USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

From Bits to Tokens: Knowledge-Driven Generative Communication of Multimodal Data

Xingyu Chen, *University of California San Diego;* Zihao Feng, *University of Southern California;* Wuqiong Zhao, *University of California San Diego;* Jianrong Ding, *Chinese University of Hong Kong;* Ke Sun, *University of Michigan Ann Arbor;* Xinyu Zhang, *University of California San Diego*

https://www.usenix.org/conference/nsdi26/presentation/chen-xingyu

This paper is included in the Proceedings of the 23rd USENIX Symposium on Networked Systems Design and Implementation.

May 4–6, 2026 • Renton, WA, USA

ISBN 978-1-939133-54-0

Open access to the Proceedings of the 23rd USENIX Symposium on Networked Systems Design and Implementation is sponsored by

جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology

From Bits to Tokens: Knowledge-Driven Generative Communication of Multimodal Data

Xingyu Chen[†], Zihao Feng[‡], Wuqiong Zhao[†], Jianrong Ding[§], Ke Sun[¶], Xinyu Zhang[†]
[†]*University of California San Diego,* [‡]*University of Southern California,*
[§]*Chinese University of Hong Kong,* [¶]*University of Michigan Ann Arbor*

Abstract

Classical communication systems strive for bit-by-bit reconstruction, yet this objective often misaligns with downstream application tasks, such as perception and decision-making with sensor data. This mismatch is amplified in wireless settings, where packet losses and channel dynamics make strict bit-fidelity both costly and fragile. In this paper, we introduce Knowledge-Driven Communication (KDC), a framework that transmits semantic knowledge rather than raw bits by leveraging pretrained knowledge bases. KDC features a task-aware transmitter, which uses multimodal foundation models to abstract source data into tokenized embeddings and prioritize semantically critical content, enabling zero-shot adaptation without task-specific retraining. On the receiver side, KDC employs pretrained knowledge bases and incrementally updated context to reconstruct task-relevant information, enabling graceful degradation even under data loss. We implement a full KDC prototype and evaluate it over diverse data modalities and wireless networks. KDC operates as an application-layer codec wrapper and receiver-side restoration module, fully compatible with existing wireless communication protocols and source/channel coding mechanisms. Experiments show that KDC consistently outperforms state-of-the-art codecs and learned baselines, achieving high task accuracy with a fraction of the transmitted data, while maintaining robustness under challenging wireless conditions.

1 Introduction

The modern Internet of Things (IoT) is characterized by a torrent of multimodal sensor data, powering applications from cooperative automotive perception [62, 65] and video surveillance [58] to volumetric holoportation [10] and smart transportation systems. These systems are no longer passive data producers but active decision-makers whose efficacy depends on a continuous, real-time understanding of their environment. However, the communication networks that underpin this evolution, particularly wireless links like 5G, WiFi, and LoRaWAN, are designed around a bit-level fidelity paradigm [57]. While Shannon's theory establishes fundamental capacity limits, practical systems built on this paradigm face a mismatch: the network's objective (minimizing bit error rate) diverges from the application's objective (maximizing task or decision success) [23].

This challenge is particularly pronounced in the dynamic wireless medium, where factors like interference and channel fading frequently lead to packet losses. The conventional retransmission-based protocols can recover the packets, yet at the cost of end-to-end latency. However, the strict bit-level accuracy is not always a prerequisite for conveying the semantic meaning of sensor data. Analogous to human conversation, even a partially received or corrupted message can often be sufficient for comprehension and decision making [7]. To leverage this observation, recent research has explored two primary pathways. The first replaces classical communication signal processing blocks with trainable, data-driven machine learning models [2, 24, 25, 47]. These systems, however, remain tethered to the bit-reconstruction paradigm, as their loss functions are defined by reconstruction accuracy. They learn to combat channel distortions without comprehending the semantic significance of the message content. The second approach, known as semantic or goal-oriented communication, optimizes for end-to-end task performance rather than bit accuracy [8, 22, 23, 64, 68]. These systems are mostly trained for joint source and channel coding, essentially optimizing task-specific data compression and error protection with limited interpretation of the data itself. In addition, they are typically trained for a specific task [24, 64] and lack the ability to adapt based on changing task relevance.

Addressing these limitations calls for a different design approach. Ideally, the transmitter should be able to represent and convey essential semantic meanings rather than merely compressing data, while the receiver should autonomously and proactively interpret that meaning rather than passively recovering data bits. This would not only improve communication efficiency and resilience but also grant the system the capacity for graceful degradation, where the receiver can incrementally build understanding and even "early exit" to reduce latency once a task is satisfied with partial information.

In this paper, we approach this vision through KDC, a novel knowledge-driven communication framework for semantic sensor data delivery. Rather than proposing new channel coding or physical-layer mechanisms, KDC operates as a systems and networking approach for task-aware data delivery that complements standard source coding and existing link/PHY-layer protocols. KDC is grounded on two fundamental principles. First, both the transmitter and receiver should be equipped with a shared knowledge base (KB)—typically a foundation model, pretrained offline. With this KB, the transmitter only needs to send the most essential semantic information, and the receiver can interpret it through its KB. This

principle facilitates zero-shot adaptation to new tasks without requiring task-specific end-to-end retraining. Secondly, the KDC receiver is a generative by nature and empowered with reasoning capabilities. It reasons about incoming partial information, extrapolates from its KB, and accumulates new knowledge as communication proceeds. In this way, *the more it receives, the less it needs to receive*. In addition to efficiency, the proactive reasoning also enables resilience against partial or corrupted information.

To materialize the principles, KDC addresses two major design challenges.

How can knowledge be effectively represented and transmitted for diverse sensor modalities? To transmit knowledge rather than raw data, we observe that multimodal foundation models [29, 51] have demonstrated strong capabilities in aligning heterogeneous sensor data with human-interpretable semantics. KDC equips both transmitter and receiver with a shared knowledge base (*SharedKB*, defined in Sec. 3.1), enabling raw sensor streams to be translated into a common semantic language of tokenized embeddings, which serve as the primary unit of transmission. This design can theoretically generalize across heterogeneous modalities, as supported by the *Platonic Representation Hypothesis* [26], which posits that sufficiently capable models learn modality-agnostic "Platonic" latent variables that capture task-relevant structure across diverse sensors. Furthermore, KDC incorporates a semantic-importance extraction method that estimates the relevance of each token for a user-defined task, contextualized by the *SharedKB*. This enables the transmitter to prioritize semantically critical content, while the receiver can form a task-relevant understanding even from a partial data stream. A key advantage of this design is its ability to perform zero-shot adaptation, *i.e.*, to adjust to new tasks without task-specific retraining [36, 51].

How can the receiver efficiently extract knowledge from partial or corrupted transmissions through reasoning? To build resilience against channel impairments, KDC introduces a knowledge-conditioned receiver architecture that intelligently reconstructs data by drawing upon both a pretrained *SharedKB* and a dynamically updated knowledge base (*PosteriorKB*) (defined in Sec. 3.1). When faced with incomplete information, the receiver can infer the missing content by referencing these knowledge stores, effectively reasoning about the data to restore its semantic integrity. Furthermore, KDC employs a progressive knowledge refinement mechanism, where the *PosteriorKB* is dynamically updated to incorporate new semantic insights gathered from ongoing communications. This incremental learning capability enables the system to progressively enhance transmission efficiency while remaining robust against channel dynamics [8].

We have realized the KDC framework through a meticulously engineered, end-to-end prototype. Our implementation encompasses a task-aware transmitter that uses offline foundation models for zero-shot semantic decomposition, modality-specific embedding pipelines for 4 representative data modalities (image, video, LiDAR, and radar data), and a novel codec wrapper that pragmatically integrates semantic prioritization with standard codecs. At the receiver side, we implement a lightweight, knowledge-conditioned restoration network. The entire system is optimized with parallel and pipelined processing to ensure real-time operation.

We have conducted a comprehensive evaluation of KDC's system-level efficacy across the 4 data modalities using real-world 5G, WiFi, and LoRa testbeds. The results demonstrate that KDC consistently and significantly outperforms state-of-the-art baselines in downstream task accuracy (*e.g.*, 0.55–0.65 Top-1 accuracy under 10% packet loss vs. < 0.2 for baselines, 0.85 Top-1 with only 10 kB data, and > 0.8 radar/LiDAR accuracy at 10% loss), particularly under wireless channel dynamics. Our micro-benchmarks confirm that KDC achieves similar or higher accuracy comparable to full-data transmission with only a fraction of the data (*e.g.*, 0.85 Top-1 accuracy with 10 kB data). Its zero-shot adaptability outperforms specialized models on diverse tasks without retraining. Furthermore, it sustains real-time performance (32 FPS on edge devices such as Jetson Orin, > 100 FPS on desktop GPUs).

Our contributions can be summarized as follows:

- We propose KDC, a novel knowledge-driven communication framework that leverages pretrained KBs for efficient transmission and a generative, reasoning-capable receiver for resilience against data loss.
- We design a task-aware transmitter that leverages multimodal foundation models to represent diverse sensor data in a unified semantic space and dynamically prioritizes content based on task relevance but without task-specific retraining.
- We design a knowledge-conditioned restoration framework for the receiver, which leverages both shared and dynamically updated local KBs to infer missing information.
- We implement a complete KDC prototype and conduct an extensive evaluation across four sensor modalities on three real-world wireless testbeds, demonstrating significant gains in task accuracy and transmission efficiency, and validating the system's real-time viability.

2 System Overview

Our KDC design aims to achieve three key objectives: *(i)* efficient task-aware data transmission, *(ii)* progressive transmission with adaptation to unstable wireless channels, including varying throughput and packet loss, and *(iii)* generalization across different data types and processing tasks.

Fig. 1 annotates the three pipeline stages in KDC and how knowledge bases (*SharedKB* and *PosteriorKB*, defined in Sec. 3.1) interact at the transmitter (Tx) and receiver (Rx). The **Task-aware Transmitter** (Sec. 3.1) embeds raw multimodal data into semantic tokens and prioritizes them by task relevance. These tokens are passed through the **Codec Wrapper** (Sec. 3.2), which integrates semantic importance into standard source coding codecs (*e.g.*, JPEG for images) to en-

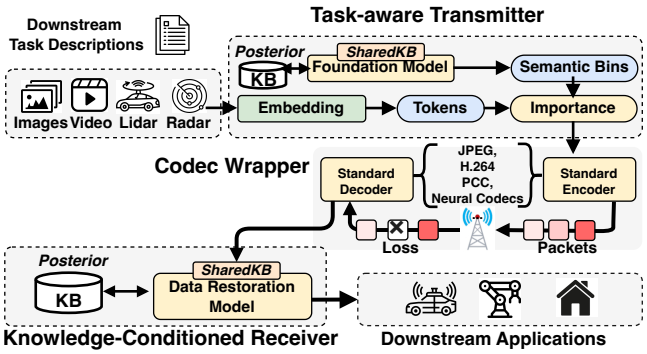

Figure 1: KDC system workflow.

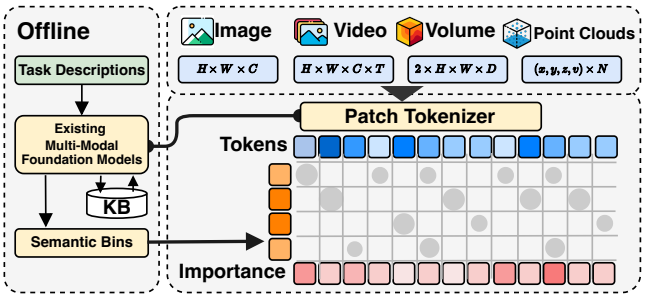

Figure 2: KDC supports multimodal data transmission by abstracting heterogeneous sensor data into a unified token space via existing multimodal foundation models.

sure compatibility. At the receiver, **Knowledge-Conditioned Restoration** (Sec. 3.3) leverages shared and posterior knowledge bases to recover corrupted or missing content, enabling resilience against channel dynamics.

Protocol-stack positioning. KDC operates entirely at the application layer, acting as a standard-compatible codec wrapper at the transmitter and a knowledge-conditioned restoration module at the receiver. The underlying PHY/MAC channel coding, retransmission mechanisms (*e.g.*, ARQ/HARQ), and transport protocols remain unchanged. KDC does not replace or modify any physical-layer reliability mechanisms. Instead, it adds a semantic prioritization and restoration layer on top of existing network stacks.

3 System Design

3.1 Knowledge Representation at Tx

3.1.1 Unified Embedding of Diverse Sensor Data Types

To enable knowledge-driven transmission of multi-modal data without task-specific retraining, KDC builds on a core observation: modern multimodal foundation models can align diverse sensor data with human-interpretable semantics. By leveraging such models, raw sensor inputs can be abstracted into unified tokenized embeddings that can then serve as *the primary unit of transmission*.

Tokenized Representation. Fig. 2 illustrates KDC's approach towards generalizable knowledge representation. Formally, let $X \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_k}$ represent raw sensor data, where k denotes the number of spatial, spectral, or temporal dimensions. For instance, X may represent a 2D grayscale image ($k = 2$), a 3D radar spectrum ($k = 3$), a point cloud, or even a time-varying tensor stream in continuous sensing applications.

KDC employs a pretrained encoder $f : \mathbb{R}^{d_1 \times \cdots \times d_k} \to \mathbb{R}^{T \times d}$ that maps X to a tokenized embedding sequence Z consisting of T tokens in a d-dimensional space:

$$Z = f(X) = [z_1, z_2, \ldots, z_T], \quad z_i \in \mathbb{R}^d. \tag{1}$$

We instantiate $f(\cdot)$ with modality-specific backbones pretrained on large-scale datasets. For images and video, we use a CLIP ViT-B/16 encoder [37], mapping each 16×16 patch into a $d = 512$ token. For 3D point clouds, we voxelize input and use sparse 3D convolutional encoders [13, 14] to produce region-level tokens. For radar volumes, we pretrain a transformer encoder [6] on synthetic and real radar datasets with CLIP-style contrastive learning, aligning radar tensors to text semantics. These encoders place heterogeneous sensor data into a unified embedding space shared by the Tx and Rx.

Knowledge Base (KB). We formalize the KB abstraction using two complementary modes.

Shared Knowledge Base (*SharedKB*) refers to the collection of pretrained multi-modal foundation model encoders and their aligned embedding spaces, jointly available to both Tx and Rx. Specifically, *SharedKB* includes the modality-specific encoders $f(\cdot)$ described above together with their associated text encoders, such that all token embeddings are projected into a common semantic space. This unified embedding space captures domain-general priors (*e.g.*, visual, spatial, and spectral semantics) that are fixed once the encoders are pretrained offline. Both Tx and Rx load the same set of encoders and embedding layers, ensuring that tokens produced by one side can be directly compared, scored, and decoded on the other. In this sense, *SharedKB* is not a single foundation model or a single encoder function, but the shared semantic backbone that aligns heterogeneous modalities into a reproducible and consistent representation. In deployment, Tx and Rx must load identical model identifiers and version numbers for SharedKB to ensure embedding consistency. We discuss practical safeguards (*e.g.*, version-ID metadata exchange) in Sec. A.

Posterior Knowledge Base (*PosteriorKB*). Unlike the static priors in *SharedKB*, *PosteriorKB* is an evolving, task-conditioned store that accumulates *session-specific* knowledge produced during communication. We implement PosteriorKB as a vector database indexed by semantic prompts (bins) and paired data embeddings. Formally, each record is a tuple $(e_{\text{text}}, e_{\text{data}}, \texttt{tags})$, where $e_{\text{text}} \in \mathbb{R}^d$ is the text embedding of a task/bin (*e.g.*, "pedestrian detection"), $e_{\text{data}} \in \mathbb{R}^d$ is a latent representation of associated sensor data (image/video token features, LiDAR voxel features, radar volume features), and `tags` annotate the modality for cross-modal retrieval. All embeddings are ℓ_2-normalized and indexed with a FAISS-backed vector store (via ChromaDB).

3.1.2 Semantic Selection (Bin-conditioned Importance)

A key innovation in KDC lies in the Tx-side *semantic selection mechanism*, which enables task-aware prioritization without retraining. Given a downstream task $\mathcal{T}$ (*e.g.*, object detection, gesture recognition, or event classification), KDC defines an importance scoring function $\alpha_i = \text{Importance}(z_i, \mathcal{T}, \textit{SharedKB})$ that quantifies how strongly each token z_i contributes to the task. Tokens with higher α_i values are prioritized for transmission:

$$Z' = \{z_i \in Z \mid \alpha_i \geq \tau\}, \tag{2}$$

where τ is a threshold controlling the aggressiveness of semantic compression. This formulation allows the Tx to focus on transmitting only semantically critical content, while the Rx leverages the *SharedKB* to reconstruct relevant task representations even under lossy channels.

Semantic Bin Extraction. The importance scoring process relies on decomposing high-level task descriptions into actionable *semantic bins*. Given a task description such as "autonomous driving" or "elder care monitoring", we leverage foundation models from the *SharedKB* during an offline preprocessing stage to extract relevant semantic entities and object categories. For example, "autonomous driving" is decomposed into semantic bins like "car", "pedestrian", "traffic light", and "road sign". This decomposition transforms abstract task descriptions into concrete semantic anchors that can be matched against sensor data embeddings.

Prompt for Semantic Bin Extraction

System: You are an expert in visual scene analysis. You have access to external knowledge retrieved from a database using "retrieve(·)". Use the retrieved context below as your primary source of truth. Your task is to identify concrete objects that cameras, LiDAR, or radar sensors would detect in real-world scenarios. List 4–8 specific objects, people, or physical elements that sensors would commonly observe in user's task scenario. Focus on visible, tangible items that appear in sensor data. Return only the object names, separated by commas.
User: Task scenario: "[TASK_DESCRIPTION]"
Assistant: [Generated semantic bins]

The foundation model (Qwen3-0.6B/1.7B/4B [67] in our implementation) is invoked only once during offline decomposition, with the sample prompt shown above, and the resulting bins are cached for the entire communication session. This eliminates the need for online LLM inference and ensures consistent semantic interpretation across multiple data transmissions. As shown in Fig. 3, without semantic decomposition, segmentation models misinterpret generic task prompts and produce incorrect importance maps, whereas decomposition via foundation models yields accurate task-aware importance estimation.

RAG-Enhanced Bin Extraction via *PosteriorKB*. To improve bin quality and coverage, we augment the offline decomposition with retrieval-augmented generation (RAG) using the *PosteriorKB*. The *PosteriorKB* contains: (i) curated label taxonomies and textual definitions from public datasets

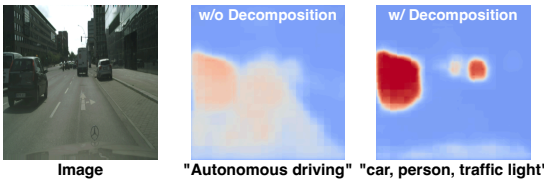

Figure 3: An example for image data transmission. The semantic segmentation fails to digest the general description of downstream tasks, resulting in an incorrect importance map. KDC uses foundation models to digest the task descriptions for segmentation models, leading to more accurate importance estimation.

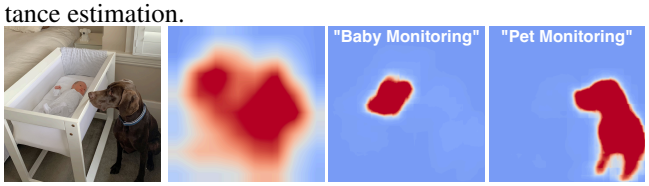

Figure 4: An example for image data transmission. The end-to-end trained model cannot adapt to downstream applications, while KDC achieves zero-shot task-aware prioritization.

(*e.g.*, Cityscapes categories for driving, human-pose labels for radar/LiDAR use cases), (ii) text snippets and exemplar images/volumes collected from domain documentation, and (iii) metadata (modality tag, environment, timestamp). We index these items as dense vectors using ChromaDB with FAISS-based indexing.

Given a task prompt, we embed it with the same text encoder used at runtime, retrieve top-k candidate entries (default k=32) from the *PosteriorKB* by cosine similarity, and pass the prompt plus retrieved snippets to the foundation model. The foundation model consolidates and canonicalizes them into the final semantic bin list (deduplicated synonyms, pruned long-tail items), which we cache for the session. This RAG-enhanced extraction produces more precise semantic bins that remain fixed throughout the communication session.

Importance Scoring and Map Construction. Given semantic bins $\mathcal{B}=\{b_j\}_{j=1}^{m}$, we compute text embeddings $t_j \in \mathbb{R}^d$ for each bin using modality-specific encoders (*e.g.*, CLIP text encoder for images/video; modality-specific text encoders for LiDAR/radar). Both token and bin embeddings are L2-normalized. For each token z_i, we compute cosine similarity:

$$s_{i,j} = \left\langle \frac{z_i}{\|z_i\|}, \frac{t_j}{\|t_j\|} \right\rangle \quad s_i = \max_j s_{i,j}. \tag{3}$$

Importance weights are obtained via temperature-normalization: $\alpha_i = \frac{\exp(s_i/T_{\text{attn}})}{\sum_u \exp(s_u/T_{\text{attn}})}$, where $T_{\text{attn}} = 0.07$ by default. Token selection follows Eq. (2) with threshold τ, which trades bandwidth for task accuracy.

Given $\{\alpha_i\}$, we generate modality-specific importance maps: a spatial mask for images/video (per-patch or per-region), a voxel-level heatmap for LiDAR, and a spectral/voxel mask for radar volumes. These maps drive transmission budgeting: units with larger α_i are allocated more tokens and finer quantization steps, while background units

receive fewer or no bits. Operationally, this bridges Sec. 3.1 and Sec. 3.2: the mask guides a pre-quantization step that retains more high-frequency coefficients in high-importance units before feeding standard codecs (JPEG/H.264/PCC), so that quantizers and entropy coders reinforce—rather than dilute—the task-aware priorities.

We instantiate Importance$(\cdot)$ with modality-aware relevance estimation. For images/video, we use CLIPSEG to propose region candidates; each region is encoded and compared (cosine similarity) with semantic-bin embeddings from the *SharedKB*, yielding per-region α_i. For LiDAR and radar, inputs are partitioned into units (voxels or point groups), unit embeddings are extracted (e.g., sparse convolution for LiDAR and a lightweight 3D encoder for radar), and cosine similarity to *SharedKB* bin embeddings produces $\{\alpha_i\}$. These scores are then converted into the importance maps described above.

The semantic bins serve as interpretable grounding units, effectively allocating transmission budget to semantically relevant regions while tokens aligned with 'background' receive fewer bits. As shown in Fig. 4, end-to-end trained codecs cannot flexibly adapt to different downstream tasks, since their importance allocation is implicitly tied to training datasets. In contrast, KDC explicitly grounds importance on semantic bins, thereby enabling zero-shot task-aware prioritization.

3.2 KDC Codec Wrapper

KDC is a general transmission paradigm designed to *interoperate* with existing source coding codecs, not replace them. This compatibility is essential because (i) standards such as H.264, JPEG, and PCC are tightly integrated with commodity hardware accelerators and network stacks–changing their internals would break interoperability; and (ii) many deployments face legacy constraints that mandate standard-compliant bitstreams. Consequently, unlike neural codecs [24, 39] that rewrite content into latent spaces (sacrificing interpretability and compatibility), KDC aims to preserve the standard data format while embedding task-aware prioritization.

Headers. KDC augments conventional codec headers with a lightweight semantic component. In addition to format-specific metadata (*e.g.*, quantization tables and encoding parameters), the header carries the word-level semantic bins generated during semantic selection (Sec. 3.1.2). This enables the receiver to interpret the task context of the incoming content without altering the standard bitstream. The header remains constant across a session (*i.e.*, throughout a sequence of transmissions) and is therefore sent once. For static scenes, adjacent frames can also reuse the same importance map, further reducing overhead. While the header provides semantic context for decoding, additional processing is required to ensure that these priorities survive the internal operations of legacy codecs.

Transform Domain Entropy Reduction. While semantic prioritization identifies task-relevant tokens, legacy codecs still apply fixed quantization tables that may override these priorities. To resolve this compatibility issue, KDC performs a lightweight pre-processing step that reshapes entropy *before* encoding in the appropriate transform domain (frequency domain for images/video/radar, spatial domain for LiDAR).

Specifically, the input is partitioned into processing units, with each unit embedded into a token, forming unit-token pairs. Tokens receive importance weights α_i using the cosine similarity and temperature-normalization process described in Sec. 3.1.2, producing a content-aware heatmap. We then apply the appropriate transform for the target codec and obtain transform coefficients (frequency coefficients for images/video/radar, spatial coefficients for LiDAR). Quantization follows the codec's predefined tables, but its granularity is modulated by the importance weights: high-importance regions preserve more detailed coefficients, while low-importance regions are aggressively quantized. Finally, coefficients are cast to appropriate bit depths, with negligible components in low-importance regions set to zero to maximize entropy reduction.

This design works with any codec that exposes a transform-quantization-entropy pipeline, allowing KDC to insert modality-specific pre-quantization steps that ensure codec-internal quantizers reinforce, rather than dilute, the semantic priorities. The specific implementation varies by modality as detailed below.

Modality-Specific Integration. KDC tailors its semantic prioritization strategy to the characteristics of each sensor modality. ***Images/Video:*** KDC prioritizes high-information regions using semantic content detection, applying finer quantization to salient areas while coarsely encoding background regions. The importance weighting ensures that semantically critical visual content receives preferential bit allocation. ***LiDAR Point Clouds:*** KDC maintains structural integrity by allocating more bits to high-density or object-boundary regions that contain semantic objects. The spatial partitioning strategy ensures geometric fidelity is preserved for task-relevant 3D structures while background regions use coarser representations. ***Radar Spectrograms:***KDC identifies task-relevant frequency components while suppressing background noise through entropy-aware filtering. The importance-driven approach ensures that motion patterns and reflective signatures critical to the task receive priority in the spectral domain.

Integration with Existing Codecs. After modality-specific preprocessing, each unit is converted back to the codec-native input domain (*e.g.*, pixel intensities for JPEG/H.264, voxel/magnitude grids for 3D spectra, or voxelized blocks for PCC), so the downstream encoder consumes a standard-conformant signal without any modification. Because low-importance regions now contain less information entropy, standard codecs *naturally* allocate fewer bits to them, achieving task-aware compression while remaining fully compatible with H.264/JPEG/PCC and existing hardware accelerators.

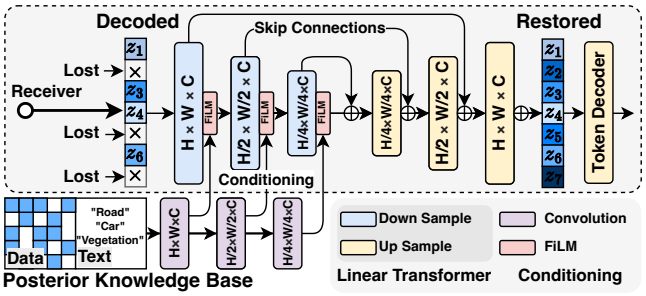

Figure 5: The restoration network at RX takes knowledge priors as conditioning to provide additional information for image restoration, hence restoring the data in terms of packet loss and resolution with satisfactory results.

3.3 Knowledge-Conditioned Restoration at Rx

In practical IoT sensor data delivery scenarios, the KDC Rx must contend with two primary challenges. The first lies in data loss due to channel dynamics. Since the tokenized knowledge eventually still has to be delivered through ordinary modulation/demodulation, random channel fading inevitably affects the transmitted tokens, leading to corrupted or missing semantic content. Unlike traditional bitwise transmission errors, such losses may disproportionately degrade KDC's performance if task-relevant semantic content is impacted.

The second challenge arises in latency-sensitive applications under communication bandwidth constraint. To meet the task deadline, a receiver may choose to terminate reception early, accepting semantic approximation in exchange for latency reduction. Consequently, the receiver only observes a partial, coarse-grained representation of the input data.

To address both problems, KDC introduces a *knowledge-conditioned restoration framework*(Fig. 5). The Rx leverages a *PosteriorKB* to reconstruct incomplete or corrupted sensor data. This process improves over time through continual accumulation of context, enabling the Rx to gradually reduce its dependence on explicit transmission.

Restoration Architecture. Let X denote the transmitted data, and $\tilde{X}$ the possibly lossy, partial version received. The Rx aims to recover a high-fidelity semantic estimate $\hat{X}$ using both $\tilde{X}$ and prior knowledge:

$$\hat{X} = \mathcal{R}(\tilde{X}, \mathit{SharedKB}, \mathit{PosteriorKB}). \tag{4}$$

The restoration function $\mathcal{R}(\cdot)$ operates through modality-specific networks: UNet for images/video and transformer architectures for LiDAR/radar volumes. These networks are *pre-trained once per modality* using generic restoration datasets, with task-specific adaptation achieved purely through runtime conditioning—eliminating per-task retraining.

For restoration conditioning, the *PosteriorKB* serves as a contextual reference database. Each *PosteriorKB* record stores $(\mathbf{e}_{\text{text}}, \mathbf{e}_{\text{data}})$ tuples, where $\mathbf{e}_{\text{text}}$ embeds the semantic bins carried in transmission headers (*e.g.*, "pedestrian detection") and $\mathbf{e}_{\text{data}}$ represents the latent features of associated sensor data. At inference time, the Rx performs k-nearest neighbor search on $\mathbf{e}_{\text{text}}$ to retrieve the most relevant contextual samples, which act as visual or structural priors for reconstruction.

Restoration is conditioned via feature-wise linear modulation (FiLM) [48] at each decoder stage:

$$\text{FiLM}(\mathbf{f}) = \gamma(\mathbf{e}_{\text{ctx}}) \cdot \mathbf{f} + \beta(\mathbf{e}_{\text{ctx}}), \tag{5}$$

where $\mathbf{f}$ is a decoder feature map, $\mathbf{e}_{\text{ctx}}$ is the concatenated embedding from retrieved *PosteriorKB* references, and $\gamma(\cdot)$, $\beta(\cdot)$ are learned affine transformations that modulate features based on retrieved context.

Knowledge Accumulation. Beyond restoration, the *PosteriorKB* itself is dynamically updated to directly address the two challenges above: additional stored context improves tolerance to missing tokens, while accumulated history enables incremental refinement over time. After each reception round, the system stores the text embedding $\mathbf{e}_{\text{text}}$ along with the restored data embedding $\mathbf{e}_{\text{data}}$, which are indexed in the *PosteriorKB* and later retrieved to aid subsequent restorations. To prevent unbounded growth, semantically similar embeddings are grouped by cosine similarity and replaced with their centroid (simple vector averaging), forming canonical references that improve restoration accuracy. This process yields a positive feedback loop: ***The more the system receives, the less it needs to receive.***

4 Implementation

4.1 Implementation of Diverse Data Modalities

We implemented a full end-to-end prototype of the KDC framework to validate its effectiveness across four representative sensor modalities: image, video, LiDAR, and radar.

Image and video modality. For image inputs, we instantiate the modality-specific *encoder backbones* introduced in Sec. 3.1, using pretrained ViT [19] and CLIP [37] models. Each image is partitioned into non-overlapping patches and embedded into semantic tokens following the standard transformer pipeline. For video inputs, we apply the same tokenization per frame and then aggregate tokens from a sliding window of consecutive frames via temporal pooling, enabling temporal-aware attention during importance estimation and restoration. To connect token importance with task semantics, we integrate CLIPSeg to generate coarse region proposals that are aligned with the *semantic bins* derived in Sec. 3.1.2. On the Rx side, image restoration is performed using a lightweight UNet decoder, conditioned by semantic context through feature-wise linear modulation (FiLM), which applies affine transformations to decoder feature maps based on embeddings retrieved from the *SharedKB*.

LiDAR and radar modality. For point-based or volumetric sensors, we develop custom embedding pipelines that follow the same tokenization principle as Sec. 3.1. LiDAR point clouds are voxelized into 3D grids and passed through a sparse convolutional encoder, which produces a sequence of region-level tokens. Radar data are treated in a similar fashion,

with domain-specific adaptations. We use a 20×20 MIMO radar [1] to collect channel responses. The raw intermediate-frequency samples are processed using the MUSIC algorithm [54] to obtain a $256 \times 256 \times 256$ spatial spectrum cube (angle, range, Doppler), which is then partitioned into voxel units and embedded as tokens. We pretrain a radar encoder using contrastive learning to embed these 3D volumes into a latent space. To enable semantic alignment (Sec. 3.1.1) with the shared embedding space, we train a transformer-based radar-to-text model on a curated dataset with human pose annotations [41], following a CLIP-style objective that aligns radar tokens with textual pose descriptors. Here "pose labels" serve as semantic tags for contrastive supervision rather than conventional segmentation masks. The LiDAR pipeline follows a similar architecture, but is supervised using per-point semantic annotations instead of pose labels.

For the restoration network on the Rx side, the radar and LiDAR inputs are restored using transformer-based backbones, with multi-head cross-attention layers to fuse corrupted tokens and retrieved references. During training, we simulate real-world degradation by introducing randomized packet drop (5%–15%) over semantic tokens. The image/video restoration network is pretrained on a combination of COCO-20 [45] and FSS-1000 [35]. The LiDAR restoration network is pretrained on ShapeNet [9] with simulated voxel dropout. The radar restoration network is pretrained on our in-house radar-pose dataset (Sec. 4.1) with synthetic packet drop augmentation. All models are trained using an L1 loss on the reconstructed signal. Optimization is performed using Adam with $\beta_1 = 0.9$, $\beta_2 = 0.999$, initial learning rate of 10^{-3}, and cosine annealing to a minimum of 2×10^{-5}. The final restoration model consists of 9.4 million parameters.

4.2 Parallel and Pipelined Processing.

The runtime processing of KDC orchestrates three key stages: KDC, encoding, and transmission.

Pipeline architecture. The three stages operate in a producer-consumer pipeline pattern, as illustrated in Fig. 6. While one frame undergoes transmission, another can be simultaneously encoded, and a third can begin KDC processing. This multi-stage pipeline ensures continuous data flow and improved resource utilization. However, the pipeline stages have unequal processing times, with KDC typically requiring more computation time than encoding, which could potentially create a bottleneck in the pipeline. KDC addresses this through parallelization combined with intelligent caching.

Parallel KDC processing. Unlike the encoding stage, which must process frames sequentially to maintain data dependencies and compression context, KDC operations are stateless across frames—the processing of a frame depends only on frame-specific parameters, not on previous frames' computational results. To optimize this further, KDC implements a mask caching mechanism that detects frame similarity and reuses KDC masks when consecutive frames show

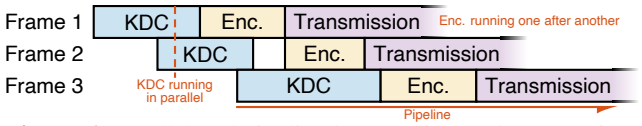

Figure 6: Parallel and pipelined processing at the transmitter.

minimal changes (Sec. 6). This cache maintains an ordered dictionary of recent masks, allowing efficient lookup of the most recent valid mask for a given frame number. The receiver adopts a symmetric pipeline architecture and hence we omit the details.

4.3 Wireless Testbeds for Sensor Data Transmission.

We deploy KDC across three wireless communication technologies, namely Cellular, WiFi, and LoRa–representing different points on the bandwidth-latency-power trade-off spectrum. The actual transmissions are conducted through a mix of trace-driven emulation and real-time communication. Appendix Table 4 summarizes the wireless trace characteristics.

Cellular. We use a 5G smartphone to continuously record high-bandwidth transmission traces via `iperf3`, which captures packet-level loss events, timestamps, and channel state metrics. We replay sensor data delivery over these traces. Packet drops are replayed at the application layer (after transport-layer retransmissions) to match the empirical loss pattern from each trace segment. To cover diverse channel conditions and mobility patterns, we collect the traces across three environments: (i) high-way driving, (ii) semi-stationary indoor office environments, and (iii) outdoor scooter riding on a university campus. Each trace spans approximately 1000 seconds. In addition, we have conducted real-time experiments through cellular networks (Sec. 6).

WiFi. We similarly evaluate KDC over a WiFi LAN using both 2.4 and 5 GHz bands. Similar to the cellular setting, we collect traces in a lab environment while moving the smartphone at walking speed.

LoRa. We implemented a LoRa wireless communication testbed using the USRP N310 devices [21] following [30]. We collected approximately 2000 seconds of LoRa transmission traces across different scenarios and distances, yielding traces with varied packet loss rates and throughput levels.

5 Real-World Evaluation

We evaluate KDC over the aforementioned 3 types of networks and 4 data modalities. For each, we run real-time transmission or trace-driven emulation. All baselines are evaluated under identical network conditions, with packet loss applied at the application layer after transport-layer recovery. Each baseline uses its native error recovery (see Appendix Table 2 for detailed configurations). Appendix Table 3 summarizes all datasets used across experiments.

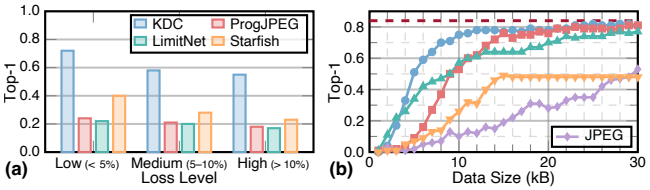

Figure 7: KDC Image data transmission over real Cellular networks.

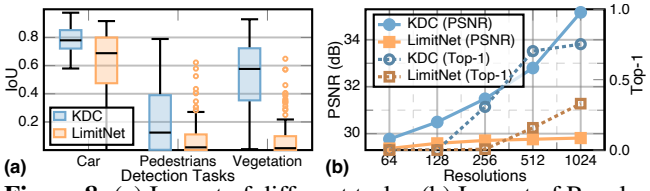

Figure 8: (a) Impact of different tasks. (b) Impact of Resolution.

5.1 Image Transmission over Cellular Network

Experimental Setup. Image transmission is evaluated assuming an edge camera captures images in urban environments and offloads visual data to cloud servers via 5G networks for classification tasks. This could correspond to robotic patrol or public safety monitoring cases. We use both ImageNet-1K [53] and Cityscapes [15] as the data sources (encompassing both single-object and multi-object scenarios), and run the image transmission through trace-driven emulation over the 5G network traces.

Baselines: (1) *Progressive JPEG (ProgJPEG)* [66]: classic bitstream-based image compression with early-exit capability; (2) *LimitNet* [24]: a learned task-aware codec fine-tuned on specific datasets; (3) *StarFish* [25]: state-of-the-art learned semantic image transmission system.

Results. Fig. 7(a) compares Top-1 accuracy under different packet loss levels. ProgJPEG and LimitNet degrade sharply ($\leq$0.2 Top-1 at 5–10% loss) because bit errors propagate to entire frames, while JPEG fails completely and is omitted. StarFish shows modest resilience ($\sim$0.3 Top-1 at 10% loss), but is bounded by the quality loss of its pretrained model. In contrast, KDC maintains 0.55–0.65 Top-1 across all loss levels.

Fig. 7(b) reports Top-1 accuracy under increasing data budgets. JPEG is the slowest to improve, since its line-by-line pixel streaming yields no usable image until the full JPEG file ($\sim$30 kB on average in the dataset) is received. ProgJPEG and LimitNet improve progressively but plateau late, as they lack task-aware semantic prioritization. StarFish saturates early but is limited to $\sim$0.6 Top-1 due to its reconstruction artifacts. KDC, by contrast, reaches near-optimal accuracy ($\sim$0.85 Top-1, close to the $\sim$0.87 full-image baseline shown by the red line) with only 10 kB of data, achieving 2–3$\times$ faster convergence.

Adaptation to Different Tasks. Existing neural image encoding and transmission methods have limited adaptability to diverse downstream tasks because they require end-to-end model training for each task with a specific dataset. In contrast, KDC enables zero-shot task-aware transmission, making it adaptable to a wide range of image processing tasks by using explicit semantic decomposition. We demonstrate this capability by evaluating KDC and LimitNet on the CityScapes dataset, which supports multiple segmentation tasks over distinct object categories. Specifically, we focus on vehicles, pedestrians, and vegetation, where the focus regions are derived from ground-truth segmentation labels and the transmitted data size budget is constrained to 30 KB.

As shown in Fig. 8(a), LimitNet yields unstable performance with IoU mostly below 0.3, especially poor for pedestrians. In contrast, KDC maintains consistently higher IoU: $\sim$0.8 for cars, $\sim$0.5 for pedestrians, and $\sim$0.7 for vegetation. Pedestrian IoU is lower for both methods due to smaller object size, but KDC still delivers a clear margin. This robustness stems from the task-conditional importance, which explicitly prioritizes semantically relevant regions instead of relying on a black-box encoder.

Generalization to Various Image Resolutions. Neural baselines such as LimitNet are tied to fixed-resolution backbones, which restricts their ability to generalize. We compare KDC against LimitNet by encoding and decoding images at different raw input resolutions. Fig. 8(b) shows that LimitNet remains nearly flat, with PSNR $<$31 dB and Top-1 $<$0.2 even at 1024 resolution. In contrast, KDC scales gracefully: PSNR surpasses 34 dB and Top-1 reaches $\sim$0.9 at 1024, improving steadily with resolution.

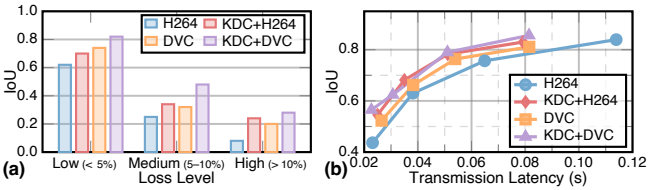

Figure 9: KDC video transmission over real Cellular networks.

5.2 Video Transmission over Cellular

Experimental Setup. We evaluate KDC on a semantic segmentation task for autonomous driving using the ACO YouTube Driving dataset [69]. Real-time video streaming is conducted over real 5G cellular traces with fluctuating throughput and average latency around 70 ms.

Baselines: We compare against: (1) *H.264*, a standard video codec; (2) *DVC* [39], a neural video compression method; and (3) *KDC +H.264* and *KDC +DVC*, where KDC pre-quantizes the data based on task-aware importance (Sec. 3.1.1) and integrates the base codec using its codec wrapper (Sec. 3.2).

Results. Fig. 9(a) reports IoU under different packet loss levels. Both H.264 and DVC degrade significantly under medium-to-high loss due to error propagation, while KDC-enhanced codecs maintain higher IoU (improving by 0.1–0.2 absolute), a substantial margin in semantic segmentation

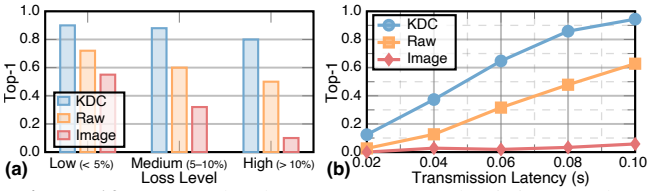

Figure 10: KDC radar data trans. over real WiFi networks.

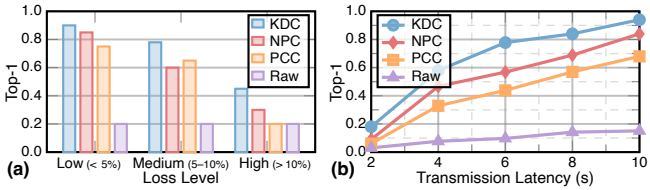

Figure 11: KDC LiDAR data trans. over real LoRA networks.

where even 0.02–0.05 gains are typically significant. Fig. 9(b) shows how IoU improves as data accumulate over time (*i.e.*, as transmission latency increases) under real cellular traces. While all methods improve as more data are delivered, H.264 lags behind, and DVC saturates early with lower accuracy. KDC+H.264 and KDC+DVC consistently achieve higher IoU (up to ∼0.85) and converge faster at low latencies (<0.05 s).

Finally, although DVC offers strong compression at high bandwidth, it is highly vulnerable to packet loss due to latent drift (*i.e.*, error accumulation in latent features). KDC+DVC alleviates this by aligning token priorities with downstream tasks and restoring corrupted segments from prior frames in the *PosteriorKB*, yielding better segmentation on small moving vehicles and fine structures such as traffic poles.

5.3 Radar Transmission over WiFi

Experimental Setup. We evaluate a radar-based human posture classification task. The input data are collected from the aforementioned MIMO radar, which form a 256^3 spectrum volume after being processed via MUSIC. 256 is the FFT bin count determined by the number of ADC samples per radar chirp. The radar data are transmitted in real-time through a WiFi network. For repeatability and fair comparison with baselines, we use the aforementioned WiFi network traces to emulate the network dynamics. The receiver performs posture classification based on the received data. A total of 10 different types of postures (*e.g.*, sitting, standing, walking, running, squatting, bending, jumping, raising hands, waving, and turning) are collected, and ground-truth human pose labels are generated via depth motion capture. For each posture, 10 minutes of continuous radar signals are recorded to provide sufficient temporal coverage.

Baselines: (1) *Raw radar*: sends full spatial spectrum; (2) *Image-based*: slices converted to 2D spectrograms and JPEG-encoded [60].

Results. Fig. 10(a) reports Top-1 accuracy for downstream application (radar human posture classification) under packet loss. Raw radar degrades quickly (dropping below 0.4 at 10% loss) due to its bitrate and vulnerability to missing samples. The image baseline performs even worse (<0.2 at high loss), since slicing discards 3D structure and spectrograms misalign with semantic meaning. In contrast, KDC maintains 0.8+ accuracy even at 10% loss by isolating pose-relevant tokens (*e.g.*, joint clusters) and restoring corrupted tokens via the KBs. Fig. 10(b) shows accuracy under increasing transmission latency (*i.e.*, early exit time). Raw radar improves only gradually, and the image baseline remains flat near zero.

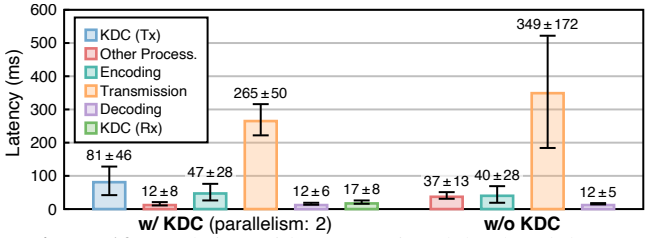

Figure 12: Real-time video streaming delay breakdown.

KDC rises sharply, exceeding 0.9 Top-1 accuracy within 0.1s latency. Performance eventually plateaus beyond 30% loss, where missing joint configurations cannot be recovered even with prior knowledge.

5.4 LiDAR Data Delivery Over LoRa

Experiment Setup. LiDAR data transmission is needed in scene reconstruction use cases, *e.g.*, a search-and-rescue scenario with robots connected through a long-range LoRa link. To experiment with this scenario, we transmit LiDAR samples (converted into voxelized sparse point clouds) from the SemanticKITTI dataset through the aforementioned LoRaWAN traces. The downstream task is LiDAR *semantic segmentation*, where we evaluate mean Intersection-over-Union (mIoU) using a standard sparse 3D segmentation head applied to the received tokens. We adopt PointNet++ [49] as the backbone model.

Baselines: (1) *Raw*: uncompressed; (2) *PCC*: MPEG standard point cloud codec; (3) *NPC*: neural point cloud compression [52].

Results. Fig. 11(a) reports IoU under packet loss. Raw transmission collapses quickly (<0.2 IoU at 5% loss) due to its large bitrate and loss sensitivity. PCC and NPC perform better, sustaining ∼0.6–0.7 at medium loss, but both degrade sharply at high loss (<0.3). KDC consistently outperforms all baselines, maintaining ∼0.8 at 5% loss and above 0.5 even with 10%+ loss, by emphasizing structural tokens (*e.g.*, roof edges, vehicle contours) and restoring corrupted ones via KB retrieval. Fig. 11(b) shows how IoU improves with transmission latency. Raw remains flat near zero, as its size makes it infeasible on the low-bandwidth LoRa link. PCC and NPC improve gradually but saturate around 0.6–0.7. In contrast, KDC rises quickly and saturates within 8–10 seconds, reaching up to 20–30% higher IoU than the baselines.

6 Real-time Performance

Experimental setup. We evaluate KDC's real-time performance using video streaming, since the video modality is

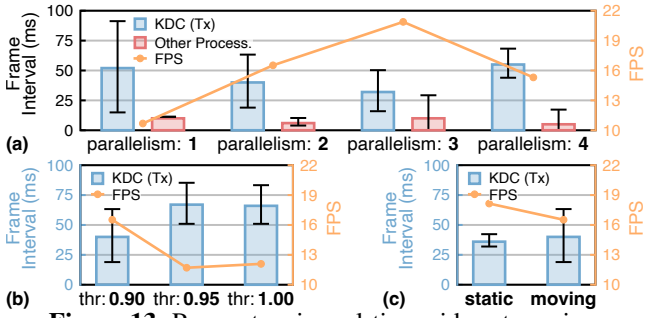

Figure 13: Parameters in real-time video streaming.

most computationally intensive. Our testbed streams 4K video (RGB 3840×2160 @ 60 FPS as USB camera [20]) from (i) an NVIDIA Jetson Orin Nano [32] (edge platform), and (ii) a laptop equipped with an RTX 4090 GPU (high-end reference) to an AWS server via 5G (Verizon network, TCL 10 5G UW phone, using USB tethering). We integrate KDC into WebRTC using aiortc [34] with modifications for: *(1)* WebSocket signaling; *(2)* adaptive rate control for FPS variations; *(3)* Nvidia hardware encoding (`h264_nvenc`); *(4)* increased jitter buffer size. GPS synchronization ensures accurate timestamps. For perceptual similarity, we employ image difference hash (dHash) [33, 43] with 8×8 hash size, defining similarity as 1 minus the normalized Hamming distance, which is efficient and has few false positives. We dynamically adjust the CLIP mask (Sec. 3.1.2) threshold based on estimated bandwidth to optimize the quality-bitrate tradeoff under varying network conditions.

Runtime / Computational Profile. We first evaluate KDC's computational cost with controlled measurements on both Tx and Rx pipelines, running PyTorch 2.3 with CUDA 12.4.

Transmitter. The main overhead stems from the foundation model query and the multi-modal segmentation network. The foundation model query and PosteriorKB retrieval are invoked *once per task session* (offline, not per frame) and cached. The per-frame critical path consists only of importance map computation via CLIPSeg, entropy pre-processing ($<$1 ms), and standard codec encoding. We benchmark the segmentation model on 1080p frames: the Jetson Orin achieves 32.1 FPS on average, while the RTX 4090 sustains over 100 FPS, confirming real-time feasibility on edge devices.

Codec. The KDC codec wrapper combines GPU and CPU stages. On the RTX 4090, pre-processing (resizing, normalization, DCT) plus GPU-accelerated JPEG encoding takes 0.74 ms per 1080p frame (averaged over 500 runs). Huffman table generation and bitstream packaging on the CPU adds another 0.18 ms. Decoding on the receiver side is symmetric, completing in under 1 ms.

Receiver. The restoration network is lightweight: inference on 1080p frames takes 0.96 ms on the RTX 4090 and 14.7 ms on the Jetson Orin (68 FPS and 32 FPS, respectively, with batch size 1). This ensures video throughput is bottlenecked only by network bandwidth, not compute.

Finally, to validate end-to-end performance, we integrate KDC into a WebRTC pipeline using hardware-accelerated `h264_nvenc`. Streaming a 1080p@30fps video shows that preprocessing and restoration stages add less than 5% overhead relative to raw WebRTC transmission, confirming the practical headroom of our design.

Latency analysis. Fig. 12 shows the end-to-end latency breakdown. With KDC using parallelism degree 2, the total end-to-end latency of 434 ms comprises KDC processing, encoding, transmission, decoding, and receiver-side KDC processing. The average KDC processing latency is 81 ± 46 ms at the transmitter and 17 ± 8 ms at the receiver, while maintaining a 40 ms frame interval through pipelining. The "other processing" component (12 ± 8 ms with KDC v.s. 37 ± 13 ms without) includes frame construction, color conversion, and frame reordering when KDC runs in parallel. Note that the encoding latency (47 ± 28 ms) includes internal buffering time and may not directly reflect the actual encoding computational load. Most importantly, the communication latency is dominant, and is reduced by 24% by using KDC, showing while KDC preserves the most important information, the end-to-end latency is also reduced by reducing the communication load.

Real-time performance optimization. We evaluate KDC's performance on 4K video streams captured during driving scenarios, including both static (vehicle stopped) and dynamic (vehicle moving) scenes. Fig. 13 demonstrates KDC's performance under different configurations:

(a) Parallelism Impact: Increasing parallelism from 1 to 3 improves both frame interval from $\sim$ 50 ms to $\sim$ 30 ms and increases FPS from 11 to 21. At parallelism degree 4, GPU saturation causes performance degradation, increasing frame interval and reducing FPS to $\sim$ 15. The optimal parallelism of 3 balances GPU utilization with frame rate, determined by the GPU's capacity to handle concurrent KDC instances while leaving headroom for encoding and other system processes.

(b) Adaptable Thresholding: We observe dHash similarity of 0.90–0.95 for static scenes, with $\sim$ 3% scene movement reducing similarity to $\sim$ 0.90. This threshold effectively balances computational efficiency with perceptual quality, enabling dynamic resource allocation based on scene complexity.

(c) Scene Dynamics: Static and moving scenes show similar frame intervals (36 ± 6 ms vs. 40 ± 23 ms respectively), with static scenes exhibiting significantly lower variance. The minimal difference in average latency indicates that KDC's processing overhead is dominated by fixed costs, including GPU-CPU transfers and mask merging operations. The higher variance in moving scenes reflects the dHash similarity detection mechanism, where some frames require mask recalculation that is relatively more computationally heavy while others can skip this step based on similarity thresholds.

7 Micro-benchmarks

Transmitter Evaluation. We compare *SharedKB*–driven semantic bin decomposition (Sec. 3.1) against directly

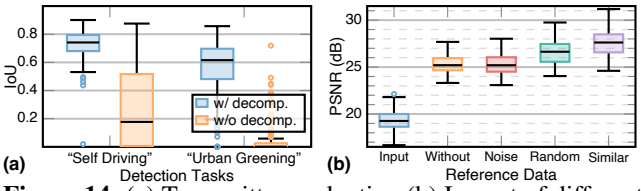

Figure 14: (a) Transmitter evaluation (b) Impact of different reference data.

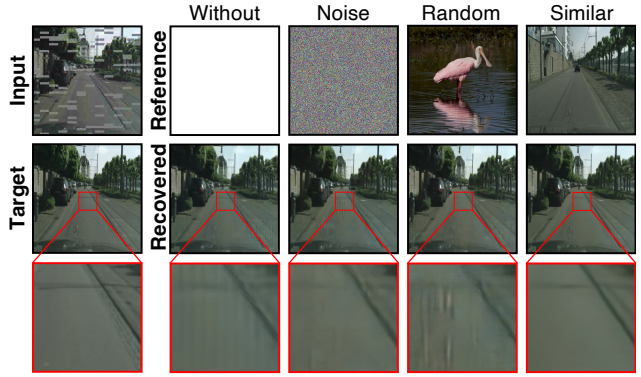

Figure 15: The effect of different reference images.

providing task descriptions to CLIPSeg, using two tasks ("autonomous driving" and "urban greening research") on CityScapes with ground-truth segmentation labels. In the *SharedKB* setting, the foundation model decomposes each task into semantic bins refined through *PosteriorKB* retrieval; in the baseline, raw task descriptions are fed to CLIPSeg directly. As shown in Fig. 14(a), IoU is 1.8× and 6.5× higher with *SharedKB*-driven decomposition, confirming that semantic bin extraction substantially improves content prioritization.

Receiver Evaluation. We evaluate the *PosteriorKB*-assisted restoration model under four reference image conditions: *(1)* no reference, *(2)* noise image, *(3)* random image, and *(4)* similar image. As shown in Fig. 14(b) and Fig. 15, even random images improve restoration, as the model leverages natural image statistics (*e.g.*, frequency distributions, color correlations) learned during training; noise images, lacking such structure, provide less benefit. Optimal performance is achieved with semantically related references that supply relevant content priors. During transmission, KDC progressively retrieves increasingly similar references for continued refinement.

Ablation: SharedKB-only vs. SharedKB+PosteriorKB. Under identical 5G traces (5% loss), adding PosteriorKB improves ImageNet Top-1 from 0.51 to 0.62 (+21.6%) and Cityscapes mIoU from 0.71 to 0.75. The gain is largest under moderate loss (3–8%), where session-accumulated references fill in missing tokens; under very low loss (<1%), both variants perform similarly. This confirms that progressive knowledge accumulation provides meaningful gains beyond the static SharedKB alone.

Impact of Loss Burstiness. We evaluate the resilience of KDC under varying levels of packet loss burstiness. We compare two loss patterns with equal average loss rates (5%): (i) *scattered* (independent packet drops) and (ii) *bursty* (consecutive losses of 100–500 ms duration). Both are cropped from real 5G cellular traces. As shown in Fig. 16(a,b), KDC maintains downstream accuracy within 80% of the full-transmission baseline under scattered loss, but degrades sharply under bursty loss spanning multiple consecutive frames, where insufficient contextual tokens prevent accurate *PosteriorKB* retrieval. No restoration method can recover extended blackouts without retransmission; for safety-critical applications, we recommend falling back to standard retransmission when estimated burst severity exceeds a configurable threshold (Sec. A).

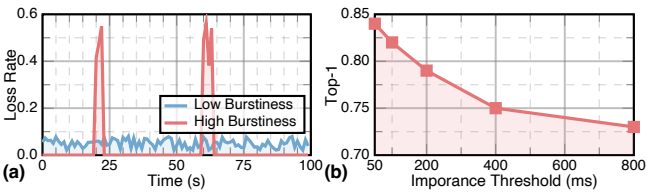

Figure 16: (a) Example of Different Loss Burstiness (b) Impact of Loss Burstiness.

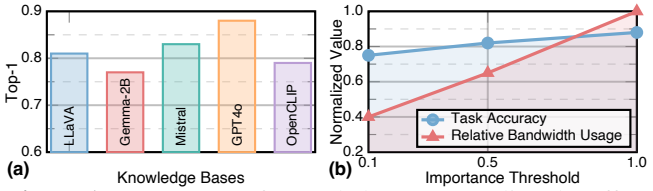

Figure 17: (a) Impact of Knowledge Base Quality. (b) Effectiveness of Knowledge Prioritization.

Impact of Task Granularity. KDC relies on task descriptions to guide semantic token prioritization. Here, we study how prompt granularity affects downstream task performance. Following the same experimental setup as Sec. 7 transmitter evaluation (Tx-side evaluation using semantic bin decomposition and CLIPSeg-based segmentation), We categorize prompts into three granularity levels—*low* (abstract task name), *medium* (scene-level with key objects), and *high* (detailed object enumeration)—and cache extracted bins for the entire session. Table 1 summarizes representative examples and their corresponding downstream metrics.

More detailed prompts consistently improve accuracy, but overly specific ones reduce generalizability. Medium-granularity prompts strike the best balance; we recommend the template "[scenario] with [2–4 key objects]," which can be auto-generated from the PosteriorKB.

Impact of Knowledge Base Quality. The effectiveness of semantic recovery in KDC depends heavily on the quality of the *SharedKB*. We evaluate this by replacing the default *SharedKB* with versions constructed using different LLMs and embedding backbones. Specifically, we compare variants using Gemma-2B, LLaVA, and Mistral-7B for semantic bin extraction, and pretrained embeddings from OpenCLIP and GPT-4o. Fig. 17(a) shows that KB quality directly impacts reconstruction performance, particularly in low-SNR and low-bandwidth conditions. The performance gap across *SharedKB*

Table 1: Impact of prompt granularity on system performance.

[Prompt Granularity] Task Description	Avg Task Acc.
[Low] "autonomous driving"	62%
[Medium] "street scene with vehicles and people"	71%
[High] "urban street with pedestrian crossing and traffic lights"	75%

variants can exceed 10% for certain tasks. These results motivate continued research in foundational world modeling, as advancements in multimodal pretraining directly translate to improved performance for knowledge-driven communication systems.

Effectiveness of Knowledge Prioritization. We quantify the impact of semantic importance thresholding during token selection. The Tx only sends tokens z_i with $\alpha_i \geq \tau$, where τ controls the sparsity of prioritized content. We compare system performance at three threshold settings: $\tau = 0.1$, $\tau = 0.5$, and $\tau = 1.0$ (*i.e.*, full transmission). As shown in Fig. 17(b), higher thresholds yield improved task accuracy at the cost of greater bandwidth usage. Notably, $\tau = 0.5$ achieves over 82% of full-task performance using only 27% of the tokens.

8 Related Work

Classical source and channel coding. Traditional systems separate source coding (compression) and channel coding (error protection), with advanced schemes introducing adaptability or redundancy to balance efficiency and resilience [3, 12]. Scalable Video Coding (SVC) enables graceful degradation via base and enhancement layers [63], while analog progressive schemes such as SoftCast [28] jointly design source and channel coding for robustness. Other techniques include superposition coding for broadcast channels [16] and rateless fountain codes for loss resilience [5]. KDC shares the incremental refinement philosophy of SVC but, leveraging knowledge bases, extends it beyond video and yields useful information from the first packets rather than requiring full reception.

Deep learning-based communication. Deep learning has been applied to improve physical-layer performance by optimizing individual components (*e.g.*, neural decoders for error-correcting codes [44]) and by training end-to-end autoencoder-like transceiver networks [18, 27, 47, 50]. These learned systems can outperform hand-crafted designs under complex channels (*e.g.*, MIMO interference), but they still act as highly optimized bit-pipes that aim to faithfully reconstruct every bit, without understanding the semantic content [38]. In contrast, KDC prioritizes transmitting **meaningful** information through knowledge-driven data representation. It enables progressive, task-relevant reconstruction even from partial and corrupted data, rather than all-or-nothing bit delivery. Unlike neural codecs (*e.g.*, DVC [39], LimitNet [24]) that encode content into task-agnostic latent spaces, KDC preserves standard codec formats and adds task-aware prioritization as a pre-processing wrapper, maintaining hardware compatibility.

Communication with side information. Sharing side information between transmitter and receiver is known to improve efficiency fundamentally [56]. One practical approach is to use a public algorithm or standard as the shared knowledge. For instance, a sender compresses data with a standard codec like JPEG and the receiver knows how to decode it [61]. Another approach is to share a large codebook or database of content, so that the sender can merely transmit a short identifier (an index or hash) for the receiver to look up the full data [40, 46, 55, 59]. While effective in reducing transmission, these methods are brittle (a single bit error in the index can cause a decoding failure) and generally oblivious to any specific task or semantic nuance of the data. KDC avoids a rigid predefined codebook. Instead, it exploits complex internal representations of data based on modern foundation models, providing much greater robustness to channel errors and adaptability to different tasks without manual re-design.

Semantic communication. Semantic communication aims to transmit meaning rather than exact bits [24, 68]. Most systems adopt implicit reasoning, where neural encoders and decoders are trained end-to-end to optimize semantic fidelity, as in DeepSC [64]. Others introduce explicit reasoning by sharing structured knowledge such as knowledge graphs [11]. While these approaches achieve bandwidth savings and task-specific gains, they lack adaptability, since retraining is needed whenever the domain, task, or channel changes. KDC addresses this gap by leveraging a fixed multimodal foundation model as a *SharedKB* [4, 17, 19] and transmitting incrementally refined semantic tokens, enabling cross-modal generalization and resilience without end-to-end retraining.

Split inference and feature transmission. Split inference [31, 42] partitions a DNN across device and server, transmitting intermediate features. Unlike split inference, KDC uses a standards-compatible codec wrapper (no shared DNN backbone required), is robust to packet loss via KB-conditioned restoration, and supports zero-shot task adaptation without re-partitioning.

9 Conclusion

We presented KDC, a framework that embeds task-aware semantics into transmission and leverages knowledge-conditioned reasoning at the receiver. Our prototype across image, video, radar, and LiDAR, deployed on cellular, WiFi, and LoRa, shows consistent gains in efficiency, robustness, and zero-shot adaptability over state-of-the-art baselines, while sustaining real-time performance. These results position KDC as a practical step toward semantic, knowledge-driven communication in future wireless systems.

Acknowledgments

We appreciate the insightful comments and feedback from the reviewers and shepherd. This work is supported by the NSF under Grants CNS-2403124, CNS-2408393, CNS-2346550, and CNS-2312715.

References

[1] Vayyar imaging - home. https://www.vayyar.com/. Accessed: 2023-1-29.

[2] Fayçal Ait Aoudia and Jakob Hoydis. End-to-end learning of communications systems without a channel model. In *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, pages 298–303. IEEE, 2018.

[3] Eirina Bourtsoulatze, David Burth Kurka, and Deniz Gündüz. Deep joint source-channel coding for wireless image transmission. *IEEE Transactions on Cognitive Communications and Networking*, 5(3):567–579, 2019.

[4] Davide Buffelli, Sowmen Das, Yu-Wei Lin, Sattar Vakili, Chien-Yi Wang, Masoud Attarifar, Pritthijit Nath, and Da-shan Shiu. Towards a foundation model for communication systems. *arXiv preprint arXiv:2505.14603*, 2025.

[5] John W Byers, Michael Luby, Michael Mitzenmacher, and Ashutosh Rege. A digital fountain approach to reliable distribution of bulk data. *ACM SIGCOMM Computer Communication Review*, 28(4):56–67, 1998.

[6] Qiming Cao, Hongfei Xue, Tianci Liu, Xingchen Wang, Haoyu Wang, Xincheng Zhang, and Lu Su. mmclip: Boosting mmwave-based zero-shot har via signal-text alignment. In *Proceedings of the 22nd ACM conference on embedded networked sensor systems*, pages 184–197, 2024.

[7] Rudolf Carnap and Yehoshua Bar-Hillel. An outline of a theory of semantic information. Technical report, Technical Report 247, Massachusetts Institute of Technology, 1952.

[8] Christina Chaccour, Walid Saad, Merouane Debbah, Zhu Han, and H Vincent Poor. Less data, more knowledge: Building next-generation semantic communication networks. *IEEE Communications Surveys & Tutorials*, 27(1):37–76, 2024.

[9] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[10] Ruizhi Cheng, Kaiyan Liu, Nan Wu, and Bo Han. Enriching telepresence with semantic-driven holographic communication. In *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*, pages 147–156, 2023.

[11] Jinho Choi, Seng W Loke, and Jihong Park. A unified approach to semantic information and communication based on probabilistic logic. *IEEE Access*, 10:129806–129822, 2022.

[12] Kristy Choi, Kedar Tatwawadi, Aditya Grover, Tsachy Weissman, and Stefano Ermon. Neural joint source-channel coding. In *International Conference on Machine Learning*, pages 1182–1192. PMLR, 2019.

[13] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

[14] Spconv Contributors. Spconv: Spatially sparse convolution library, 2022.

[15] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Scharwächter, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset. In *CVPR Workshop on the Future of Datasets in Vision*, volume 2, pages 1–8. IEEE, 2015.

[16] Thomas Cover. Broadcast channels. *IEEE Transactions on Information Theory*, 18(1):2–14, 2003.

[17] Henghui Ding, Chang Liu, Suchen Wang, and Xudong Jiang. Vision-language transformer and query generation for referring segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16321–16330, 2021.

[18] Sebastian Dörner, Sebastian Cammerer, Jakob Hoydis, and Stephan Ten Brink. Deep learning based communication over the air. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):132–143, 2017.

[19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[20] ELP. ELP HDMI 4K USB3.0 camera 60FPS wide angle no distortion lens H.264 high speed IMX415 ultra HD mini PC webcam for live streaming. https://www.svpro.cc/product/elp-hdmi-4k-usb3-0-camera-60fps-wide-angle-no-distortion-lens-h-264-high-speed-imx415-ultra-hd-mini-pc-webcam-for-live-streaming/, 2024.

[21] Ettus Research. USRP N310. https://www.ettus.com/all-products/usrp-n310/, 2019.

[22] Nariman Farsad, Milind Rao, and Andrea Goldsmith. Deep learning for joint source-channel coding of text. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2326–2330. IEEE, 2018.

[23] Deniz Gündüz, Zhijin Qin, Inaki Estella Aguerri, Harpreet S Dhillon, Zhaohui Yang, Aylin Yener, Kai Kit Wong, and Chan-Byoung Chae. Beyond transmitting bits: Context, semantics, and task-oriented communications. *IEEE Journal on Selected Areas in Communications*, 41(1):5–41, 2022.

[24] Ali Hojjat, Janek Haberer, Tayyaba Zainab, and Olaf Landsiedel. 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 (MobiSys)*, pages 519–533, 2024.

[25] Pan Hu, Junha Im, Zain Asgar, and Sachin Katti. Starfish: Resilient image compression for AIoT cameras. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems (SenSys)*, pages 395–408, 2020.

[26] Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. The platonic representation hypothesis. *arXiv preprint arXiv:2405.07987*, 2024.

[27] Nazmul Islam and Seokjoo Shin. Deep learning in physical layer: Review on data driven end-to-end communication systems and their enabling semantic applications. *IEEE Open Journal of the Communications Society*, 5:4207–4240, 2024.

[28] Szymon Jakubczak and Dina Katabi. SoftCast: Clean-slate scalable wireless video. In *Proceedings of the 2010 ACM workshop on Wireless of the students, by the students, for the students*, pages 9–12, 2010.

[29] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International conference on machine learning*, pages 4904–4916. PMLR, 2021.

[30] Tapparel Joachim and Andreas Burg. Design and implementation of LoRa physical layer in GNU radio. In *Proceedings of the GNU Radio Conference*, volume 9, 2024.

[31] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGARCH Computer Architecture News*, 45(1):615–629, 2017.

[32] Leela S Karumbunathan. NVIDIA Jetson AGX Orin series. Technical brief, NVIDIA, July 2022.

[33] Neal Krawetz. Kind of like that. https://www.hackerfactor.com/blog/?/archives/529-Kind-of-Like-That.html, January 2013. Accessed: March 13, 2025.

[34] Jeremy Lainé. aiortc: WebRTC and ORTC implementation for python using asyncio. https://github.com/aiortc/aiortc, February 2025. Version 1.10.1, Accessed: March 13, 2025.

[35] Xiang Li, Tianhan Wei, Yau Pun Chen, Yu-Wing Tai, and Chi-Keung Tang. Fss-1000: A 1000-class dataset for few-shot segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2869–2878, 2020.

[36] Le Liang, Hao Ye, Yucheng Sheng, Ouya Wang, Jiacheng Wang, Shi Jin, and Geoffrey Ye Li. Large language models for wireless communications: From adaptation to autonomy. *arXiv preprint arXiv:2507.21524*, 2025.

[37] Yuqi Lin, Minghao Chen, Wenxiao Wang, Boxi Wu, Ke Li, Binbin Lin, Haifeng Liu, and Xiaofei He. Clip is also an efficient segmenter: A text-driven approach for weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15305–15314, 2023.

[38] Zikun Liu, Changming Xu, Emerson Sie, Gagandeep Singh, and Deepak Vasisht. Exploring practical vulnerabilities of machine learning-based wireless systems. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 1801–1817, 2023.

[39] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. DVC: An end-to-end deep video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11006–11015, 2019.

[40] Mohammad Ali Maddah-Ali and Urs Niesen. Fundamental limits of caching. *IEEE Transactions on Information Theory*, 60(5):2856–2867, 2014.

[41] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. AMASS: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5442–5451, 2019.

[42] Yoshitomo Matsubara, Marco Levorato, and Francesco Restuccia. Split computing and early exiting for deep learning applications: Survey and research challenges. *ACM Computing Surveys*, 55(5):1–30, 2022.

[43] Vishal Monga and Brian L Evans. Perceptual image hashing via feature points: performance evaluation and tradeoffs. *IEEE Transactions on Image Processing*, 15(11):3452–3465, 2006.

[44] Eliya Nachmani, Elad Marciano, Loren Lugosch, Warren J Gross, David Burshtein, and Yair Be'ery. Deep learning methods for improved decoding of linear codes. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):119–131, 2018.

[45] Khoi Nguyen and Sinisa Todorovic. Feature weighting and boosting for few-shot segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 622–631, 2019.

[46] Urs Niesen and Mohammad Ali Maddah-Ali. Coded caching for delay-sensitive content. In *2015 IEEE International Conference on Communications (ICC)*, pages 5559–5564. IEEE, 2015.

[47] Timothy O'shea and Jakob Hoydis. An introduction to deep learning for the physical layer. *IEEE Transactions on Cognitive Communications and Networking*, 3(4):563–575, 2017.

[48] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[49] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30, 2017.

[50] Zhijin Qin, Hao Ye, Geoffrey Ye Li, and Biing-Hwang Fred Juang. Deep learning in physical layer communications. *IEEE Wireless Communications*, 26(2):93–99, 2019.

[51] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PmLR, 2021.

[52] Hongning Ruan, Yulin Shao, Qianqian Yang, Liang Zhao, and Dusit Niyato. Point cloud compression with implicit neural representations: A unified framework. In *2024 IEEE/CIC International Conference on Communications in China (ICCC)*, pages 1709–1714. IEEE, 2024.

[53] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[54] Ralph Schmidt. Multiple emitter location and signal parameter estimation. *IEEE Transactions on Antennas and Propagation*, 34(3):276–280, 1986.

[55] Karthikeyan Shanmugam, Negin Golrezaei, Alexandros G Dimakis, Andreas F Molisch, and Giuseppe Caire. Femtocaching: Wireless content delivery through distributed caching helpers. *IEEE Transactions on Information Theory*, 59(12):8402–8413, 2013.

[56] Claude E Shannon. Channels with side information at the transmitter. *IBM Journal of Research and Development*, 2(4):289–293, 1958.

[57] Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[58] Jiawei Shao, Xinjie Zhang, and Jun Zhang. Task-oriented communication for edge video analytics. *IEEE Transactions on Wireless Communications*, 23(5):4141–4154, 2023.

[59] David Slepian and Jack Wolf. Noiseless coding of correlated information sources. *IEEE Transactions on Information Theory*, 19(4):471–480, 2003.

[60] Michael Stephan, Thomas Stadelmayer, Avik Santra, Georg Fischer, Robert Weigel, and Fabian Lurz. Radar image reconstruction from raw ADC data using parametric variational autoencoder with domain adaptation. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 9529–9536. IEEE, 2021.

[61] Gregory K Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4):30–44, 1991.

[62] Junjie Wang and Tomas Nordström. Latency robust cooperative perception using asynchronous feature fusion. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1–10. IEEE, 2025.

[63] Thomas Wiegand, Heiko Schwarz, and Detlev Marpe. Overview of the scalable video coding extension of H. 264/AVC. In *Dortmunder Fernsehseminar 2006*, 2007.

[64] Huiqiang Xie, Zhijin Qin, Geoffrey Ye Li, and Biing-Hwang Juang. Deep learning enabled semantic communication systems. *IEEE Transactions on Signal Processing*, 69:2663–2675, 2021.

[65] Runsheng Xu, Hao Xiang, Xin Xia, Xu Han, Jinlong Li, and Jiaqi Ma. OPV2V: An open benchmark dataset and fusion pipeline for perception with vehicle-to-vehicle communication. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2583–2589. IEEE, 2022.

[66] Eddie Yan, Kaiyuan Zhang, Xi Wang, Karin Strauss, and Luis Ceze. Customizing progressive JPEG for efficient image storage. In *9th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 17)*, 2017.

[67] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[68] Wanting Yang, Hongyang Du, Zi Qin Liew, Wei Yang Bryan Lim, Zehui Xiong, Dusit Niyato, Xuefen Chi, Xuemin Shen, and Chunyan Miao. Semantic communications for future internet: Fundamentals, applications, and challenges. *IEEE Communications Surveys & Tutorials*, 25(1):213–250, 2022.

[69] Qihang Zhang, Zhenghao Peng, and Bolei Zhou. Learning to drive by watching youtube videos: Action-conditioned contrastive policy pretraining. In *European Conference on Computer Vision*, pages 111–128. Springer, 2022.

A Discussion

Task-optimized delivery vs. full reconstruction. KDC is designed to maximize downstream task utility under bandwidth and loss constraints, not to achieve pixel-perfect reconstruction. This is the intended operating point for latency-sensitive IoT applications where timely, task-relevant information is more valuable than bit-exact fidelity. KDC's progressive delivery and early-exit interface naturally support time-bounded tasks: the receiver can terminate reception once task confidence exceeds a threshold. When full-fidelity reconstruction is required (*e.g.*, archival storage, forensic analysis), KDC can be bypassed and the standard codec used alone; alternatively, the importance threshold τ can be set to 0 to transmit all tokens without semantic filtering.

Deployment safeguards. KDC assumes that Tx and Rx load identical SharedKB model versions. In practice, this is enforced by embedding model identifiers (name + version hash) in the KDC header exchanged at session initialization. If a version mismatch is detected, the system falls back to standard codec-only delivery. For gradual model upgrades, both sides can maintain a small set of supported versions and negotiate at connection time, similar to TLS cipher-suite negotiation.

Limitations. KDC's restoration relies on SharedKB priors that are domain-general but not domain-universal. For specialized sensor domains (*e.g.*, medical ultrasound, underwater sonar) where pretrained foundation models have limited coverage, the SharedKB may provide weak priors, leading to degraded restoration quality. In such cases, domain-specific fine-tuning of the SharedKB encoders is recommended before deployment.

Power and compute trade-offs. Our current evaluation focuses on throughput and latency rather than power consumption. KDC reduces communication power by transmitting significantly less data than traditional approaches, but introduces additional computational power consumption for semantic processing. This trade-off is hardware-dependent: on a 15 W Jetson Orin, KDC's compute overhead is small relative to radio power at typical 5G transmission levels, but the balance shifts on ultra-low-power LoRa devices. A systematic power analysis across hardware platforms is an important direction for future work.

Multi-task streams and modality–network pairings. The current evaluation uses a single downstream task per experiment. In multi-task deployments (*e.g.*, simultaneous object detection and tracking), KDC can partition the transmission budget across tasks by assigning importance scores from multiple bin sets and scheduling tokens by their maximum importance across tasks. We leave detailed multi-task scheduling policies to future work. The modality–network pairings in our evaluation (image/video over 5G, radar over WiFi, LiDAR over LoRa) reflect typical deployment assumptions and testbed practicality; the KDC mechanism itself is modality-agnostic and network-agnostic, as demonstrated by consistent gains across all tested configurations.

B Pseudocode

Algorithm 1 Tx: Task-Aware Token Selection

Require: Raw sensor data X, task description $\mathcal{T}$, threshold τ
Ensure: Prioritized bitstream $\mathcal{S}$

(Offline, once per session:)
$\mathcal{B} \leftarrow \textsc{BinExtract}(\mathcal{T}, \mathit{SharedKB})$ ▷ LLM + RAG
Cache $\mathcal{B}$ for session
(Per frame:)
$Z \leftarrow f(X)$ ▷ Modality-specific encoder
for each token $z_i \in Z$ **do**
    $\alpha_i \leftarrow \max_{b_j \in \mathcal{B}} \cos(z_i, t_j)$ ▷ Importance
end for
$\{\alpha_i\} \leftarrow \textsc{TempNorm}(\{\alpha_i\}, T_{\text{attn}})$
$M \leftarrow \textsc{ImportanceMap}(\{\alpha_i\}, \tau)$ ▷ Spatial mask
$X' \leftarrow \textsc{EntropyReduce}(X, M)$ ▷ Pre-quantize
$\mathcal{S} \leftarrow \textsc{Codec.Encode}(X')$ ▷ JPEG/H.264/PCC
return $\mathcal{S}$ with header $\{\mathcal{B}, M\}$

C Supplementary Tables

Algorithm 2 Rx: Knowledge-Conditioned Restoration

Require: Received bitstream $\tilde{\mathcal{S}}$, header $\{\mathcal{B}, M\}$
Ensure: Restored data $\hat{X}$
$\tilde{X} \leftarrow \textsc{Codec.Decode}(\tilde{\mathcal{S}})$
$e_{\text{text}} \leftarrow \textsc{TextEnc}(\mathcal{B})$
$\{r_k\} \leftarrow \textsc{PosteriorKB.Retrieve}(e_{\text{text}}, k)$ ▷ Top-k refs
$e_{\text{ctx}} \leftarrow \textsc{Concat}(\mathit{SharedKB}(\tilde{X}),\ \{r_k\})$
$\hat{X} \leftarrow \mathcal{R}(\tilde{X}, e_{\text{ctx}})$ ▷ FiLM-conditioned network
(Update PosteriorKB:)
$e_{\text{data}} \leftarrow \textsc{Enc}(\hat{X})$
$\textsc{PosteriorKB.Insert}(e_{\text{text}}, e_{\text{data}})$
return $\hat{X}$

Table 2: Baseline codec and configuration settings.

Baseline	Key Settings	Recovery
JPEG	Quality 75, baseline	None (fails on loss)
ProgJPEG	Quality 75, progressive	Native scan ordering
LimitNet	Pretrained, λ=0.01	Learned decoder
StarFish	Default config	Learned restoration
H.264	CRF 23, baseline profile	Frame-copy concealment
DVC	λ=256, default	None (latent drift)
PCC (MPEG)	Octree, QP=30	Missing-point zeroing
NPC	Default config	Neural interpolation

Table 3: Dataset summary for all evaluation experiments.

Modality	Dataset	Size	Task
Image	ImageNet-1K	50K val	Classification
Image	Cityscapes	5K val	Segmentation
Video	ACO YouTube Driving	10 clips	Sem. Segmentation
Radar	In-house MIMO	100 min	Posture Classif.
LiDAR	SemanticKITTI	23K scans	Sem. Segmentation

Table 4: Summary of wireless trace characteristics.

Network	Environment	Duration	Avg. Loss	Avg. Thr.
5G Cellular	Highway driving	∼1000 s	2–8%	5–35 Mbps
5G Cellular	Outdoor scooter	∼1000 s	3–12%	10–31 Mbps
WiFi 2.4/5 GHz	Indoor lab	∼600 s	1–5%	20–80 Mbps
LoRa	Indoor/Outdoor	∼2000 s	5–20%	5–50 kbps

Table 5: Hardware configurations for real-time experiments.

	Jetson Orin Nano	RTX 4090 Laptop
GPU	1024-core Ampere	AD102, 16384 CUDA
Memory	8 GB shared	16 GB GDDR6X
TDP	15 W	150 W
PyTorch	2.3	2.3
CUDA	12.4	12.4