

DynaFilter: Cloud-driven Dynamic Filtering for Satellite Edge Intelligence

Ziyang Zhang
Politecnico di Milano

Jie Liu
Harbin Institute of Technology

Luca Mottola
Politecnico di Milano, RISE,
and Uppsala University

Abstract

Modern satellite edge systems, including those performing remote sensing tasks such object detection and tracking, are characterized by severely limited bandwidth and intermittent connections, making continuous data transmission to the cloud impractical. Existing edge-cloud systems, however, either require heavy pre-processing before analysis, for instance, full decompression of imagery data, or transmit all compressed data regardless of relevance. To address these challenges, we design DynaFilter, a dynamic filtering technique that enables satellite edge devices to perform selective region-of-interest (RoI) inference directly in the compressed-domain, without full decompression. Our key insight is that low-level compression syntax, specifically DC coefficients/AC energy in JPEG images and motion vectors in video streams, exhibits strong correlations with high-level semantic queries. By establishing a precise mapping between cloud query semantics and multimodal compressed-domain features, DynaFilter enables the edge to identify and transmit only relevant data associated to RoIs. Extensive evaluations show that DynaFilter reduces the total volume of pixel data for decoding and subsequent inference by $1.6\times\sim 7.1\times$ for images, and achieves 92.0% bandwidth savings for video streams compared to state-of-the-art baselines. Furthermore, it decreases energy consumption by 43.1~88.6% on target devices and achieves a $1.6\times\sim 3.0\times$ speedup in inference latency.

CCS Concepts

• **Networks** → **Mobile networks**; • **Human-centered computing** → **Ubiquitous and mobile computing**.

Keywords

Edge Intelligence, Satellite Computing, Dynamic Filtering, Deep Learning Inference

ACM Reference Format:

Ziyang Zhang, Jie Liu, and Luca Mottola. 2026. DynaFilter: Cloud-driven Dynamic Filtering for Satellite Edge Intelligence. In *The 32nd Annual International Conference on Mobile Computing and Networking (MobiCom '26)*, October 26–30, 2026, Austin, TX, USA. ACM, New York, NY, USA, 16 pages. <https://doi.org/10.1145/3795866.3844496>

1 Introduction

Edge computing is making it into space systems, especially those built with off-the-shelf hardware and thus inherently resource constrained [5, 14, 15, 40]. This evolution is particularly transformative for remote sensing tasks, such as environmental monitoring [44] and aerial surveillance [48].

Satellite edge computing. In modern satellite edge computing architectures [14, 15, 47], visual data captured by orbiting devices is typically compressed using standard codecs like JPEG or H.264 before transmission to cloud servers on the ground for analysis. However, most existing systems transmit all compressed data regardless of relevance [3, 33]. This is fundamentally impractical in severely bandwidth-constrained environments, or in the presence of intermittent connections, as in most modern satellite systems, especially those built with off-the-shelf hardware or exploiting opportunistic connections to the ground [5, 14, 15, 43, 57].

The fundamental challenge arises because edge devices must transmit compressed data through constrained channels, often providing kbps of bandwidth, if at all [10, 37, 39], while the cloud's analytical requirements, such as identifying specific object categories with minimum confidence thresholds, dynamically evolve based on mission objectives. Thus, edge devices are forced to transmit large volumes of compressed data that may prove irrelevant to the cloud's current queries, wasting precious communication resources. The challenge manifests in three critical dimensions.

First, the inherent mismatch between how data is transmitted from edge devices and how the data of interest is queried at the cloud causes *data drift* [38, 45]. In edge-cloud systems where bandwidth is extremely limited, transmission efficiency is key. Existing works [24, 25, 58] cannot adapt to dynamically changing query requirements, forcing systems to either transmit all compressed data regardless of relevance or repeatedly retransmit data when queries change.



This work is licensed under a Creative Commons Attribution 4.0 International License.

MobiCom '26, Austin, TX, USA

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2505-0/2026/10

<https://doi.org/10.1145/3795866.3844496>

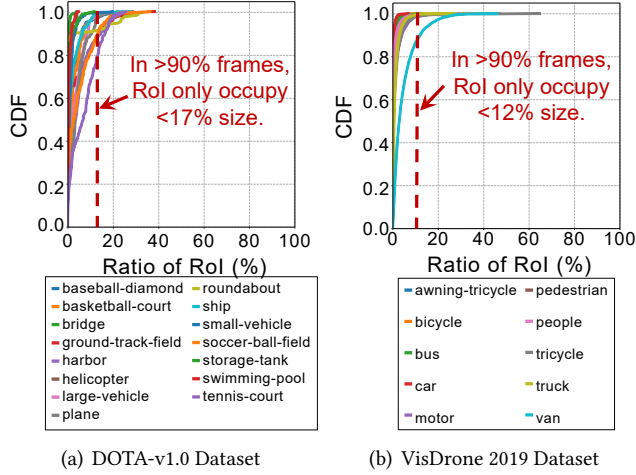


Figure 1: In over 90% of frames over DOTA-V1.0 and VisDrone 2019 datasets, the region of interest (RoI) for object detection tasks only occupies 12~17% of the spatial area of a frame.

Second, the overhead of *decompress-infer-recompress* (DIR) at the edge creates a significant bottleneck [18, 22, 63], which holds for both static imagery and videos. As shown in Figure 1, the analysis of the widely used DOTA-v1.0 (satellite imagery) and VisDrone 2019 (aerial imagery) datasets reveals that the regions of interest (RoI) occupies merely 12~17% of the area in over 90% of frames. Figure 2 further reveals that on off-the-shelf edge hardware platforms like the NVIDIA Jetson Orin Nano, full decompression of high-resolution images consumes up to 35% of the total processing time and 42% of energy consumption. Even if combined with lightweight inference, that cost remains. Figure 2 does not even report recompression costs, which may even be higher than decompression and would add on top.

For videos, DIR includes video transcoding, shown to be computation-intensive [6, 17], especially at the edge. For videos, recompression is the bottleneck: video encoding dominates end-to-end latency [19]. This process would also be CPU-hungry [6], consuming 60-90% of the CPU resource and adding 5~10% battery drain [17]. Consequently, in environments where connections are intermittent and video streaming is impractical, this sparsity coupled with the high cost of pixel-domain processing would lead to a massive waste of computational resources and bandwidth.

Third, the limitations of existing edge-cloud systems [3, 25, 33] are further exacerbated by the *resource constraints* of satellite edge devices, where continuous operation is crucial, yet battery life and energy budgets are limited [34]. Existing work [24, 26, 32, 58] fully decompress or upload compressed data to the cloud, without adapting to dynamically changing queries. For instance, DeepCOD [58] processes all compressed features without dynamic adaptation. LimitNet [24]

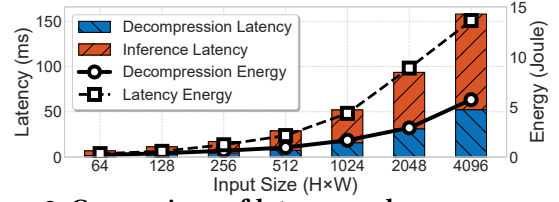


Figure 2: Comparison of latency and energy consumption when performing YOLOv8 inference with different input sizes on NVIDIA Jetson Orin Nano.

requires uploading all compressed data to the cloud, resulting in significant energy consumption. Although conserving energy is not a primary objective of existing edge computing literature, it becomes so when edge computing is deployed on a resource-constrained device such as a satellite built with off-the-shelf hardware [5, 14, 15, 40, 57].

Contribution. To address these challenges, we design DynaFilter, a cloud-driven dynamic filtering technique. DynaFilter targets extremely bandwidth-constrained environments where data is represented by either static images or video streaming is applied for limited time intervals, for instance, whenever a reliable connection to the ground is available [43]. It enables filtering data that is relevant to a cloud's query before transmission, enabling analysis in a highly-constrained communication setting.

Unlike prior work built on a paradigm where the edge uploads data regardless of the specific application interests, while the cloud analyzes it, DynaFilter fosters a different execution model. The cloud dynamically configures filtering parameters based on current queries representing application interests. The edge executes compressed-domain filtering to identify potential RoIs without full decompression.

Our key insight enabling this functionality is that low-level JPEG/Video features, such as DC coefficients/AC energy, and motion vectors in the compressed-domain exhibit strong correlations with high-level semantics, for instance, object categories in a classification task. Backed by an experimental foundation described next, we establish a mapping between compressed data and dynamic filtering, enabling accurate RoI prediction with minimal computational overhead. Extensive evaluations show that DynaFilter reduces decompressed data size by $1.6\times\sim 7.1\times$, compared to state-of-the-art baselines. Furthermore, it decreases energy consumption by 43.1~88.6% and achieves $1.6\times\sim 3.0\times$ speedup in inference latency.

Overall, we make the following contributions:

- (1) We reveal the data drift challenge in satellite edge computing, where compressed data stored at the edge does not align with dynamically changing queries at the cloud, representing application interests.
- (2) We propose a new cloud-driven dynamic filtering technique, enabling the cloud to dynamically configure

filtering parameters for edge devices and eliminating the need for full inference or recompression.

- (3) We design a compressed-domain feature mapper that establishes a precise mapping between high-level semantics and low-level compressed-domain features.
- (4) We develop a multimodal feature fusion filter that enhances filtering accuracy by incorporating temporal-spatial features for dynamic video scenarios.

In the rest of the paper, Section 2 provides background information on compressed-domain features. Section 3 motivates the work, whereas Section 4 describes the system overview. Section 5 details the design. Section 6 presents the implementation, and Section 7 reports experimental results. Section 8 presents related work, while Section 9 discusses limitations and extensions. Section 10 ends the paper.

2 Background

We design DynaFilter to leverage the inherent organization of discrete cosine transform (DCT) coefficients [9] within the JPEG standard [51], which systematically partitions image information into frequency-domain components [46]. This design principle enables DynaFilter to perform semantic analysis directly in the compressed domain.

Discrete cosine transform (DCT). JPEG divides images into 8×8 pixel blocks and applies DCT to each block:

$$F(u, v) = C(u)C(v) \sum_{x=0}^7 \sum_{y=0}^7 f(x, y) \cos\left[\frac{(2x+1)u\pi}{16}\right] \cos\left[\frac{(2y+1)v\pi}{16}\right] \quad (1)$$

where $f(x, y)$ represents the pixel value at spatial position (x, y) , $F(u, v)$ is the DCT coefficient at frequency position, $C(u)$ and $C(v)$ are normalization factors.

Discrete cosine (DC) coefficient. The DC coefficient, located at position $(0, 0)$ in the DCT matrix, represents the average brightness of the 8×8 block and is calculated as:

$$DC_{coefficient} = F(0, 0) = \frac{1}{4} \sum_{x=0}^7 \sum_{y=0}^7 f(x, y) \quad (2)$$

The DC coefficient [42] difference between neighboring blocks is often encoded rather than the absolute value, leveraging this spatial correlation for additional compression.

Alternating current (AC) energy. The remaining 63 coefficients (where $u > 0$ or $v > 0$) are AC coefficients [59], representing higher frequency details within the block (e.g., edges, textures, and fine-grained features). The AC energy is defined as the sum of absolute values of all AC coefficients:

$$AC_{energy} = \sum_{u=0}^7 \sum_{v=0}^7 |F(u, v)| - |F(0, 0)| \quad (3)$$

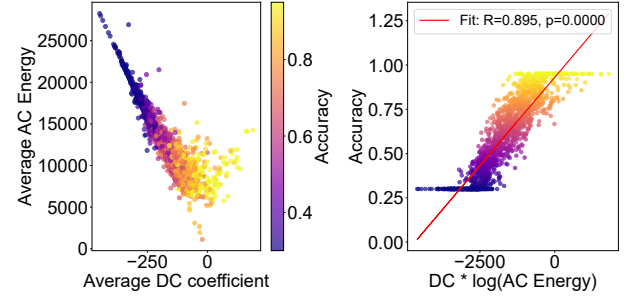


Figure 3: Relationship between JPEG features, that is, DC coefficient and AC energy, and inference accuracy.

The AC energy effectively measures the texture richness of an image. Regions with high AC energy typically contain edges, textures, or object boundaries, while background regions have low AC energy.

3 Motivation

Our work is motivated by the challenges in satellite edge computing, where orbiting devices must analyze high-resolution imagery and video streams under severe bandwidth on top of possibly intermittent communication channels, and subject to energy constraints. By analyzing the characteristics of remote sensing data and the processing overhead on edge devices, we derive four key observations.

First, we find a strong relationship between low-level JPEG features, that is, DC coefficients and AC energy, and high-level semantics, for example, inference accuracy, exist, as shown in Figure 3. Accuracy here refers to the inference mAP, that is, mean average precision, achieved when the DNN model is run only on regions matching specific AC/DC signatures. This correlation is consistent across different object categories in Figure 4, where the JPEG features for each category have distinct distributions. Note that the DC coefficients for object regions tend to cluster within specific ranges depending on the object category, creating identifiable patterns in the compressed-domain.

Takeaway 1: Low-level compressed-domain features are closely related to high-level semantics.

As shown in Figure 5(a) as an example, the original image contains multiple RoIs. Figure 5(b) shows that object regions typically exhibit AC energy values $15 \sim 25 \times$ higher than background regions, creating a clear distinguishable boundary in the compressed-domain. Similarly, Figure 5(c) shows that DC coefficients provide complementary information about illumination conditions and large-scale structures. To leverage both features, we combine them as $DC \times \log(AC \text{ Energy})$. Figure 5(d) shows a highly accurate representation of potential RoI by preserving the high-frequency detail sensitivity of AC

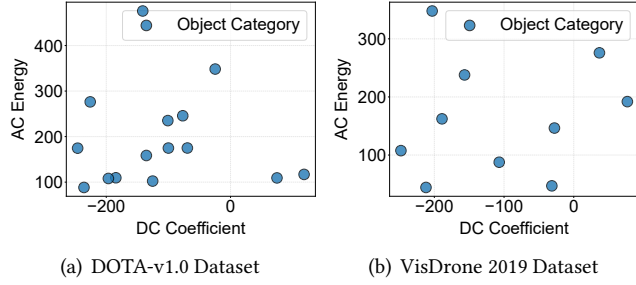


Figure 4: Distribution of JPEG features, that is, DC coefficient and AC energy, for each category.

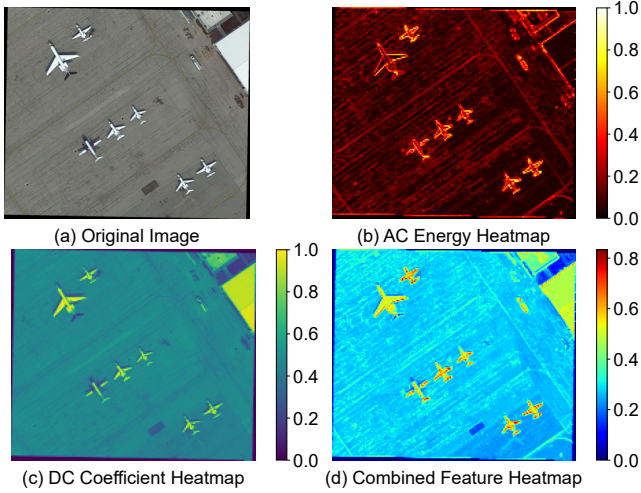


Figure 5: JPEG features for the plane category.

energy, while incorporating the illumination and structural information from DC coefficients.

Takeaway 2: Low-level JPEG compressed-domain features enable accurate RoI-based inference.

To confirm this intuition, we compare full inference against RoI-based inference. As shown in Figure 6(a), RoI-based inference achieves a $2.0\times\sim 2.5\times$ speedup in inference latency across different JPEG quality factors. Figure 6(b) confirms that this is not detrimental to accuracy, showing a less than 2% loss. This demonstrates that processing only the regions identified by compressed-domain analysis is a viable strategy. Note that DynaFilter aims to spare the prohibitive overhead of the DIR pipeline rather than just skipping background blocks. Even in scenarios where RoIs occupy a large area, DynaFilter remains effective by selectively decoding only the relevant blocks as determined by the cloud-driven filters, thus avoiding the recompression overhead.

Takeaway 3: RoI-based inference can achieve high energy efficiency without compromising accuracy.

For satellite video monitoring, traditional pixel-domain methods, for example, optical flow techniques [28, 29], are

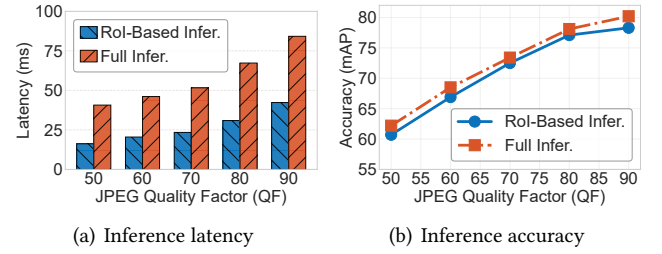


Figure 6: Comparison of RoI-based Inference and Full Inference with different JPEG quality factors.

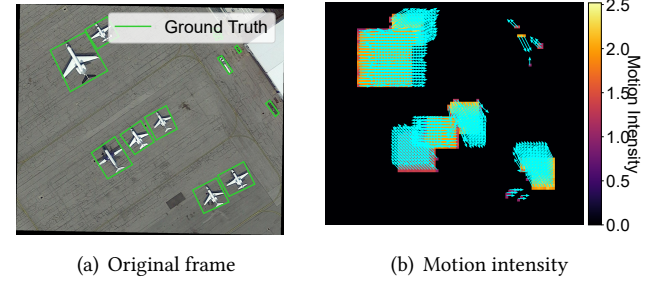


Figure 7: VisDrone 2019 dataset: Spatial correlation between ground objects and compressed-domain MVs.

computationally prohibitive for edge devices. However, we observe that video codecs, such as H.264/HEVC, inherently encode temporal dynamics as motion vectors during compression. To study this aspect, we extracted motion vectors from compressed video sequences in the VisDrone 2019 dataset. As shown in Figure 7, the magnitude of motion vectors exhibits a strong spatial correlation with moving targets, separating them apart from the static background. This implies that motion can be identified directly from the compressed bitstream, enabling DynaFilter to locate dynamic RoIs with negligible computational cost before full decoding.

Takeaway 4: Motion semantics in video streams are natively encoded in bitstream motion vectors.

The observations we articulated provide a stepping stone for the design of DynaFilter, described next.

4 System Overview

Figure 8 shows the architecture of DynaFilter, which consists of two stages designed to bridge the semantic gap between cloud queries and compressed data at the edge.

Filter generation at the cloud. This stage aims to translate high-level semantic queries into multimodal compressed-domain filter configurations. In step ❶, the compressed-domain feature mapper takes object categories and confidence thresholds as input. It utilizes a neural network to determine corresponding baseline thresholds for both spatial features, that is, AC energy and DC coefficients, and

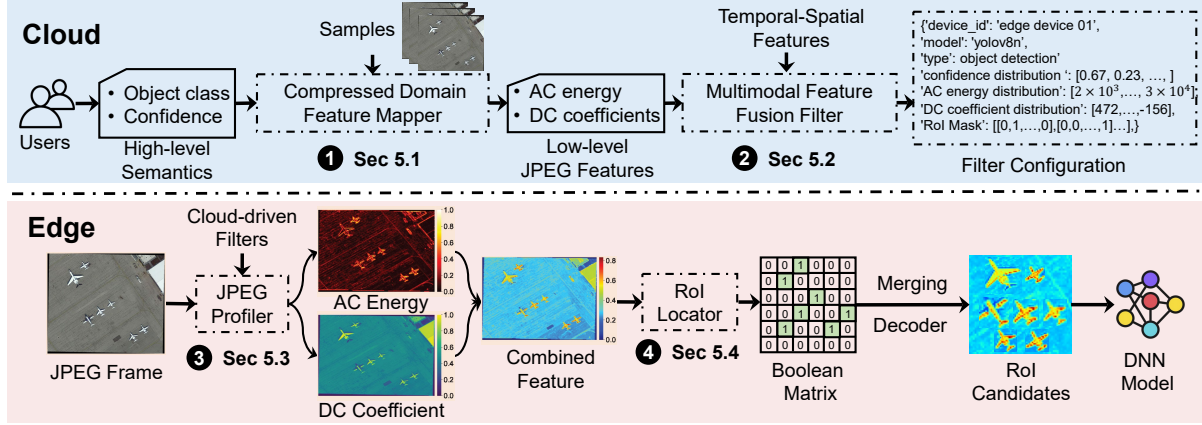


Figure 8: Two stages in DynaFilter: multimodal filter generation at the cloud for converting high-level semantics queries into compressed-domain filter configurations, and RoI-based inference at the edge for applying filters directly in the compressed-domain and performing partial decompression and inference only on potential RoIs.

temporal features, that is, motion vectors and residual energy. In step ②, the multimodal feature fusion filter enhances adaptability by incorporating dynamic context. It directly analyzes motion vector statistics from historical bitstreams to generate motion-aware thresholds and refine tracking parameters, creating a robust filter configuration that adapts to both texture and motion changes.

Compressed-domain filtering at the edge. This stage executes selective RoI-based inference directly on the input bitstreams. In step ③, the JPEG/video codec profiler parses the bitstream header to identify the modality and routes data accordingly. It extracts DCT coefficients for static content (JPEG or I-frames) to capture spatial texture, and extracts motion vectors (MVs) and residuals for dynamic video content (P/B-frames) to capture temporal motion intensity. In step ④, the RoI locator applies the filters configured at the cloud to extract these features. It identifies potential RoIs using adaptive threshold matching, merges them into spatially coherent regions, and formats them for partial decompression.

5 DynaFilter Design

We detail the design of the single components in DynaFilter, according to Figure 8.

5.1 Compressed-Domain Feature Mapper

As shown in Figure 9, we develop a streamlined neural network architecture to implement the mapper while maintaining accuracy and efficiency. To support the analysis of both static images and video streams, as detailed next, we extend the mapper’s capability to learn cross-modal representations.

The input to the mapper is an $(N+10)$ -dimensional feature vector that includes three elements: (1) an N -dimensional one-hot encoding of the object categories (N represents the

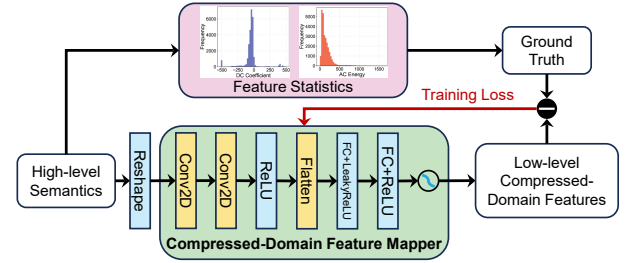


Figure 9: Compressed-domain feature mapper.

number of object categories); (2) a 1-dimensional confidence threshold value required by the user; (3) 9-dimensional historical statistics. The latter includes the mean and standard deviation of the AC energy, DC coefficients, MVs, and residual energy, along with a global normalization factor. These inputs collectively represent the semantic parameters and contextual information—covering both texture and motion priors—necessary for accurate threshold prediction.

The first stage of the mapper is an input reshaping operation that transforms the $(N+10)$ -dimensional feature vector into a structured tensor. For practical implementation, we reshape this vector into a $\lceil \sqrt{N+10} \rceil \times \lceil \sqrt{N+10} \rceil \times 1$ tensor. This operation recognizes that semantic parameters possess an inherent spatial structure where related features are positioned in proximity to each other.

After reshaping, the first hidden layer is a convolutional layer with two filters of 3×3 size with padding of one. This layer applies spatial filters that capture local feature interactions. The second hidden layer begins with a flattening operation to integrate the spatially extracted features into a holistic representation. This flattened vector then passes through a fully connected layer with 16 neurons using LeakyReLU [55]. The third hidden layer consists of a fully connected layer

with 8 neurons using ReLU [2], extracting the most essential information needed for the final prediction. The output consists of four neurons with linear activation, corresponding to the predicted thresholds for AC energy (T_{ac}), DC coefficients (T_{dc}), MVs (T_{mv}), and residual energy (T_{res}). Post-processing is applied to ensure these values fall within valid ranges, for example, ensuring energy is not negative.

We implement two alternative approaches for comparison: a statistical analysis approach and a random forest approach. Table 1 compares these approaches on the DOTA-v1.0 and VisDrone 2019 datasets. The NN-based approach achieves the highest prediction accuracy across both datasets. This stems from the NN's ability to model complex non-linear relationships between high-level semantics and low-level compression signatures, both spatial and temporal, while leveraging feature interactions through the convolutional operation. The NN-based approach remains lightweight, requiring only minimal memory for parameters and achieving low execution times (12.6~13.1ms), significantly outperforming the random forest approach. Note that our NN-based approach learns the specific compressed-domain features for any query. For instance, a low-AC smooth surface could still be identified if that is the query semantics.

5.2 Multimodal Feature Fusion Filter

Real-world edge intelligence applications, especially video streams, require adaptive handling of complex dynamics such as object motion, occlusion, and varying coding structures. We address this by integrating multiple feature modalities through a Transformer-based architecture [50]. This enables DynaFilter to generate a robust, context-aware configuration that includes not only spatial thresholds (T_{ac} , T_{dc}) but also temporal thresholds (T_{mv} , T_{res}) for video analysis. Figure 10 illustrates the filter operation, whereas Algorithm 1 details the complete process.

Transformer-based feature encoding. We transform heterogeneous feature inputs into a unified representation (line 1 in Algorithm 1). To support video streams, we expand the input modalities: (1) compressed-domain features: historical statistics of AC energy and DC coefficients; (2) temporal context features: historical statistics of motion vectors (MV)

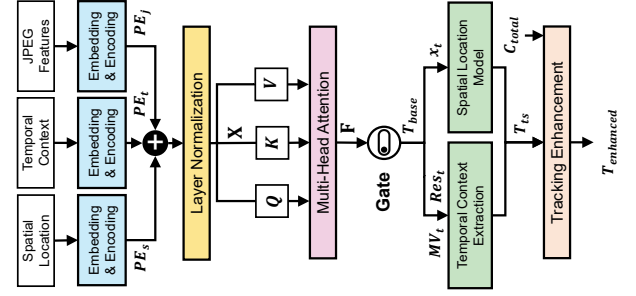


Figure 10: Architectural overview of the multimodal feature fusion filter.

Algorithm 1 Multimodal Feature Fusion Process

Input: Semantic parameters S , Compressed features C , Motion stats M , Spatial features L

Output: Enhanced filter configuration $T_{enhanced}$

- 1: $X \leftarrow \text{LayerNorm}(\text{Concat}(E_c + PE_j, E_m + PE_m, E_l + PE_l))$
- 2: $F \leftarrow \text{MultiHead}(X)$
- 3: $g \leftarrow \sigma(W_g \cdot F + b_g)$
- 4: $T_{ac/dc} \leftarrow \text{Compute spatial thresholds}(g, S)$
- 5: $T_{mv} \leftarrow \mu_{mv} - g_{mv} \cdot \sigma_{mv}$
- 6: $T_{res} \leftarrow \mu_{res} - g_{res} \cdot \sigma_{res}$
- 7: $T_{base} \leftarrow \{T_{ac}^{min/max}, T_{dc}^{min/max}, T_{mv}, T_{res}\}$
- 8: $I_{motion} \leftarrow \frac{1}{N} \sum ||MV_t(i)||_2$
- 9: $x_t \leftarrow \text{Kalman filter}(x_{t-1}, z_t)$
- 10: $T_{ts} \leftarrow \text{Enhance thresholds}(T_{base}, I_{motion}, x_t)$
- 11: **if** task is object tracking **then**
- 12: $C_{total} \leftarrow \alpha \cdot C_{space} + \beta \cdot C_{mv} + \gamma \cdot C_{reid}$
- 13: $T_{enhanced} \leftarrow T_{ts} \cup \text{Tracking enhancement}(C_{total})$
- 14: **else**
- 15: $T_{enhanced} \leftarrow T_{ts}$
- 16: **end if**
- 17: **return** $T_{enhanced}$

and residual energy derived from the video streams. It allows the model to sense the motion context (e.g., high-speed movement and stationary hovering); (3) spatial location: the normalized coordinates of detected objects. The unified representation X is created by concatenating the encoded features from each modality with their positional encodings:

$$X = \text{LayerNorm}(\text{Concat}(E_j + PE_j, E_t + PE_t, E_s + PE_s)) \quad (4)$$

where E_j , E_t , and E_s represent the embedded compressed-domain features, temporal context features, and spatial location information, respectively, while PE_j , PE_t , and PE_s are their corresponding positional encodings.

Multi-head attention fusion. We integrate the encoded features using multi-head attention [13] to dynamically weight the importance of different modalities (line 2 in Algorithm 1). For instance, in high-motion video scenes, the attention mechanism automatically assigns higher weights to the temporal context features (E_t) to prioritize motion-based filtering over static texture matching. The multi-head attention mechanism is formalized as:

$$F = \text{MultiHead}(X) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \quad (5)$$

Table 1: Prediction accuracy of mappers.

Dataset	Approach	Accuracy	Memory	Time
DOTA-v1.0	Statistical Analysis	75.3%	0.5MB	8.3ms
	Random Forest	82.7%	4.2MB	15.7ms
	NN (Ours)	91.8%	0.8MB	12.6ms
VisDrone 2019	Statistical Analysis	76.2%	0.5MB	8.4ms
	Random Forest	83.5%	4.3MB	15.9ms
	NN (Ours)	92.3%	0.9MB	13.1ms

where $\text{head}_i = \text{Attention}(\mathbf{XW}_i^Q, \mathbf{XW}_i^K, \mathbf{XW}_i^V)$.

Adaptive threshold generation. We implement an adaptive gating mechanism [64] that generates confidence-aware thresholds (line 3 in Algorithm 1). Crucially, we extend the gating function to output thresholds for motion vectors T_{mv} and residual energy T_{res} (line 4~7 in Algorithm 1). The gating function is formalized as:

$$\mathbf{g} = \sigma(\mathbf{W}_g \cdot \mathbf{F} + \mathbf{b}_g) \quad (6)$$

where $\mathbf{F}$ is the feature representation from the attention computation stage, $\mathbf{W}_g$ and $\mathbf{b}_g$ are learnable parameters, and σ is the sigmoid function. The gating values then modulate the historical feature statistics to produce the final filter configuration $T_{base} = \{T_{ac}^{min}, T_{ac}^{max}, T_{dc}^{min}, T_{dc}^{max}, T_{mv}, T_{res}\}$. $T_{mv} = \mu_{mv} - g_{mv} \cdot \sigma_{mv}$ represents the minimum motion intensity required to trigger a dynamic RoI.

Temporal-spatial feature enhancement strategy. This strategy aims to support object tracking tasks (line 8~10 in Algorithm 1). The temporal component utilizes the motion intensity (I_{motion}) derived directly from the bitstream:

$$I_{motion} = \frac{1}{N} \sum_{k=1}^N \sqrt{(mv_x^{(k)})^2 + (mv_y^{(k)})^2} \quad (7)$$

where mv_x and mv_y are the motion vector components extracted by the profiler. It allows the Kalman filter-based state estimation to predict object trajectories with significantly higher accuracy, ensuring that the generated filters are spatially aligned with the predicted location of moving targets.

Tracking association enhancement. For tracking tasks, we introduce a multimodal fusion cost function (line 11~13 in Algorithm 1) that incorporates motion consistency based on the extracted MVs. Standard block-matching motion vectors (MV) in video coding standards (e.g., H.264/HEVC) are optimized for rate-distortion efficiency rather than true optical flow, rendering them inherently prone to high-frequency noise and pseudo-motion vectors in textureless regions. To mitigate this, our multimodal cost function explicitly filters out high-frequency MV noise through multi-source validation boundaries:

$$C_{total} = \alpha \cdot C_{space} + \beta \cdot C_{mv} + \gamma \cdot C_{reid} \quad (8)$$

where $C_{mv} = 1 - \cos(\vec{v}_{pred}, \vec{v}_{det})$ measures the cosine similarity between the predicted velocity vector and the detected motion vector. Crucially, to prevent high-frequency noise from corrupting association, candidate MVs are strictly cross-validated against spatial-prediction bounding boxes generated by the Kalman filter and constrained by the residual energy parameter E_{res} . MVs falling outside the spatial validation boundaries or lacking corresponding residual energy support are flagged as noise and suppressed, ensuring robust tracking association even under noisy compressed streams. This direct utilization of bitstream motion data significantly

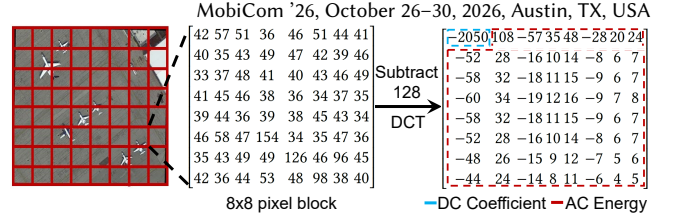


Figure 11: JPEG profiler: it processes JPEG bitstream incrementally to extract DCT coefficients, then calculates DC coefficients and AC energy for each 8×8 block without full decompression.

reduces ID switching during occlusion compared to relying on pixel-domain inference alone. The final enhanced configuration $T_{enhanced}$ is then transmitted to the edge (line 14~16 in Algorithm 1).

5.3 JPEG/Video Codecs Profiler

To seamlessly support both static imagery (e.g., JPEG) and dynamic video streams (e.g., H.264/HEVC), our profiler adopts a unified branching architecture. It first parses the bitstream header to identify the data modality, that is, image frame or video stream, and subsequently routes the data into the corresponding feature extraction pipeline. Crucially, our design explicitly circumvents the most computationally intensive stages of the traditional decoding loop, including inverse quantization, inverse discrete cosine transform (IDCT), and specifically for video codecs, loop filtering and motion compensation. Our profiler drastically reduces computational overhead compared to full decoding.

5.3.1 Static Feature Extraction for JPEG Streams. Modern commercial satellite payloads natively enforce standard hardware level compression directly on sensor components to limit local storage footprints [5, 14, 15]. As a direct consequence, uncompressed raw pixel frames are physically unavailable at the data ingestion phase, making conventional pixel-domain pipelines impractical. This hardware constraint enforces the necessity of compressed-domain processing. As shown in Figure 11, the profiler uses libjpeg-turbo [1] to process JPEG bitstream incrementally to extract DCT coefficients, then calculates DC coefficients and AC energy for each 8×8 block without full decompression. DC coefficients capture the overall brightness of regions, which is useful for distinguishing between different object categories based on their typical luminance patterns. AC energy, on the other hand, measures the texture complexity within a block, which is indicative of whether a region contains meaningful content (such as object edges) or uniform background. These values would then be compared against the filter configuration thresholds ($T_{ac}, T_{dc}^{min}, T_{dc}^{max}$) to determine if this block is part of a potential RoI.

5.3.2 Dynamic Feature Extraction for Video Streams. The conventional DIR paradigm incurs prohibitive end-to-end

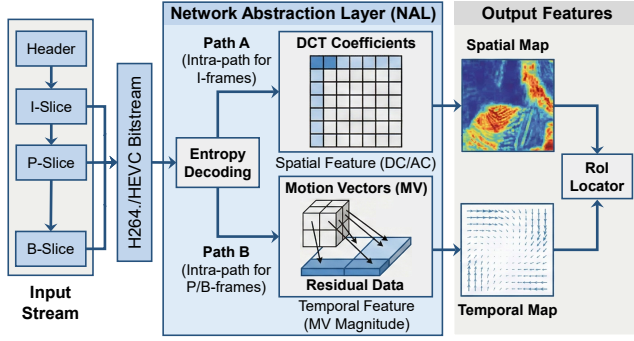


Figure 12: Video profiler: it extracts DCT and MV features from I-Frames and P/B-Frames by parsing NAL units and entropy-decoded data, avoiding the overhead of pixel-level reconstruction and loop filtering.

latency. Furthermore, since resource-constrained satellite edge devices often lack dedicated hardware encoders, the re-compression step consumes substantial CPU resources. Our Profiler eliminates this bottleneck by extending native support to H.264/HEVC bitstreams. As shown in Figure 12, our profiler parses network abstraction layer (NAL) units and applies distinct feature extraction strategies based on the slice type, either keyframes Intra-coded Frame (I-Frame) or non-keyframes Predicted/Bi-predictive Frames (P/B-Frames)).

I-Frame processing. The processing logic for I-Frames mirrors that of JPEG, leveraging intra-prediction and DCT transforms. Our profiler extracts DCT residual coefficients and prediction modes. These features effectively capture the spatial texture distribution of video keyframes, which is used to identify RoIs in static backgrounds.

P/B-Frames processing. P/B-frames encode temporal changes by referencing preceding or succeeding frames. Our profiler extracts MVs and prediction residuals from the bitstream without requiring pixel-level reconstruction, by means of two separate techniques.

- **Motion intensity analysis:** we calculate the per-block magnitude of the MV as $|MV| = \sqrt{mv_x^2 + mv_y^2}$. Significant MVs indicate dynamic regions within the scene, such as moving objects, providing semantic cues that static image features cannot offer.
- **Spatiotemporal consistency verification:** By integrating residual energy, our profiler effectively distinguishes between moving objects and illumination changes. A region exhibiting both large MV magnitude and high residual energy is identified with high probability as a dynamic RoI. We calculate the L1 norm of the quantized transform coefficients (QTC) as the residual energy using $E_{res}^{(k)} = \sum_i |QTC_i|$.

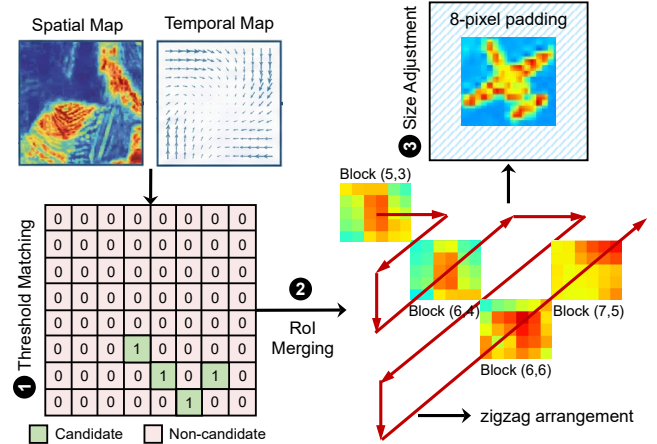


Figure 13: The workflow of the RoI Locator processing a 32×32 pixel image (4×4 blocks with 8×8 pixels).

5.4 RoI Locator

The RoI Locator aims to translate the raw feature maps extracted by the profiler into actionable binary masks. As shown in Figure 13, the locator processes the feature grid through three pipelined stages: threshold matching, RoI merging, and size adjustment.

To support concurrent multi-category queries, the cloud coordinates filtering parameters by computing a spatial-temporal threshold union. Specifically, given a set of active queries $Q = \{q_1, q_2, \dots, q_K\}$, the cloud aggregates their individual low-level features into a unified filter configuration:

$$T_{\text{union}} = \bigcup_{k \in Q} T^{(k)} = \bigcup_{k \in Q} \left\{ T_{dc}^{(k)}, [T_{dc}^{\min(k)}, T_{dc}^{\max(k)}], T_{mv}^{(k)}, T_{res}^{(k)} \right\} \quad (9)$$

The cloud transmits T_{union} as a single compact configuration payload to the edge device. The RoI Locator at the edge then performs a single-pass thresholding operation to identify candidate blocks matching *any* of the target categories in Q . Overlapping or co-located targets belonging to different query classes are subsequently disentangled and categorized by the downstream DNN classifier during the inference stage. For prioritized missions, the cloud can dynamically adjust T_{union} to cover only high-priority targets.

Note that DynaFilter is designed for object- or region-centric queries, the compressed-domain filtering paradigm naturally excels at semantic queries bound to distinct localized textures, structural shapes, and motion intensities. Conversely, queries that demand global contextualization while lacking descriptive local features, such as estimating the total cloud cover percentage over an entire frame, fall outside the scope of localized compressed-domain filtering and require full decoding or a separate global-scene estimator.

5.4.1 Adaptive Threshold Matching. The locator first generates a binary candidate map Θ by comparing block-level

features against the cloud-configured filters. Figure 13 (stage 1) visualizes this process for a 32×32 pixel area.

Spatial matching for JPEG & I-Frames. For static content, the locator evaluates spatial texture features. In Figure 13, a block (i, j) is marked as a candidate ("1") if its DC coefficient $D[i, j]$ and AC energy $A[i, j]$ fall within the specific ranges defined by the cloud (e.g., $T_{ac} \in [1500, 2000]$). It can be formalized as:

$$\Theta_{spatial}(i, j) = \begin{cases} 1, & \text{if } D[i, j] \in [T_{dc}^{min}, T_{dc}^{max}] \wedge A[i, j] \in [T_{ac}^{min}, T_{ac}^{max}] \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

Temporal matching for P/B-Frames. The matching logic shifts to motion semantics. Instead of texture, the locator evaluates the MV magnitude $|MV[i, j]|$ and residual energy $E_{res}[i, j]$. A block is flagged if it exhibits significant motion intensity or prediction error:

$$\Theta_{temporal}(i, j) = \begin{cases} 1, & \text{if } |MV[i, j]| \geq T_{mv} \wedge E_{res}[i, j] \geq T_{res} \\ 0, & \text{otherwise} \end{cases} \quad (11)$$

This ensures that we capture dynamic objects that may not have distinct static textures but strong motion signatures.

5.4.2 Rol Merging. The raw binary map often contains fragmented blocks. Figure 13 (stage 2) demonstrates how the RoI Locator aggregates these fragments into coherent regions. We apply connected component analysis using 4-connectivity, that is, merging blocks sharing horizontal or vertical edges. The connectivity function Ψ merges two candidate blocks p and q if:

$$\Psi(p, q) = \begin{cases} 1, & \text{if } |p_x - q_x| + |p_y - q_y| \leq 1 \\ 0, & \text{otherwise} \end{cases} \quad (12)$$

After merging, the blocks are organized in a zigzag arrangement to align with standard JPEG entropy coding orders, as shown by the red arrows in Figure 13. Empirically, the zigzag pattern preserves the spatial frequency distribution of objects better than standard raster scanning, facilitating coordinate mapping for the subsequent decoding step [8, 51].

5.4.3 Size Adjustment and Temporal Smoothing. Figure 13 (stage 3) shows the padding process. If a merged region is smaller than the model's minimum receptive field (e.g., 32×32 pixels), the locator automatically expands the bounding box with padding pixels. This is crucial for detecting small objects like distant vehicles in aerial imagery.

For video content, to prevent RoI flickering caused by compression artifacts while avoiding boundary trailing for fast-moving targets, we implement an adaptive recursive smoothing mechanism:

$$\Theta_{final}^{(t)}(i, j) = \alpha \cdot \Theta_{final}^{(t-1)}(i, j) + (1 - \alpha) \cdot \Theta_{current}^{(t)}(i, j) \quad (13)$$

Table 2: Hardware configuration of edge devices.

Edge GPU	AI Performance	DRAM	Power
Jetson Orin Nano	67TOPS (INT8)	8GB 102GB/s	7-25W
Jetson AGX Orin	275TOPS (INT8)	64GB 204.8GB/s	15-60W

where $\Theta_{final}^{(t-1)}$ represents the smoothed feature map from the previous frame. To eliminate trailing artifacts under rapid target motion or abrupt direction changes, the smoothing factor $\alpha \in [0, 1)$ is not static; instead, it dynamically scales inversely with the target's predicted velocity metric $\|\bar{v}_{pred}\|$ and the bounding box Intersection-over-Union (IoU) ratio between consecutive frames:

$$\alpha = \alpha_0 \cdot \exp(-\lambda \cdot \|\bar{v}_{pred}\| \cdot (1 - \text{IoU})) \quad (14)$$

where α_0 and λ are scaling hyperparameters. When a target moves at high speed or undergoes sudden displacement (resulting in a low IoU), α rapidly approaches zero, forcing the locator to rapidly adapt the RoI mask to the current frame $\Theta_{current}^{(t)}$ and eliminating trailing drift. Conversely, for slow or static targets, a higher α effectively suppresses frame-to-frame feature fluctuations caused by quantization noise.

6 Implementation

We implemented DynaFilter in approximately 3,800 lines of Python 3.9, along with roughly 900 lines of C++ code for low-level codec bitstream processing. We deploy the system on the widely used NVIDIA Jetson edge devices. Table 2 shows the hardware configurations of the edge devices, running Ubuntu 20.04 LTS with NVIDIA JetPack 5.1.4.

The components at the cloud are implemented as a microservice architecture with RESTful APIs for interaction with edge devices. During training of the compressed domain feature mapper, the NN-based model is trained using the Adam optimizer with a learning rate of 0.001 and batch size of 64, requiring approximately 150 epochs to converge on an NVIDIA GeForce GTX 3080 GPU. The multimodal feature fusion filter is implemented as a Transformer-based encoder, including a positional encoding module and a multi-head attention mechanism. The filter generates a configuration in JSON format and sends it to the edge device with a payload size of only a few bytes, minimizing network overhead.

The JPEG/video codecs profiler integrates two distinct backends to support dual modalities. For static images, we utilized an extended version of libjpeg-turbo 3.0 [1] to process JPEG bitstreams incrementally. We modified the library internals to expose the quantized DCT coefficients directly after the entropy decoding stage. We integrated FFmpeg's libavcodec [49] for video streams with custom parsing hooks. Specifically, we intercept MVs and residuals directly from H.264/HEVC NAL units. To improve performance on edge devices, the RoI Locator includes SIMD optimizations, such

as NEON instructions on ARM architectures, for efficient threshold comparisons and bitmask operations. The component outputs RoI coordinates in pixel units, which are used for partial decompression and inference.

7 Performance Evaluation

We describe the experimental setup first, followed by a discussion of the experimental results.

7.1 Experiment Setup

We train RoI-based YOLOv8 [30] for object detection using the DOTA-v1.0 dataset. For object tracking, we integrate ByteTrack [62] into the RoI-based YOLOv8 and train it using the VisDrone 2019 dataset. Note that DynaFilter is model-agnostic and compatible with newer implementations, such as YOLOv11. Even with these, the fundamental trade-offs remain and thus the conclusions we draw from our experiments do not change, either. We use the following datasets:

- **DOTA-v1.0** [16, 53]: A large-scale dataset of aerial images containing 2,806 images across 15 object categories. We use this dataset for object detection.
- **VisDrone 2019** [66]: A dataset for object detection and tracking with drones, containing 261,908 video frames across 10 object categories. We use it for object tracking.

We compare DynaFilter applied to JPEG static imagery with the following baselines:

- **Full Inference**: The edge device fully decompresses each frame and runs inference on the entire image.
- **DeepCOD** [58]: An offloading framework that designs an encoder, trained in a data-driven manner, using compressive sensing, and then performs offloading.
- **LimitNet** [24]: An efficient offloading technique; the content-aware encoder prioritizes critical data based on image content and then performs offloading.
- **EFilter**: A variant of DynaFilter without RoI positioning, where the edge performs compressed-domain feature extraction but processes all features.

For video streaming, instead, we consider two baselines over H.264/HEVC streams:

- **Full Streaming**: transmitting the complete original bitstream to the cloud for full decompression and inference.
- **DIR**: the complete *decompress-infer-recompress* pipeline we outlined in the Introduction.

In the following, we first evaluate DynaFilter performance with JPEG imagery data. Next, we deep dive into the performance with video data, and conclude by discussing figures related to system overhead.

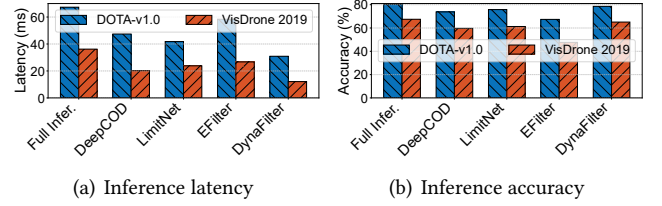


Figure 14: Inference latency and accuracy comparison on two datasets. We set the JPEG quality factor to 80.

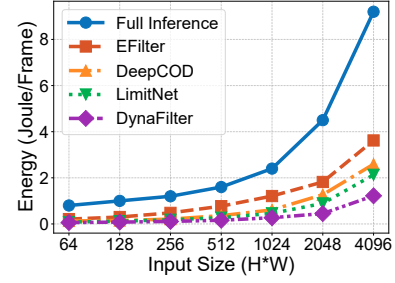


Figure 15: Comparison of energy consumption across different input sizes.

7.2 Accuracy, Latency, and Energy

Figure 14(a) shows that DynaFilter achieves a 2.2 $\times$ and 3.0 $\times$ speedup over Full Inference on DOTA-v1.0 and VisDrone 2019, respectively, and consistently outperforms DeepCOD, LimitNet, and EFilter by 1.6~2.2 $\times$. This efficiency stems from eliminating full decompression and focusing solely on RoIs.

Figure 14(b) shows that DynaFilter maintains performance within 1.8% on DOTA-v1.0 and 2.4% on VisDrone of Full Inference. It outperforms DeepCOD, which lacks RoI filtering, by 5% and LimitNet, which suffers from cloud-side decompression artifacts, by 3%. EFilter performs worst due to background noise interference.

Energy efficiency is critical for edge satellite computing. Figure 15 demonstrates that DynaFilter reduces energy consumption by 88.6% compared to Full Inference. By selectively processing only potential RoIs, it achieves 72.1% savings over EFilter. DeepCOD and LimitNet consume 2.3 $\times$ and 1.8 $\times$ more energy than DynaFilter, respectively, due to non-adaptive feature processing.

7.3 Dynamic Query Filtering

Figure 16 demonstrates the system's adaptability on DOTA-v1.0. DynaFilter achieves a 3.1 $\times$ ~7.7 $\times$ improvement in adaptation latency (Figure 16(a)), due to its compact cloud-driven configurations. It maintains high accuracy even at high query frequencies (Figure 16(b)) via robust multimodal feature fusion. Furthermore, inference latency remains significantly lower than the baselines (Figure 16(c)), and object switching latency is minimized (Figure 16(d)), confirming DynaFilter's effectiveness in handling data drift.

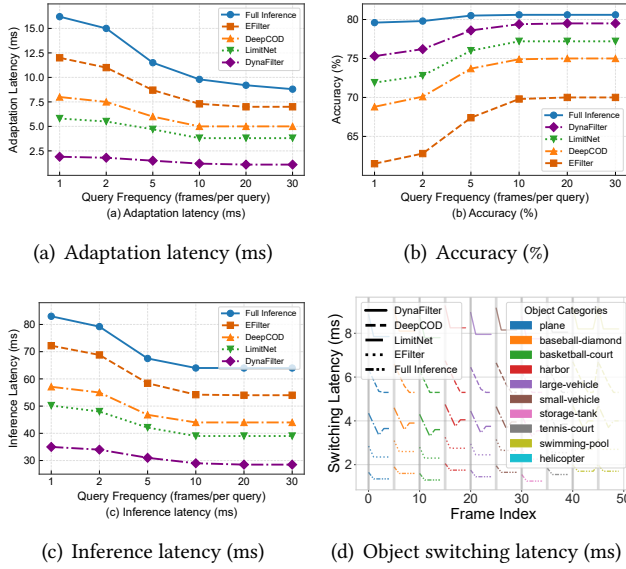


Figure 16: The impact of dynamic query filtering on the DOTA-v1.0 dataset.

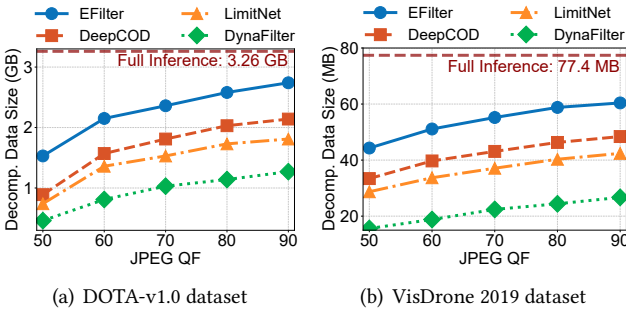


Figure 17: Decompressed data size with different JPEG quality factors (QFs) across two datasets.

7.4 JPEG Quality and RoI Locator

We use decompressed data size for validating bandwidth efficiency across varying JPEG quality factors (QFs). As shown in Figure 17, DynaFilter reduces decompressed data size by $2.6\times \sim 7.1\times$ on DOTA-v1.0 and $2.9\times \sim 5.0\times$ on VisDrone compared to Full Inference. Lower QFs actually enhance filtering effectiveness as coding artifacts distinguish background from RoIs. DynaFilter outperforms DeepCOD ($1.8\times \sim 2.5\times$) and LimitNet ($1.6\times \sim 2.2\times$) by avoiding the transmission of irrelevant compressed features or full frames.

Figure 18 shows the performance of the RoI Locator across different JPEG QFs. The results show that the locator maintains consistent performance across the entire quality spectrum, with only minor variations in accuracy and memory overhead, while incurring negligible latency and energy consumption compared to Full Inference. While lower QF settings

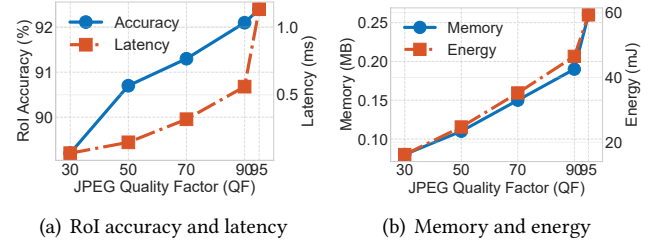


Figure 18: Performance of RoI Locator across different JPEG quality factors.

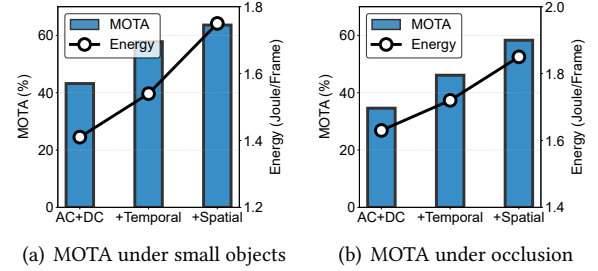


Figure 19: Performance comparison of multimodal feature fusion filter with different feature modality.

result in more aggressive quantization, the fundamental patterns that distinguish object regions from background are preserved, enabling the analyzer to effectively identify RoIs.

7.5 Multimodal Feature Fusion Filter

We evaluate the contributions of each feature modality to multi-object tracking accuracy (MOTA) for the object tracking task, with special attention to challenging scenarios including occlusion and small object tracking.

In Figure 19, for objects smaller than 32×32 pixels, compressed-domain features alone achieve only 43.2% MOTA due to weak and easily confused signatures in the compressed-domain. In comparison, temporal context features provide a dramatic 14.6% improvement by capturing motion signatures that persist despite limited spatial footprint. Spatial location modeling contributes an additional 5.8% MOTA improvement through Kalman filtering, which maintains trajectory consistency during brief visibility gaps.

For occlusion, compressed-domain features alone struggle during occlusion, with MOTA dropping to 34.6% as objects temporarily disappear. Temporal context features provide a 11.5% MOTA improvement by identifying objects through their motion history, while spatial location modeling contributes 12.2% by predicting object positions during occlusion periods. For severe occlusion, the system increases emphasis on temporal continuity, while during motion blur, it shifts to spatial consistency.

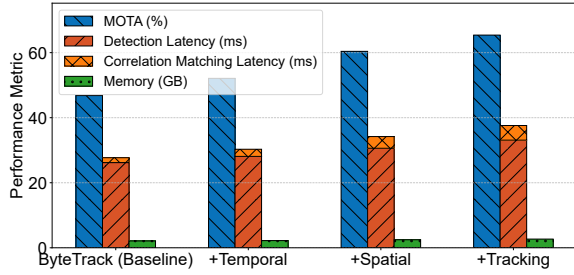


Figure 20: Performance of RoI-based object tracking with different enhancement strategies.

7.6 Enhancement Strategy

We evaluate the impact of different strategies on the performance of the object tracking tasks, with comprehensive metrics including MOTA, tracking latency accounting for detection latency and correlation matching latency, and memory usage. The temporal-spatial feature enhancement strategy significantly improves tracking robustness by incorporating motion intensity analysis and spatial prediction. The tracking association enhancement strategy addresses the critical challenges of target occlusion and ID switching.

As shown in Figure 20, the combination of the two enhancement strategies achieves significant performance improvements across all metrics. DynaFilter achieves a MOTA of 65.4%, while the ByteTrack [62] achieves a MOTA of 46.8%. Meanwhile, the tracking latency and memory usage only increase by 26.3% and 21.6%, respectively.

7.7 Video Streaming

Figure 21 shows that DIR suffers high latency, about 100ms, due to the recompression bottleneck. In contrast, DynaFilter achieves 13.5ms latency, corresponding to a 7.4 $\times$ speedup over DIR, by leveraging bitstream features. DynaFilter maintains 67.8% MOTA, matching Full Streaming (68.5%) and outperforming again DIR (62.0%).

For energy consumption, DynaFilter consumes just 0.4 J/frame, which is 83.3% lower than DIR (2.4 J/frame), and achieves 92% bandwidth savings by transmitting only query-relevant partial bitstreams. Furthermore, as shown in Appendix A, DynaFilter remains effective under dense RoI scenarios because it selectively decodes only the relevant compressed blocks, completely bypassing the expensive video recompression bottleneck of the standard DIR pipeline.

7.8 System Overhead

We use 1080p video input for breaking down the different latency factors on Jetson Orin Nano. As shown in Figure 22, DynaFilter’s processing pipeline demonstrates exceptional time efficiency. Specifically, inference remains the most computationally intensive component (52% for DOTA-v1.0, 64%

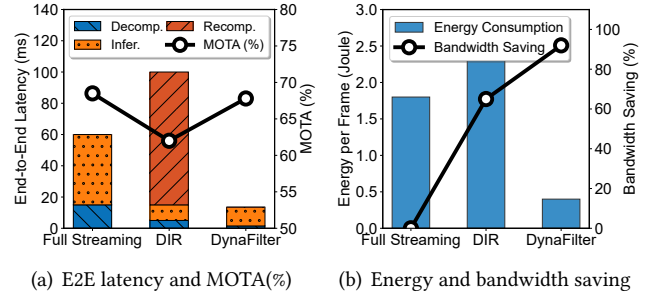


Figure 21: Performance comparison of JPEG/video codecs profiler for video streaming.

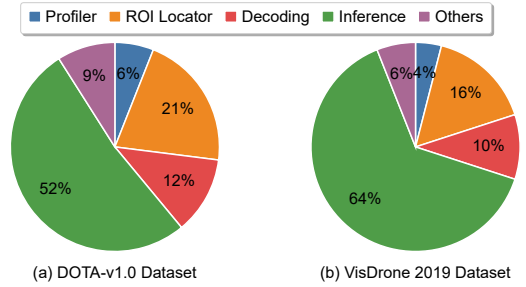


Figure 22: Execution latency of each component.

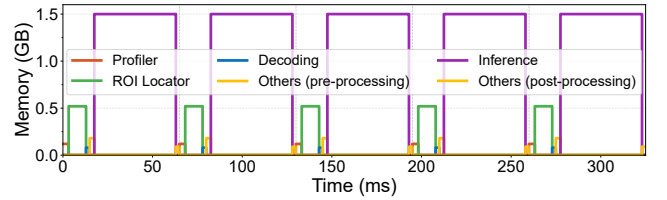


Figure 23: Memory usage of the DynaFilter pipeline.

for VisDrone), while our innovative compressed-domain filtering components contribute minimally to overall latency. The profiler accounts for only 4~6% of processing time, and the RoI Locator adds just 16~21%, demonstrating the lightweight nature of our compressed-domain analysis.

As for memory consumption, Figure 23 reveals that DynaFilter achieves a peak usage of only 2.49GB, representing a 36% reduction compared to full inference’s 3.87GB. By processing only the identified RoIs, DynaFilter reduces the memory required for the inference stage. The consistent memory profile of DynaFilter (± 0.12 GB across different scene types) ensures system robustness.

8 Related work

Our work intersects different areas, whose existing literature we compare with our efforts next.

Computational offloading. Recent advances focus on computational offloading strategies [25, 26, 58, 65] to address resource constraints of edge devices. DeepCOD [58] and LimitNet [24] design encoders with different architectures to

achieve efficient offloading. Elf [61] accelerates high-resolution visual processing by partitioning video frames and offloading them to multiple servers in parallel. AgileNN [26] and E3 [36] uses explainable AI to achieve on-device inference. However, these works cannot address data drift. DynaFilter introduces a cloud driven-edge inference paradigm, enabling edge devices to perform filtering directly in the compressed-domain without full decompression.

Compressed-domain processing. Recent advances focus on compressed-domain processing using deep learning. For example, CoVA [27] splits video frames between compressed and pixel domains to address decoding bottlenecks. MFCD-Net [4] uses imitation learning to mimic raw domain networks in the compressed-domain for action recognition. CD-VSR [11] enhances video quality directly in the compressed-domain. EBM [54] optimizes enhancement bias towards compressed-domain to improve the quality of compressed images. However, these works treat compression as a bottleneck, rather than a computational medium. DynaFilter fundamentally differs by establishing a precise mapping between semantic parameters and compressed-domain features.

Dynamic filtering. Recent advances explore various approaches for dynamic filtering. EfficientVIS [52] uses temporal information to improve segmentation accuracy, but cannot adapt to changing queries in the cloud. Meta-Filter [56] predicts position- and channel-specific filter weights for few-shot learning. SQ-MG [60] presented an object detection framework with query-support mutual guidance, but operates exclusively in the pixel domain. Reducto [35] implements frame-level filtering using low-level video features, but focuses on temporal filtering. However, these works treat filters as static, meaning they need to be retrained or reanalyzed when queries change. In contrast, DynaFilter enables the cloud to dynamically generate new filter configurations based on changing queries.

9 Discussion

We articulate limitations of our work and possible extensions.

Cold-start mitigation strategy. A key consideration for DynaFilter is its execution behavior in scenarios where historical tracking or feature statistics are initially unavailable. To mitigate this cold-start problem, we introduce a dedicated strategy detailing two explicit initialization methods: (1) a statistical warm-start, which initializes the system using generalized base statistics extracted from existing baseline benchmarks, and (2) a temporary cloud-driven calibration, where the cloud temporarily requests full decompression of the first few initial frames. It enables the edge device to rapidly calibrate localized baseline statistics, with the guarantee that the filtering performance will rapidly converge to optimal theoretical bounds post-initialization.

Extension to modern and neural codecs. The landscape of image and video compression is continuously evolving toward modern formats (e.g., WebP, AVIF, HEIC) and emerging neural video compression standards. Our core insight, which leverages the strong correlation between compression syntax and high-level semantics, remains fundamentally applicable to these standards. For block-based formats like AVIF, DynaFilter can be adapted to profile its specific partition structures and frequency transforms. For neural video codecs that compress data into latent feature representations, the pixel-domain DCT extraction module can be bypassed entirely in neural configurations. Instead, the compressed-domain filtering logic can be mapped directly onto the latent feature space. This configuration fully retains the core cloud-driven paradigm of DynaFilter, where the cloud dynamically updates parameters to allow the edge to skip complete latent-to-pixel decoding.

Adaptive filter generation. The filter configurations of DynaFilter rely on historical data. We can integrate online learning [23] to enable adaptation based on immediate feedback from edge devices. One promising approach involves a federated meta-learning [20]. For instance, integrating model-agnostic meta-learning [21] could enable the cloud to rapidly adapt filter configurations to new object categories. Additionally, developing neural architecture search (NAS) [7] specifically for filter generation could optimize the trade-off between accuracy and computational overhead.

Cross-modal extension. The cloud driven-edge execution paradigm of DynaFilter is fundamentally applicable to any bandwidth-constrained multimodal application, such as underwater exploration or remote sensing. Our principle of mapping high-level semantics to low-level compression features could process diverse data types with consistent efficiency by abstracting them into a unified semantic feature space. We can develop cross-modal learning [41] that enables knowledge transfer between different data types. For instance, contrastive learning [12, 31] could be employed to learn unified representations across modalities, allowing the system to leverage complementary information from different sensor types to improve overall accuracy.

10 Conclusion

This paper proposes DynaFilter, a cloud-driven technique for dynamic filtering. By accurately mapping low-level compressed-domain features with high-level semantics, DynaFilter can identify RoI without full decompression. Extensive experiments show that DynaFilter delivers significant improvement over state-of-the-art baselines in decompressed data size, energy consumption and inference latency.

Acknowledgment

We thank the anonymous reviewers and shepherd for the comments and feedback. This work is partly supported by the Swedish Foundation for Strategic Research (SSF).

References

- [1] 2025. libjpeg-turbo. <https://github.com/libjpeg-turbo/libjpeg-turbo> Version 3.1.1.
- [2] Abien Fred Agarap. 2018. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375* (2018).
- [3] Amin Banitalebi-Dehkordi, Naveen Vedula, Jian Pei, Fei Xia, Lanjun Wang, and Yong Zhang. 2021. Auto-split: A general framework of collaborative edge-cloud AI. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2543–2553.
- [4] Barak Battash, Haim Barad, Hanlin Tang, and Amit Bleiweiss. 2020. Mimic the raw domain: Accelerating action recognition in the compressed domain. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 684–685.
- [5] J. Berck, S. Pedrotty, J. Wisbiski, M. Alexander, K. Knesek, R. May, J. Nunez-Kearny, R. Stoddard, T. Kirven, R. Onsay, et al. 2025. Launch Fast Learn Fast: Lessons Learned from the R5-S2 and R5-S4 Missions. In *Proc. Small Satellite Conf.* 1–20.
- [6] Syed Muhammad Ammar Hassan Bukhari, Kashif Bilal, Aiman Erbad, Amr Mohamed, and Mohsen Guizani. 2023. Video transcoding at the edge: cost and feasibility perspective. *Cluster Computing* 26, 1 (2023), 157–180.
- [7] Han Cai, Ligeng Zhu, and Song Han. 2018. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332* (2018).
- [8] Robby Candra, Sarifuddin Madenda, Sunny Arief Sudiro, and Muhammad Subali. 2017. The implementation of an efficient zigzag scan. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)* 9, 2 (2017), 95–98.
- [9] Shih-Fu Chang and David G. Messerschmitt. 1995. Manipulation and compositing of MC-DCT compressed video. *IEEE Journal on selected areas in communications* 13, 1 (1995), 1–11.
- [10] Bo Chen, Ali Bakhshi, Gustavo Batista, Brian Ng, and Tat-Jun Chin. 2022. Update compression for deep neural networks on the edge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3076–3086.
- [11] Peilin Chen, Wenhan Yang, Meng Wang, Long Sun, Kangkang Hu, and Shiqi Wang. 2021. Compressed domain deep video super-resolution. *IEEE Transactions on Image Processing* 30 (2021), 7156–7169.
- [12] Ching-Yao Chuang, Joshua Robinson, Yen-Chen Lin, Antonio Torralba, and Stefanie Jegelka. 2020. Debiased contrastive learning. *Advances in neural information processing systems* 33 (2020), 8765–8775.
- [13] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. 2020. Multi-head attention: Collaborate instead of concatenate. *arXiv preprint arXiv:2006.16362* (2020).
- [14] B. Denby and B. Lucia. 2019. Orbital edge computing: Machine inference in space. *IEEE Computer Architecture Letters* 18, 1 (2019), 59–62.
- [15] B. Denby and B. Lucia. 2020. Orbital edge computing: Nanosatellite constellations as a new class of computer system. In *Proc. Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 939–954.
- [16] Jian Ding, Nan Xue, Gui-Song Xia, Xiang Bai, Wen Yang, Michael Ying Yang, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, et al. 2021. Object detection in aerial images: A large-scale benchmark and challenges. *IEEE transactions on pattern analysis and machine intelligence* 44, 11 (2021), 7778–7796.
- [17] Pradeep Dogga, Sandip Chakraborty, Subrata Mitra, and Ravi Netravali. 2019. Edge-based transcoding for adaptive live video streaming. In *2nd USENIX workshop on hot topics in edge computing (HotEdge 19)*.
- [18] Kuntai Du, Yihua Cheng, Peder Olsen, Shadi Noghbi, and Junchen Jiang. 2025. Earth+: On-board satellite imagery compression leveraging historical earth observations. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*. 361–376.
- [19] Kuntai Du, Qizheng Zhang, Anton Arapin, Haodong Wang, Zhengxu Xia, and Junchen Jiang. 2022. Accmpeg: Optimizing video encoding for video analytics. *arXiv preprint arXiv:2204.12534* (2022).
- [20] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. 2020. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in neural information processing systems* 33 (2020), 3557–3568.
- [21] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*. PMLR, 1126–1135.
- [22] Alireza Furutanpey, Qiyang Zhang, Philipp Raith, Tobias Pfandzelter, Shangguang Wang, and Shahram Dustdar. 2025. Fool: Addressing the downlink bottleneck in satellite computing with neural feature compression. *IEEE Transactions on Mobile Computing* (2025).
- [23] Steven CH Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. 2021. Online learning: A comprehensive survey. *Neurocomputing* 459 (2021), 249–289.
- [24] Ali Hojjat, Janek Haberger, Tayyaba Zainab, and Olaf Landsiedel. 2024. Limitnet: Progressive, content-aware image offloading for extremely weak devices & networks. In *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services*. 519–533.
- [25] Pan Hu, Junha Im, Zain Asgar, and Sachin Katti. 2020. Starfish: Resilient image compression for AIoT cameras. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*. 395–408.
- [26] Kai Huang and Wei Gao. 2022. Real-time neural network inference on extremely weak devices: agile offloading with explainable AI. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*. 200–213.
- [27] 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.
- [28] Eddy Ilg, Nikolaus Mayer, Tommo Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. 2017. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2462–2470.
- [29] Wonjoon Jin, Qi Dai, Chong Luo, Seung-Hwan Baek, and Sunghyun Cho. 2025. Flovd: Optical flow meets video diffusion model for enhanced camera-controlled video synthesis. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 2040–2049.
- [30] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. 2023. *Ultralytics YOLOv8*. <https://github.com/ultralytics/ultralytics>
- [31] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *Advances in neural information processing systems* 33 (2020), 18661–18673.
- [32] Stefanos Laskaridis, Stylianos I Venieris, Mario Almeida, Ilias Leontiadis, and Nicholas D Lane. 2020. SPINN: Synergistic progressive inference of neural networks over device and cloud. In *Proceedings of the 26th annual international conference on mobile computing and networking*. 1–15.
- [33] Min Li, Yu Li, Ye Tian, Li Jiang, and Qiang Xu. 2021. AppealNet: An efficient and highly-accurate edge/cloud collaborative architecture for DNN inference. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 409–414.

- [34] Qing Li, Shangguang Wang, Xiao Ma, Ao Zhou, Yue Wang, Gang Huang, and Xuanzhe Liu. 2024. Battery-aware energy optimization for satellite edge computing. *IEEE Transactions on Services Computing* 17, 2 (2024), 437–451.
- [35] 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*. 359–376.
- [36] Changyao Lin, Zhenming Chen, Ziyang Zhang, and Jie Liu. 2025. E3: Early Exiting with Explainable AI for Real-Time and Accurate DNN Inference in Edge-Cloud Systems. In *Proceedings of the 23rd ACM Conference on Embedded Networked Sensor Systems*. 385–397.
- [37] Sidi Lu, Xin Yuan, and Weisong Shi. 2020. Edge compression: An integrated framework for compressive imaging processing on cavs. In *2020 IEEE/ACM symposium on edge computing (SEC)*. IEEE, 125–138.
- [38] Ankur Mallick, Kevin Hsieh, Behnaz Arzani, and Gauri Joshi. 2022. Matchmaker: Data drift mitigation in machine learning for large-scale systems. *Proceedings of Machine Learning and Systems* 4 (2022), 77–94.
- [39] Yoshitomo Matsubara, Ruihan Yang, Marco Levorato, and Stephan Mandt. 2022. Supervised compression for resource-constrained edge computing systems. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 2685–2695.
- [40] S. Narayana, R. V. Prasad, V. Rao, L. Mottola, and T. V. Prabhakar. 2020. Hummingbird: energy efficient GPS receiver for small satellites. In *Proc. Int. Conf. on Mobile Computing and Networking (MobiCom)*.
- [41] Xiaomin Ouyang, Xian Shuai, Jiayu Zhou, Ivy Wang Shi, Zhiyuan Xie, Guoliang Xing, and Jianwei Huang. 2022. Cosmo: contrastive fusion learning with small data for multimodal human activity recognition. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*. 324–337.
- [42] Chuan Qin, Jinchuan Hu, Fengyong Li, Zhenxing Qian, and Xinpeng Zhang. 2022. JPEG image encryption with adaptive DC coefficient prediction and RS pair permutation. *IEEE Transactions on Multimedia* 25 (2022), 2528–2542.
- [43] SatNOGS. 2025. Open Source Global Network of Satellite Ground-stations. <https://www.satnogs.org>
- [44] Jayanth Shenoy, Om Chabra, Tusher Chakraborty, Suraj Jog, Deepak Vasisht, and Ranveer Chandra. 2024. CosMAC: Constellation-Aware Medium Access and Scheduling for IoT Satellites. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 724–739.
- [45] Sudipta Saha Shubha and Haiying Shen. 2023. Adainf: Data drift adaptive scheduling for accurate and slo-guaranteed multiple-model inference serving at edge servers. In *Proceedings of the ACM SIGCOMM 2023 Conference*. 473–485.
- [46] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. 2002. The JPEG 2000 still image compression standard. *IEEE Signal processing magazine* 18, 5 (2002), 36–58.
- [47] Bill Tao, Om Chabra, Ishani Janveja, Indranil Gupta, and Deepak Vasisht. 2024. Known knowns and unknowns: Near-realtime earth observation via query bifurcation in serval. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 809–824.
- [48] Bill Tao, Maleeha Masood, Indranil Gupta, and Deepak Vasisht. 2023. Transmitting, fast and slow: Scheduling satellite traffic through space and time. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–15.
- [49] Suramya Tomar. 2006. Converting video formats with FFmpeg. *Linux journal* 2006, 146 (2006), 10.
- [50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [51] Gregory K Wallace. 1991. The JPEG still picture compression standard. *Commun. ACM* 34, 4 (1991), 30–44.
- [52] Jialian Wu, Sudhir Yarram, Hui Liang, Tian Lan, Junsong Yuan, Jayan Eledath, and Gerard Medioni. 2022. Efficient video instance segmentation via tracklet query and proposal. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 959–968.
- [53] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. 2018. DOTA: A large-scale dataset for object detection in aerial images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3974–3983.
- [54] Qunliang Xing, Mai Xu, Shengxi Li, Xin Deng, Meisong Zheng, Huaida Liu, and Ying Chen. 2024. Enhancing quality of compressed images by mitigating enhancement bias towards compression domain. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 25501–25511.
- [55] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. 2015. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853* (2015).
- [56] Chengming Xu, Yanwei Fu, Chen Liu, Chengjie Wang, Jilin Li, Feiyue Huang, Li Zhang, and Xiangyang Xue. 2021. Learning dynamic alignment via meta-filter for few-shot learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5182–5191.
- [57] A. E. Yaacoub, T. Voigt, P. Ruemmer, and L. Mottola. 2025. Fault Tolerance in Space with Heterogeneous Hardware: Experiences from a 68-day CubeSat Deployment in LEO. In *Proc. Int. Conf. on Embedded Wireless Systems and Networks (EWSN)*. 1–12.
- [58] Shuochao Yao, Jinyang Li, Dongxin Liu, Tianshi Wang, Shengzhong Liu, Huajie Shao, and Tarek Abdelzaher. 2020. Deep compressive offloading: Speeding up neural network inference by trading edge computation for network latency. In *Proceedings of the 18th conference on embedded networked sensor systems*. 476–488.
- [59] Chunqiang Yu, Shichao Cheng, Xianquan Zhang, Xinpeng Zhang, and Zhenjun Tang. 2024. Reversible data hiding in shared JPEG images. *ACM Transactions on Multimedia Computing, Communications and Applications* 20, 12 (2024), 1–24.
- [60] Lu Zhang, Shuigeng Zhou, Jihong Guan, and Ji Zhang. 2021. Accurate few-shot object detection with support-query mutual guidance and hybrid loss. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*. 14424–14432.
- [61] 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.
- [62] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. 2022. Bytetrack: Multi-object tracking by associating every detection box. In *European conference on computer vision*. Springer, 1–21.
- [63] Ziyuan Zhang, Han Qiu, Maosen Zhang, Jun Liu, Bin Chen, Tianwei Zhang, and Hewu Li. 2024. Cosmic: Compress satellite images efficiently via diffusion compensation. *Proceedings of the NeurIPS, Vancouver, BC, Canada (2024)*, 10–15.
- [64] Ziyang Zhang, Yang Zhao, Ming-Ching Chang, Changyao Lin, and Jie Liu. 2025. E4: Energy-Efficient DNN Inference for Edge Video Analytics Via Early Exiting and DVFS. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 1165–1173.

- [65] Ziyang Zhang, Yang Zhao, Huan Li, Changyao Lin, and Jie Liu. 2024. DVFO: Learning-based DVFS for energy-efficient edge-cloud collaborative inference. *IEEE Transactions on Mobile Computing* 23, 10 (2024), 9042–9059.
- [66] 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.

A Dense RoI Evaluation

In this section, we present the comprehensive benchmarks of DynaFilter across two datasets (i.e., DOTA-v1.0 and VisDrone 2019) for dense RoI scenarios. We evaluate latency and energy consumption across scalable RoI spatial fractions (10%~90%) on both target edge devices specified in Table 2 (i.e., the power-constrained NVIDIA Jetson Orin Nano and the high-performance NVIDIA Jetson AGX Orin).

As shown in Figure 24, on the Jetson Orin Nano, the standard Decompress-Infer-Recompress (DIR) pipeline suffers from extreme latency (77.2 ms~118.8 ms) and energy drain (1.95 J~2.71 J per frame). This stems from Orin Nano’s hardware architecture, which lacks a dedicated NVENC encoder block, forcing DIR to perform CPU-bound software re-encoding. On the Jetson AGX Orin, while dedicated NVENC hardware acceleration reduces DIR’s latency to 16.0 ms~30.4 ms, the full decode-encode loop still incurs substantial computational overhead.

Across both devices, DynaFilter maintains a clear performance advantage across all RoI spatial fractions. By performing direct bitstream profiling and selectively decoding only query-relevant blocks, DynaFilter completely bypasses the recompression stage. On Jetson Orin Nano, DynaFilter achieves a $2.5\times\sim 5.6\times$ latency speedup and $2.5\times\sim 5.3\times$ energy reduction over DIR. Even on the highly capable AGX Orin, DynaFilter achieves a $2.7\times\sim 4.7\times$ speedup and $2.3\times\sim 4.3\times$ energy savings. This confirms that DynaFilter is hardware-agnostic and resilient to dense RoI.

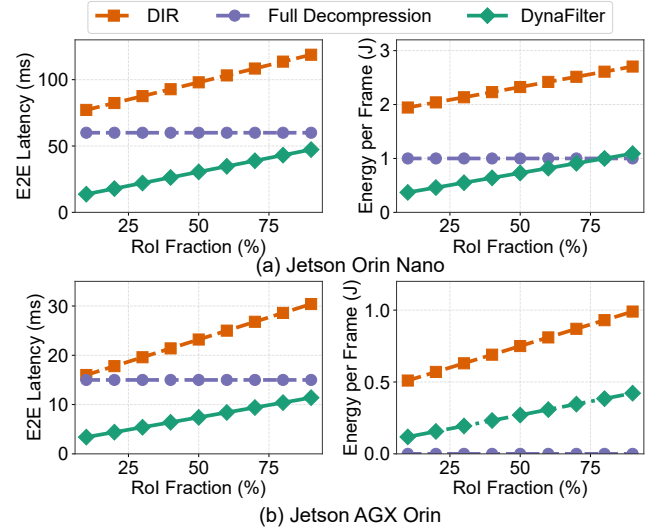


Figure 24: Performance scaling under varying RoI spatial fractions (10%~90%) on (a) Jetson Orin Nano and (b) Jetson AGX Orin.