How2Compress: Scalable and Efficient Edge Video Analytics via Adaptive Granular Video Compression

Yuheng Wu
yuhengwu@kaist.ac.kr
KAIST
Daejeon, Republic of Korea

Thanh-Tung Nguyen
tungnt@kaist.ac.kr
KAIST
Daejeon, Republic of Korea

Lucas Liebe
lucasliebe@kaist.ac.kr
KAIST
Daejeon, Republic of Korea

Quang Tau
quangntau1223@kaist.ac.kr
KAIST
Daejeon, Republic of Korea

Pablo Espinosa Campos
pabloe@kaist.ac.kr
KAIST
Daejeon, Republic of Korea

Jinghan Cheng
chengjh@kaist.ac.kr
KAIST
Daejeon, Republic of Korea

Dongman Lee*
dlee@kaist.ac.kr
KAIST
Daejeon, Republic of Korea

ABSTRACT

With the rapid proliferation of the Internet of Things, video analytics has become a cornerstone application in wireless multimedia sensor networks. To support such applications under bandwidth constraints, learning-based adaptive quantization for video compression have demonstrated strong potential in reducing bitrate while maintaining analytical accuracy. However, existing frameworks often fail to fully exploit the fine-grained quality control enabled by modern blockbased video codecs, leaving significant compression efficiency untapped.

In this paper, we present How2Compress, a simple yet effective framework designed to enhance video compression efficiency through *precise, fine-grained* quality control at the macroblock level. How2Compress is a plug-and-play module and can be seamlessly integrated into any existing edge video analytics pipelines. We implement How2Compress on the H.264 codec and evaluate its performance across diverse real-world scenarios. Experimental results show that How2Compress achieves up to 50.4% bitrate savings and outperforms baselines by up to 3.01× without compromising accuracy, demonstrating its practical effectiveness and efficiency. *Code is available at link and a reproducible docker image at link.*

1 INTRODUCTION

Video analytics has emerged as a critical application in edge computing, enabling intelligent services such as traffic management [12, 21, 32, 42] and urban surveillance [1, 11, 41, 46, 62]. As illustrated in Fig. 1, typical edge video analytics systems compress video data at edge cameras and offload it to nearby edge clusters for inference [15, 33, 35, 36, 62, 69, 70]. These systems must operate under stringent bandwidth constraints and fluctuating scene complexities [12, 42, 62, 69], while simultaneously adhering to strict Service Level Objectives (SLOs) that demand both high inference accuracy and low end-to-end latency [4, 15, 33, 36, 42, 62, 69]. To meet these requirements, video compression becomes a critical enabler. To meet these requirements, video compression becomes a critical enabler. However, there is a key misalignment in current edge video analytics system. Video codec is primarily designed for human visual perception and often overlook semantically important features that are essential for machine-centric analytics tasks.

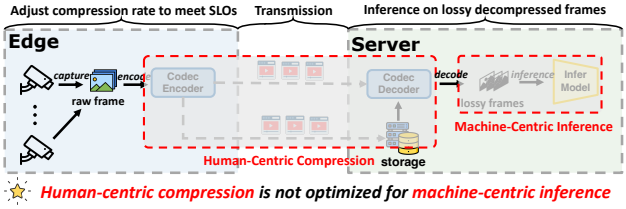


Figure 1: Machine-centric video compression is a key component to unleash wireless sensor network efficiency.

To address this gap, prior work has explored machine-centric adaptive quality assignment [15, 62, 67, 69], differing in spatial granularity. As shown in Fig. 2, Frame-level methods [62, 69] apply a uniform Quantization Parameter (QP) per frame, deciding **when to compress** based on global scene importance. In contrast, macroblock-level methods [15, 67] distinguish salient from non-salient regions, assigning binary (high/low) QPs per macroblock to decide **where to compress**. Modern codecs [18, 22, 44, 49, 59, 61], however, support multi-level QP control at the macroblock level, enabling finer-grained and more expressive quality modulation. Yet, the exponentially large decision space at this scale poses a significant challenge, limiting existing methods from fully leveraging the codec's configurability.

Goal and Insight. In this work, we aim to improve video compression through fine-grained macroblock-level quality assignment while preserving downstream analytical accuracy. Our key insight is that macroblocks contribute *unequally* to analytical performance and exhibit *content-dependent resilience* to compression. By exploring this variability, we can assign *just enough quality* to each macroblock (**how to compress**), minimizing bitrate without sacrificing accuracy. Specifically, in this paper, we first assign a low QP as a base and then selectively apply fine-grained *emphasis* (QP offset) to

*Corresponding Author

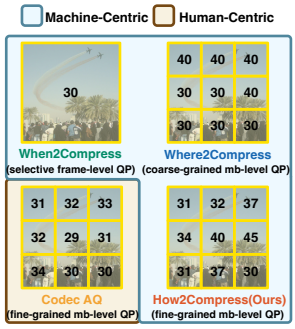


Table 1: Comparison between our proposed framework and prior works.

Category	Framework	Macroblock-level Assignment	Fine-grained Assigment	Low Training Overhead*	Low Inference Overhead*†	Machine-Centric
Uniform QP (When2Compress)	CASVA [69]	✗	✓	✗	✓	✓
	ILCAS [62]	✗	✓	✓	✗	✓
Coarse-grained (Where2Compress)	AccMPEG [15]	✓	✗	✗	✗	✓
Adaptive Quant (Codec AQ)	Blockbased Codec [2, 9, 18, 22, 49, 61] (H.264, H.265, H.266, VP9, AV1)	✓	✓	/	/	✗
Fine-grained (How2Compress)	How2Compress (Ours)	✓	✓	✓	✓	✓

* evaluated on 1080p resolution videos
† e2e overhead latency ≤ 33.3ms (30fps) on the edge-level device (*e.g.*, Nvidia Jetson Orin Nano)

Figure 2: Comparison between our proposed framework and prior works. How2Compress is a superset of both *When2Compress* and *Where2Compress*. Each macroblock is 16×16 pixels.

adjust macroblock-level quality. We refer to the optimal *emphasis* decision for each macroblock as *golden emphasis*.

Challenges. We identify two key and non-trivial challenges:

1) How can we determine a content adaptive but task agnostic signal? Each macroblock contributes differently to downstream task performance, depending on its content and context. This creates a local-to-global optimization problem, where per-macroblock quality decisions collectively influence frame-level detection accuracy. However, the relationship between local quality and global task utility is non-linear and content-dependent, making it infeasible to predetermine macroblock importance through deterministic heuristics [25, 67].

2) How can we efficiently navigate quality assignment in the exponential decision space? Encoding a 1080p frame involves ~ 8160 macroblocks (M), each with N (*e.g.*, 5) possible emphasis levels. This leads to an exponential decision space of size N^M. A natural formulation is to treat this as a Markov Decision Process (MDP) and use planning or RL to discover optimal emphasis assignments. However, this approach faces three major difficulties: 1) *Long-horizon decisions* [16, 38, 48], where a full sequence of 8160 decisions must be made before evaluating the outcome. 2) *Sparse reward signal* [43, 47, 64, 65], as performance feedback is only available after the entire assignment is completed. 3) *Expensive exploration*, exploring such a large space requires extensive trial-and-error, yet the encoding process is computationally expensive. The inefficiency and scalability bottlenecks of MDP-based approaches in our setting necessitate a different solution strategy.

Proposed Framework. In this paper, we propose How2Compress, a self-supervised video compression framework that performs fine-grained quality assignment at the macroblock level. Rather than formulating emphasis assignment as a sequential decision-making problem, How2Compress reframes it as a segmentation task, which enables parallel and scalable inference. The core component of How2Compress is the Region-aware Emphasis Routing (RER) module, which predicts the optimal quality emphasis for each macroblock without requiring manual labels or exhaustive preprofiling [67]. RER is co-trained with a *proxy emphasis target*, which serves as a dynamic pseudo-label that evolves based on downstream task feedback and spatiotemporal priors. This co-evolutionary training strategy allows the emphasis assignment model to efficiently explore the quality assignment space and converge toward high-utility compression strategies.

We integrate How2Compress as a plug-and-play module into the standard H.264 pipeline, which is the most widely adopted codec in practical edge deployments, and validate its performance across diverse real-world settings.

Contributions. Our main contributions are as follows:

1) We introduce the **Proxy Emphasis Target**, a dynamic supervisory signal that evolves during training and enables fine-grained quality assignment without relying on pre-profiling.

2) We design **Region-aware Emphasis Routing**, a self-supervised module that exploits spatial and temporal priors to efficiently assign compression quality across macroblocks in an exponential decision space.

3) We integrate How2Compress into the H.264 codec and demonstrate up to 50.4% bitrate reduction and a 3.01× improvement over state-of-the-art baselines, with negligible computational overhead on edge hardware.

2 BACKGROUND

How2Compress is designed to be codec-agnostic, provided the codec supports encoding pixel blocks at varying quality levels (*e.g.*, H.264 [61], H.265 (HEVC) [49], VP8 [44], VP9 [22], H.266 (VVC) [59] and AV1 [18]). In this paper, we integrate How2Compress with the H.264 codec and evaluate it on object detection, tracking and keypoint detection (pose estimation). Both are most common practice in edge computing. To motivate our design, we first analyze the compression mechanism of H.264, and then examine the limitations of prior approaches in enabling fine-grained quality control at the macroblock level.

2.1 Preliminaries on Video Compression

Video compression aims to reduce spatial and temporal redundancies while preserving visual quality. The encoding process of H.264 consists of three main stages. First, each frame is *typically* divided into 16×16 macroblocks. Each macroblock is predicted using either intra (within the current frame) or inter (motion-compensated from reference frames) modes. The difference between the predicted and actual macroblock forms the residual. Next, the residual undergoes a Discrete Cosine Transform (DCT), converting the data from the spatial domain to the frequency domain, concentrating the energy

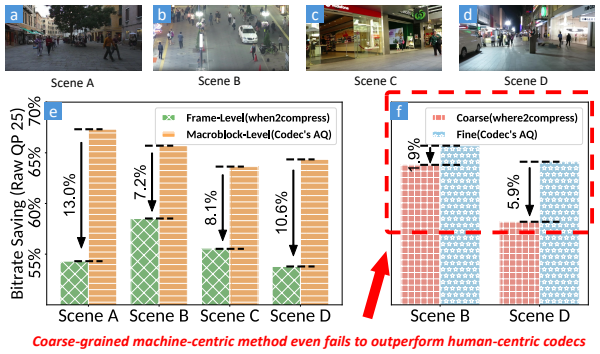


Figure 3: Samples from MOT dataset (a-d). Bitrate savings over raw video using human-centric fine-grained (e) and machine-centric coarse-grained (f) compression.

into fewer significant coefficients. Quantization is then applied to these coefficients, where they are divided by a quantization step, reducing precision. The degree of quantization is controlled by the QP, allowing a balance between compression efficiency and quality at the macroblock level. Finally, the quantized coefficients are further compressed through lossless entropy coding.

2.2 Related Work

Prior efforts to reduce bandwidth in video analytics span three main strategies: (1) *spatial pruning* (*e.g.*, cropping RoIs)[3, 35, 55, 66, 70, 72], (2) *temporal pruning* (*e.g.*, frame filtering or semantic skipping)[26, 34, 36], and (3) *tuning global quality knobs* (*e.g.*, framerate, resolution, CRF) [62, 69]. While effective, these approaches are often tightly coupled to specific downstream models or system pipelines, limiting their generalizability.

In contrast, our work focuses solely on fine-grained quality assignment which adjusts macroblock-level QP to reduce bitrate while remaining agnostic to downstream tasks and models. This strategy can be easily integrated with other system-level optimizations for additional gains.

#When2Compress: Frame-level quality assignment. Methods like CASVA [69] and ILCAS [62] dynamically adjust frame-level video settings (*e.g.*, resolution, framerate, QP) to balance latency and accuracy. However, they lack spatial granularity, often preserving high-quality macroblocks that are irrelevant to task performance, resulting in inefficient compression.

#Where2Compress: Region- or Macroblock-level quality assignment. The most closely related work to ours is research that has explored quality assignment at the macroblock level [15, 67]. Compared to frame-level approaches, this method offers significant advantages. Ideally, it allows for reducing the quality of macroblocks that are not directly associated with performance, while assigning necessary higher quality to those that influence the performance. Despite this potential, existing approaches fall short, failing to fully exploit the granular quality levels available at the macroblock level. For instance, AccMPEG [15] estimates the impact of each macroblock on accuracy and assigns quality based on its importance. But its reliance on an *accuracy gradient* limits the quality assignment to a coarse binary decision (*i.e.*, high (QP 30) or low (QP 40)), thereby underutilizing the codec's full range of quantization levels. AccelIR [67] targets on image restoration and conducts exhaustive pre-profiling of the benefit of each macroblock. While effective in restoration tasks, this approach is impractical for analytics tasks, where complex local-to-global dependencies cannot be captured by such pre-profiling.

3 MEASUREMENT STUDY

Experimental Setup. All measurement experiments are conducted on NVIDIA RTX 3090 GPU. For video encoding, we use FFMPEG with libx264 [17]. To explore macroblock-level fine-grained quality adjustments, we activate the AQ mode in the *variance* setting, which enables dynamic refinement of the QP at macroblock level based on content variance. Each video chunk has a framerate of 30 FPS and a GOP size of 30, consisting of 1 I-frame and up to 3 consecutive B-frames per group. Four representative video sequences from the MOT dataset [40] are selected as shown in Fig. 3(a-d) to represent a diverse range of conditions, including varying view angles (horizontal *vs.* vertical), object density (number of objects in a frame), luminance levels (daytime *vs.* nighttime), and camera dynamics (stationary *vs.* moving). To ensure a robust comparison, we first conduct a thorough search of the frame-level QP for each video sequence, identifying the highest QP value. Raw videos are encoded at QP 25 and all parameters are optimized to ensure that accuracy fluctuations remain within a margin of 2%.

Observation #1: Human-centric fine-grained AQ improves compression but struggles with content dynamics. As shown in Fig. 3e, our results indicate that H.264's built-in AQ enables more aggressive compression while maintaining comparable accuracy. However, its effectiveness diminishes in videos containing large, low-complexity flat regions (*e.g.*, Fig. 3b), where visual quality is more sensitive to degradation and thus requires additional bits to preserve perceptual fidelity. This limitation stems from AQ's original design [53], which prioritizes human visual perception rather than task-relevant machine perception. As a result, it may allocate bits inefficiently from an analytics perspective.

Observation #2: Potential of machine-centric, finer-grained macroblock-level quality assignment is promising but remains underexplored. As shown in Fig. 3f, machine-centric coarse-grained methods (*i.e.*, AccMPEG) even fail to outperform built-in codec's AQ. This inefficiency underscores an untapped opportunity for macroblock-level, task-aware quality assignment to enhance both compression efficiency and task accuracy.

Motivation. These observations reveal a critical gap: existing built-in AQ techniques are misaligned with the needs of video analytics, and coarse-grained approaches fail to leverage the codec's expressive capabilities. This motivates our design of a machine-centric, fine-grained macroblock-level quality assignment framework.

4 PROBLEM FORMULATION

Given an input image of resolution (H, W), we divide it into a grid of $n_w \times n_h$ non-overlapping macroblocks, where $n_w = \lceil \frac{H}{16} \rceil$ and $n_h = \lceil \frac{W}{16} \rceil$. Each macroblock covers a region of 16×16 pixels. Instead of predicting a unique QP for each macroblock, we employ a fixed base QP (*i.e.*, 45 in our setup) and assign an emphasis level

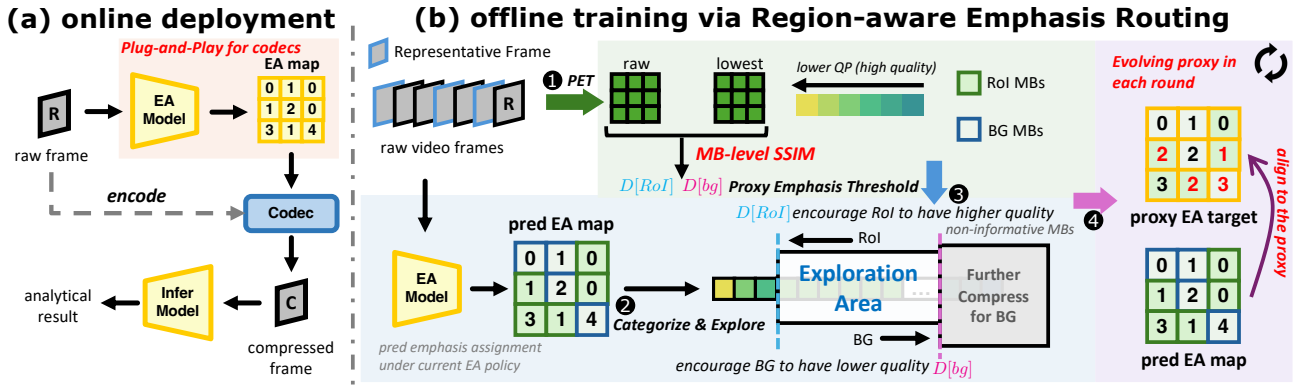


Figure 4: Overview of How2Compress.

$q_{i,j} \in \{0, 1, 2, 3, 4\}$ to each macroblock $M_{i,j}$[1]. This emphasis level serves as a proxy for quality refinement: a higher $q_{i,j}$ implies a more negative offset to the base QP, thus improving the quality of that macroblock.

We define the resulting emphasis configuration over the frame as an emphasis map $EM \in \mathbb{R}^{n_w \times n_h}$, where each entry $EM(i, j)$ denotes the emphasis level assigned to macroblock $M_{i,j}$. The optimization objective is to reduce the total emphasis cost (a surrogate for bitrate) while maintaining the application-level accuracy within a margin τ of the ground-truth performance under maximum quality. Formally, the objective is:

$$\max_{\mathbf{q}} \mathbb{E}[A(\mathbf{q})] \quad \text{s.t.} \quad \sum_{i=1}^{n_w} \sum_{j=1}^{n_h} EM(i, j) \text{ is minimized}, \quad \text{and} \quad |\mathbb{E}(A(\mathbf{q})) - \mathbb{E}(G)| \leq \tau, \tag{1}$$

Here, $A(\mathbf{q})$ denotes the application-level accuracy for frames decoded using the predicted emphasis map $\mathbf{q}$, while $\mathbf{E}(G)$ is the reference accuracy computed from frames encoded at the highest possible quality. The constraint ensures that the degradation in accuracy remains within an acceptable bound τ. Intuitively, the goal is to learn an emphasis assignment strategy that minimally allocates quality to accuracy-insensitive macroblocks, while preserving or boosting quality in regions critical to the application's performance.

5 METHODOLOGY

Overview. As shown in Fig. 4, How2Compress consists of two stages: (a) online deployment and (b) offline training via Region-aware Emphasis Routing. In the deployment stage (Fig. 4a), the *Emphasis Assignment (EA) Model* (§5.1) predicts macroblock-level emphasis in parallel, producing an *emphasis map*. This map guides QP refinement to control bitrate. The encoded video is then offloaded to a nearby edge server for inference. During training (Fig. 4b), we first sample representative frames and compress them at the lowest quality to estimate per-macroblock SSIM. Based on this, two *proxy emphasis thresholds* are derived to guide the exploration ❶. The EA map is then partitioned into RoI and Background regions ❷, enabling the *Region-aware Emphasis Routing (RER)* module (§5.2)

[1]This formulation is designed to be compatible with the *Emphasis Map* feature of NVIDIA's Video Codec SDK [8]. (More design rationale in Appendix A.2)

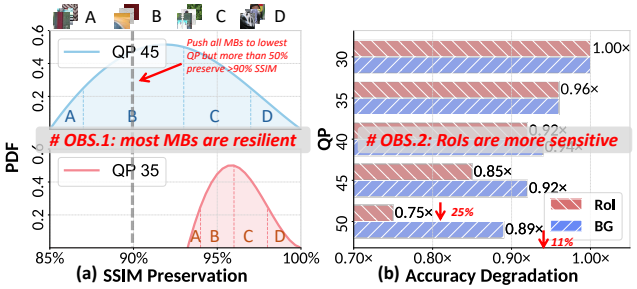


Figure 5: Design motivation: (a) Different macroblocks exhibit varying resilience to compression based on content characteristics. (b) Macroblocks in RoI are more sensitive.

to apply *region-aware exploration* ❸, generating a *proxy emphasis target* that evolves over iterations. This proxy target serves as adaptive supervision ❹, encouraging the model to assign lower quality to non-informative regions while preserving task-relevant details. Through iterative updates, the EA Model learns to make fine-grained, task-aware quality decisions.

5.1 Emphasis Assignment Model (EA Model)

As discussed in §1, similar to the approach in [15], the emphasis assignment problem can be cast as a segmentation task, where each 16×16 macroblock serves as an independent prediction unit. Solving this task requires two core capabilities: 1) the ability to capture both short-range and long-range semantic context [5, 50, 63], and 2) the computational efficiency necessary for real-time inference on edge devices (*e.g.*, sustaining about 30fps even for high-resolution inputs such as 1080p) [15, 67, 69]. Therefore, the backbone network must strike a balance between computational efficiency and performance.

To this end, How2Compress is designed to be backbone-agnostic as long as the model meets above requirements. Leveraging recent advancements in lightweight architectures [5, 6, 14, 37, 39, 52, 63, 71, 74], we adopt MobileViT v2 as our backbone due to two key advantages: 1) 16×16 pixel macroblocks aligns naturally with the patch-based design of vision transformers backbones, facilitating the modeling of relationships between these macroblocks and 2) its optimization tailored to mobile platforms. The Emphasis Assignment Model is trained offline on a centralized server using pre-encoded videos and distributed to edge devices for deployment.

Table 2: Notation used in Region-aware Emphasis Routing

Symbol	Description
I_r, I_c	Raw and compressed input frames
raw_mb, low_mb	Macroblocks from I_r and I_c
EM	Emphasis Assignment model
$em[i, j]$	Predicted emphasis level at macroblock (i, j)
$proxy\_em[i, j]$	Proxy target emphasis at macroblock (i, j)
p	Exploration probability
τ_{roi}, τ_{bg}	Thresholds for RoI and BG macroblocks
$\mathcal{E}(a, b)$	Exponential sampling between a and b
Acc_r, Acc_c	Accuracy on raw and compressed input
λ_1, λ_2	Weights for loss terms
$penalty$	Scaling factor for alignment loss

5.2 Region-aware Emphasis Routing (RER)

Region-aware Emphasis Routing (RER) is a simple yet effective mechanism for fine-grained, macroblock-level emphasis assignment in video compression. It is designed to tackle two key challenges outlined in §1: **C1**: how to determine a content-aware but task-agnostic *proxy emphasis target* for each macroblock, and **C2**: how to efficiently navigate the large decision space of possible emphasis configurations. To address these challenges, RER builds on the exploration-exploitation paradigm, with two core designs that accelerate and structure this process: (1) *Proxy Emphasis Threshold*, which narrows the exploration area by filtering out trivially saturated macroblocks. (2) *Region-aware Exploration*, which focuses exploration differently across foreground (RoI) and background regions to efficiently search the QP assignment space. To further improve training efficiency, we utilize hardware-optimized codec implementations during offline training. We ensure that the QP assignment behavior is aligned with standard codec semantics, enabling seamless deployment across heterogeneous edge devices.

5.2.1 *Proxy Emphasis Threshold (PET).*

Challenge #1: How to identify a content-aware yet task-agnostic supervisory signal? In real-world scenarios, switching between tasks or analytical models is common. To accommodate this, we seek a supervision strategy that remains broadly applicable across diverse tasks and models, without requiring task-specific annotations or retraining.

Observation #1: Structural and textural information is generally useful, and most macroblocks are resilient to compression. Prior studies [13, 19, 20, 27, 29, 45, 51, 68] emphasize that preserving fine-grained structural and textural details is essential for sustaining the performance of downstream tasks such as object detection. Motivated by this, our design aims to retain such cues during compression. To approximate their preservation, we adopt SSIM [58] as a proxy, recognizing that while it does not perfectly capture task relevance, it provides a reasonable content-aware signal. Importantly, the overall compression extent is ultimately guided by downstream task performance (see Appendix E). Furthermore, as shown in Fig. 5a, we observe that even when macroblocks are compressed at the lowest quality, a substantial portion of structural cues remains intact. This suggests that many macroblocks are inherently resilient to aggressive compression. This insight enables us to shrink the exploration space by focusing efforts on the smaller subset of MBs that are more sensitive to quality loss.

Algorithm 1: Proxy Emphasis Threshold

Input : I_r, I_c, raw_mb, low_mb, EM, em
Config: τ_{roi}, τ_{bg}

Function ProxyEmphasisThreshold(I_r, I_c):
  // SSIM threshold for RoI and BG
  D ← [ ];
  for $i = 0, 1, \ldots$ **do**
    for $j = 0, 1, \ldots$ **do**
      $D \leftarrow D \cup \{\text{SSIM}(raw\_mb[i, j], low\_mb[i, j])\}$;
  $D \leftarrow \text{sort}(D)$;
  return $D[\tau_{roi}], D[\tau_{bg}]$;

Design #1: Proxy Emphasis Threshold. To operationalize this insight, we introduce the *Proxy Emphasis Threshold* mechanism. As shown in Algorithm 1 and Fig. 4, we begin by compressing a representative raw frame to its lowest quality and compute SSIM scores for each macroblock to estimate its compression resilience ❶. These scores are sorted to define two percentile-based thresholds: one for regions of interest (RoI) and another for background (BG) regions. The thresholds determine which macroblocks are structurally important and guide the quality assignment during training, encouraging higher quality (lower QP) for RoI and more aggressive compression for resilient background blocks ❸ later.

This threshold-driven process provides a content-aware and task-agnostic supervisory signal, enabling adaptive bitrate allocation without exhaustive search. Moreover, to ensure computational efficiency, this profiling step is performed only once on a small set of representative frames. The resulting thresholds are then reused across the video stream, taking advantage of the spatial and temporal redundancy common in edge video streams.

5.2.2 *Region-aware Dual Exploration.*

Challenge #2: How to efficiently explore and exploit the macroblock decision space? Not all macroblocks contribute equally to accuracy. Efficient exploration requires a strategy that adapts emphasis decisions based on the task relevance of each macroblock, ensuring computational effort is directed where it matters most.

Observation #2: RoI MBs are generally more critical than those in Background. Empirical analysis (Fig. 5b) shows that compression artifacts in RoI regions significantly degrade accuracy, while aggressive compression in BG regions has less effect. Prioritizing RoI macroblocks during exploration can potentially improve efficiency and accuracy retention.

Design #2: Region-aware Dual Exploration Strategy. To accelerate macroblock-level search, we adopt a region-aware dual exploration strategy that treats RoI and BG macroblocks differently. RoI regions, typically more critical for analysis, are guided toward higher quality via upward sampling, while BG regions are steered toward more aggressive compression ❷.

This region-aware design serves only as a heuristic to improve exploration efficiency. It remains broadly applicable across diverse tasks, as RoI-like regions (*e.g.*, objects, humans) tend to be structurally important regardless of the specific objective (*e.g.*, detection,

Algorithm 2: Region-aware Emphasis Routing

Input : I_r, I_c, raw_mb, low_mb, EM, em
Config: p, τ_{roi}, τ_{bg}, λ_1, λ_2, $penalty$

Function `RegionAwareEmphasisRouting`(I_r, I_c)**:**
  `// efficiently explore decision space`
  $\tau_{roi}, \tau_{bg} \leftarrow$ `ProxyEmphasisThreshold`(I_r, I_c);
  repeat
    $em[i,j] \leftarrow EM(I_r)$;
    `// Dual-Exploration for RoI mb`
    for $mb \in RoI$ **do**
      if $p' \sim \mathcal{U} < p \wedge em[i,j] \leq \tau_{roi}$ **then**
        $proxy\_em[i,j] \leftarrow \mathcal{E}\,((em[i,j], high)$;
    `// Dual-Exploration for BG mb`
    for $mb \in BG$ **do**
      if $p' \sim \mathcal{U} < p \wedge em[i,j] \geq \tau_{bg}$ **then**
        $proxy\_em[i,j] \leftarrow \mathcal{E}\,(low, em[i,j])$;
      else
        $proxy\_em[i,j] \leftarrow proxy\_em[i,j] - 1$;
    $loss1 \leftarrow \mid Acc_c - Acc_r \mid$;
    $loss2 \leftarrow CE\,(em, proxy\_em) \cdot penalty$;
    $loss \leftarrow \lambda_1 \cdot loss1 + \lambda_2 \cdot loss2$;
    Update EM;
    decay p;
  until *accuracy requirement not met*;

tracking, pose estimation). Even in the absence of ground-truth boxes, pseudo-RoIs can be extracted using any pretrained object detectors, making the approach practical and generalizable. The final emphasis decisions are not hardcoded by region type but are supervised by the downstream task performance. As shown in Algorithm 2, the EA model iteratively refines its emphasis map based on proxy thresholds and region-specific sampling, while a combined loss ensures the learned policy preserves task accuracy ❹. We refer to Appendix B for the theoretical justification of RER.

6 EVALUATION

We evaluate How2Compress with a comprehensive real-world evaluation across 1) diverse edge devices, 2) varying content dynamics, 3) all baseline categories and 4) thorough ablations, revealing the following key findings:

1) How2Compress is compression-efficient. It achieves up to **50.4%** bitrate reduction and up to **3.01**× improvement over baselines without compromising accuracy by preserving task-relevant structure and discarding redundant detail (§6.2.1).

2) How2Compress is lightweight and scalable. Its pipelined design interleaves QP assignment and encoding, keeping latency under 6 ms per 1080p frame and enabling real-time performance on resource-constrained devices (§6.2.2).

3) How2Compress is robust and generalizable. Region-aware Emphasis Routing adapts across backbones, downstream models, and hyperparameters, enabling stable convergence and effective navigation of the fine-grained emphasis decision space (§6.3).

6.1 Experimental Setup

Task and Datasets. We evaluate analytical accuracy across three representative tasks: object detection, multi-object tracking, and keypoint detection. We mainly report YOLOv8-X [30] for detection and tracking, and YOLOv8-Pose for keypoint estimation. Experiments are conducted on the MOT17 [40], NVIDIA AI City [57], and VisDrone [73] datasets.

Implementation. We adopt two H.264 codec implementations: Nvidia Video SDK [8] and FFMPEG with libx264 [61]. Nvidia Video SDK support Advanced Emphasis Map feature [10], which provides five emphasis levels to adjust quality at the macroblock level. From our empirical observations, setting a base QP of 45 and a minimum QP of 30 aligns Nvidia's five emphasis levels approximately to libx264 QP values of $[45, 43, 37, 34, 30]$. We adopt this empirical alignment to unify the behavior of both implementations. Note that How2Compress is a plug-and-play module that is agnostic to specific codec implementations. We utilize Nvidia Video SDK mainly for accelerated video encoding during training, while deploying libx264 for evaluation to ensure robust and practical performance across diverse edge devices. Additional details are in Appendix A.2.

Evaluation Metrics. We evaluate performance across four dimensions: 1) computational cost, measured in MACs per pixel [7], 2) latency overhead, 3) bitrate savings and 4) training efficiency. Accuracy is measured by standard task-specific metrics: mAP for object detection, MOTA/MOTP/IDF1 for tracking, and OKS, PCK@0.2, and PCK@0.5 for keypoint detection. In scenarios where ground-truth annotations are unavailable, we treat results obtained on the raw (uncompressed) video as the reference baseline.

Training Details. Refer to Appendix A.6.

Baselines. We benchmark How2Compress with three categories of methods. 1) *When2Compress*: We adapt all methods to a frame-level QP assignment approach, while other video configurations (*e.g.*, resolution, framerate) remain orthogonal and can serve as complementary optimizations. 2) *Where2Compress*: AccMPEG is the state-of-the-art method which directly assign binary QP configurations to distinguished (non)-informative regions. 3) *H.264 Adaptive Quantization mode*: A human-centric fine-grained QP assignment mechanism. In our experiments, for When2Compress, we exhaustively search for the minimum frame-level QP that satisfies the accuracy requirement. For Where2Compress, we use the SOTA AccMPEG [15]. For H.264 AQ mode, we adopt the *variance* mode, which dynamically adjusts the QP based on the variance of macroblocks to improve SSIM [53]. For all other advanced codecs (*i.e.*, H.265, VP9, H.266 and AV1), we use their most practical configurations as recommended for real-world deployment (Appendix C).

6.2 Evaluation Results

6.2.1 Better Bitrate Savings across Diverse Scenes. We evaluate How2Compress across a variety of video scenes to validate its robustness and show its consistent compression efficiency.

Bitrate Savings. As shown in Table 3, How2Compress achieves a bitrate reduction of up to 50.4%, outperforming the baseline methods by as much as 3.01×. Figure 6a further shows that, when paired with vanilla H.264, How2Compress matches H.265 (HEVC) [49] and surpasses advanced codecs like VP9 [22], H.266 (VVC) [59], and AV1 [18]. We integrate How2Compress with H.264 due to its widespread adoption in edge video analytics scenarios. Since it only

Table 3: Bitrate (Mbps) comparison over MOT, Nvidia AI City, and VisDrone datasets. The accuracy drop remains within 2% for detection and tracking tasks, and within 5% for keypoint detection. Bold/Red highlights the best compression performance. Values indicate the second-best compression.

Method	Multiple Object Tracking [40]						Nvidia AI City [57]			VisDrone [73]						
	1702	1704	1709	1710	1711	1713	S01	S02	S03	086	117	137	182	268	305	339
Frame-Level (baseline)																
Uniform QP [62, 69] (When2Compress)	4.076	1.646	3.986	5.027	5.117	3.711	3.323	4.313	7.439	4.27	3.59	5.53	8.52	9.08	3.81	2.52
Macroblock-Level																
AccMPEG* [15] (Where2Compress)	3.354 (17.7%↓)	1.569 (4.7%↓)	3.622 (9.1%↓)	4.518 (10.1%↓)	4.469 (12.7%↓)	3.300 (11.1%↓)	2.707 (18.5%↓)	3.446 (20.1%↓)	6.457 (13.2%↓)	3.61 (15.5%↓)	3.02 (15.9%↓)	4.66 (15.7%↓)	7.13 (16.3%↓)	7.21 (20.6%↓)	3.31 (13.1%↓)	2.14 (15.1%↓)
Adaptive Quantization* [9, 53] (Codec AQ)	2.914 (28.5%↓)	**1.362** (17.3%↓)	3.258 (18.3%↓)	3.878 (22.9%↓)	4.101 (19.9%↓)	3.144 (15.3%↓)	2.971 (10.6%↓)	3.912 (9.3%↓)	6.662 (10.4%↓)	2.91 (31.9%↓)	2.92 (18.7%↓)	3.72 (32.7%↓)	5.88 (31.0%↓)	**4.64** (48.9%↓)	3.09 (18.9%↓)	2.06 (18.3%↓)
How2Compress (Machine-centric fine-grained)	**2.023** (50.4%↓)	1.388 (15.7%↓)	**2.943** (26.2%↓)	**3.469** (31.0%↓)	**3.488** (31.8%↓)	**2.759** (25.7%↓)	**2.202** (33.7%↓)	**3.182** (26.2%↓)	**4.475** (39.8%↓)	**2.53** (40.7%↓)	**2.30** (35.9%↓)	**3.39** (38.7%↓)	**4.38** (48.6%↓)	5.47 (39.8%↓)	**2.25** (40.9%↓)	**1.60** (36.5%↓)
δ bitrate saving over second-best	1.77×	0.91×	1.43×	1.35×	1.60×	1.68×	1.82×	1.30×	3.01×	1.28×	1.91×	1.18×	1.57×	0.81×	2.16×	1.99×

Table 4: Post-compression SSIM comparisons of MOT dataset.

Method	1702	1704	1709	1710	1711	1713
H.264 AQ [17]	0.981	0.983	0.971	0.984	0.982	0.985
AccMPEG [15]	0.945	0.949	0.900	0.948	0.930	0.930
How2Compress	**0.907**	**0.948**	**0.871**	**0.939**	**0.919**	**0.928**

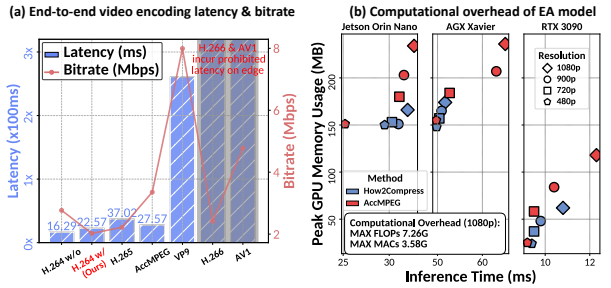
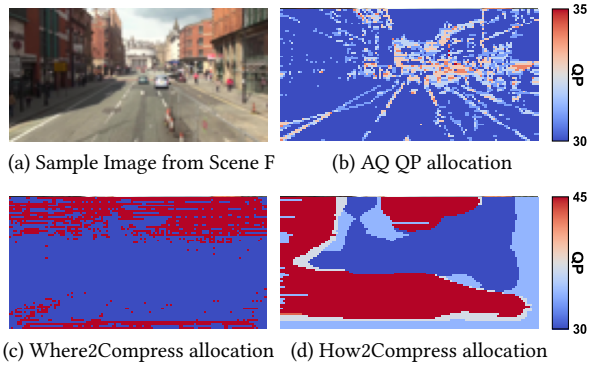


Figure 6: (a) E2E video encoding latency and bitrate comparison of advanced codecs. (b) Computational resource and latency overhead of EA model.



(a) Sample Image from Scene F (b) AQ QP allocation

(c) Where2Compress allocation (d) How2Compress allocation

Figure 7: QP allocation heatmaps for different methods across macroblocks. (zoom in for better visualization.)

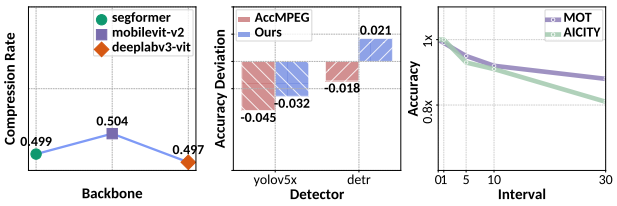


Figure 8: Ablation study on (a) Different Emphasis Assignment Model backbones. (b) Different detection backends. (c) Different execution intervals.

refines codec's QP assignment, integrating it with more advanced codecs is expected to yield even greater compression gains.

Structural and Textural Information Preservation. The primary objective of How2Compress is to eliminate redundant structural and textural information that does not affect the performance of the detection task. As presented in Table 4, How2Compress yields a lower SSIM compared to AccMPEG [15] and the codec's AQ while achieving comparable accuracy. This result indicates that How2Compress more effectively retains the essential structural features required by the downstream model while discarding non-essential details, thereby optimizing bit allocation for accuracy-critical macroblocks.

6.2.2 Overhead. We evaluate it across heterogeneous devices to assess its scalability and suitability for real-world deployment.

Computational Latency Overhead. In standard video analytics pipelines, cameras typically operate at 30 frames per second (fps), resulting in an inter-frame interval of approximately 33.3 milliseconds. How2Compress is explicitly implemented to interleave emphasis assignment within this interval. This interleaved architecture enables emphasis assignment decision-making and encoding to be performed in parallel with frame acquisition, thereby amortizing additional latency overhead. As shown in Fig. 6, integrating with H.264, our framework incurs no more than 6 milliseconds of latency per frame in average. This is lower than that of advanced codecs while achieving superior bitrate savings.

Computational Resource Overhead. As shown in Fig. 6b, our How2Compress requires only up to 7.26 GFLOPS (Floating Point Operations per Second) and 3.58 GMACs (Multiply-Add Operations

Table 5: Effect of exploration probability decay and RER mechanism. gray indicates accuracy deviation ≥ 10%.

Method	Perf*	Probability Decay 0.1	0.15	0.2	0.25	0.3
How2Compress (w/ RER)	Comp†	1×	0.98×	0.99×	0.96×	0.97×
	Time‡	1×	1×	1.3×	1.6×	1.6×
How2Compress (w/o RER)	Comp†	0.83×	0.87×	0.81×	0.83×	0.88×
	Time‡	2×	2×	2×	2×	2×

* Perf: performance metric category.
† Comp: bitrate after compression (normalized).
‡ Time: # epochs to convergence (normalized).

per Second) when processing 1080p video input. The per-pixel computational cost is limited to 1726 MACs, which is significantly below the hard constraint of 2000 MACs per pixel for mobile platforms [7].

6.2.3 Emphasis Assignment In-depth Analysis. In this case study, we compare the quality assignment strategies of different methods and present qualitative results that highlight the advantages of How2Compress. As shown in Fig. 7,we visualize the QP allocation on a sampled I-frame. Figure 7b shows the QP allocation of the standard H.264 codec using the AQ variance mode [53], which employs a conservative adjustment strategy designed to preserve human visual perception. Figure 7c illustrates the allocation pattern of *Where2Compress*, which primarily lowers the quality in non-object regions. In contrast, Fig. 7d demonstrates that How2Compress not only identifies *where* to compress but also determines *how much* compression each macroblock can tolerate. This fine-grained and adaptive strategy leads to higher compression efficiency without sacrificing detection accuracy. Additional analysis of Table 3 and Table 4, along with qualitative results, is provided in Appendix F & G.3.

6.3 Ablation Study

We validate the effectiveness of our key contribution, RER, by ablating How2Compress under four settings: 1) different Emphasis Assignment Model backbones, 2) detection backends, 3) execution intervals, and 4) with *vs.* without RER.

Backbone-agnostic. As shown in Fig. 8a, How2Compress consistently captures macroblock importance across different backbones, validating that RER generalizes well and is agnostic to the choice of Emphasis Assignment Model architecture.

Backend-agnostic. We pretrain the Emphasis Assignment Model and evaluate the accuracy on different object detectors (*i.e.*, DETR [74] and YOLOv5 [31]) without retraining EA model. As illustrated in Fig. 8b, How2Compress maintains high detection accuracy across backends, suggesting the generalizability of using the *proxy emphasis target* as a proxy backend-agnostic supervision signal.

Adaptive Temporal Execution. Video streams often exhibit strong temporal coherence, especially in slow-motion scenarios. We leverage this property by selectively recomputing the emphasis assignment at adaptive intervals. This strategy aligns with the principle behind *When2Compress* and can be viewed as a complementary mechanism. As shown in Fig. 8c, in slow-motion scenarios such as those in MOT datasets, increasing the execution interval (≤ 5) does not significantly degrade accuracy.

Effect of RER. Table 5 shows that removing RER and only use *proxy emphasis target* leads to a significant drop in accuracy (≥ 10%), despite achieving higher compression via fine-grained QP assignment. This is because without RER, the model fails to emphasize critical regions thereby treating all macroblocks equally. This results in the loss of informative regional features. Moreover, RER proves robust to changes in the exploration probability decay schedule, indicating that it is stable and hyperparameter-insensitive.

7 DISCUSSION

Generalization to Other Tasks. Although the EA model is pretrained on detection, it generalizes well to tracking (within 2% accuracy drop) and keypoint detection (within 5%), due to the shared focus on foreground regions. Quantitative results are provided in Appendix G.1.

Generalization to Other Blockbased Codecs. A wide range of widely adopted video codecs, including H.264 [61], H.265 [49], VP8 [44], VP9 [22], H.266 [2] and AV1 [18], rely on blockbased compression algorithms. How2Compress is inherently compatible with these codecs, as it does not alter the fundamental encoding process but instead refines the quality assignment strategy. We provide the implementation examples in Appendix H.

8 CONCLUSION

We present How2Compress, a novel plug-and-play module that enhances video compression for analytics through fine-grained, macroblock-level quality assignment. It dynamically allocates the minimum necessary quality to each macroblock via a progressive, self-guided *Region-aware Emphasis Routing* mechanism. Integrated with the H.264 codec, How2Compress achieves up to 50.4% bitrate savings and outperforms baselines by up to 3.01×, all while preserving task performance. Extensive evaluations prove that How2Compress is lightweight, backbone-agnostic, backend-agnostic, and training efficient, making it both practical and scalable for real world deployment in edge video analytics systems.

ACKNOWLEDGMENTS

This work is supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP), funded by the Ministry of Science and ICT (MSIT) of the Republic of Korea, under grants No. RS-2019-II191126 and No. RS-2024-00459749.

REFERENCES

[1] Ganesh Ananthanarayanan, Paramvir Bahl, Peter Bodík, Krishna Chintalapudi, Matthai Philipose, Lenin Ravindranath, and Sudipta Sinha. 2017. Real-Time Video Analytics: The Killer App for Edge Computing. *Computer* 50, 10 (2017), 58–67.

[2] Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J. Sullivan, and Jens-Rainer Ohm. 2021. Overview of the Versatile Video Coding (VVC) Standard and its Applications. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 10 (2021), 3736–3764. https://doi.org/10.1109/TCSVT.2021.3101953

[3] Shubham Chaudhary, Aryan Taneja, Anjali Singh, Purbasha Roy, Sohum Sikdar, Mukulika Maity, and Arani Bhattacharya. 2024. TileClipper: Lightweight Selection of Regions of Interest from Videos for Traffic Surveillance. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*. USENIX Association, Santa Clara, CA, 967–984. https://www.usenix.org/conference/atc24/presentation/chaudhary

[4] Bo Chen, Zhisheng Yan, and Klara Nahrstedt. 2022. Context-aware image compression optimization for visual analytics offloading. In *Proceedings of the 13th ACM Multimedia Systems Conference*. 27–38.

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. 2017. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence* 40, 4 (2017), 834–848.

[6] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. 2017. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587* (2017).

[7] Marcos V Conde, Eduard Zamfir, Radu Timofte, Daniel Motilla, Cen Liu, Zexin Zhang, Yunbo Peng, Yue Lin, Jiaming Guo, Xueyi Zou, et al. 2023. Efficient deep models for real-time 4k image super-resolution. NTIRE 2023 benchmark and report. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 1495–1521.

[8] NVIDIA Corporation. 2021. NVIDIA Video Codec SDK 11.0.1. https://developer.nvidia.com/nvidia-video-codec-sdk.

[9] NVIDIA Corporation. 2021. NVIDIA Video Codec SDK 11.0.1 - Adaptive Quantization. https://docs.nvidia.com/video-technologies/video-codec-sdk/11.0/nvenc-video-encoder-api-prog-guide/index.html.

[10] NVIDIA Corporation. 2024. Nvidia Video SDK - Emphasis Map Feature. https://docs.nvidia.com/video-technologies/video-codec-sdk/11.1/nvenc-video-encoder-api-prog-guide/index.html. Available at https://docs.nvidia.com/video-technologies/video-codec-sdk/11.1/nvenc-video-encoder-api-prog-guide/index.html.

[11] Sokèmi René Emmanuel Datondji, Yohan Dupuis, Peggy Subirats, and Pascal Vasseur. 2016. A Survey of Vision-Based Traffic Monitoring of Road Intersections. *IEEE Transactions on Intelligent Transportation Systems* 17, 10 (2016), 2681–2698.

[12] Sina G. Davani and Nabil J. Sarhan. 2021. Experimental Analysis of Optimal Bandwidth Allocation in Computer Vision Systems. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 10 (2021), 4121–4130.

[13] Philip De Rijk, Lukas Schneider, Marius Cordts, and Dariu Gavrila. 2022. Structural knowledge distillation for object detection. *Advances in Neural Information Processing Systems* 35 (2022), 3858–3870.

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *ICLR* (2021).

[15] Kuntai Du, Qizheng Zhang, Anton Arapin, Haodong Wang, Zhengxu Xia, and Junchen Jiang. 2022. AccMPEG: Optimizing Video Encoding for Accurate Video Analytics. In *Proceedings of Machine Learning and Systems*, D. Marculescu, Y. Chi, and C. Wu (Eds.), Vol. 4. 450–466.

[16] Aleksandra Faust, Kenneth Oslund, Oscar Ramirez, Anthony Francis, Lydia Tapia, Marek Fiser, and James Davidson. 2018. Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning. In *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 5113–5120.

[17] FFmpeg. 2024. FFmpeg. https://ffmpeg.org/. Available at https://ffmpeg.org/.

[18] Alliance for Open Media. 2025. AV1 Codec Library. https://aomedia.googlesource.com/aom/ Accessed: 2025-04-01.

[19] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. 2018. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International conference on learning representations*.

[20] Robert Geirhos, Pascal Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. 2019. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations (ICLR)*.

[21] Mahshid Ghasemi, Sofia Kleisarchaki, Thomas Calmant, Jiawei Lu, Shivam Ojha, Zoran Kostic, Levent Gürgen, Gil Zussman, and Javad Ghaderi. 2023. Real-time Multi-Camera Analytics for Traffic Information Extraction and Visualization. In *2023 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. 288–290. https://doi.org/10.1109/PerComWorkshops56833.2023.10150224

[22] Robert Glover, Debargha Mukherjee, John Bankoski, Peter Wilkins, Yaowu Xu, Jani Han, Rajat Joshi, Pascal de Rivaz, and Bill Rose. 2013. VP9 – An Open Video Codec for Next-Generation Web Video. https://www.webmproject.org/vp9/. *Google White Paper* (2013). Accessed: 2025-03-18.

[23] Junpeng Guo, Shengqing Xia, and Chunyi Peng. 2022. VPPlus: Exploring the Potentials of Video Processing for Live Video Analytics at the Edge. In *2022 IEEE/ACM 30th International Symposium on Quality of Service (IWQoS)*. 1–11. https://doi.org/10.1109/IWQoS54832.2022.9812896

[24] Alain Horé and Djemel Ziou. 2010. Image Quality Metrics: PSNR vs. SSIM. In *2010 20th International Conference on Pattern Recognition*. 2366–2369.

[25] Danlan Huang, Feifei Gao, Xiaoming Tao, Qiyuan Du, and Jianhua Lu. 2023. Toward Semantic Communications: Deep Learning-Based Image Semantic Coding. *IEEE Journal on Selected Areas in Communications* 41, 1 (2023), 55–71.

[26] Jinwoo Hwang, Minsu Kim, Daeun Kim, Seungho Nam, Yoonsung Kim, Dohee Kim, Hardik Sharma, and Jongse Park. 2022. {CoVA}: Exploiting {Compressed-Domain} analysis to accelerate video analytics. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*. 707–722.

[27] Md Amirul Islam, Matthew Kowal, Patrick Esser, Sen Jia, Bjorn Ommer, Konstantinos G Derpanis, and Neil Bruce. 2021. Shape or texture: Understanding discriminative features in CNNs. In *International Conference on Learning Representations*.

[28] Anil K. Jain. 1989. *Fundamentals of Digital Image Processing*. Prentice-Hall.

[29] Deyi Ji, Haoran Wang, Mingyuan Tao, Jianqiang Huang, Xian-Sheng Hua, and Hongtao Lu. 2022. Structural and statistical texture knowledge distillation for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16876–16885.

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

[31] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, ChristopherSTAN, Liu Changyu, Laughing, tkianai, Adam Hogan, lorenzomammana, yxNONG, AlexWang1900, Laurentiu Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, Francisco Ingham, Frederik, Guilhen, Hatovix, Jake Poznanski, Jiacong Fang, Lijun Yu, changyu98, Mingyu Wang, Naman Gupta, Osama Akhtar, PetrDvoracek, and Prashant Rai. 2020. *ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements*. https://doi.org/10.5281/zenodo.4154370

[32] Kwihoon Kim and Yong-Geun Hong. 2019. Autonomous network traffic control system based on intelligent edge computing. In *2019 21st International Conference on Advanced Communication Technology (ICACT)*. 164–167.

[33] Seyeon Kim, Kyungmin Bin, Donggyu Yang, Sangtae Ha, Song Chong, and Kyunghan Lee. 2023. ENTRO: Tackling the Encoding and Networking Trade-off in Offloaded Video Analytics. In *Proceedings of the 31st ACM International Conference on Multimedia* (Ottawa ON, Canada) *(MM '23)*. Association for Computing Machinery, New York, NY, USA, 9115–9123.

[34] Yuxin Kong, Peng Yang, and Yan Cheng. 2023. Edge-Assisted On-Device Model Update for Video Analytics in Adverse Environments. In *Proceedings of the 31st ACM International Conference on Multimedia* (Ottawa ON, Canada) *(MM '23)*. Association for Computing Machinery, New York, NY, USA, 9051–9060. https://doi.org/10.1145/3581783.3612585

[35] Jingzong Li, Libin Liu, Hong Xu, Shudeng Wu, and Chun Jason Xue. 2023. Cross-Camera Inference on the Constrained Edge. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*. 1–10. https://doi.org/10.1109/INFOCOM53939.2023.10229045

[36] Yuanqi Li, Arthi Padmanabhan, Pengzhan Zhao, Yufei Wang, Guoqing Harry Xu, and Ravi Netravali. 2020. Reducto: On-Camera Filtering for Resource-Efficient Real-Time Video Analytics. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '20)*. Association for Computing Machinery, 359–376.

[37] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. 2022. A ConvNet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 11976–11986.

[38] Chris Lu, Yannick Schroecker, Albert Gu, Emilio Parisotto, Jakob Foerster, Satinder Singh, and Feryal Behbahani. 2024. Structured state space models for in-context reinforcement learning. *Advances in Neural Information Processing Systems* 36 (2024).

[39] Sachin Mehta and Mohammad Rastegari. 2022. Separable self-attention for mobile vision transformers. *arXiv preprint arXiv:2206.02680* (2022).

[40] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. 2016. MOT16: A Benchmark for Multi-Object Tracking. (March 2016). http://arxiv.org/abs/1603.00831

[41] Thanh-Tung Nguyen, Si Young Jang, Boyan Kostadinov, and Dongman Lee. 2023. PreActo: Efficient Cross-Camera Object Tracking System in Video Analytics Edge Computing. In *2023 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. 101–110. https://doi.org/10.1109/PERCOM56429.2023.10099298 ISSN: 2474-249X.

[42] Thanh-Tung Nguyen, Lucas Liebe, Nhat-Quang Tau, Yuheng Wu, Jinghan Cheng, and Dongman Lee. 2025. OctopInf: Workload-Aware Inference Serving for Edge Video Analytics. In *2025 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. 128–137. https://doi.org/10.1109/PerCom64205.2025.00032 ISSN: 2474-249X.

[43] Alexander Ororbia and Ankur Mali. 2023. Active predictive coding: Brain-inspired reinforcement learning for sparse reward robotic control problems. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 3015–3021.

[44] WebM Project. 2010. VP8 Data Format and Decoding Guide. https://www.webmproject.org/vp8/. Accessed: 2025-03-18.

[45] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. 2021. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10213–10224.

[46] Charushila Raskar and Shikha Nema. 2018. Smart Traffic monitoring system. In *2018 Second International Conference on Green Computing and Internet of Things (ICGCIoT)*. 120–123.

[47] Desik Rengarajan, Gargi Vaidya, Akshay Sarvesh, Dileep Kalathil, and Srinivas Shakkottai. 2022. Reinforcement learning with sparse rewards using guidance

from offline demonstration. *arXiv preprint arXiv:2202.04628* (2022).

[48] Jimmy Smith, Shalini De Mello, Jan Kautz, Scott Linderman, and Wonmin Byeon. 2024. Convolutional state space models for long-range spatiotemporal modeling. *Advances in Neural Information Processing Systems* 36 (2024).

[49] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. 2012. Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE Transactions on Circuits and Systems for Video Technology* 22, 12 (2012), 1649–1668.

[50] Shiyu Tang, Ting Sun, Juncai Peng, Guowei Chen, Yuying Hao, Manhui Lin, Zhihong Xiao, Jiangbin You, and Yi Liu. 2023. PP-MobileSeg: Explore the Fast and Accurate Semantic Segmentation Model on Mobile Devices. arXiv:2304.05152 [cs.CV] https://arxiv.org/abs/2304.05152

[51] Anju Jose Tom and Sudhish N. George. 2020. Simultaneous Reconstruction and Moving Object Detection From Compressive Sampled Surveillance Videos. *IEEE Transactions on Image Processing* 29 (2020), 7590–7602. https://doi.org/10.1109/TIP.2020.3004696

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

[53] VideoLAN. 2008. X264. https://code.videolan.org/videolan/x264/-/commit/b59440f09b7eb7e6f30c1131d56843ee92e3751d.

[54] Marko Viitanen, Joose Sainio, Alexandre Mercat, Ari Lemmetti, and Jarno Vanne. 2022. From HEVC to VVC: the First Development Steps of a Practical Intra Video Encoder. *IEEE Transactions on Consumer Electronics* (2022). https://doi.org/10.1109/TCE.2022.3146016

[55] Hanling Wang, Qing Li, Heyang Sun, Zuozhou Chen, Yingqian Hao, Junkun Peng, Zhenhui Yuan, Junsheng Fu, and Yong Jiang. 2023. VaBUS: Edge-Cloud Real-Time Video Analytics via Background Understanding and Subtraction. *IEEE Journal on Selected Areas in Communications* 41, 1 (2023), 90–106.

[56] Junjue Wang, Ziqiang Feng, Zhuo Chen, Shilpa George, Mihir Bala, Padmanabhan Pillai, Shao-Wen Yang, and Mahadev Satyanarayanan. 2018. Bandwidth-Efficient Live Video Analytics for Drones Via Edge Computing. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. 159–173. https://doi.org/10.1109/SEC.2018.00019

[57] Shuo Wang, David C. Anastasiu, Zheng Tang, Ming-Ching Chang, Yue Yao, Liang Zheng, Mohammed Shaiqur Rahman, Meenakshi S. Arya, Anuj Sharma, Pranamesh Chakraborty, Sanjita Prajapati, Quan Kong, Norimasa Kobori, Munkhjargal Gochoo, Munkh-Erdene Otgonbold, Ganzorig Batnasan, Fady Alnajjar, Ping-Yang Chen, Jun-Wei Hsieh, Xunlei Wu, Sameer Satish Pusegaonkar, Yizhou Wang, Sujit Biswas, and Rama Chellappa. 2024. The 8th AI City Challenge. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[58] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* 13, 4 (2004), 600–612.

[59] Adam Wieckowski, Jens Brandenburg, Tobias Hinz, Christian Bartnik, Valeri George, Gabriel Hege, Christian Helmrich, Anastasia Henkel, Christian Lehmann, Christian Stoffers, Ivan Zupancic, Benjamin Bross, and Detlev Marpe. [n. d.]. VVenC: An Open And Optimized VVC Encoder Implementation. In *Proc. IEEE International Conference on Multimedia Expo Workshops (ICMEW)* (2021). 1–2. https://doi.org/10.1109/ICMEW53276.2021.9455944

[60] Adam Wieckowski, Jens Brandenburg, Tobias Hinz, Christian Bartnik, Valeri George, Gabriel Hege, Christian Helmrich, Anastasia Henkel, Christian Lehmann, Christian Stoffers, Ivan Zupancic, Benjamin Bross, and Detlev Marpe. 2021. Proc. IEEE International Conference on Multimedia Expo Workshops (ICMEW). github.com/fraunhoferhhi/vvenc/source/Lib/EncoderLib/BitAllocation.cpp. , 2 pages. https://doi.org/10.1109/ICMEW53276.2021.9455944

[61] Thomas Wiegand, Gary J. Sullivan, Gisle Bjøntegaard, and Ajay Luthra. 2003. Overview of the H.264/AVC Video Coding Standard. *IEEE Transactions on Circuits and Systems for Video Technology* 13, 7 (2003), 560–576. https://doi.org/10.1109/TCSVT.2003.815165

[62] Duo Wu, Dayou Zhang, Miao Zhang, Ruoyu Zhang, Fangxin Wang, and Shuguang Cui. 2024. ILCAS: Imitation Learning-Based Configuration- Adaptive Streaming for Live Video Analytics With Cross-Camera Collaboration. *IEEE Transactions on Mobile Computing* 23, 6 (2024), 6743–6757.

[63] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. 2021. SegFormer: Simple and efficient design for semantic segmentation with transformers. *Advances in neural information processing systems* 34 (2021), 12077–12090.

[64] Kai Yan, Alex Schwing, and Yu-Xiong Wang. 2022. CEIP: combining explicit and implicit priors for reinforcement learning with demonstrations. *Advances in Neural Information Processing Systems* 35 (2022), 7614–7627.

[65] Hanlin Yang, Chao Yu, Siji Chen, et al. 2024. Hybrid policy optimization from Imperfect demonstrations. *Advances in Neural Information Processing Systems* 36 (2024).

[66] Kichang Yang, Juheon Yi, Kyungjin Lee, and Youngki Lee. 2022. FlexPatch: Fast and Accurate Object Detection for On-device High-Resolution Live Video Analytics. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*. 1898–1907. https://doi.org/10.1109/INFOCOM48880.2022.9796984

[67] Juncheol Ye, Hyunho Yeo, Jinwoo Park, and Dongsu Han. 2023. AccelIR: Task-Aware Image Compression for Accelerating Neural Restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 18216–18226.

[68] Lu Zhang and Laurens van der Maaten. 2013. Structure preserving object tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1838–1845.

[69] Miao Zhang, Fangxin Wang, and Jiangchuan Liu. 2022. CASVA: Configuration-Adaptive Streaming for Live Video Analytics. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*. 2168–2177.

[70] Wuyang Zhang, Zhezhi He, Luyang Liu, Zhenhua Jia, Yunxin Liu, Marco Gruteser, Dipankar Raychaudhuri, and Yanyong Zhang. 2021. Elf: accelerate high-resolution mobile deep vision with content-aware parallel offloading. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*. 201–214.

[71] Wenqiang Zhang, Zilong Huang, Guozhong Luo, Tao Chen, Xinggang Wang, Wenyu Liu, Gang Yu, and Chunhua Shen. 2022. Topformer: Token pyramid transformer for mobile semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12083–12093.

[72] Tianxiong Zhong, Zhiwei Zhang, Guo Lu, Ye Yuan, Yu-Ping Wang, and Guoren Wang. 2023. TVM: A Tile-based Video Management Framework. *Proc. VLDB Endow.* 17, 4 (Dec. 2023), 671–684. https://doi.org/10.14778/3636218.3636224

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

[74] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. 2020. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159* (2020).

How2Compress: Efficient Edge Video Analytics via Adaptive Granular Video Compression

Supplementary Material

A ARTIFACTS

To ensure full reproducibility of our experiments, we provide both the source code and the complete software environment. The source code is available at https://github.com/wyhallenwu/how2compress. Additionally, we provide a Docker image, available at dockerhub https://hub.docker.com/r/wuyuheng/how2compress, which contains all necessary dependencies, and scripts for reproducing our results.

A.1 Hardware

How2Compress is trained on a compute cluster and evaluated on real-world deployable edge devices.

The compute cluster used for training is configured as follows:

- **CPU:** Dual Intel(R) Xeon(R) Silver 4210R @ 2.40GHz
- **GPU:** 4× NVIDIA RTX 3090 (only 3 GPUs are used due to limitations of the Nvidia Video Codec)
- **Memory:** 192 GB RAM
- **Cuda:** CUDA 12.6 and Driver 560.28.03

For edge deployment evaluation, we use the NVIDIA Jetson Orin Nano 4GB. Detailed specifications for this device can be found at: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/.

A.2 Software

The implementation of H.264 used in our work is a customized version of the libx264 encoder that supports macroblock-level Quantization Parameter (QP) assignment. This customized encoder is built upon the repository available at https://github.com/Alex-q-z/myh264. We use FFMPEG [17] version 3.4.8 and libx264 version 0.163.x (paired with libswscale version 4.8.100).

To accelerate training, we utilize the NVIDIA Video Codec SDK v11.0.10, which supports GPU-accelerated video encoding and includes an *Emphasis Map* feature. Detailed documentation is available at: https://docs.nvidia.com/video-technologies/video-codec-sdk/11.1/nvenc-video-encoder-api-prog-guide/index.html#emphasis-map. Although the SDK provides five emphasis levels, their behavior differs slightly from that of libx264. Through extensive empirical testing, we determine that, when using a base QP of 45 and a minimum QP of 30, these emphasis levels approximately correspond to effective QP values of $[45, 43, 37, 34, 30]$.

The compiled binary tools used in our experiments are included in the repository. For completeness, we also evaluate the encoder without the use of the *Emphasis Map* feature.

A.3 Datasets

How2Compress is evaluated on a large-scale collection of diverse, real-world edge video stream datasets.

Multiple Object Tracking Benchmark. The Multiple Object Tracking 2017 (MOT17) dataset is a widely used benchmark in the computer vision community, particularly for evaluating multi-object tracking algorithms. It consists of seven distinct video sequences recorded in both indoor and outdoor public environments, with a primary focus on pedestrian tracking. Each sequence is divided into training and testing subsets, enabling comprehensive algorithm development and evaluation under diverse real-world conditions. The dataset poses various challenges such as varying crowd densities, illumination changes, and camera motion. In total, MOT17 comprises 15,948 frames, corresponding to approximately 645 seconds (or 10.75 minutes) of video data. Details are available at https://motchallenge.net/data/MOT17Det/.

NVIDIA AI City Challenge. We primarily utilize the dataset from the 2021 edition of the NVIDIA AI City Challenge, a large-scale urban traffic video benchmark. This dataset comprises approximately 9 hours of video footage captured from 20 distinct urban locations, including single-direction intersection approaches, full intersections, highway segments, and city streets. The dataset reflects a broad range of environmental conditions such as different times of day (*e.g.*, dawn). Details are available at https://www.aicitychallenge.org/2021-data-and-evaluation/.

VisDrone. While the previous two datasets utilize static cameras, the VisDrone dataset [73] features cameras mounted on drones, thereby capturing more dynamic scenes and diverse visual content. This enables evaluation under more realistic and challenging conditions involving camera motion. For further details, please refer to the official repository at https://github.com/VisDrone/VisDrone-Dataset.

A.4 Downstream Models

We employ three representative object detectors in our evaluation: YOLOv5 [31], YOLOv8 [30], and DETR [74]. These models span both one-stage convolutional architectures and transformer-based paradigms, offering a comprehensive basis for assessing the generalization of our framework across diverse detector designs.

Specifically, we fine-tune YOLOv5 and YOLOv8 on the two datasets introduced above using high-quality video frames (encoded with a Quantization Parameter of 25) to establish strong baseline detection performance. During the training and evaluation of our compression framework, these fine-tuned detectors are kept fixed (frozen), while input frames are re-encoded at lower quality levels using various baseline and proposed methods. Our objective is to maximize compression efficiency while minimizing any degradation in detection accuracy with respect to these fixed detectors.

For pose estimation tasks, we employ various versions of YOLO-Pose provided by Ultralytics, and for tracking, we utilize their corresponding tracking models. We primarily focus on detection and

tracking tasks, as they represent the most widely adopted applications in real-world scenarios.

Encoding Configurations

```
# uniform QP (When2Compress)
ffmpeg -i <input> -c:v libx264 -qp <qp> -x264-params
↪  aq-mode=0 <output>
# AccMPEG (Where2Compress)
# qp is placeholder
# and it will load QP assignment from
↪  <qp_matrix_file>
/myh264/ffmpeg -y -i <input> -qp <useless
↪  placeholder> -pix_fmt yuv420p <output>
# Codec AQ (Codec's How2Compress)
ffmpeg -i <input> -c:v libx264 -qp <qp> -x264-params
↪  aq-mode=<aq-mode> <output>
# How2Compress (Ours)
# load QP assignment from <qp_matrix_file>
/myh264/ffmpeg -y -i <input> -qp <useless
↪  placeholder> -pix_fmt yuv420p <output>
# x265
ffmpeg -benchmark -framerate 30 -i <input> -c:v
↪  libx265 -tune zerolatency -preset veryfast
↪  -x265-params "pools=1:qp=<qp>:aq-mode=<mode>"
↪  -threads 1 <output>
# VP9
ffmpeg -benchmark -framerate 30 -i <input> -c:v
↪  libvpx-vp9 -crf <config> -deadline realtime
↪  -threads 1 <output>
# VVC encoding
ffmpeg -benchmark -framerate 30 -i <input> -c:v
↪  libvvenc -preset 0 -qp <qp>  -pix_fmt yuv420p
↪  -threads 1 <output>
# AV1 encoding
ffmpeg -benchmark -framerate 30 -i <input> -c:v
↪  libaom-av1 -crf <config> -b:v 0 -cpu-used 4
↪  -usage realtime -row-mt 1 -threads 1 -g 30
↪  -pix_fmt yuv420p <output>
```

A.5 Emphasis Assignment Model Implementation

It is important to note that How2Compress is not tied to a specific backbone architecture. Our core contribution lies in the proposed **Region-aware Emphasis Routing** mechanism, which efficiently guides fine-grained macroblock-level QP assignment. The main results reported in the paper are based on the MobileViTv2 backbone, and we conduct ablation studies with alternative backbones to demonstrate the robustness and generality of our approach.

For the Emphasis Assignment model, we implement a prediction head on top of MobileViTv2, SegFormer, or DeepLabV3 backbones. The head consists of a single Conv2D layer that produces a probability distribution of shape $(B, H, W, 5)$, where (H, W) corresponds to the number of macroblocks, and the last dimension represents the five discrete emphasis levels.

To accommodate the constraints of various edge devices, the input resolution can be downsampled by a factor of D ($[1, 4]$) along both spatial dimensions. If $D > 1$, the resulting low-resolution emphasis map is then upscaled to the original resolution using bilinear interpolation to match the macroblock grid.

A.6 Training Details

The training procedure, detailed in Algorithm 2, employs the following default hyperparameters: the exploration probability p starts at 0.8 and decays by 0.1 per epoch; SSIM threshold percentiles are set at 90% for τ_{roi} and 50% for τ_{bg}. The model is optimized using AdamW with a CosineAnnealing learning rate schedule, decaying from 1×10^{-3} to 1×10^{-6}. The loss function is balanced with weights $\lambda_1 = 10$ and $\lambda_2 = 5$, and emphasis levels are penalized progressively with factors [1, 1.3, 1.6, 1.9, 2.2] to discourage excessive quality allocation.

For training and evaluation, the dataset is split into training, validation, and test sets using a 70:20:10 ratio. To avoid temporal dependencies, each frame is encoded as an I-frame with its corresponding predicted emphasis map during training, ensuring the model focuses on spatial frame-level decisions. In deployment, the video stream follows a GOP structure of 30 frames, comprising 1 I-frame and up to 3 B-frames between P-frames.

B PROOF

We provide a *soft theoretical bound for convergence and compression efficiency*. Let $\mathcal{E}^{(t)} \in \mathbb{Z}_+^{n_w \times n_h}$ denote the predicted emphasis map at iteration t, where each entry $\mathcal{E}_{i,j}^{(t)} \in \{0, 1, \ldots, K-1\}$ corresponds to the QP offset (*i.e.*, emphasis level) assigned to macroblock $M_{i,j}$. Let $\mathcal{P}^{(t)}$ denote the proxy emphasis target at iteration t, obtained through percentile-based PET thresholds and Region-aware Emphasis Routing (RER). The training loss is defined as:

$$\mathcal{L}^{(t)} = \lambda_1 \cdot \text{AccLoss}^{(t)} + \lambda_2 \cdot \text{AlignLoss}^{(t)},$$

where $\text{AccLoss}^{(t)} := \left|\text{Acc}(\mathcal{E}^{(t)}) - \text{Acc}(\mathcal{E}^{\max})\right|$ quantifies the task performance deviation from maximum quality, and $\text{AlignLoss}^{(t)} := \text{CE}(\mathcal{E}^{(t)}, \mathcal{P}^{(t)})$ measures the divergence between the model output and the proxy.

Assumptions.

(1) *Proxy Improvement:* The proxy $\mathcal{P}^{(t)}$ improves over time, *i.e.*, $\text{CE}(\mathcal{P}^{(t)}, \mathcal{E}^*) \to 0$ as $t \to T$, where $\mathcal{E}^*$ is the (unknown) optimal emphasis map.

(2) *Temporal Smoothness:* For adjacent frames, macroblock assignments evolve slowly: $\|\mathcal{E}^{(t)} - \mathcal{E}^{(t-1)}\|_1 \le \delta$.

(3) *Loss Contraction:* The model learns to align with the proxy at each round:

$$\mathcal{L}^{(t+1)} \le \mathcal{L}^{(t)} - \eta \cdot \|\mathcal{E}^{(t+1)} - \mathcal{P}^{(t)}\|_1,$$

for some $\eta > 0$.

Proposition. Under the assumptions above, for any accuracy tolerance $\tau > 0$, the number of rounds T needed to obtain an emphasis map $\mathcal{E}^{(T)}$ satisfying

$$\left|\text{Acc}(\mathcal{E}^{(T)}) - \text{Acc}(\mathcal{E}^{\max})\right| \le \tau \quad \text{and} \quad \mathbb{E}[R(\mathcal{E}^{(T)})] \ge R_{\min}(\tau),$$

is bounded by

$$T \le \frac{\mathcal{L}^{(0)}}{\eta \cdot \epsilon},$$

where ϵ denotes the minimal improvement in alignment per round, and $R(\cdot)$ is the achieved bitrate reduction.

Interpretation. This bound shows that convergence is linear in the initial loss and inversely proportional to the progress made in each round. In practice, due to temporal consistency and the design of RER, we empirically observe convergence in only 1–3 rounds.

C COMPARISON WITH ADVANCED CODECS

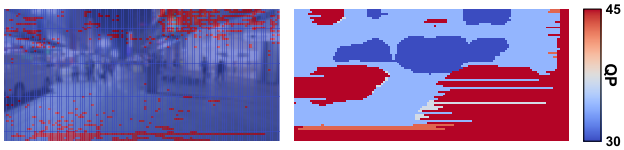
How2Compress is designed to be codec-agnostic, assuming the underlying codec supports block-level quality control *i.e.*, the ability to encode pixel blocks at varying quality levels. This includes widely adopted codecs such as H.264 [61], H.265 [49], VP9 [22], H.266 (VVC) [2] and AV1 [18]. In this work, we primarily use H.264 as a case study, as it remains the most prevalent video codec in edge computing scenarios due to its widespread hardware support and deployment maturity. To ensure experimental reproducibility, the encoding configurations used for each codec are provided in Box A.4. All codecs were evaluated using the FFMPEG toolchain [17], with default parameters unless otherwise specified. It is important to note that we deliberately disable multi-threaded encoding during our comparisons. This decision reflects a realistic edge deployment scenario, where devices typically feature limited CPU cores (2–6 threads) and are concurrently burdened with computationally intensive tasks, such as real-time video analytics or other deep learning inference workloads.

D CONSTANT RATE FACTOR (CRF)

Some research [23, 56] leverage CRF in modern video encoders to enable implicit macroblock-level QP adjustment. It operates by targeting a perceptually constant visual quality across frames and spatial regions. Internally, it modulates the QP at the macroblock level based on content characteristics such as texture complexity, luminance masking, and motion. Regions deemed perceptually less sensitive (*e.g.*, flat or low-motion areas) are assigned higher QPs, while visually salient areas are preserved with lower QPs. This implicit adjustment is guided by heuristics rooted in human visual system modeling.

Despite its adaptive nature, CRF is not considered suitable for edge video analytics scenarios. First, CRF is inherently optimized for human visual perception rather than machine-centric tasks such as object detection or activity recognition. Consequently, it may prioritize quality in visually salient but analytically irrelevant regions, while under-allocating bitrate to semantically important areas (*e.g.*, small moving objects). Second, the mechanism is opaque and nondeterministic from external control. It does not expose a clear interface for frame- or region-level quality manipulation, making it ill-suited for latency-sensitive edge systems that require direct and responsive QP control in reaction to scene dynamics or detection confidence.

In contrast, offering explicit macroblock-level QP assignment facilitates fine-grained and machine-centric quality adaptation. It allows encoders to prioritize bitrate allocation to analytically critical regions—identified using objectness maps, motion vectors, or model-informed saliency while compressing less relevant background areas more aggressively. This capability is particularly beneficial in edge-cloud collaborative systems, where uplink bandwidth



(a) AccMPEG QP allocation (Sence D)

(b) Ours QP allocation (Scene D)

Figure 9: Comparison of QP allocation between AccMPEG and our method. AccMPEG requires careful online threshold tuning to prevent over-allocation of high-quality regions in object-sparse scenes.

is limited and quality-budget precision is essential to maintain detection performance across a broad range of video scenes. Moreover, such direct control improves system responsiveness, enabling real-time adaptation to content changes and ensuring robust operation under stringent resource constraints.

E PSNR *vs.* SSIM

Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) are two widely adopted metrics for evaluating image and video quality. While both aim to quantify the degradation introduced by lossy compression, they differ fundamentally in their design principles and perceptual alignment.

PSNR is a pixel-wise fidelity metric that computes the logarithmic ratio between the maximum possible pixel value and the mean squared error (MSE) between the original and compressed images [24, 28]. It assumes an independent and identically distributed error model across pixels, making it analytically convenient and computationally efficient. However, PSNR is agnostic to spatial correlations and human perceptual sensitivity. As a result, it often fails to capture structural distortions or texture degradation that are critical to downstream tasks.

In contrast, SSIM evaluates image similarity by modeling luminance, contrast, and structural information in localized regions [58]. It correlates more strongly with human visual perception by emphasizing structural consistency rather than absolute pixel accuracy. Despite being originally designed for perceptual quality assessment, SSIM better captures the degradation patterns relevant to DNN-based video analytics, particularly in the presence of structural artifacts and compression-induced distortions.

It is important to note that neither PSNR nor SSIM is designed to measure the specific feature retention of DNNs used in video analytics. DNNs rely not on perceptual quality *per se*, but on the preservation of task-relevant features. In this regard, both PSNR and SSIM may misrepresent the true impact of compression on detection accuracy. PSNR can over-penalize visually insignificant pixel differences, while SSIM may overemphasize structures that are perceptually relevant but semantically irrelevant to the analytical models.

Nevertheless, among the two, SSIM remains more adoptable in analytics-aware compression systems. Its emphasis on structural preservation provides a better proxy for semantic fidelity, especially in scenarios where object shapes and boundaries are critical. Empirically, in our experiments, SSIM tends to correlate more strongly

than PSNR with downstream performance metrics such as mean Average Precision (mAP) in object detection tasks. Furthermore, SSIM's localized and differentiable structure [24] makes it useful for guiding encoder decisions or supervising quality estimation models when model-in-the-loop feedback is computationally expensive or unavailable.

F FURTHER IN-DEPTH ANALYSIS

F.1 Disentangling optimization of streaming and inference

As discussed in §2.2, beyond controlling video quality, there exist several alternative strategies to reduce video bitrate, including: 1) filtering redundant frames (temporal redundancy), 2) reducing video resolution (spatial downsampling), and 3) cropping regions of interest (RoI) for targeted inference.

Differentiate from them, How2Compress focuses exclusively on adjusting quality within each frame, *i.e.*, exploring the quality dimension at a fine-grained level. This design choice offers three primary advantages:

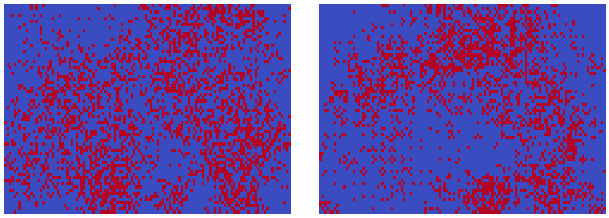
- ✓ **Modularity and decoupled optimization:** How2Compress is a plug-and-play module that decouples the optimization of the frontend (edge device) from the backend (edge cluster). In comparison, strategies such as 2) resolution reduction and 3) RoI cropping often generate video frames of varying resolutions. To maintain high throughput in downstream inference (*e.g.*, with batch processing) such variability necessitates joint optimization between edge devices and analytical backends. In contrast, How2Compress preserves consistent frame structure, simplifying deployment and is compatible with backend inference optimization.
- ✓ **Complementarity:** How2Compress is orthogonal to the above methods. It can be seamlessly integrated with 1), 2), and 3) to further enhance bitrate reduction, offering additive benefits when combined with existing system optimization.
- ✓ **Effectiveness in densely populated scenes:** In scenarios where frames contain numerous objects, spatial pruning methods (*e.g.*, resolution reduction or RoI cropping) often become less effective, as most regions are deemed important. In such cases, How2Compress remains effective by adaptively reducing information along the quality dimension, selectively lowering the fidelity of less critical regions within crowded frames.

F.2 Comparison with AccMPEG and Its Potential Extensions

Our proposed framework, *How2Compress*, is fundamentally distinct from AccMPEG in both methodological philosophy and implementation strategy. Below, we outline two core differences that enable *How2Compress* to achieve better scalability, generalization, and runtime efficiency compared to AccMPEG.

(1) **Model/Task Independence.** AccMPEG requires extensive offline gradient profiling for each specific model and task to estimate the relative importance of individual macroblocks (MBs) for downstream performance. This process involves backpropagating gradients from the downstream task (*e.g.*, object detection or tracking) through the video processing pipeline, which is computationally expensive and must be repeated for every new model, dataset, or task configuration. Consequently, AccMPEG's applicability is limited to fixed, pre-profiled scenarios.
In contrast, our framework eliminates the need for any downstream task gradients by leveraging *structural cues* (*e.g.*, edges, textures, and spatial layouts) that are inherently present in video content. These cues serve as a general and task-agnostic signal to guide quality allocation decisions. As a result, *How2Compress* is naturally **model- and task-independent**, allowing it to generalize seamlessly across various analytical models and tasks without re-profiling. This advantage is especially valuable in real-world edge deployment scenarios where the underlying models may change over time or differ across devices.

(2) **Global Correlation Modeling.** Another key distinction lies in how inter-macroblock relationships are handled. AccMPEG uses *AccGrad*, a heuristic metric that evaluates each macroblock's contribution to performance in isolation, based solely on the magnitude of its associated gradient. This per-MB analysis neglects the complex correlations and contextual dependencies between macroblocks, such as object continuity, motion coherence, or scene layout consistency.
By contrast, our framework adopts a learning-based strategy that models **global correlations across macroblocks**. Rather than assigning quality in a greedy or local manner, *How2Compress* employs a self-supervised, attention-guided mechanism that captures long-range dependencies. This enables the system to allocate bitrate in a globally optimized fashion, taking into account the overall impact on scene understanding and task accuracy.



YOLO DETR

Figure 10: Macroblocks identified as important by AccMPEG using accuracy gradients from different detection models. The variation indicates model-specific dependence, necessitating re-profiling for each detector.

On Potential Extensions to AccMPEG. In an attempt to reduce AccMPEG's complexity, one may consider the following two extensions that discretize gradients into QP bins:

- **(Ext@1): Fixed Scalar Thresholds.** One straightforward extension involves defining a set of scalar thresholds to partition the gradient values into discrete QP bins. For instance, using 5 equally spaced thresholds across the observed gradient range $[\min, \max]$, one could assign coarser or finer compression quality per macroblock accordingly. However,

Table 6: Benchmark Results Across Different Hardware Platforms (Mean Time in ms)

	RTX3090			Orin Nano1			Xavier NX1			AGX Xavier1		
	NumPy	Numba	PyTorch	NumPy	Numba	PyTorch	NumPy	Numba	PyTorch	NumPy	Numba	PyTorch
480 × 640 (480p)	18.06	24.72	3.47	24.96	16.41	5.61	37.13	25.98	7.96	26.10	15.31	15.79
720 × 1280 (720p)	54.76	55.61	8.81	70.77	41.89	13.60	110.04	65.76	14.21	72.81	45.39	10.64
900 × 1600 (900p)	89.05	59.77	6.83	115.76	70.86	19.74	161.78	68.37	19.68	102.76	49.20	15.14
1080 × 1920 (1080p)	129.45	73.55	7.13	167.92	103.00	27.43	230.56	99.13	19.22	148.38	63.38	22.00

Table 7: Benchmark Results Across Different Hardware Platforms (Standard Deviation in ms)

	RTX3090			Orin Nano1			Xavier NX1			AGX Xavier1		
	NumPy	Numba	PyTorch	NumPy	Numba	PyTorch	NumPy	Numba	PyTorch	NumPy	Numba	PyTorch
480 × 640 (480p)	4.42	8.98	1.23	0.10	0.21	0.12	1.06	1.90	1.14	3.14	1.71	0.78
720 × 1280 (720p)	1.92	5.59	1.38	0.09	0.16	0.18	2.46	14.72	0.23	2.05	12.83	0.49
900 × 1600 (900p)	1.20	6.18	2.14	0.08	0.42	0.08	1.72	1.30	5.33	2.36	4.16	1.13
1080 × 1920 (1080p)	2.94	3.91	0.54	0.14	0.21	0.07	1.16	1.09	0.39	2.36	0.84	1.76

Table 8: Normalized Performance Comparison (Lower is Better)

	RTX3090			Orin Nano1			Xavier NX1			AGX Xavier1		
	NumPy	Numba	PyTorch	NumPy	Numba	PyTorch	NumPy	Numba	PyTorch	NumPy	Numba	PyTorch
480 × 640 (480p)	5.2×	7.1×	**1.0×**	4.4×	2.9×	**1.0×**	4.7×	3.3×	**1.0×**	1.7×	**1.0×**	1.0×
720 × 1280 (720p)	6.2×	6.3×	**1.0×**	5.2×	3.1×	**1.0×**	7.7×	4.6×	**1.0×**	6.8×	4.3×	**1.0×**
900 × 1600 (900p)	13.0×	8.8×	**1.0×**	5.9×	3.6×	**1.0×**	8.2×	3.5×	**1.0×**	6.8×	3.2×	**1.0×**
1080 × 1920 (1080p)	18.2×	10.3×	**1.0×**	6.1×	3.8×	**1.0×**	12.0×	5.2×	**1.0×**	6.7×	2.9×	**1.0×**

such an approach suffers from severe practical limitations: scalar thresholds must be hand-tuned for each scene, model, and task. Without careful tuning, the bin boundaries may poorly reflect the actual semantic importance of content regions.

To empirically validate this limitation, we implemented a variant of AccMPEG using 5-bin discretization over the per-scene gradient range. The resulting bitrate performance (measured in Megabits) is reported below for the MOT17 and AI CITY datasets:

Dataset	Bitrate (Mbits)
MOT17	[2.732, 1.441, 3.312, 4.132, 3.818, 2.918]
AI CITY	[2.544, 3.289, 5.452]

These results demonstrate clear performance degradation when compared to our method, which uses a learned, soft-assignment strategy without requiring manual threshold tuning.

- **(Ext@2): Quantile-Based Thresholds.** An alternative approach replaces fixed scalar thresholds with quantiles derived from the empirical distribution of gradients. This method computes percentiles (*e.g.*, 20th, 40th, 60th, and 80th) to dynamically form QP bins, adapting to the actual gradient distribution shape rather than relying on arbitrary fixed boundaries. While this strategy offers improved theoretical robustness by naturally accounting for gradient distribution characteristics, it introduces prohibitive computational overhead that severely limits practical deployment. Our empirical evaluation on the Jetson Orin Nano platform reveals substantial runtime penalties for quantile computation per frame. As shown in our benchmark results (Table 6), quantile computation requires 27.43 ms when implemented using PyTorch on GPU and 103.00 ms using CPU with Numba acceleration. These latencies represent significant bottlenecks that violate real-time processing constraints essential for edge video analytics pipelines. For context, at 30 FPS video processing, the total frame budget is only 33.33 ms, making the quantile computation overhead alone consume 82% of the available processing time on GPU or exceed the frame budget by 3× on CPU. Furthermore, the computational complexity scales poorly with gradient tensor size, creating additional challenges for higher resolution inputs. The overhead becomes even more pronounced when considering that quantile computation must be performed for every frame in the video stream, leading to cumulative performance degradation that renders this approach impractical for real-time edge deployment scenarios.

In summary, both proposed extensions to AccMPEG fail to meet practical requirements: (Ext@1) lacks generality due to the need for manual tuning, while (Ext@2) is too computationally expensive for real-time applications. By contrast, *How2Compress* **circumvents both limitations** by learning macroblock-level emphasis directly

from structural content, without relying on downstream gradients or expensive runtime operations. This enables fast, generalizable, and efficient video compression that is well-suited for modern edge intelligence systems.

F.3 Advantages over Codec's Adaptive Quantization

An interesting observation, which may initially appear paradoxical, arises from Table 3 and Fig. 7: AccMPEG consistently outperforms the codec's AQ across all scenarios of the NVIDIA AI City dataset, yet underperforms compared to AQ on the MOT17 dataset. In the QP assignment heatmap, the QP range produced by the codec's AQ ([30, 35]) is noticeably narrower than that of AccMPEG ([30, 45]). Nonetheless, codec AQ achieves superior compression efficiency in MOT17. This might cause by two reasons: (1) differences in object density across the datasets, and (2) the codec's AQ's more effective exploitation of skip-mode macroblocks.

The NVIDIA AI City dataset generally exhibits low object density, with most frames containing fewer than five detectable objects and many frames containing none. In such cases, AccMPEG (*Where2Compress*) can efficiently assign lower quality to the majority of macroblocks, focusing quality on a sparse set of informative regions, which leads to greater compression efficiency than the codec's AQ. In contrast, the MOT17 dataset features a significantly higher object density, with frequent occurrences of densely packed targets within a frame. This necessitates a larger number of high-quality macroblocks to maintain detection accuracy. While AccMPEG does allocate high quality to critical regions, the codec's AQ often designates even more macroblocks with higher quality. However, when AQ is enabled in the H.264 codec, it adjusts the QP by increasing it in flat or low-activity regions. This raises the likelihood that these regions will closely match their motion predictions, resulting in negligible residuals. Consequently, the encoder can exploit skip mode, which omits residual coding altogether and significantly reduces the number of bits required. This leads to increased sparsity, which in turn enhances the efficiency of entropy coding, thereby lowering the bitrate even if the QP is relatively low.

Our proposed method, How2Compress, consistently outperforms both AccMPEG and the codec's AQ across datasets in most scenarios, due to its capacity for more aggressive, fine-grained, and region-aware QP modulation. Unlike traditional heuristic-based approaches or coarse quantization control, How2Compress dynamically allocates quality at the macroblock level based on the localized visual importance of regions. This allows it to preserve object-relevant areas with higher fidelity while aggressively compressing uninformative background regions. As a result, it can also leverage the benefits of skip-mode macroblocks and achieves superior accuracy–bitrate trade-offs across diverse video scenarios.

G FURTHER RESULTS

G.1 Generalize to Other Tasks

See Table 9 and Table 10.

Table 9: Pose estimation performance comparison. The table presents evaluation results using standard pose estimation metrics: Object Keypoint Similarity (OKS), Percentage of Correct Keypoints at 0.2 threshold (PCK@0.2), and Percentage of Correct Keypoints at 0.5 threshold (PCK@0.5).

Pose Estimation (OKS / PCK@0.2 / PCK@0.5)			
Sequence	**AQ**	**ACCMPEG**	**Ours**
MOT17-02	0.956 / 0.862 / 0.940	0.954 / 0.854 / 0.936	0.947 / 0.839 / 0.924
MOT17-04	0.968 / 0.921 / 0.966	0.948 / 0.894 / 0.951	0.939 / 0.880 / 0.943
MOT17-09	0.970 / 0.900 / 0.948	0.964 / 0.892 / 0.944	0.954 / 0.886 / 0.937
MOT17-10	0.927 / 0.898 / 0.929	0.926 / 0.895 / 0.932	0.919 / 0.886 / 0.934
MOT17-11	0.975 / 0.948 / 0.965	0.975 / 0.948 / 0.966	0.976 / 0.952 / 0.973
MOT17-13	0.950 / 0.893 / 0.946	0.933 / 0.870 / 0.934	0.935 / 0.875 / 0.941

Table 10: Multi-object tracking performance comparison. The table presents evaluation results using standard tracking metrics: Multiple Object Tracking Accuracy (MOTA), Multiple Object Tracking Precision (MOTP), and ID F1 Score (IDF1).

Multi-Object Tracking (MOTA / MOTP / IDF1)			
Sequence	**AQ**	**ACCMPEG**	**Ours**
MOT17-02	0.910 / 0.949 / 0.938	0.882 / 0.949 / 0.934	0.891 / 0.941 / 0.927
MOT17-04	0.982 / 0.971 / 0.989	0.983 / 0.962 / 0.985	0.982 / 0.961 / 0.982
MOT17-09	0.978 / 0.977 / 0.982	0.974 / 0.971 / 0.976	0.972 / 0.967 / 0.964
MOT17-10	0.930 / 0.955 / 0.950	0.909 / 0.951 / 0.942	0.909 / 0.944 / 0.935
MOT17-11	0.970 / 0.973 / 0.972	0.960 / 0.965 / 0.961	0.965 / 0.963 / 0.945
MOT17-13	0.928 / 0.950 / 0.941	0.923 / 0.943 / 0.932	0.919 / 0.949 / 0.911

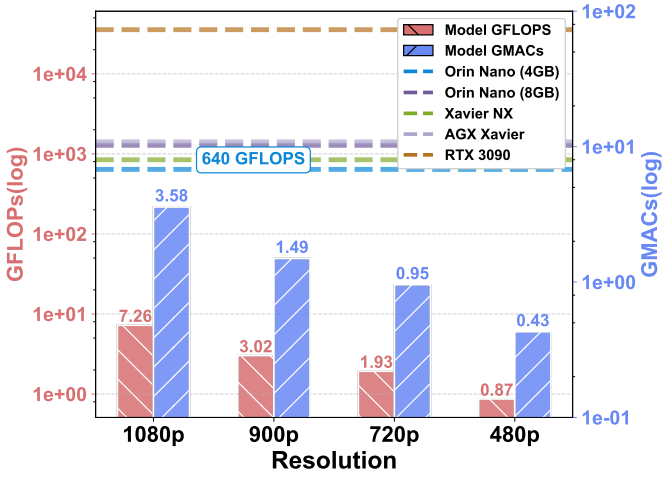


Figure 11: Computational overhead across varying input resolutions and device power profiles

G.2 Overhead of Different Resolution Input

We evaluate How2Compress across different input resolutions (1080p, 900p, 720p, and 480p). As shown in Fig. 11, the model requires only 7.21 GFLOPS even for 1080p videos, rendering it suitable

for resource-constrained edge devices. The model exhibits low computational complexity, with only 3.56 GMACs. In comparison to the lightweight detection model YOLOV5-s [31], the Emphasis Assignment model uses just 0.4× the resources, highlighting its efficient across different platforms.

G.3 Qualitative Result

See Fig. 12.

G.4 Lower SSIM means Better Compression?

We elaborate the rationale in detail in Appendix D. In our framework, lower SSIM arises primarily from aggressive compression of task-irrelevant regions (*e.g.*, background), while preserving information in task-relevant areas (*e.g.*, object boundaries or foreground objects). This localized degradation leads to lower global SSIM, yet the information critical to the downstream task is retained. Because most vision models depend heavily on structural cues (*e.g.*, edges, contours), this selective compression often maintains (if not improves) task accuracy. Thus, in our context, lower SSIM is a sign of more efficient task-aware compression

Figure 12 presents representative visualizations of frames from the MOT17 dataset after compression using different methods. These examples qualitatively illustrate how each method affects spatial quality and the allocation of compression across regions.

Our proposed method introduces more visually apparent artifacts and pushes a larger proportion of macroblocks toward lower quality levels. Despite this aggressive compression strategy, the object detection accuracy is well preserved, as demonstrated in Fig. 13. This result underscores the effectiveness of our framework in making precise, content-aware decisions regarding both the spatial distribution and magnitude of compression.

H GENERALIZATION

In the main paper, we implemented How2Compress using the H.264 codec, focusing on two of its most widely adopted implementations: libx264[61] and the NVIDIA Video SDK[8]. Importantly, How2Compress is not inherently tied to H.264, as it operates independently of the core encoding logic and instead focuses on enhancing the codec's quality assignment strategy. In this section, we first review the compression mechanisms of modern codecs, then illustrate how How2Compress can be seamlessly integrated into each of them with minimal modification.

H.1 compression mechanism of different codecs

Modern video codecs (*i.e.*, H.265 (HEVC)[49], VP9[22], AV1 [18], and H.266 (VVC)) have evolved significantly from the rigid, fixed-block architectures of earlier standards. These codecs employ increasingly flexible and content-adaptive partitioning strategies that enable more efficient encoding of diverse visual scenes.

H.264/AVC typically uses a fixed 16×16 macroblock structure and follows a relatively rigid pipeline of intra/inter prediction, transformation, quantization, and entropy coding. While effective in its time, its limited spatial adaptivity reduces efficiency, especially for high-resolution or visually complex content.

H.265/HEVC introduces Coding Tree Units (CTUs) that support variable sizes up to 64×64. These CTUs are hierarchically partitioned into Coding Units (CUs), Transform Units (TUs), and Prediction Units (PUs), allowing finer spatial granularity. This design improves coding efficiency compared to H.264, particularly for scenes with heterogeneous texture or motion.

VP9 adopts a similar superblock-based design (up to 64×64) and further enhances adaptivity through variance-based adaptive quantization (AQ) and segmentation maps. Additional features such as tile-based parallel decoding and cyclic refresh support better temporal robustness and hardware parallelism.

AV1 extends these ideas by supporting 128×128 superblocks, multiple transform modes, and block-wise quantization control via segmentation and delta QIndex. VVC (H.266) further advances this trajectory with additional partitioning modes and generalized CTU structures, maintaining high efficiency across various content types.

Despite their advantages, the deployment of HEVC, VP9, AV1, and VVC in real-time edge environments is limited by their computational demands and the scarcity of mature, power-efficient hardware support. In contrast, H.264 continues to offer a favorable tradeoff between performance and efficiency. As shown in Fig. **??**, our interleaved QP assignment further reduces H.264's encoding latency, enabling it to achieve faster encoding while maintaining competitive quality and bitrate.

H.2 Integration with H.265 (HEVC)

H.265 [49] supports spatially adaptive quantization via the *delta QP (dQP)*. Each Coding Tree Unit (CTU) is assigned a base QP, and Coding Units (CUs) within the CTU can receive QP adjustments through signed dQP offsets. These offsets are determined based on visual features such as texture complexity or motion, allowing perceptually important regions to receive finer quantization.

How2Compress naturally aligns with this architecture by generating content-aware QP maps that can be translated into dQP values for each CU. These values are explicitly signaled in the bitstream, enabling precise control over spatial quality.

To implement this in practice, minor modifications to HEVC encoders are sufficient. For example, in the x265 implementation, How2Compress can override the QP assignment logic in the rate control module at https://github.com/videolan/x265/blob/master/source/encoder/frameencoder.cpp#L604. Similarly, in the reference HEVC Test Model (HM), CU-level QP control can be integrated by modifying https://vcgit.hhi.fraunhofer.de/jvet/HM/-/blob/master/source/Lib/TLibEncoder/TEncCu.cpp#L1217.

H.3 Integration with H.266 (VVC)

H.266 [2], as an extension of HEVC, increases the maximum CTU size to 128×128 and retains native support for delta QP-based quantization control. Given How2Compress 's ability to predict per-block QP values, its integration into VVC is straightforward.

Several VVC implementations support external QP map injection. For example, the UVG266 encoder [54] allows custom QP assignments via the –roi flag, enabling direct input of region-based QP maps produced by How2Compress. This functionality is realized in the codebase through fine-grained QP control logic (uvg266.h at https://github.com/ultravideo/uvg266/blob/master/src/uvg266.h#L418).

Figure 12: Qualitative visualization of post-compression frames across all methods. (zoom in for better visualization)

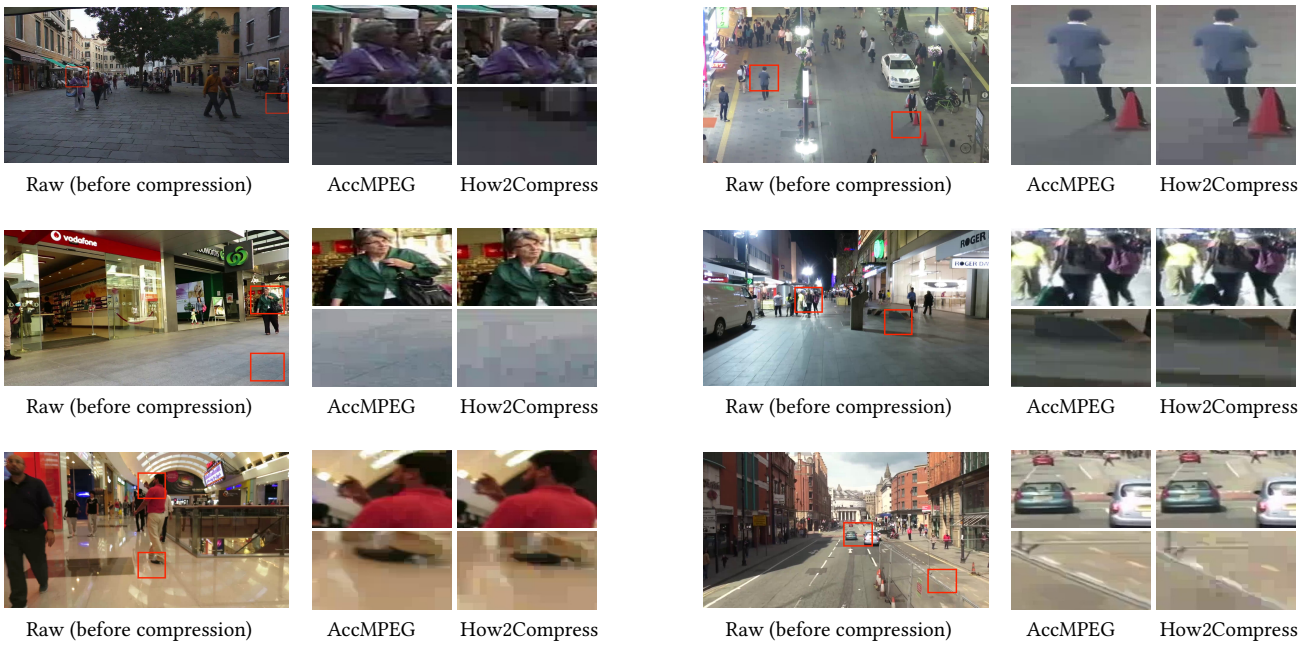


Figure 13: Qualitative visualization of compression artifacts. (Zoom in for better visualization)

Other implementations, such as VVENC [60], do not provide direct support but can be adapted with minimal changes. In particular, the rate control logic at https://github.com/fraunhoferhhi/vvenc/blob/master/source/Lib/EncoderLib/BitAllocation.cpp#L508 can be modified to accommodate externally specified QP values.

H.4 Integration with VP9

The VP9 codec [22], as implemented in libvpx, supports adaptive quantization at the segment level through AQ modes. Internally, the encoder converts QP values into qindex parameters, which control quantization strength.

To integrate How2Compress, externally predicted QP values can be translated to qindex equivalents and injected into the encoder by modifying the rate control mechanism. Specifically, the quantization decision logic at https://github.com/webmproject/libvpx/blob/main/vp9/encoder/vp9_quantize.c#L185 can be extended to reflect How2Compress 's per-segment quality assignments.

H.5 Integration with AV1

AV1, the successor to VP9, also supports block-wise quantization control using qindex values. How2Compress 's predicted QP values can be directly mapped to these indices to guide the encoder's decisions.

The reference encoder libaom [18] provides mechanisms to adjust quantization at the block level, as defined in the AV1 specification at https://aomediacodec.github.io/av1-spec/#quantizer-index-delta-syntax. Practical integration can be realized by modifying the quantization logic in at https://aomedia.googlesource.com/aom/+/refs/tags/v3.12.0/av1/encoder/av1_quantize.c#764, allowing injection of How2Compress-generated quality signals.

I LIMITATIONS

In this section, we discuss the limitations of both the existing frameworks and our proposed How2Compress. Additionally, we outline promising directions for addressing these limitations, which we leave as future work.

Marginal Compression Gains in Certain Scenes. Although How2Compress consistently achieves notable bitrate savings without sacrificing accuracy in most scenarios, it demonstrates only marginal improvement for certain video content (*e.g.*, as shown in Table 3 for the MOT17-04 sequence). In such cases, the built-in codec mechanisms may already achieve efficient compression due to their ability to organize QP values in a way that enhances inter-prediction and increases the use of skip mode macroblocks, thereby improving compression efficiency. One of the key advantages of video compression over image compression lies in the exploitation of temporal redundancy through inter-frame prediction. In H.264, skip mode macroblocks (*i.e.*, inter-predicted macroblocks that omit motion vectors and residuals) can be used when the content exhibits minimal motion and negligible prediction error. However, externally controlling QP at a fine granularity may disrupt the codec's internal optimization pipeline, such as motion estimation and residual prediction, ultimately reducing the proportion of skip mode macroblocks. We believe this issue can be mitigated by introducing temporally-aware QP adjustment mechanisms, which jointly consider visual saliency and motion consistency to align better with the codec's prediction model, thereby improving the ratio of skipped macroblocks without compromising accuracy.

Unstable Bitrate Behavior in Streaming. In real-world video streaming scenarios, especially under bandwidth-constrained or latency-sensitive conditions, maintaining a stable and predictable bitrate is crucial for smooth transmission and adaptive bitrate (ABR) control. However, external fine-grained control of QP, while effective for accuracy-aware compression, which may lead to large fluctuations in bitrate across frames, depending on the spatial and temporal complexity of the scene. This variability may result in buffer underflows, unstable transmission, or degraded Quality of Experience (QoE) in live offloading scenarios. Incorporating bitrate regularization techniques or post-hoc rate control can help address this issue.