA, which enables effective fine-tuning of only a small portion of the decoder at test time. While prior works utilized SVD-LoRA for language task-specific adaptation [41,42], LoRA-comp focuses on test-time decoder fine-tuning for a fully instance-specific optimized compression model. Consequently, LoRA-comp can

flexibly handle diverse, arbitrary target tasks and is compatible with various backbone LIC architectures, including both CNN and Transformer-based models.

Figure 2 illustrates the LoRA-comp framework utilizing SVD-LoRA, which consists of two stages. In the first stage, task-specific tuning is performed to initialize the SVD-LoRA parameters. Under the closed-set setting, LoRA-comp uses the target downstream task for this initialization. For the open-set setting, LoRA-comp leverages a ResNet-50 model pre-trained on the ImageNet dataset, which provides strong generalization across a wide range of machine vision applications. The next stage involves performing TTFT to optimize each test sample according to the target downstream task. The precise specification of the loss network used during this stage, under both closed-set and open-set settings, is detailed in Section 5.1. During this process, all of the SVD-LoRA weights in the encoder and only the SVs in the decoder are fine-tuned. While full fine-tuning of the encoder does not increase bit-rate overhead or decoding latency, it requires the updating of a large number of parameters for each TTFT instance, thereby necessitating more training epochs and resulting in increased TTFT time overhead. Therefore, LoRA-comp applies SVD-LoRA to the encoder instead of performing full fine-tuning. As a result, this approach significantly reduces training costs and enables rapid adaptation to new downstream tasks. Additionally, fine-tuning only the SVs on the decoder side further reduces bitstream overhead compared to full LoRA fine-tuning. While increasing the number of epochs during TTFT improves performance, it also increases the encoding time. This issue can be addressed by adjusting the number of epochs according to the compression scenario, thereby balancing performance improvement with encoding time.

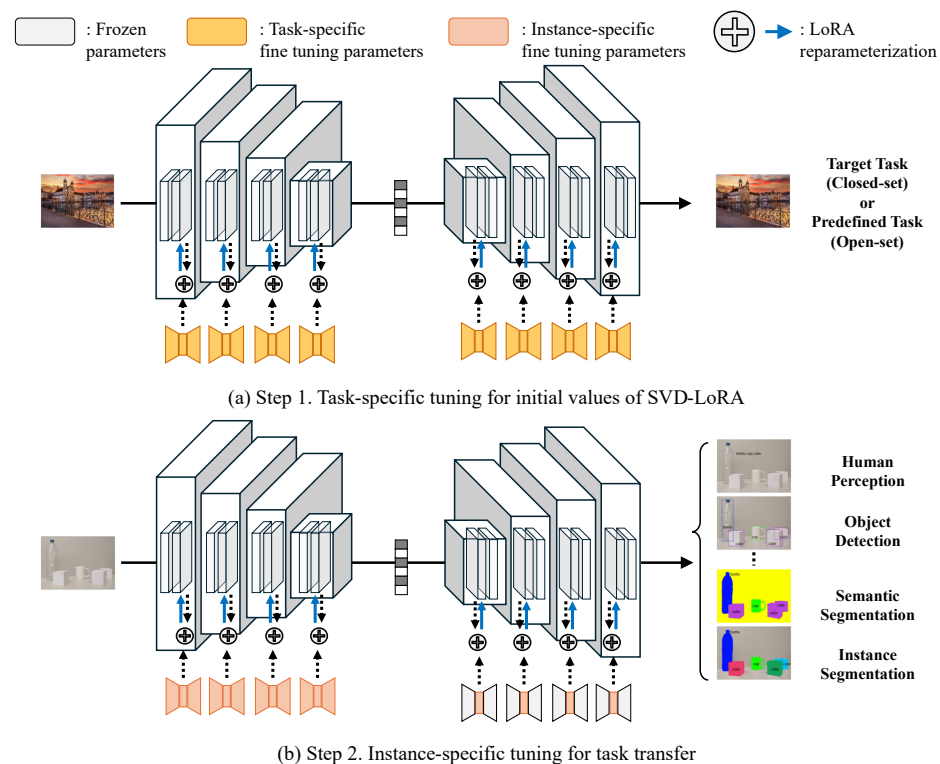


Figure 2. Overview of the LoRA-comp framework. LoRA-comp adapts a frozen backbone codec to diverse downstream tasks via instance-specific TTFT. The TTFT procedure has two stages: (a) initialization of SVD-LoRA parameters and (b) per-instance test-time optimization that fine-tunes each input to the target task. Reprinted with permission from Ref. [11]. 2025, IEEE.

The loss function of SVD-LoRA during TTFT is defined by Equation (1):

$$L = -\log p(\hat{z}) - \log p(\hat{y} | \hat{z}) + \lambda d(x, \hat{x}), \quad (1)$$

where x , y , and z represent the input image, encoded latent feature, and hyperprior, respectively. Additionally, $\hat{x}$, $\hat{y}$, and $\hat{z}$ represent their reconstruction. The $p(\hat{z})$ and $p(\hat{y} | \hat{z})$ terms represent the probabilities of the compressed latent variables, which correspond to the bit rate used for compression of the input (x). $d(x, \hat{x})$ represents the distortion between the original and the reconstructed images. Mean squared error (MSE) is used as the distortion metric when targeting human perception, as it measures the pixel-wise difference between the original and reconstructed images. For machine vision tasks such as image classification, perceptual loss is used as the distortion metric by comparing intermediate features extracted from the original and reconstructed images via the target network [17]. The λ hyperparameter adjusts the balance between rate and distortion. This loss function allows for optimization that balances data compression and reconstruction while tailoring to the objective of each task. All experiments conducted in this work are formulated as rate-distortion optimization, with different values of λ and distortion functions designed according to the characteristics of each downstream task.

3.2. SVD-Based LoRA

SVD-LoRA is represented by Equation (2):

$$W = W_0 + \Delta W = W_0 + U \cdot \Sigma \cdot V^T, \quad (2)$$

where $U \in \mathbb{R}^{d \times r}$, $\Sigma \in \mathbb{R}^{r \times r}$, $V \in \mathbb{R}^{k \times r}$, and $r \ll \min(d, k)$. Here, U and V learn the intrinsic ranks for columns and rows, respectively, while Σ is a diagonal matrix that learns the SVs.

Although the structure includes additional SVs compared to conventional LoRA, the increase in the number of parameters is negligible due to the typically small rank used. Figure 3 illustrates the process of reconstructing W (adapted weights) through the reparameterization of LoRA and SVD-LoRA. By learning additional SVs, SVD-LoRA facilitates the learning of more complex intrinsic dimensions of adaptation weights compared to the original LoRA. In cases where both LoRA and SVD-LoRA have the same small rank, experimental results indicate that SVD-LoRA better captures the intrinsic dimensions of the adaptation weights [11].

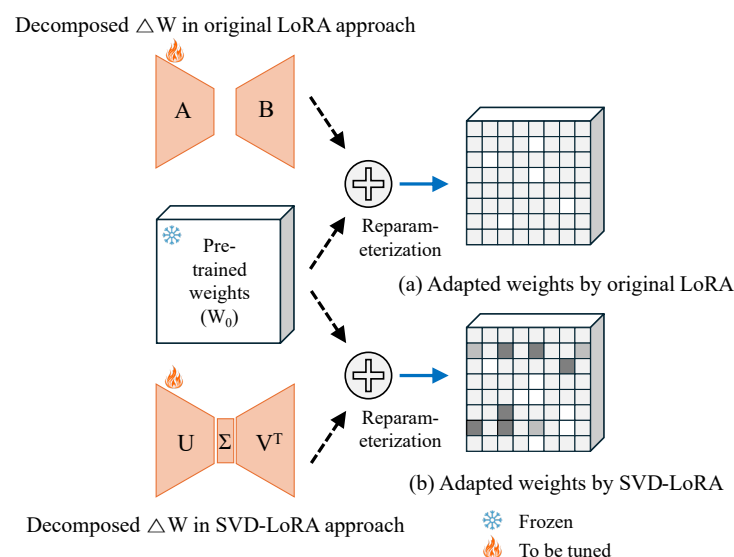


Figure 3. Comparison between original LoRA and SVD-LoRA. Original LoRA represents the adaptation as a low-rank update ($\Delta W = B \cdot A$), which is merged into the frozen pre-trained weights (W_0). In contrast, SVD-LoRA decomposes ΔW into $U \cdot \Sigma \cdot V^T$, enabling more effective capture of the intrinsic adaptation subspace, particularly under small-rank constraints. Reprinted with permission from Ref. [11]. 2025, IEEE.

Figure 4 illustrates the process of embedding the decoder-side SVD-LoRA weights into the compressed bitstream for instance-specific fine-tuning at the decoder stage. While the bitstream size increases due to the additional SV component, the overhead is negligible. Specifically, under the LoRA-comp experimental setting with rank 4, the additional overhead accounts for less than 1% of the compressed image bitstream size, on average, ranging from 0.29% to 2.05% depending on the target task and quality level.

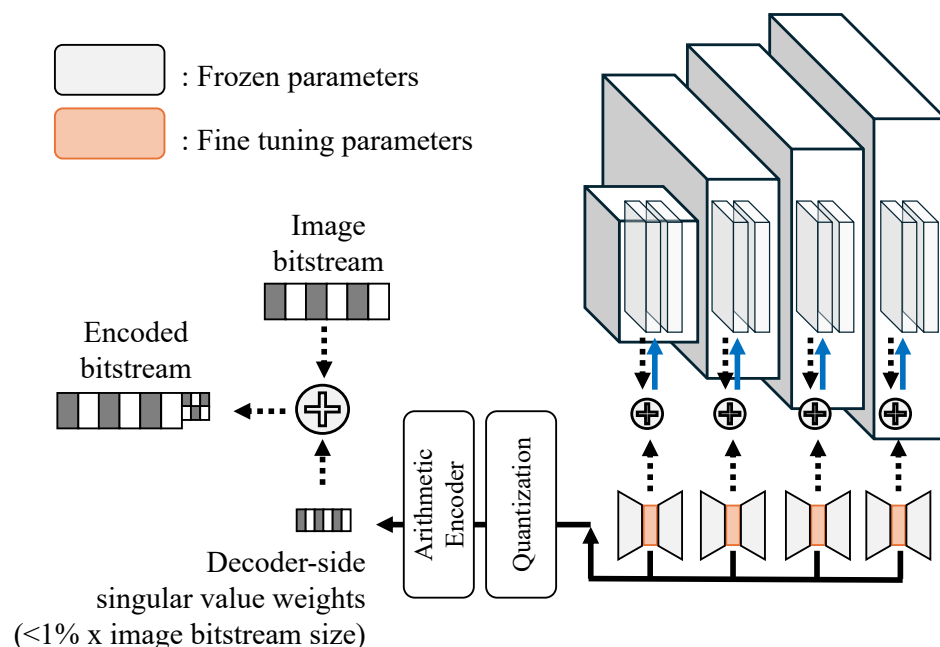


Figure 4. Bitstream generation process. To enable instance-specific decoder tuning with minimal bitstream overhead, only the decoder-side singular values are encoded and appended to the image bitstream. This additional signaling introduces an average bit-rate overhead of less than 1% relative to the original image bitstream size. Reprinted with permission from Ref. [11]. 2025, IEEE.

4. ISER: Fully Instance-Specific TTFT

4.1. Framework Overview

ISER retains the LoRA-comp framework, which performs instance-level optimization following task-specific fine-tuning. However, as illustrated in Figure 5, ISER introduces a modified instance-specific fine-tuning framework designed to overcome the limitations of LoRA-comp.

Although LoRA-comp demonstrates strong adaptability to open-set tasks relative to task-specific approaches, its TTFT process remains instance-agnostic, applying a uniform LoRA architecture, fixed training epochs, and a single image resolution to all instances. A fixed LoRA architecture hinders effective optimization for individual instances, while allocating an identical number of training epochs introduces unnecessary encoding-time overhead from an instance-specific perspective. Moreover, the use of a fixed image resolution may induce training inefficiency by limiting scale-aware optimization. To address these limitations, we incorporate three complementary components. First, we introduce ISRA, which improves instance-level optimality by dynamically tailoring the LoRA architecture for each input. Second, we adopt ISES to reduce the computational burden of TTFT through instance-aware epoch scheduling. Third, we employ an MSTs, which leverages multi-resolution inputs to promote rapid convergence and stable optimization while improving performance by capturing global contextual information. These components are jointly applied within a single instance-specific TTFT process.

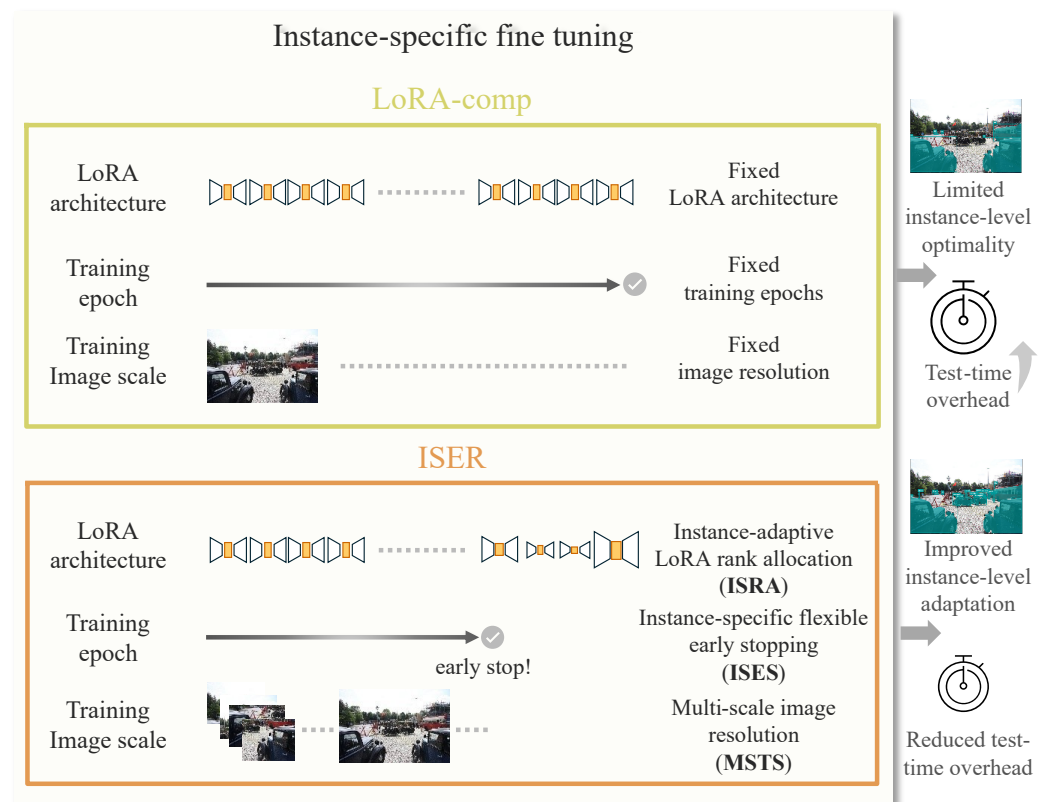


Figure 5. Conceptual comparison of instance-specific fine-tuning in LoRA-comp and ISER. LoRA-comp adopts fixed LoRA architectures, a fixed image resolution, and fixed training epochs, which may limit instance-level optimality and increase test-time overhead. In contrast, ISER enables instance-adaptive LoRA rank allocation, multi-scale training, and instance-specific early stopping, improving multi-task adaptability while reducing per-instance tuning cost.

While Figure 5 outlines the conceptual differences between LoRA-comp and ISER, Table 1 further summarizes these distinctions across six design axes. Although ISER introduces ISRA, ISES, and MSTs to enhance instance-level adaptability, it preserves the transmitted information and inference-time computational complexity of LoRA-comp, ensuring that these gains incur no additional deployment cost.

Table 1. Comparison between LoRA-comp and ISER across key design axes. While ISER builds upon the instance-specific TTFT framework of LoRA-comp, it introduces instance-adaptive rank allocation, early stopping, and multi-scale training without increasing transmitted information or inference-time complexity.

Axis	LoRA-Comp	ISER (Ours)
Optimization variables	Encoder: full SVD-LoRA (U, Σ, V); Decoder: SVs (Σ) only	Same as LoRA-comp
Rank-allocation strategy	Fixed, uniform rank for all layers/instances	Adaptive rank allocation by ISRA ($r^{(0)} = 8 \rightarrow r_{\text{target}} = 4$)
Stopping criterion	Fixed epoch budget	Adaptive early stopping by ISES
Training scale	Single, fixed full resolution	Multi-scale resolution training by MSTs
Transmitted information	Decoder-side SVs (<1% bitstream overhead on average)	Same as LoRA-comp
Computational complexity (inference)	3.72 M/3.94 M params; 142.31/142.53 kMACs/pixel	Same as LoRA-comp

4.2. Dynamic Instance-Specific LoRA Architecture Design and Early Stopping

Determining an appropriate LoRA rank has long been recognized as a challenge in the LoRA research literature. To address this issue, AdaLoRA [42] dynamically adjusts the rank of each LoRA layer during fine-tuning by estimating layer-wise importance scores and allocating ranks accordingly. However, AdaLoRA designs the LoRA architecture in a task-specific manner under the assumption of a fixed backbone model and a predefined target task. As a result, it is not well suited for instance-specific LoRA architecture design, and its training cost increases substantially under target-task shifts, which limits its applicability to open-set scenarios. Beyond AdaLoRA, more recent dynamic-rank LoRA approaches have also been proposed, including gradient-driven rank adjustment [43] and variance-optimal initialization with adaptive rank allocation [44]. However, like AdaLoRA, these approaches determine rank allocation once, prior to or at the start of fine-tuning, based on statistics aggregated over a training dataset for a fixed task. They therefore remain task-level- and pre-training-dependent and do not address instance-specific adaptation at test time without task-specific pre-training.

To overcome these limitations, we introduce ISRA, which constructs an instance-specialized LoRA architecture by jointly reflecting both target-task characteristics and instance-specific properties during the adaptation process, without requiring task-specific pre-training. Furthermore, we integrate ISES into ISRA to adaptively allocate the training budget per instance, thereby reducing the overall ISRA time overhead. While the preceding discussion motivates ISRA in contrast to AdaLoRA, we further clarify the specific technical distinctions between ISRA and AdaLoRA, a representative dynamic-rank LoRA approach. First, AdaLoRA determines rank importance under a fixed target task, so its importance scores and pruning schedule are tied to that task. When the task changes, this process must be repeated from scratch. In contrast, ISRA requires no task-specific pre-training. The same procedure is applied directly at test time to each new instance and task, regardless of whether the task was seen during training. Second, AdaLoRA allocates ranks per task, implicitly assuming that all instances belonging to a task share a similar optimal architecture, whereas ISRA allocates ranks per instance, allowing different images of the same task to converge to different layer-wise rank configurations depending on their individual gradient dynamics. Third, ISRA deliberately restricts rank adaptation to the decoder SVs rather than applying it across the full network, as in AdaLoRA. This design choice not only stabilizes optimization under the short TTFT horizon but also ensures that rank adaptation neither affects the encoder nor introduces additional bitstream overhead. Together, these distinctions allow ISRA to achieve instance-level architectural adaptivity within a single test-time optimization pass, a capability that AdaLoRA's task-level, pre-training-dependent design cannot support.

ISRA initializes the LoRA modules with a higher rank and progressively reduces the rank during TTFT by suppressing less important components, thereby converging to a target rank. However, applying this procedure across all layers would require frequent rank reconfiguration across all LoRA modules, which can compromise optimization stability. Therefore, we restrict ISRA to the decoder SVs. Focusing adaptation on the decoder rather than the encoder is advantageous because the decoder directly determines the reconstructed output that is provided to downstream tasks. Moreover, the stability of decoder-side LoRA parameters serves as a reliable indicator of TTFT optimization stability, as confirmed by our experimental analysis. Finally, SVs that collapse to zero can be pruned at decoding time, increasing the effective decoder rank without introducing additional bit-rate overhead.

Algorithm 1 provides an overview of the proposed dynamic instance-specific LoRA design process, which integrates ISRA and ISES. Given a set of layers ($\mathcal{K}$) and their low-rank singular value components ($\sigma_{k \in \mathcal{K}}$), ISRA iteratively evaluates the importance of low-rank

components and removes less important ones to meet a target rank budget ($\mathcal{K} \cdot r_{\text{target}}$). ISES regulates the convergence speed toward target ranks on a per-instance basis.

Algorithm 1 Dynamic instance-specific LoRA design

Require: Input instance x ; codec model $D(x; w_o, w_u, w_\sigma, w_v)$; learning rate δ ; epoch budget T ; initial ranks $\{r_k^{(0)}\}_{k \in \mathcal{K}}$; decoder SV layer set $\mathcal{K}$; importance function $s(\cdot)$ in (3); threshold τ ; reduction ratio ρ ; per-layer target rank r_{target}

Ensure: Updated LoRA weights (w'_u, w'_σ, w'_v) and instance-optimized ranks $\{r_k\}_{k \in \mathcal{K}}$

```

1: for  $t = 1$  to  $T$  do
    {Model update (Lines 2–5)}
2:  $\hat{x}, \hat{y}, \hat{z} \leftarrow D(x; w_o, w_u, w_\sigma, w_v)$ 
3:  $\ell \leftarrow L(x, \hat{x}, \hat{y}, \hat{z})$  in (1)
4:  $\mathcal{G} \leftarrow \nabla_{(w_u, w_\sigma, w_v)} \ell$ 
5:  $(w_u, w_\sigma, w_v) \leftarrow (w_u, w_\sigma, w_v) - \delta \mathcal{G}$ 
    {Instance-specific LoRA design (Lines 6–20)}
6: extract  $\{w_{\sigma,k}\}_{k \in \mathcal{K}}$  and  $\{\nabla_{w_{\sigma,k}}\}_{k \in \mathcal{K}}$ 
7: for each  $k \in \mathcal{K}$  do
8:   Compute importance scores in (3):  $S_{k,i}^{(t)} \leftarrow s(\sigma_{k,i}^{(t)}), \forall i \in \{1, \dots, r_k\}$ 
9: end for
10: if  $t \geq 3$  then
11:   for each  $k \in \mathcal{K}$  do
12:     Compute second-order differences in (4):  $\Delta^2 S_{k,i}^{(t)}, \forall i \in \{1, \dots, r_k\}$ 
13:   end for
14:   Compute convergence indicator in (5):  $C^{(t)}$ 
15:   if  $C^{(t)} < \tau$  and  $\sum_{k \in \mathcal{K}} r_k^{(t-1)} > \mathcal{K} \cdot r_{\text{target}}$  then
16:     Compute global rank budget in (6):  $m^{(t)}$ 
17:     Sort all candidates  $(k, i)$  by ascending  $\Delta^2 S_{k,i}^{(t)}$ 
18:     Prune the globally smallest  $\sum_{k \in \mathcal{K}} r_k^{(t-1)} - m^{(t)}$  components and update ranks
         $\{r_k^{(t)}\}_{k \in \mathcal{K}}$ 
19:   end if
20: end if
    {Early stopping (Line 21)}
21: early stop if  $\sum_{k \in \mathcal{K}} r_k^{(t)} \leq \mathcal{K} \cdot r_{\text{target}}$ 
22: end for
23: Set  $(w'_u, w'_\sigma, w'_v) \leftarrow (w_u, w_\sigma, w_v)$ 
  
```

At each training epoch (t), the algorithm first updates the model parameters following the same procedure as in LoRA-comp. It then performs instance-specific LoRA design using the updated parameters and the corresponding gradients obtained during the update step. During this instance-specific design process, the algorithm computes importance scores for each low-rank component in every adapted layer. Specifically, for the i -th rank component in layer k , an importance score ($S_{k,i}^{(t)}$) is defined as the contribution of the corresponding singular values ($\sigma_{k,i}^{(t)}$), i.e.,

$$S_{k,i}^{(t)} = s(\sigma_{k,i}^{(t)}), \quad (3)$$

where $s(\cdot)$ denotes the predefined importance functions used in AdaLoRA. Specifically, $s(\cdot)$ measures sensitivity based on the gradients of the weights.

From the third epoch onward, the algorithm evaluates the convergence stability of low-rank components by computing the second-order temporal difference of importance scores [45], i.e.,

$$\Delta^2 S_{k,i}^{(t)} = |S_{k,i}^{(t)} - 2S_{k,i}^{(t-1)} + S_{k,i}^{(t-2)}|. \quad (4)$$

The overall convergence indicator ($C^{(t)}$) is then obtained by summing the second-order differences across all layers and active rank components, i.e.,

$$C^{(t)} = \sum_{k \in \mathcal{K}} \sum_{i=1}^{r_k^{(t-1)}} \Delta^2 S_{k,i}^{(t)}. \quad (5)$$

If the convergence indicator ($C^{(t)}$) falls below a predefined threshold (τ), the algorithm interprets this as an indication of a stable training state and activates rank reduction with a reduction ratio of ρ . Otherwise, no rank pruning is performed, and the current ranks are preserved. When rank reduction is activated, a new rank budget is defined as

$$m^{(t)} = \max\left(\left\lfloor \rho \cdot \sum_{k \in \mathcal{K}} r_k^{(t-1)} \right\rfloor, \mathcal{K} \cdot r_{\text{target}}\right) \quad (6)$$

where $\rho \in (0, 1)$, $r_k^{(t-1)}$ denotes the rank allocated to layer k at epoch $t - 1$, and r_{target} represents the predefined per-layer target rank.

Finally, for each layer (k), low-rank components are pruned according to the newly defined rank budget ($m^{(t)}$) and the second-order temporal difference ($\Delta^2 S_{k,i}^{(t)}$). Specifically, only $m^{(t)}$ low-rank components are retained, while components with small $\Delta^2 S_{k,i}^{(t)}$ values are removed. This instance-specific LoRA design process is repeated at each training epoch until the rank budget ($m^{(t)}$) reaches the target rank ($\mathcal{K} \cdot r_{\text{target}}$). It is worth noting that the rank range (from the initial rank ($r^{(0)}$) to the target rank (r_{target})) is shared across all instances as a global configuration. The instance-specificity of ISRA therefore lies in which layers undergo rank reduction (layer selection) and when the reduction is triggered (pruning timing), both of which are determined adaptively based on the importance scores and convergence indicator of each individual instance.

An example of the dynamic instance-specific LoRA design procedure is illustrated in Figure 6. To ensure stable optimization, we incorporate both a warm-up training phase to prevent premature rank reduction and a post-training phase to further stabilize the adapted model.

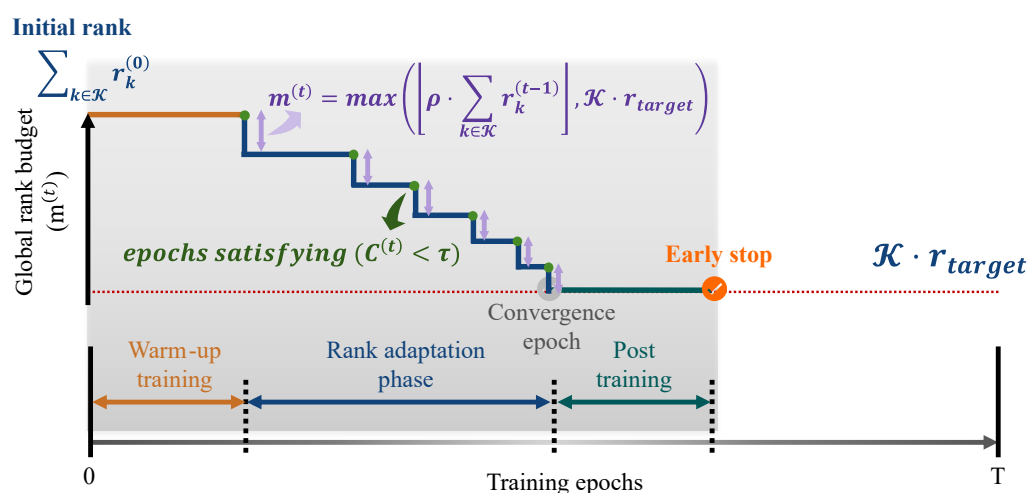


Figure 6. An example of the proposed dynamic instance-specific LoRA design procedure. To ensure stable optimization, we incorporate a warm-up phase to prevent premature rank reduction and a post-training phase to further stabilize the adapted model.

4.3. Multi-Scale Training Strategy

Image resolution directly influences the balance between global context modeling and optimization stability during TTFT [46,47]. While full-resolution training maintains global spatial coherence, it is prone to unstable and slow optimization due to single-instance gradients. Patch-based training mitigates these issues through mini-batch optimization and implicit spatial augmentation, at the expense of reduced global contextual awareness. Therefore, to reduce training time while improving performance, we adopt an MSTs that leverages these complementary properties. Specifically, during the early training phase prior to convergence, we employ lower-resolution inputs with batch-based optimization to promote stability and rapid convergence. In the subsequent post-training phase, we switch to the original image resolution to enhance the model's capacity to capture global contextual information. Concretely, during the warm-up and rank-adaptation phases, each input image is divided into non-overlapping 256×256 patches, and mini-batches of patches are used for optimization, where the batch size is set according to the input resolution of each dataset. Upon entering of the post-training phase (i.e., once early stopping is triggered), training switches to the original full-resolution image with a batch size of 1 for the remaining 10 post-training epochs. Dataset-specific configurations are available in our public repository.

5. Experimental Results

5.1. Experimental Settings

Our approach is architecture-agnostic and can be applied to various LIC backbones. To validate its effectiveness, we adopt two representative models, the Transformer-based LIC [6] and the CNN-based LIC [7]. Pretrained weights provided by the CompressAI library [48] are used as base models, referred to as TIC and Cheng2020.

Figure 7 illustrates the application of our approach to the TIC (a) and Cheng2020 (b) models. Both models adopt an autoencoder architecture consisting of a main encoder–decoder pair and a hyper encoder–decoder pair, designed to optimize the trade-off between distortion and bit rate for human perceptual quality. In our work, to maximize task-switching efficiency while minimizing the number of tunable parameters, we apply SVD-LoRA exclusively to the main encoder and decoder, as these components have the most significant impact on task adaptation [17]. For the TIC model, SVD-LoRA is applied exclusively to the Swin Transformer Blocks (STBs). In contrast, for the Cheng2020 model, SVD-LoRA is applied only to the CNN layers.

Target Tasks. While both closed-set and open-set scenarios are considered in our evaluation, we focus primarily on the open-set setting, which better captures the practical challenges of supporting multiple and diverse vision tasks in real-world deployments. Specifically, we conduct performance comparisons under the same experimental settings as those used in prior LoRA-comp studies in order to maintain continuity with its reported contributions. Accordingly, when comparing against LoRA-comp, we evaluate object detection [26], instance segmentation [49], and image classification [50] under the closed-set setting, while open-set evaluations are performed for object detection and instance segmentation tasks. Furthermore, in the ISER study, we additionally consider depth estimation [51] as an open-set task in order to evaluate conditions that are more distant in characteristics from the basis task (i.e., image classification in our work).

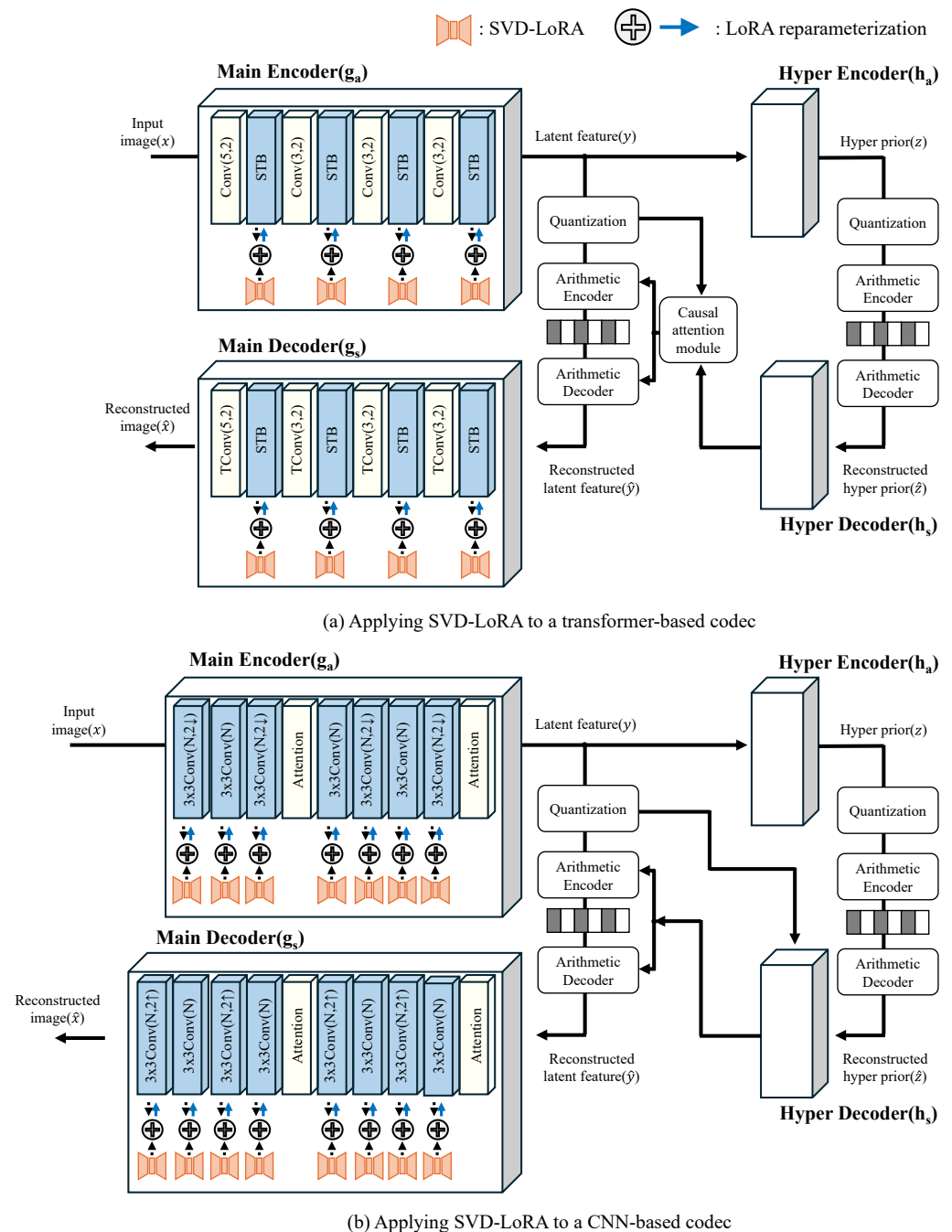


Figure 7. An example of the application of SVD-LoRA to the TIC (a) and Cheng2020 (b) codecs. While SVD-LoRA can be applied to both CNN and STB layers, we restrict fine-tuning to the STB layers in the main encoder/decoder of TIC and to the CNN layers in the main encoder/decoder of Cheng2020. This design improves task-switching efficiency while keeping the number of tunable parameters minimal. Reprinted with permission from Ref. [11]. 2025, IEEE.

Vision Models, Datasets, and Evaluation Metrics. To evaluate performance across diverse downstream tasks, we employ pre-trained vision models [26,49–51] as task-specific evaluators. Specifically, ResNet50 [50] is used for image classification, Faster R-CNN [26] for object detection, Mask R-CNN [49] for instance segmentation, and Depth Anything [51] for depth estimation. For evaluation, we use the entire COCO-2017 validation set (5000 images) [52] for object detection and instance segmentation, the entire ImageNet-1K validation set (50,000 images) [53] for image classification, and the full KITTI Eigen test split (697 images) [54] for depth estimation, rather than sampled subsets. All

compared methods are evaluated on identical image sets to ensure fair comparison. Performance is reported using mean average precision (mAP) for detection and segmentation, top-1 accuracy for classification, and the absolute relative error metric for depth estimation. In addition, numerical comparisons are conducted using the Bjøntegaard Delta (BD) metric [55].

We clarify the information used during TTFT under the open-set condition. Stage 1 (task-specific fine-tuning) always uses the task-agnostic ImageNet-pretrained ResNet50, irrespective of the eventual task. During Stage 2 (instance-specific TTFT), the loss in Equation (1) is computed via feature-level MSE against the target task's frozen evaluator network, without accessing ground-truth labels or task outputs. Critically, no task-specific parameters are pretrained or stored offline, and each instance's LoRA weights are initialized from this task-agnostic basis and further optimized per instance at test time, then discarded after encoding. We use "open-set" to denote the absence of stored, dataset-level task-specific adaptation, not the absence of task identity at test time.

Configuration Settings. We evaluate each method at four quality levels to illustrate its rate-distortion behavior. The quality levels are determined by the rate-distortion parameter (λ) of the base models. Following prior work, λ is set to $\{2, 1, 0.5, 0.2\}$ for both TIC and Cheng2020. While the training epoch budget (T) is adjustable, we fix it at 40 to match the experimental setting used in LoRA-comp. Within the ISER framework, however, the effective number of training epochs is automatically adapted according to the characteristics of each individual instance. The learning rate (δ) is set to 5×10^{-3} , consistent with the configuration used in LoRA-comp. For dynamic rank adaptation, we initialize the decoder rank to $r^{(0)} = 8$ and set the per-layer target rank to $r_{\text{target}} = 4$ while fixing the encoder rank at 4. The target rank ($r_{\text{target}} = 4$) is adopted from LoRA-comp, where rank 4 was identified as the optimal configuration for the compression scenarios considered in this work (LIC models, datasets, and target tasks), providing the best balance between task performance and TTFT overhead. The initial rank of $r^{(0)} = 8$ follows the empirical guideline of AdaLoRA, which recommends an initial-to-target ratio of $1.5 \times - 2 \times$ for stable rank pruning. Our $2 \times$ choice is further validated by the ablation study in Section 5.4. The rank reduction ratio (ρ) is set to 0.8, and the threshold (τ) is defined according to the quality level as $4 \times 10^{-7}, 2 \times 10^{-7}, 9 \times 10^{-8}, 3 \times 10^{-8}$, which are empirically determined in our experiments (see the hyperparameter sensitivity analysis in Section 5.4). In addition, the warm-up and post-training phases are set to 5 and 10 epochs, respectively, as determined empirically through preliminary experiments.

Hardware Configuration. Experiments were performed on a machine with four NVIDIA RTX 4090 GPUs (24 GB each), an AMD EPYC 7313 16-Core CPU, and 256 GB RAM. Both encoding and decoding were performed on a single GPU. The reported encoding time includes image patching, full-resolution post training, instance-specific fine-tuning (including dynamic rank pruning via ISRA and early stopping via ISER), auxiliary bit-stream generation (including singular-value quantization), and downstream task inference. The reported decoding time includes backbone codec inference and downstream task inference. Under the TIC backbone, the peak memory configuration occurs for the instance segmentation task, where the peak GPU memory usage is 17.79 GB during encoding and 5.94 GB during decoding.

Compared Methods. The following methods are compared in our work:

- **(1) TIC [6]:** TIC is a Transformer-based LIC model that is pre-trained with a focus on human perceptual quality. We use the publicly available weights trained on the Flickr2W dataset [56].

- **(2) Cheng2020 [7]:** Cheng2020 is a CNN-based LIC model that is also pre-trained with a focus on human perceptual quality. We use the publicly available weights trained on the ImageNet dataset [53].
- **(3) TransTIC [17]:** TransTIC applies VPT to adapt a backbone TIC model originally optimized for human perceptual quality to downstream machine vision tasks.
- **(4) Adapt-ICMH [10]:** Adapt-ICMH employs an adapter-based fine-tuning framework that jointly considers spatial and frequency domains to adapt the compression model to a target machine vision task.
- **(5) LoRA-comp [11]:** LoRA-comp adopts an instance-specific TTFT strategy based on SVD-LoRA to adapt a backbone compression codec to diverse machine vision tasks.

5.2. Instance-Specific Architectural Design: ISER vs. LoRA-Comp

LoRA-comp has already demonstrated superior performance over prior methods in both closed-set and open-set scenarios through its instance-specific adaptation strategy. Accordingly, we conduct performance comparisons under the same experimental settings as those used in prior LoRA-comp studies in order to maintain continuity with its reported contributions.

Figures 8 and 9 compare the performance of ISER and LoRA-comp. Figure 8 reports results using the TIC backbone, while Figure 9 presents results using the Cheng2020 backbone. Across all experimental settings considered in LoRA-comp, ISER consistently reduces encoding time while simultaneously improving performance relative to LoRA-comp. In particular, ISER achieves up to a 7% improvement in BD-Rate and up to a 44% reduction in average per-instance encoding time (in seconds). These results indicate that ISER effectively addresses open-set scenarios through fully instance-specific adaptation while substantially mitigating time overhead, making it well suited for practical compression applications that require both adaptability and efficiency. Table 2 summarizes the corresponding numerical results.

Table 2. Numerical results corresponding to Figure 8. Performance comparison between ISER and LoRA-comp using the TIC backbone. Bold indicates the best result.

Method	Closed-Set						Open-Set			
	Object Detection		Instance Seg.		Image Classification		Object Detection		Instance Seg.	
	BD-Rate ↓	BD-mAP ↑	BD-Rate ↓	BD-mAP ↑	BD-Rate ↓	BD-Acc. ↑	BD-Rate ↓	BD-mAP ↑	BD-Rate ↓	BD-mAP ↑
LoRA-comp	−54.96	2.98	−56.49	2.68	−70.04	0.14	−48.94	2.69	−49.51	2.43
ISER (ours)	−57.86	3.26	−58.39	2.77	−70.85	0.14	−50.49	2.93	−52.86	2.67

5.3. Analysis Under Open-Set Condition

LIC is a core component of modern vision systems. As vision applications continue to expand, LIC must increasingly address the challenge of supporting multiple open-set machine vision tasks. While maintaining strong performance on closed-set tasks remains essential, robust generalization to open-set tasks has become a critical requirement for practical deployment. To quantitatively characterize the properties of open-set tasks, we employ the CKA metric. This allows us to analyze performance variations as a function of the representational distance from the basis task.

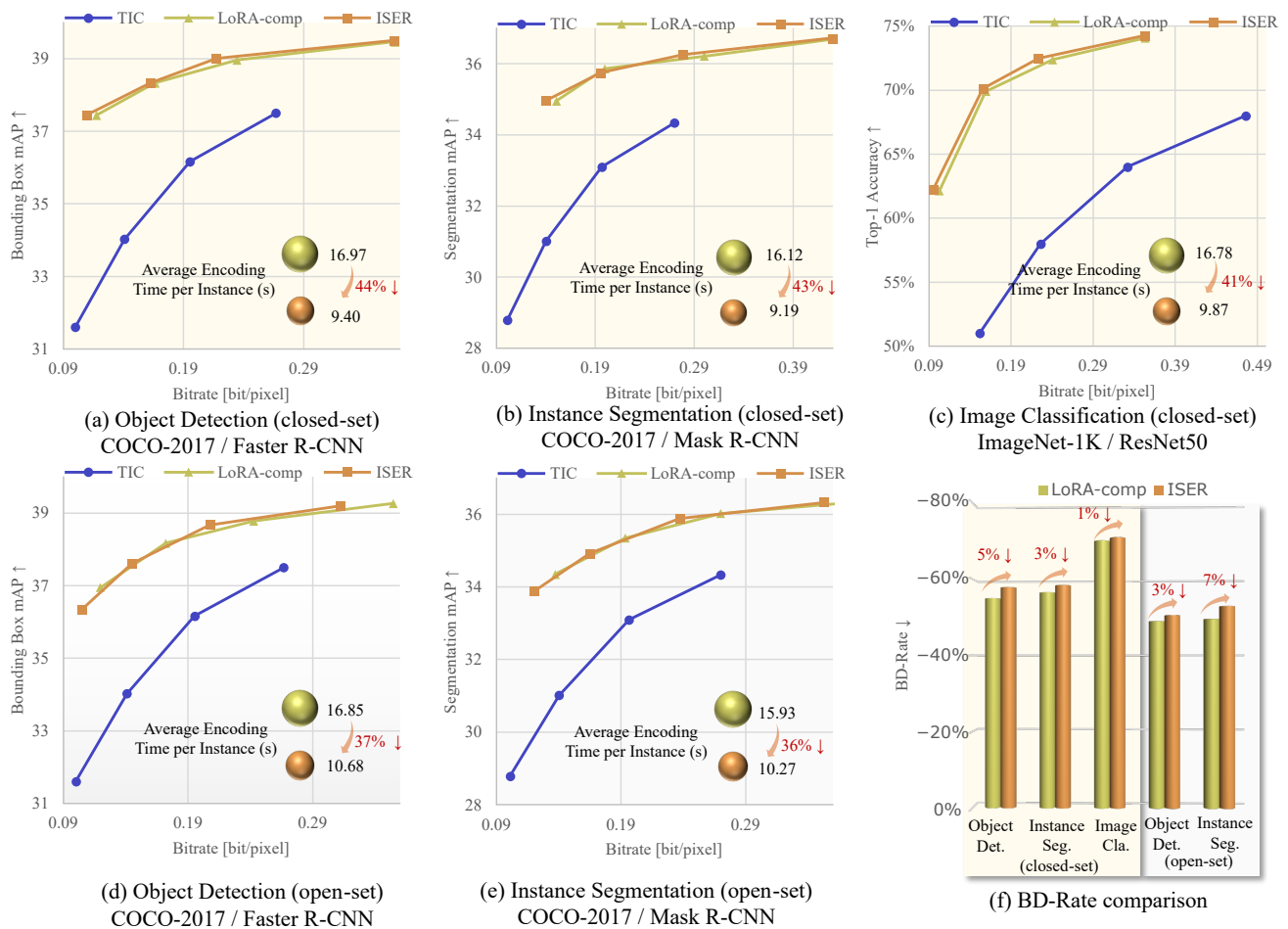


Figure 8. Performance comparison between ISER and LoRA-comp using the TIC backbone. Following the experimental settings of LoRA-comp, panels (a–c) present results under the closed-set condition, whereas panels (d,e) report results under the open-set condition. Panel (f) summarizes the BD-Rate of both methods across all five settings. Across all settings, ISER consistently reduces encoding time while simultaneously improving performance compared to LoRA-comp.

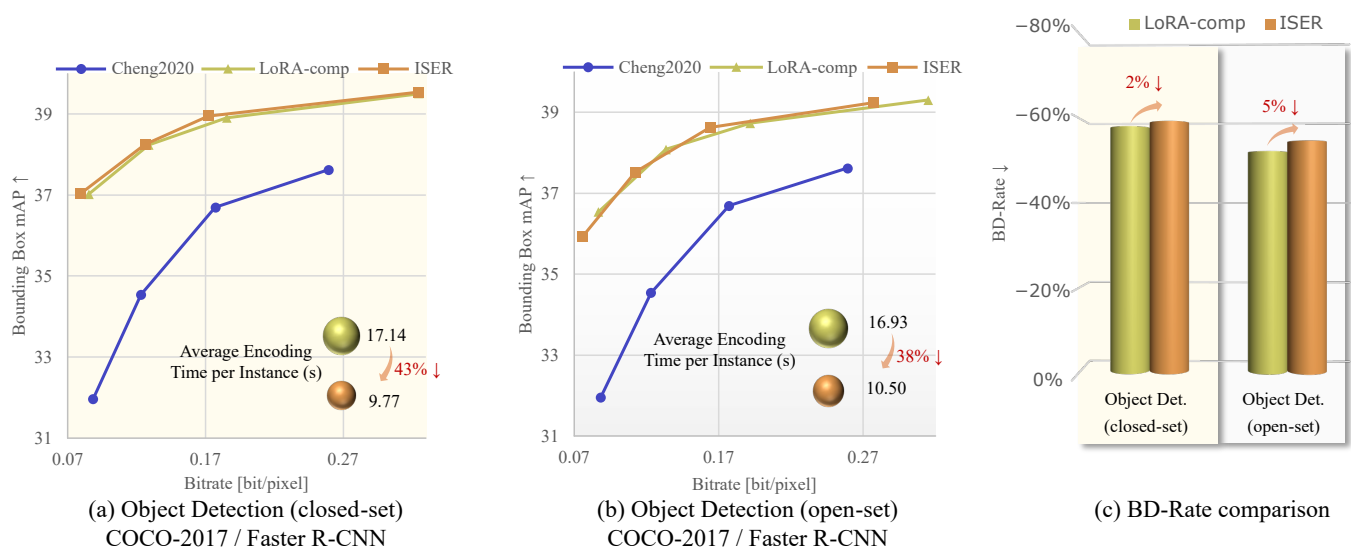


Figure 9. Performance comparison between ISER and LoRA-comp using the Cheng2020 backbone. ISER can be efficiently integrated into CNN-based LIC backbones and consistently reduces encoding time while simultaneously improving performance compared to LoRA-comp.

Representation Similarity Analysis for Open-Set Tasks. To quantify the representational distance among open-set tasks, we employ linear CKA, a widely adopted metric for analysis of similarities between representations learned by different neural network architectures [25]. For representation similarity analysis, we use ResNet50 as the reference model and measure its representational similarity with Depth Anything, Faster R-CNN, and Mask R-CNN. To ensure a consistent comparison, we focus on layers responsible for hierarchical multi-scale feature representations across all models. Specifically, we use layers 1–4 of ResNet50, layers 1–4 of the backbone networks in Faster R-CNN and Mask R-CNN, and the reassembly layers (1_rn–4_rn) of Depth Anything. All representation similarity analyses are conducted using the CIFAR-10 dataset [57].

We adopt CIFAR-10 as a common probing dataset for two reasons. First, Kornblith et al. [25] established CIFAR-10 as a standard benchmark for CKA-based analysis. They further showed that CKA reliably identifies layer correspondences not only within a single architecture but also across entirely different architectures, which supports the cross-architecture layer alignment described above. Second, Cui et al. [58] showed that CKA is confounded by the population structure of the input data used to compute it. In our setting, ResNet50, Faster R-CNN, Mask R-CNN, and Depth Anything are trained on different datasets that vary substantially in resolution, domain, and class distribution. Computing CKA separately on each model’s native dataset would therefore conflate genuine task-specific representational differences with differences arising purely from the input distributions. Using a single, fixed probing dataset across all models removes this confound and isolates representational differences that stem from the task itself.

Figure 10 visualizes representation similarity using linear CKA, where each axis index denotes an individual sub-block within layers 1–4 of the compared models. To quantitatively summarize the visualized results, we compute the mean CKA value across all compared layers, denoted as $mCKA$. Furthermore, we report a relative metric, r_mCKA , which normalizes $mCKA$ with respect to a reference comparison. Faster R-CNN and Mask R-CNN, which share an identical ResNet50 backbone, achieve a high representational similarity of about 86%. By contrast, Depth Anything exhibits markedly lower similarity, with $r_mCKA = 53.5\%$.

Quantitative Results. Figure 11 presents a comparative evaluation of open-set task performance and computational efficiency against competing methods. As shown in Figure 11a–c, ISER consistently outperforms the competing approaches across individual open-set tasks. In terms of latency, ISER achieves up to a 19.2% reduction in decoding time relative to TransTIC, since the reparameterization property of LoRA leaves the decoding path unaffected by instance-specific adaptation. This decoding-side efficiency comes at the cost of encoding time. Whereas TransTIC and Adapt-ICMH require no per-instance test-time optimization and thus incur negligible encoding overhead beyond a single forward pass, ISER’s instance-specific TTFT introduces additional encoding-time cost as a trade-off for its adaptability gains. This trade-off is favorable in the common deployment pattern where an image is compressed once but decoded repeatedly, such as archival storage, medical imaging, or streaming to multiple downstream task consumers, where the one-time encoding investment is amortized over many latency-critical decoding operations.

Figure 11d compares the BD-Rate performance across individual open-set tasks. All methods achieve performance improvements over the human-perceptual-quality-oriented TIC backbone. However, the competing methods exhibit limited gains on tasks that are more distant from the basis task, such as depth estimation, whereas ISER achieves a substantially larger improvement, reaching a 98% larger BD-Rate gain than TransTIC on the depth estimation task. Table 3 summarizes the corresponding numerical results.

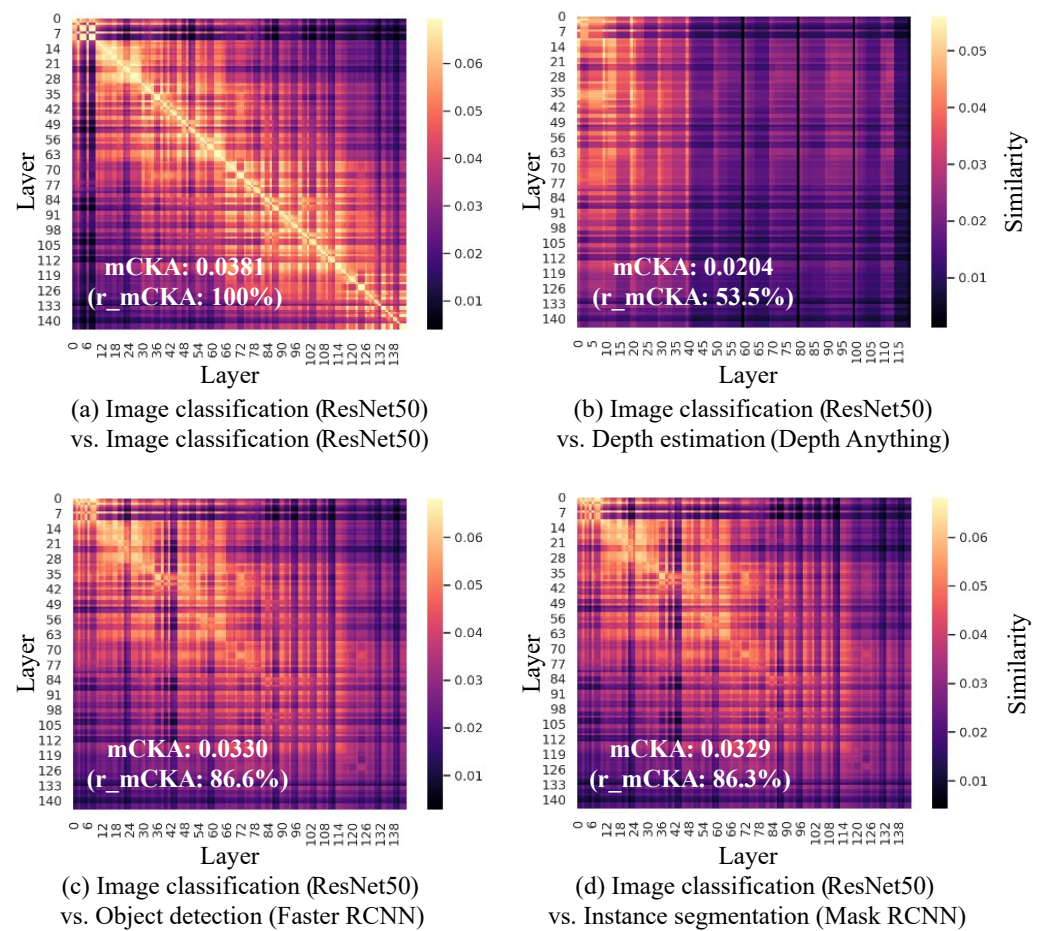


Figure 10. Representation similarity analysis for open-set tasks. We employ linear CKA to analyze representational similarities across different open-set tasks. To quantitatively summarize the results, we report a relative metric, r_mCKA . Faster R-CNN and Mask R-CNN, which share the same ResNet-50 backbone, exhibit high representational similarity ($\approx 86\%$). In contrast, Depth Anything shows substantially lower similarity (53.5%).

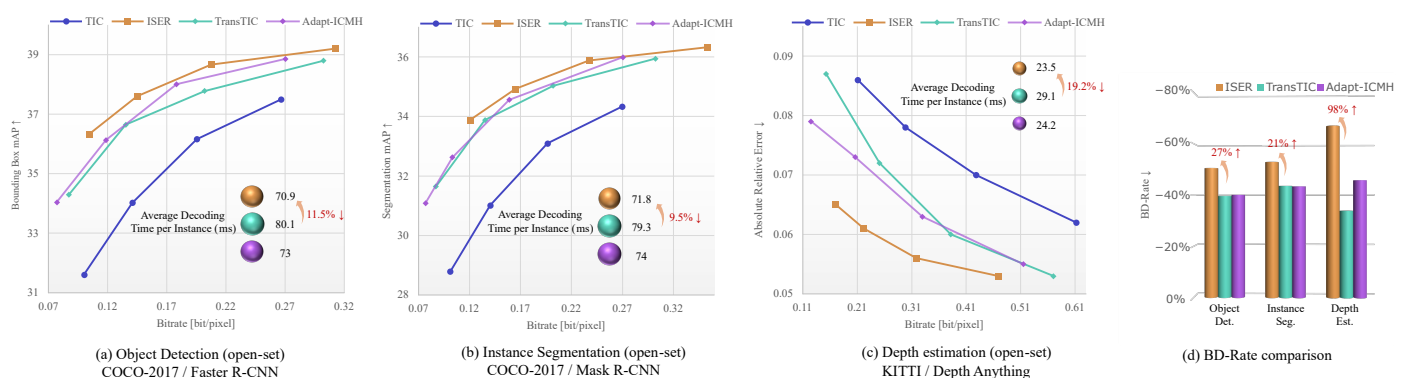


Figure 11. Performance comparison against competing methods on open-set tasks using the TIC backbone. ISER consistently outperforms competing approaches across individual open-set tasks while also providing faster decoding. In particular, ISER achieves a substantially larger gain in depth estimation, delivering an additional 98% performance improvement over TransTIC.

Table 3. Numerical results corresponding to Figure 11. Performance comparison against competing methods on open-set tasks using the TIC backbone. Bold indicates the best result, and underline indicates the second-best result.

Method	Object Detection		Instance Seg.		Depth Estimation	
	BD-Rate ↓	BD-mAP ↑	BD-Rate ↓	BD-mAP ↑	BD-Rate ↓	BD-Error ↓
ISER (ours)	−50.49	2.93	−52.86	2.67	−66.87	−0.02
TransTIC	−39.60	2.23	<u>−43.55</u>	2.28	−33.81	−0.011
Adapt-ICMH	<u>−40.09</u>	<u>2.58</u>	−43.35	<u>2.35</u>	<u>−45.73</u>	<u>−0.012</u>

While Table 3 reports task-level metrics such as BD-mAP, these metrics alone do not verify whether the underlying feature representations are genuinely improved rather than reflecting effects specific to the downstream task head. To assess this directly, we quantify the representation shift between closed-set and open-set adaptation at the feature level. For each input image, we obtain reconstructions from codecs adapted under closed-set and open-set conditions, extract activations from the final layer of the Faster R-CNN backbone, and compute linear CKA between the two. This layer is the last module of the ResNet-50 bottom-up pathway, immediately preceding the FPN and detection-specific heads, and therefore reflects the quality of the backbone representation itself rather than task-head behavior. Because the closed-set condition is pre-trained on the target task, it serves as a reference for the best achievable adaptation, and a higher CKA indicates that open-set adaptation recovers a representation closer to this reference without task-specific pre-training. CKA is computed per image by treating spatial positions within the feature map as samples, which yields one similarity value per image and allows the distribution across images to be characterized. Table 4 reports these statistics over the COCO-2017 validation set. The two instance-specific methods attain the highest similarity, with ISER at 0.4400 and LoRA-comp at 0.4230, followed by the task-specific TransTIC at 0.3717 and Adapt-ICMH at 0.2940. This ordering matches the task-level results in Table 3, indicating that the reported improvements are reflected in the underlying feature representation and not only in the final task output.

Table 4. Feature-level representation shift between closed-set and open-set adaptation. Per-image linear CKA computed on the final backbone layer of Faster R-CNN between reconstructions obtained under closed-set and open-set conditions, evaluated on the COCO-2017 validation set (object detection, quality level 1). Since the closed-set condition is pre-trained on the target task, higher similarity indicates that open-set adaptation recovers a representation closer to the task-informed optimum. Bold indicates the best result, and underline indicates the second-best result.

Method	Mean	Std. Dev.	Median	Min	Max
ISER (ours)	0.4400	0.3395	0.3293	0.0172	0.9446
LoRA-comp	<u>0.4230</u>	0.3574	<u>0.2796</u>	0.0089	0.9583
TransTIC	0.3717	0.3132	0.2697	0.0053	0.9412
Adapt-ICMH	0.2940	0.3078	0.1306	0.0116	0.9040

To assess the reliability of the reported efficiency metrics, we analyze both the instance-to-instance and run-to-run variation of the stopping epoch and encoding time under the open-set object detection setting (TIC backbone, quality level 1). Table 5 reports the results over the full COCO-2017 validation set ($n = 5000$). Within a single run, the stopping epoch varies substantially across instances (mean of 25.70 ± 6.35 and range of $[14, 40]$), and the encoding time varies accordingly (mean of 9.92 ± 2.29 s and range of $[6.61, 16.00]$ s), confirming that ISER adapts the training budget on a per-instance basis rather than converging to a fixed epoch. To determine whether this variation reflects genuine instance-specific behavior

or optimization stochasticity, we repeat the same experiment over five independent runs and compute the run-to-run standard deviation of the per-run average. The run-to-run standard deviation is an order of magnitude smaller than the instance-to-instance standard deviation for both metrics (0.66 vs. 6.35 epochs and 0.15 s vs. 2.29 s). This indicates that the observed variation is predominantly driven by instance-specific characteristics rather than by randomness in the TTFT optimization procedure, supporting the reliability of the average efficiency metrics reported throughout this paper.

Qualitative Results. Figure 12 presents qualitative visualizations of the results for open-set tasks. While the visual quality of the reconstructed outputs does not directly indicate machine vision task performance, the results reveal that all evaluated methods exhibit common qualitative characteristics across tasks throughout the experiments. Relative to the TIC backbone, TransTIC yields blurrier reconstructions, whereas Adapt-ICMH exhibits structured noise artifacts. By contrast, ISER produces reconstructions with enhanced sharpness and contrast. As shown in Figure 12a,b, the corresponding bit allocation maps reveal that ISER distributes bits more selectively toward task-relevant image regions.

Table 5. Distributional statistics of stopping epoch and encoding time under the open-set object detection setting (TIC backbone, quality level 1, and $n = 5000$ instances). Instance-to-instance variation substantially exceeds run-to-run variation, indicating that the observed spread predominantly reflects instance-specific characteristics rather than optimization stochasticity.

Statistic	Stopping Epoch	Enc. Time (s)
Instance-to-instance distribution (single run, $n = 5000$)		
Mean	25.70	9.92
Std. Dev.	6.35	2.29
Min	14	6.61
Max	40	16.00
Run-to-run distribution (mean over $n = 5000$ instances, 5 independent runs)		
Mean across runs	24.78	9.66
Std. Dev. across runs	0.66	0.15

5.4. Ablation Studies

We perform an ablation study to quantify the contribution of each component within the ISER framework and to evaluate the validity of the overall design. All ablation experiments are conducted using the TIC-based ISER framework.

Ablation on ISER components. We conduct an ablation study to evaluate the contribution of each component of the proposed ISER framework. Table 6 summarizes the ablation results. When only ISRA is applied, rank adaptation is performed throughout the entire predefined training epoch budget (T), which leads to relatively larger performance gains but incurs higher encoding-time overhead. In contrast, incorporating ISES and MSTs slightly reduces the magnitude of performance improvement while providing substantially greater benefits in terms of encoding-time reduction.

Table 6. Summary of the ablation study on ISER components. ISRA provides stronger performance gains at the cost of increased encoding time. Adding ISES and MSTs trades a small portion of performance for a substantial reduction in encoding time.

ISER Components	Avg. Enc. Time (s/Instance)	OD (Open)		IS (Open)		Depth (Open)	
		BD-Rate ↓	BD-mAP ↑	BD-Rate ↓	BD-mAP ↑	BD-Rate ↓	BD-Error ↓
ISRA only	12.6	−52.46	3.09	−54.66	2.87	−70.25	−0.022
ISRA + ISES	11.5	−51.33	3.02	−53.49	2.73	−67.62	−0.021
ISRA + ISES + MSTs	8.16	−50.49	2.93	−52.86	2.67	−66.87	−0.020

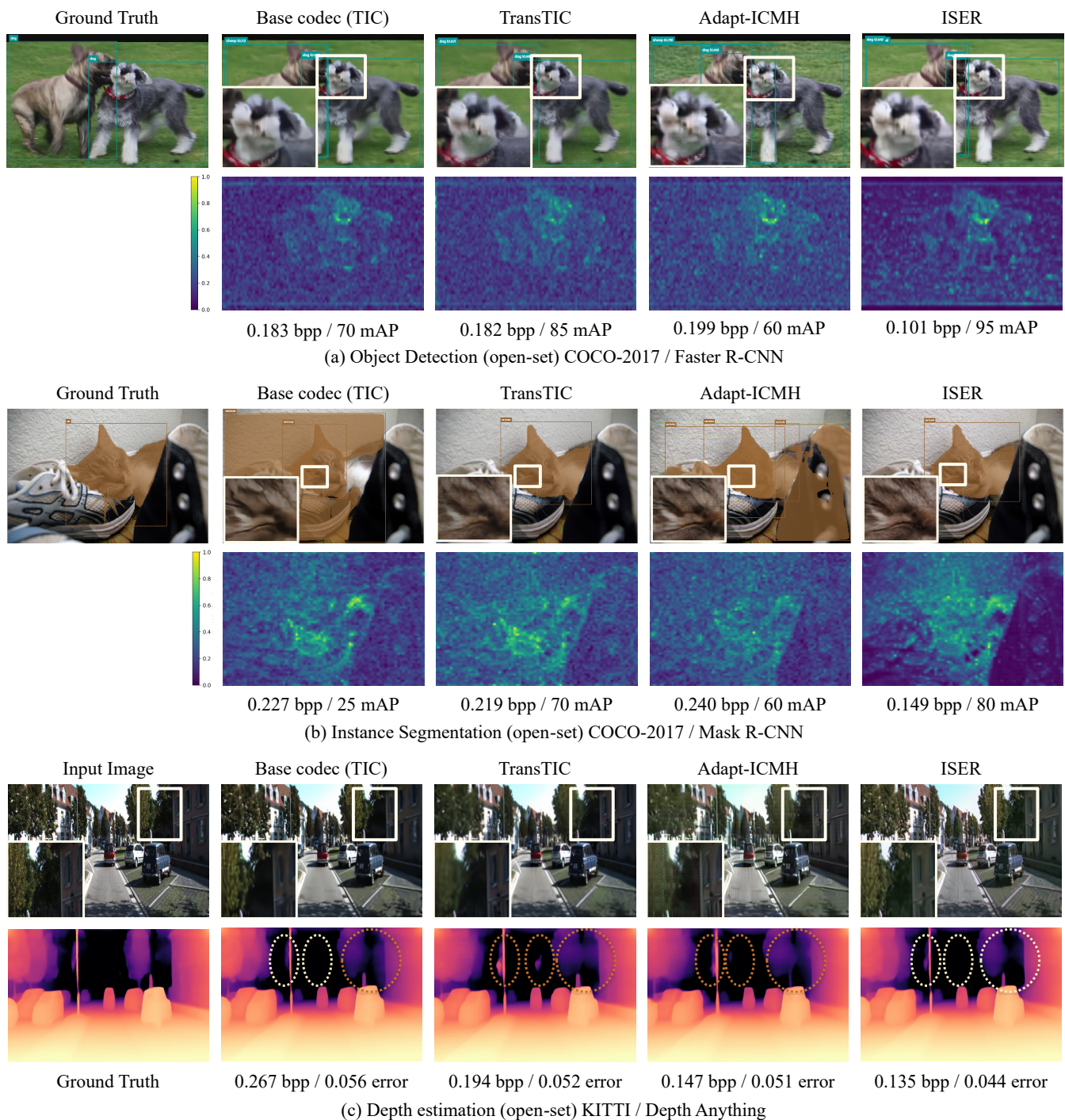


Figure 12. Qualitative visualizations of open-set task results. Compared to the TIC backbone, TransTIC produces blurrier reconstructions, whereas Adapt-ICMH introduces structured noise artifacts. In contrast, ISER yields reconstructions with improved sharpness and contrast. The corresponding bit-allocation maps further indicate that ISER allocates bits more effectively to task-relevant regions.

We clarify why the ablation in Table 6 adopts an additive rather than a full factorial structure. The additive structure directly reflects the architectural dependency among the three components. ISRA, ISES, and MSTs are not independent modules but are designed to build on one another within a single test-time fine-tuning process. ISES monitors the convergence of the ISRA optimization loop on a per-instance basis and determines when training stops. It therefore operates on signals produced internally by ISRA and cannot be

evaluated in isolation from it. MSTs uses this ISes stopping point to schedule the transition from low-resolution to full-resolution training. Evaluating MSTs without ISes would require its replacement with an arbitrary fixed epoch, which would not reflect ISes’s actual design. It would instead evaluate an unrelated, non-instance-specific baseline rather than isolate the contribution of the MSTs. For this reason, we adopt the additive configuration (ISRA only $\rightarrow$ ISRA + ISes $\rightarrow$ ISRA + ISes + MSTs) as the most faithful decomposition of each component’s contribution under ISes’s intended design.

Ablation on flexible architectural design. To validate the effectiveness of instance-specific flexible architectures, we compare ISRA against fixed LoRA architectures. In the fixed-architecture setting, a predefined subset of layers is assigned a fixed rank that remains constant across all training epochs. Using the TIC-based ISes framework, in which SVD-LoRA is applied to 28 layers, we consider three fixed configurations: Fixed-Front, which assigns ranks to layers 0–13; Fixed-Mid, which assigns ranks to layers 7–20; and Fixed-Back, which assigns ranks to layers 14–27. The ablation results are summarized in Table 7, with qualitative comparisons shown in Figure 13. While Fixed-Back generally yields stronger performance due to its proximity to the final reconstruction, Figure 13 reveals that no single fixed configuration is optimal across all input images. By contrast, ISRA adapts its architecture based on instance-specific image characteristics, resulting in consistently superior performance.

Table 7. Summary of the ablation study on flexible architectural design. This ablation study demonstrates that the proposed instance-specific flexible architecture design consistently outperforms any fixed-rank architecture. Bold indicates the best result, and underline indicates the second-best result.

Architecture Type	OD (Open)		IS (Open)		Depth (Open)	
	BD-Rate $\downarrow$	BD-mAP $\uparrow$	BD-Rate $\downarrow$	BD-mAP $\uparrow$	BD-Rate $\downarrow$	BD-Error $\downarrow$
ISRA (ours)	−52.46	3.09	−54.66	2.87	−70.25	−0.0221
Fixed-Front	−49.64	2.81	−46.72	2.49	−66.57	−0.019
Fixed-Mid	−50.71	3.00	−50.07	2.79	−68.95	−0.020
Fixed-Back	<u>−51.82</u>	<u>3.05</u>	<u>−54.50</u>	<u>2.84</u>	<u>−70.15</u>	<u>−0.0220</u>

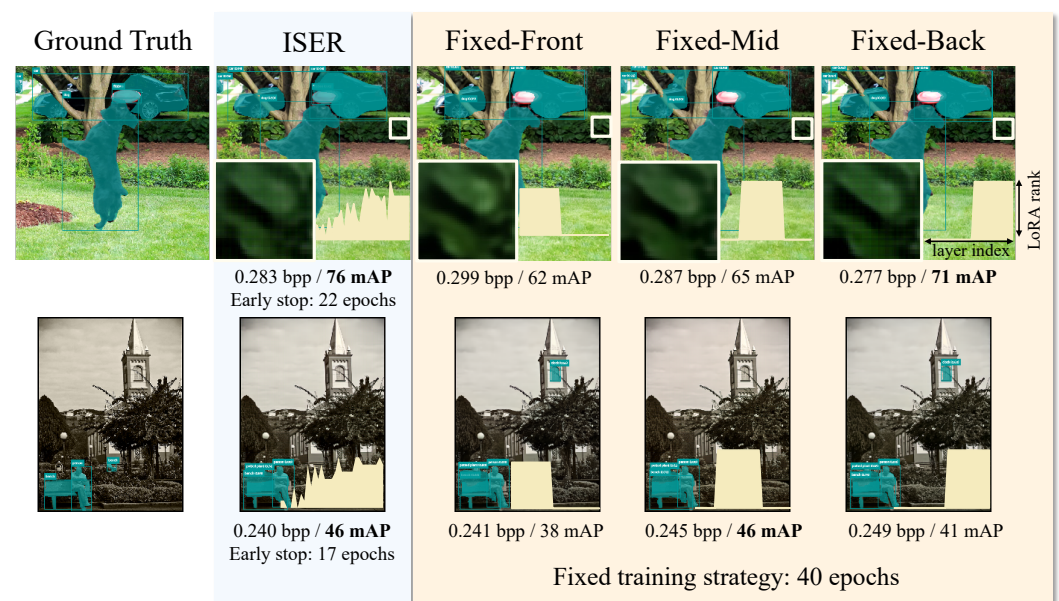


Figure 13. Visualization of the ablation study on flexible architectural design. This study shows that no single fixed-rank configuration is optimal across all input images. Therefore, the proposed instance-specific approach provides a more effective and robust adaptation strategy.

Ablation on warm-up and post training. As illustrated in Figure 6, warm-up and post-training phases are incorporated to ensure stable ISRA optimization. Table 8 summarizes the performance under different warm-up and post-training configurations. Extending the warm-up phase delays early stopping, leading to increased encoding time, whereas shortening the post-training phase limits achievable performance gains. Based on these observations, we adopt a 5-epoch warm-up and a 10-epoch post-training phase.

Table 8. Summary of the ablation study on warm-up and post training. Extending the warm-up phase delays early stopping, resulting in increased encoding time, whereas shortening the post-training phase limits the achievable performance gains. Enc. time refers specifically to the open-set OD task and is therefore not directly comparable to the task-averaged value in Table 6. Bold indicates the best result, and underline indicates the second-best result.

Scenario	OD (Open)		
	BD-Rate ↓	BD-mAP ↑	Enc. Time (s)
Warm-up: 5 epochs Post training: 10 epochs (ours)	<u>−50.49</u>	<u>2.93</u>	10.68
Warm-up: 10 epochs Post training: 10 epochs	−51.17	2.94	13.52
Warm-up: 10 epochs Post training: 5 epochs	−46.25	2.72	<u>13.09</u>

Ablation on TTFT iteration budget. LoRA-comp and ISER share the same underlying TTFT optimization loop, allowing the number of TTFT epochs to be adjusted to match the latency requirements of the target application. Table 9 reports this trade-off under the open-set object detection setting. Reducing the TTFT budget consistently lowers the encoding time at the cost of task performance. This trade-off is more favorable for ISER than for LoRA-comp. At 40 epochs, ISER requires only 9.92 s, a 30% reduction relative to LoRA-comp’s 14.13 s, while achieving a higher mAP (36.343 vs. 36.245). In the extreme case of zero TTFT epochs (w/o TTFT), both frameworks reduce to their task-specific initialization stage, requiring only a single backbone inference pass with no per-instance optimization. Even in this setting, ISER attains an mAP of 34.021, which already exceeds the raw, unadapted TIC backbone (mAP 31.603, Table 9) without any instance-specific adaptation. This indicates that a real-time-capable configuration of ISER—one that entirely forgoes per-instance TTFT—still provides a meaningful performance improvement over the unadapted backbone. It offers a practical operating point for latency-critical applications such as real-time optical imaging or edge deployment, where per-instance fine-tuning is infeasible.

Ablation on convergence indicator. We assess whether $C^{(t)}$, computed only from decoder-side importance scores ((3)–(5)), still reflects the stability of the full TTFT process, which also involves encoder-side updates. ISRA restricts rank adaptation to the decoder, since it directly determines the reconstructed output for downstream tasks. This raises a concern that a decoder-only signal might miss instability arising from encoder-side optimization, which is jointly captured in the total loss (ℓ) in (1). To address this concern, we compute the per-instance Pearson correlation between $C^{(t)}$ and ℓ across all test instances in the open-set object detection setting. The correlation averages 0.71 ± 0.09 (mean $\pm$ standard deviation), confirming that $C^{(t)}$ serves as a consistent proxy for training stability. Figure 14 illustrates two representative instances. These results indicate that focusing rank adaptation on the decoder does not compromise the stability of the overall optimization process.

Table 9. Ablation on the number of TTFT iterations for LoRA-comp and ISER under the open-set object detection setting (TIC backbone, quality level 1). Both frameworks allow the TTFT budget to be adjusted according to the target application, trading a modest amount of task performance for a substantial reduction in encoding time. The unadapted TIC backbone is included as a reference point for both frameworks.

TTFT Epochs	LoRA-Comp			ISER (Ours)		
	Bitrate (Bit/px)	mAP	Enc. Time (s)	Bitrate (Bit/px)	mAP	Enc. Time (s)
TIC (unadapted)	0.1000	31.603	—	0.1000	31.603	—
w/o TTFT	0.0959	34.021	—	0.0960	34.021	—
5	0.1063	35.278	1.18	0.1050	35.083	1.47
10	0.1059	35.838	4.28	0.1050	35.613	2.93
15	0.1066	35.883	5.86	0.1050	36.013	4.40
40	0.1058	36.245	14.13	0.1050	36.343	9.92

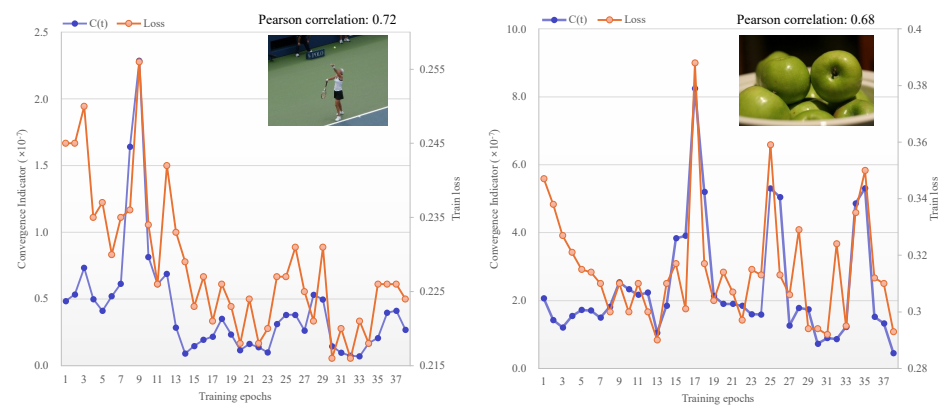


Figure 14. Visualization of the ablation study on the convergence indicator for two representative instances. The proposed convergence indicator ($C^{(t)}$) closely tracks the training loss (ℓ). Across the full test set, the per-instance Pearson correlation is 0.71 ± 0.09 , indicating that $C^{(t)}$ reliably reflects training stability (source images: COCO-2017 dataset [52]).

Ablation on initial rank ratio. To validate the selection of $r^{(0)} = 8$, we evaluate three initial-to-target rank ratios following the empirical guideline of AdaLoRA: $r^{(0)} = 6$ ($1.5\times$), $r^{(0)} = 8$ ($2\times$, ours), and $r^{(0)} = 12$ ($3\times$). As shown in Table 10, the $1.5\times$ ratio yields slightly degraded performance due to insufficient parameter exploration during early adaptation, while the $3\times$ ratio increases encoding-time overhead without a commensurate performance gain. The $2\times$ ratio achieves the best trade-off, confirming the suitability of $r^{(0)} = 8$.

Table 10. Ablation on initial rank ratio. The $2\times$ initial-to-target rank ratio ($r^{(0)} = 8$) achieves the best balance between task performance and encoding efficiency. Enc. time refers specifically to the open-set OD task and is therefore not directly comparable to the task-averaged value in Table 6. Bold indicates the best result, and underline indicates the second-best result.

$r^{(0)}$ (Ratio)	OD (Open)		
	BD-Rate ↓	BD-mAP ↑	Enc. Time (s)
6 ($1.5\times$)	−49.03	2.67	8.31
8 ($2\times$, ours)	−50.49	2.93	<u>10.68</u>
12 ($3\times$)	<u>−50.37</u>	<u>2.90</u>	13.65

Ablation on hyperparameter sensitivity. We evaluate sensitivity to the rank-reduction ratio (ρ) and the convergence threshold (τ). Table 11 reports OD (open) performance under

different ρ and τ configurations. A smaller ρ (0.6) aggressively reduces the rank budget, causing premature pruning and slight performance degradation. A larger ρ (0.9) retains more ranks per step, increasing encoding time without significant performance gain. Our choice of $\rho = 0.8$ provides the best trade-off. For τ , a single global threshold fails to account for the differing magnitudes of importance scores across quality levels. A loose threshold terminates convergence prematurely and degrades compression performance, while a tight threshold prolongs encoding without proportional gain. Our quality-level-adaptive τ avoids this sensitivity and achieves the best balance across all metrics.

Table 11. Ablation on rank-reduction ratio (ρ) and convergence threshold (τ). Both $\rho = 0.8$ and quality-level-adaptive τ achieve the best balance between performance and encoding efficiency. Bold indicates the best result, and underline indicates the second-best result.

Configuration	OD (Open)		
	BD-Rate ↓	BD-mAP ↑	Enc. Time (s)
<i>Rank-reduction ratio (ρ)</i>			
$\rho = 0.6$	−48.79	2.60	9.24
$\rho = 0.8$ (ours)	−50.49	2.93	<u>10.68</u>
$\rho = 0.9$	<u>−50.25</u>	<u>2.88</u>	12.12
<i>Convergence threshold (τ)</i>			
$\tau = 2 \times 10^{-7}$ (single global)	−48.23	2.67	9.45
$\tau = 9 \times 10^{-8}$ (single global)	<u>−50.42</u>	<u>2.90</u>	12.71
Quality-level-adaptive (ours)	−50.49	2.93	<u>10.68</u>

Ablation on input degradation robustness. Since ISER derives its adaptation signal from gradients computed on each test input, we evaluate how its effectiveness changes when the input image itself is degraded. We apply Gaussian blur and additive Gaussian noise to the input prior to compression and evaluate on 100 randomly sampled images from the COCO-2017 validation set under the open-set object detection setting. Table 12 reports the results. ISER maintains a consistent mAP improvement over the TIC backbone across all degradation conditions, and no abrupt collapse in adaptation performance is observed. Under blur, the improvement is preserved and, in fact, widens with severity. Under the stronger blur setting, the backbone drops from 39.93 to 30.06 mAP, whereas ISER drops only from 44.58 to 38.31, increasing the gain from 4.65 to 8.25 mAP. This suggests that instance-specific adaptation compensates for the loss of high-frequency content that the backbone alone cannot recover. Under additive noise, the gain narrows progressively with noise strength, from 4.65 under the clean setting to 3.15 and 1.46 under the two noise levels, indicating that the per-instance gradient signal becomes less discriminative as the input becomes noisier. ISER nonetheless retains a positive gain under every evaluated condition.

Table 12. Ablation on input degradation robustness. Performance under Gaussian blur and additive Gaussian noise applied to the input image prior to compression, evaluated on 100 randomly sampled images from the COCO-2017 validation set under the open-set object detection setting (TIC backbone, quality level 1). The uncompressed column reports mAP on the degraded image without compression, serving as an upper bound for each condition.

Condition	Uncomp.	TIC (Backbone)		ISER (Ours)		Δ mAP
	mAP	bpp	mAP	bpp	mAP	
None	48.83	0.0924	39.93	0.119	44.58	+4.65
Blur 1 ($k = 5, \sigma = 1$)	46.94	0.0740	40.50	0.108	45.17	+4.67
Blur 2 ($k = 9, \sigma = 2.5$)	42.04	0.0552	30.06	0.077	38.31	+8.25

Table 12. Cont.

Condition	Uncomp.	TIC (Backbone)		ISER (Ours)		Δ mAP
	mAP	bpp	mAP	bpp	mAP	
Noise 1 ($\sigma = 0.02$)	45.96	0.0931	41.58	0.114	44.73	+3.15
Noise 2 ($\sigma = 0.05$)	42.74	0.0946	39.58	0.100	41.04	+1.46

5.5. Computational Complexity

Table 13 highlights the cost efficiency of ISER in terms of both computation and memory. Despite employing LoRA-based adaptation, ISER and LoRA-comp introduce negligible overhead due to their low-rank configurations (rank 4 or 8), with no noticeable increase in computational complexity relative to the backbone and only modest parameter growth (2% and 3% increases for the encoder and decoder, respectively). Moreover, although ISER temporarily adopts a higher initial rank (8) during ISRA, hard pruning is applied at inference time so that the final decoder operates at the target rank (4), thereby preserving inference-time efficiency and model compactness. In contrast, TransTIC incurs substantially higher complexity, increasing computational cost by 133% in the encoder and 42% in the decoder relative to the backbone, along with a 44% increase in encoder parameters. Adapt-ICMH introduces moderate overhead, increasing computational complexity by 10% for both the encoder and decoder and increasing parameters by 5% for both components.

Table 13. Summary of the computational complexity comparison. This result highlights the cost efficiency of ISER in terms of both computational overhead and parameter size. Bold indicates the best result, and underline indicates the second-best result.

Method	kMACs/Pixel		Params (M)	
	Encoder	Decoder	Encoder	Decoder
TIC (backbone)	142.31	142.53	3.64	3.86
TransTIC	332.03	202.60	5.24	3.89
Adapt-ICMH	157.16	157.38	3.84	4.06
LoRA-comp	142.31	142.53	3.72	<u>3.94</u>
ISER (Enc-side TTFT)	142.31	142.53	3.72	3.99
ISER (Dec-side inference)	-	142.53	-	<u>3.94</u>

Beyond computational and memory overhead, we further verify the bitstream overhead introduced by ISER's decoder-side rank adaptation. Although ISRA prunes ranks along an instance-specific path during TTFT, all decoder SV layers converge to the same target rank ($r_{\text{target}} = 4$), regardless of the instance, which is fixed as part of the model configuration shared by both the encoder and decoder. Consequently, the decoder does not require any additional layer-wise rank configuration, pruning mask, or index information to reconstruct the transmitted SVs. Only the resulting rank-4 SVs, structurally identical to those of LoRA-comp, are encoded into the bitstream. Table 14 reports the measured SV overhead ratio across all open-set tasks and quality levels. The overhead ratio ranges from 0.29% to 2.05%, with an average of 0.98% across all tasks and quality levels, confirming that the additional signaling required by ISER remains below 1% of the total bitstream, on average, and does not introduce meaningful overhead beyond LoRA-comp.

Table 14. Auxiliary bitstream overhead introduced by the decoder-side SVs across open-set tasks and quality levels. Since all decoder SV layers converge to the same target rank ($r_{\text{target}} = 4$), regardless of the instance-specific pruning path taken by ISRA, no additional layer-wise rank configuration, pruning mask, or index information needs to be transmitted. Only the resulting rank-4 SVs are encoded into the bitstream, identical in structure to LoRA-comp.

Task	Quality Level	Total Bitrate (Bit/Pixel)	Image Bitrate (Bit/Pixel)	Decoder SV Bitrate (Bit/Pixel)	SV Overhead Ratio (%)
OD (open)	1	0.105	0.103	0.0021	2.05
	2	0.145	0.143	0.0021	1.46
	3	0.208	0.205	0.0021	1.04
	4	0.312	0.310	0.0021	0.67
IS (open)	1	0.121	0.119	0.0021	1.73
	2	0.165	0.163	0.0020	1.24
	3	0.237	0.235	0.0020	0.86
	4	0.353	0.351	0.0020	0.58
Depth (open)	1	0.170	0.169	0.0014	0.80
	2	0.222	0.220	0.0014	0.62
	3	0.318	0.317	0.0014	0.43
	4	0.469	0.468	0.0014	0.29

6. Conclusions

This paper presented Instance-Specific Early Stopping with Dynamic Rank Adaptation (ISER), an instance-specific TTFT framework that extends LoRA-comp by reducing TTFT overhead and enabling the design of an instance-adaptive LoRA architecture. ISER consistently outperforms competing approaches under both closed-set and open-set settings, achieving up to a 7% BD-Rate improvement and up to a 44% reduction in encoding time over LoRA-comp, as well as up to a 98% relative improvement in BD-Rate gain and up to a 19.2% reduction in decoding time over TransTIC. The proposed framework incurs negligible additional computational complexity relative to the backbone, with only modest parameter growth (2% and 3% for the encoder and decoder, respectively). In addition, ISER supports plug-and-play integration across CNN- and Transformer-based backbones, allowing for scalable performance gains as backbone architectures advance. Finally, we provide systematic insights into open-set task-oriented compression by quantifying representational distances using Centered Kernel Alignment (CKA), both to characterize the distances among open-set tasks and to verify at the feature level that the reported improvements reflect genuine gains in representation quality.

The encoding-time cost of ISER is incurred once per image, and the framework is designed for deployment patterns in which encoding is performed offline or as a backend process. Representative settings include archival storage, medical imaging repositories, and airborne or satellite imaging pipelines, where acquired data is typically re-encoded at a ground station or backend facility and subsequently analyzed many times over its lifetime. In these settings, it is the decoding side that operates under latency constraints, and ISER is favorable in this respect, since the reparameterization property of LoRA leaves the decoding path unaffected by instance-specific adaptation. Deployments that instead require encoding on embedded hardware under a real-time constraint fall outside this design target and would use the reduced TTFT budget analyzed in Table 9, where encoding reduces to a single forward pass while still outperforming the unadapted backbone.

Despite these contributions, the current framework has room for improvement. As shown in Table 12, ISER retains a positive gain over the backbone under blurred and noisy inputs, but this gain narrows progressively as noise strength increases. This

reflects a structural property of the method. Because ISER derives its adaptation signal from gradients computed on each individual test input, its effectiveness depends on that signal remaining informative, and it therefore becomes less effective as the input itself degrades. Architecture-level approaches that embed structural compensation directly into the network offer a complementary direction. For example, a self-compensating graph convolutional network has been proposed for skeleton-based action recognition under data constraints [59]. This approach does not depend on an instance-specific optimization signal at test time and may therefore provide robustness precisely in the regimes where ISER's gradient-based signal becomes less discriminative. Exploring how such structural compensation mechanisms could be incorporated into the TTFT framework to improve robustness under more severe data constraints is a promising direction for future work.

Author Contributions: Conceptualization, U.P., G.-M.P. and J.H.K.; methodology, U.P.; software, U.P. and S.J.; formal analysis, U.P. and S.J.; resources, S.K. and J.L.; writing—original draft preparation, U.P.; writing—review and editing, G.-M.P. and J.H.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Technology Innovation Program (or Industrial Strategic Technology Development Program) (RS-2023-00235718, 23040-15FC) funded by the Ministry of Trade, Industry & Energy (MOTIE, Korea) (1415187505); by a National Research Foundation of Korea (NRF) grant (RS-2025-02216217); by an Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2025-25442569, AI Star Fellowship Support Program (Sungkyunkwan Univ.)); by an IITP grant funded by the Korea government (MSIT) (RS-2019-II190421, Artificial Intelligence Graduate School Program (Sungkyunkwan University)); by an IITP-ICT Creative Consilience Program grant funded by the Korea government (MSIT) (IITP-2026-RS-2020-II201821); by an IITP-ITRC (Information Technology Research Center) grant funded by the Korea government (Ministry of Science and ICT) (RS-2021-II212052); and by a Korea Research Institute for defense Technology planning and advancement (KRIT) grant funded by the Korean government, Defense Acquisition Program Administration (DAPA), under Grant KRIT-CT-23-021, Development of Multi-Modal Information Fusion and High-Realization Battlefield Data Generation Technology, in 2025.

Informed Consent Statement: Not applicable. The image shown in Figure 14 was sourced from the publicly available COCO-2017 benchmark dataset, and the depicted individual was anonymized (face-blurred) prior to publication. No identifiable human subject appears in the manuscript.

Data Availability Statement: The code implementing the proposed method is publicly available at <https://github.com/unkipark/ISER> (accessed on 17 August 2026).

Conflicts of Interest: Author Unki Park is employed by the Samsung Institute of Technology. Authors Sangmin Kim and Jeungsub Lee are employed by Hanwha Systems. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Ballé, J.; Laparra, V.; Simoncelli, E.P. End-to-End Optimized Image Compression. In Proceedings of the International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
2. Jiao, L.; Zhao, J. A Survey on the New Generation of Deep Learning in Image Processing. *IEEE Access* **2019**, *7*, 172231–172263. [CrossRef]
3. Esenlik, S.; Wu, Y.; Zhang, Z.; Wang, Y.-K.; Zhang, K.; Zhang, L.; Ascenso, J.; Liu, S. An Overview of the JPEG AI Learning-Based Image Coding Standard. *IEEE Trans. Circuits Syst. Video Technol.* **2026**, *36*, 2520–2537. [CrossRef]
4. Duan, L.; Liu, J.; Yang, W.; Huang, T.; Gao, W. Video Coding for Machines: A Paradigm of Collaborative Compression and Intelligent Analytics. *IEEE Trans. Image Process.* **2020**, *29*, 8680–8695. [CrossRef]
5. Yang, W.; Huang, H.; Hu, Y.; Duan, L.-Y.; Liu, J. Video Coding for Machines: Compact Visual Representation Compression for Intelligent Collaborative Analytics. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 5174–5191. [CrossRef]

6. Lu, M.; Guo, P.; Shi, H.; Cao, C.; Ma, Z. Transformer-Based Image Compression. In Proceedings of the 2022 Data Compression Conference (DCC), Snowbird, UT, USA, 22–25 March 2022; p. 469.
7. Cheng, Z.; Sun, H.; Takeuchi, M.; Katto, J. Learned Image Compression with Discretized Gaussian Mixture Likelihoods and Attention Modules. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 7936–7945.
8. Choi, H.; Bajić, I.V. Scalable Image Coding for Humans and Machines. *IEEE Trans. Image Process.* **2022**, *31*, 2739–2754. [[CrossRef](#)]
9. Guo, H.; Sui, L.; Zhang, C.; Chen, Z.; Yang, W.; Duan, L. A Unified Image Compression Method for Human Perception and Multiple Vision Tasks. In Proceedings of the European Conference on Computer Vision (ECCV), Milan, Italy, 29 September–4 October 2024; pp. 342–359.
10. Li, H.; Li, S.; Ding, S.; Dai, W.; Cao, M.; Li, C.; Zou, J.; Xiong, H. Image Compression for Machine and Human Vision with Spatial-Frequency Adaptation. In Proceedings of the European Conference on Computer Vision (ECCV), Milan, Italy, 29 September–4 October 2024; pp. 382–399.
11. Park, U.; Jeong, S.; Jang, Y.; Park, G.-M.; Ko, J.H. Test-Time Fine-Tuning of Image Compression Models for Multi-Task Adaptability. In Proceedings of the 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 11–15 June 2025; pp. 4430–4440.
12. Harell, A.; Foroutan, Y.; Ahuja, N.; Datta, P.; Kanzariya, B.; Somayazulu, V.S.; Tickoo, O.; de Andrade, A.; Bajić, I.V. Rate-Distortion Theory in Coding for Machines and Its Applications. *IEEE Trans. Pattern Anal. Mach. Intell.* **2025**, *47*, 5501–5519. [[CrossRef](#)]
13. Le, N.; Zhang, H.; Cricri, F.; Ghaznavi-Youvalari, R.; Rahtu, E. Image Coding for Machines: An End-to-End Learned Approach. In Proceedings of the ICASSP 2021—2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, 6–11 June 2021; pp. 1590–1594.
14. Le, N.; Zhang, H.; Cricri, F.; Ghaznavi-Youvalari, R.; Rezazadegan Tavakoli, H.; Rahtu, E. Learned Image Coding for Machines: A Content-Adaptive Approach. In Proceedings of the 2021 IEEE International Conference on Multimedia and Expo (ICME), Shenzhen, China, 5–9 July 2021; pp. 1–6.
15. Liu, L.; Hu, Z.; Chen, Z.; Xu, D. ICMH-Net: Neural Image Compression towards Both Machine Vision and Human Vision. In Proceedings of the 31st ACM International Conference on Multimedia (MM '23), Ottawa, ON, Canada, 29 October–3 November 2023; pp. 8047–8056.
16. Hu, Y.; Yang, S.; Yang, W.; Duan, L.-Y.; Liu, J. Towards Coding for Human and Machine Vision: A Scalable Image Coding Approach. In Proceedings of the 2020 IEEE International Conference on Multimedia and Expo (ICME), London, UK, 6–10 July 2020; pp. 1–6.
17. Chen, Y.-H.; Weng, Y.-C.; Kao, C.-H.; Chien, C.; Chiu, W.-C.; Peng, W.-H. TransTIC: Transferring Transformer-Based Image Compression from Human Perception to Machine Perception. In Proceedings of the 2023 IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 1–6 October 2023; pp. 23240–23250.
18. Liu, K.; Liu, D.; Li, L.; Yan, N.; Li, H. Semantics-to-Signal Scalable Image Compression with Learned Reversible Representations. *Int. J. Comput. Vis.* **2021**, *129*, 2605–2621. [[CrossRef](#)]
19. Yan, N.; Gao, C.; Liu, D.; Li, H.; Li, L.; Wu, F. SSSIC: Semantics-to-Signal Scalable Image Coding with Learned Structural Representations. *IEEE Trans. Image Process.* **2021**, *30*, 8939–8954. [[CrossRef](#)]
20. Duan, Z.; Ma, Z.; Zhu, F. Unified Architecture Adaptation for Compressed Domain Semantic Inference. *IEEE Trans. Circuits Syst. Video Technol.* **2023**, *33*, 4108–4121. [[CrossRef](#)] [[PubMed](#)]
21. Feng, R.; Jin, X.; Guo, Z.; Feng, R.; Gao, Y.; He, T.; Zhang, Z.; Sun, S.; Chen, Z. Image Coding for Machines with Omnipotent Feature Learning. In Proceedings of the European Conference on Computer Vision (ECCV), Tel Aviv, Israel, 23–27 October 2022; pp. 510–528.
22. Liu, J.; Sun, H.; Katto, J. Learning in Compressed Domain for Faster Machine Vision Tasks. In Proceedings of the 2021 International Conference on Visual Communications and Image Processing (VCIP), Munich, Germany, 5–8 December 2021; pp. 1–5.
23. Liu, J.; Sun, H.; Katto, J. Improving Multiple Machine Vision Tasks in the Compressed Domain. In Proceedings of the 2022 26th International Conference on Pattern Recognition (ICPR), Montreal, QC, Canada, 21–25 August 2022; pp. 331–337.
24. Song, M.; Choi, J.; Han, B. Variable-Rate Deep Image Compression through Spatially Adaptive Feature Transform. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 11–17 October 2021; pp. 2380–2389.
25. Kornblith, S.; Norouzi, M.; Lee, H.; Hinton, G. Similarity of Neural Network Representations Revisited. In Proceedings of the International Conference on Machine Learning (ICML), Long Beach, CA, USA, 9–15 June 2019.
26. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 1137–1149. [[CrossRef](#)] [[PubMed](#)]
27. Abdi, H.; Williams, L.J. Principal Component Analysis. *Wiley Interdiscip. Rev. Comput. Stat.* **2010**, *2*, 433–459.
28. Ballé, J.; Minnen, D.; Singh, S.; Hwang, S.; Johnston, N. Variational Image Compression with a Scale Hyperprior. In Proceedings of the International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.

29. Liu, J.; Sun, H.; Katto, J. Learned Image Compression with Mixed Transformer-CNN Architectures. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023; pp. 14388–14397.
30. Minnen, D.; Ballé, J.; Toderici, G.D. Joint Autoregressive and Hierarchical Priors for Learned Image Compression. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 10794–10803.
31. He, D.; Yang, Z.; Peng, W.; Ma, R.; Qin, H.; Wang, Y. ELIC: Efficient Learned Image Compression with Unevenly Grouped Space-Channel Contextual Adaptive Coding. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 5718–5727.
32. Wallace, G.K. The JPEG Still Picture Compression Standard. *Commun. ACM* **1991**, *34*, 30–44. [\[CrossRef\]](#)
33. Marcellin, M.W.; Gormish, M.J.; Bilgin, A.; Boliek, M.P. An Overview of JPEG-2000. In *Proceedings of the Data Compression Conference (DCC 2000), Snowbird, UT, USA, 28–30 March 2000*; IEEE: Piscataway, NJ, USA, 2000; pp. 523–541.
34. Klopp, J.P.; Liu, K.; Chen, L.; Chien, S. How to Exploit the Transferability of Learned Image Compression to Conventional Codecs. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 16165–16174.
35. Gao, Y.; Shi, X.; Zhu, Y.; Wang, H.; Tang, Z.; Zhou, X.; Li, M.; Metaxas, D.N. Visual Prompt Tuning for Test-Time Domain Adaptation. *arXiv* **2022**, arXiv:2210.04831.
36. Jia, M.; Tang, L.; Chen, B.; Cardie, C.; Belongie, S.; Hariharan, B.; Lim, S. Visual Prompt Tuning. In Proceedings of the European Conference on Computer Vision (ECCV), Tel Aviv, Israel, 23–27 October 2022; pp. 709–727.
37. Chavan, A.; Liu, Z.; Gupta, D.; Xing, E.; Shen, Z. One-for-All: Generalized LoRA for Parameter-Efficient Fine-Tuning. *arXiv* **2023**, arXiv:2306.07967.
38. Sun, Y.; Wang, X.; Liu, Z.; Miller, J.; Efros, A.; Hardt, M. Test-Time Training with Self-Supervision for Generalization under Distribution Shifts. In Proceedings of the 37th International Conference on Machine Learning (ICML), Virtual, 13–18 July 2020; Volume 119, pp. 9229–9248.
39. Hardt, M.; Sun, Y. Test-Time Training on Nearest Neighbors for Large Language Models. In Proceedings of the Twelfth International Conference on Learning Representations (ICLR), Vienna, Austria, 7–11 May 2024.
40. Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. LoRA: Low-Rank Adaptation of Large Language Models. In Proceedings of the International Conference on Learning Representations (ICLR), Virtually, 25–29 April 2022.
41. Han, L.; Li, Y.; Zhang, H.; Milanfar, P.; Metaxas, D.; Yang, F. SVDiff: Compact Parameter Space for Diffusion Fine-Tuning. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 2–3 October 2023; pp. 7323–7334.
42. Zhang, Q.; Chen, M.; Bukharin, A.; He, P.; Cheng, Y.; Chen, W.; Zhao, T. Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. In Proceedings of the International Conference on Learning Representations (ICLR), Kigali, Rwanda, 1–5 May 2023.
43. He, H.; Ye, P.; Ren, Y.; Yuan, Y.; Zhou, L.; Ju, S.; Chen, L. GoRA: Gradient-Driven Adaptive Low Rank Adaptation. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), San Diego, CA, USA, 2–7 December 2025.
44. Paischer, F.; Hauzenberger, L.; Schmied, T.; Alkin, B.; Deisenroth, M.P.; Hochreiter, S. Parameter Efficient Fine-Tuning via Explained Variance Adaptation. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), San Diego, CA, USA, 2–7 December 2025.
45. Yuan, S.; Lin, R.; Feng, L.; Han, B.; Liu, T. Instance-Dependent Early Stopping. In Proceedings of the International Conference on Learning Representations (ICLR), Singapore, 24–28 April 2025.
46. Bakhtiarnia, A.; Zhang, Q.; Iosifidis, A. Efficient High-Resolution Deep Learning: A Survey. *ACM Comput. Surv.* **2024**, *56*, 1–35. [\[CrossRef\]](#)
47. Dede, A.; Nunoo-Mensah, H.; Tchao, E.T.; Agbemenu, A.S.; Adjei, P.E.; Acheampong, F.A.; Kponyo, J.J. Deep Learning for Efficient High-Resolution Image Processing: A Systematic Review. *Intell. Syst. Appl.* **2025**, *26*, 200505. [\[CrossRef\]](#)
48. Bégaïnt, J.; Racapé, F.; Feltman, S.; Pushparaja, A. CompressAI: A PyTorch Library and Evaluation Platform for End-to-End Compression Research. *arXiv* **2020**, arXiv:2011.03029.
49. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2961–2969.
50. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
51. Yang, L.; Kang, B.; Huang, Z.; Xu, X.; Feng, J.; Zhao, H. Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 16–22 June 2024; pp. 10371–10381.
52. Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In *Computer Vision—ECCV 2014*; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2014; Volume 8693, pp. 740–755.

53. Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Miami, FL, USA, 20–25 June 2009; pp. 248–255.
54. Geiger, A.; Lenz, P.; Urtasun, R. Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA, 16–21 June 2012; pp. 3354–3361.
55. Bjøntegaard, G. Calculation of Average PSNR Differences between RD-Curves. In Proceedings of the ITU-T SG16/Q6, 13th VCEG Meeting, Austin, TX, USA, 2–4 April 2001.
56. Liu, J.; Lu, G.; Hu, Z.; Xu, D. A Unified End-to-End Framework for Efficient Deep Image Compression. *arXiv* **2020**, arXiv:2002.03370.
57. Krizhevsky, A.; Hinton, G. *Learning Multiple Layers of Features from Tiny Images*; Technical Report; University of Toronto: Toronto, ON, Canada, 2009.
58. Cui, T.; Kumar, Y.; Marttinen, P.; Kaski, S. Deconfounded Representation Similarity for Comparison of Neural Networks. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 19138–19151. [[CrossRef](#)]
59. Li, X.; Geng, Q.; Huang, Q.; Li, X.; Tang, J.; Ye, Q. Spatial–Temporal Self-Compensating Graph Convolutional Network for Skeleton-Based Action Recognition Under Data Constraints. *IEEE Trans. Image Process.* **2026**, *35*, 5818–5833. [[CrossRef](#)] [[PubMed](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.