Ultra-Low-Power Semantic Edge Intelligence: Ternary Multimodal Transformers on FPGA for IoT and Space Platforms

Sunny
Reconfigurable Space Computing Lab (RSCL)
Department of Electrical and Computer Engineering
California State Polytechnic University, Pomona

October 13, 2026

Abstract

This research introduces a novel ultra-low-power edge intelligence architecture that integrates ternary multimodal transformer-based models with semantic communication on a Field Programmable Gate Array (FPGA) for resource-constrained IoT and space platforms. By mapping heterogeneous inputs such as text, images, and scalar telemetry into a shared latent space via 1.58-bit ternary weights, the system bypasses traditional bit-exact transmission in favor of compact semantic tokenization. We investigate this framework through a dual contribution: first, by analyzing the foundational Bandwidth–Distortion–Energy trade-off triangle, quantifying how extreme model quantization and semantic compression reduce transmission overhead and energy consumption against acceptable reconstruction distortion under noisy wireless channels; and second, through a hardware–software co-design on FPGA fabric, where power-intensive floating-point multiplications are replaced by addition-based sparse logic to maximize throughput and minimize resource utilization. Designed to be generalizable across diverse bandwidth-limited environments, the approach enables real-time multimodal inference and substantially lowers power overhead for edge nodes. To operate under strict channel constraints, the architecture adds semantic joint source–channel coding (JSCC) trained over simulated AWGN and Rayleigh channels, a priority-aware attention-gating mechanism that reallocates bandwidth across modalities in real time. The same ternary backbone is reused for both transmit and receive paths, so a node can encode data for downlink and decode uplink commands and inter-satellite tokens from a single stored weight image.

Keywords: semantic communication, ternary neural networks, 1.58-bit quantization, FPGA, multimodal transformers, PocketQube, LoRa, edge AI.

1. Introduction

Sensing is scaling faster than communication. A single low-cost imager can produce tens of megabits per frame, while the radio links available to small satellites and battery-powered IoT nodes deliver kilobits per second. The conventional "capture everything, downlink everything" pipeline therefore fails on three fronts at once:

- **Bandwidth saturation.** High-resolution imaging and continuous telemetry quickly exhaust the limited radio-frequency spectrum.
- **Ground-infrastructure cost.** Backhauling raw data demands large, expensive ground-station networks.
- **Power budgets.** Processing and transmitting raw data exceeds what a small spacecraft or a coin-cell sensor can sustain.

A paradigm shift is needed in which the node decides *what is worth sending*, and sends its *meaning* rather than its bits. This is the premise of seman-

tic communication [2, 3]: the transmitter extracts task-relevant features, and the receiver reconstructs or acts on them, with fidelity measured against the task rather than bit-exactness.

Two obstacles keep this idea from reaching the smallest platforms. First, semantic encoders are neural networks, typically with floating-point arithmetic that is expensive in energy and silicon area. Second, most semantic-communication work treats the radio and the compute engine separately, even though on a small node the energy spent *thinking* and the energy spent *transmitting* come from the same tiny budget.

Contributions. This paper makes the following contributions:

(1) **A unified BDE framework.** We formalize the Bandwidth–Distortion–Energy trade-off for ternary-quantized semantic encoders over noisy channels (Section 3).

(2) **A ternary multimodal architecture.** We describe a transformer that maps text, images, and scalar telemetry into one shared latent space using $\{-1, 0, +1\}$ weights and emits compact semantic tokens (Section 4).

(3) **An FPGA co-design.** We replace floating-point multipliers with add/subtract/skip logic and exploit weight sparsity for throughput and resource savings (Section 8).

(4) **Semantic JSCC with dynamic allocation.** We train the ternary model over simulated noisy channels and add a priority-aware attention gate that reallocates bandwidth across modalities in real time (Sections 5 and 6).

(5) **A two-way shared codec.** We reuse one ternary backbone for transmit and receive, covering downlink, uplink, and inter-satellite links with a single stored weight image (Section 7).

(6) **A memory and reliability analysis.** We quantify weight and runtime RAM savings (Section 9) and show how the unused fourth state of the 2-bit ternary code can carry error correction at no extra storage (Section 10).

(7) **A deployment blueprint.** We instantiate the system in a PocketQube constellation with Sub-GHz links, in-fabric cryptography, and an inter-satellite mesh (Section 11), and give an evaluation plan (Section 12).

2. Background and Related Work

Semantic and task-oriented communication. Shannon separated the problem of conveying symbols from that of conveying meaning [1]. Modern deep joint source–channel coding (JSCC) revisits the question: learned encoders map images directly to channel symbols and degrade gracefully with channel quality [4], and transformer-based semantic systems extend this to text and multimodal data [3, 2]. Section 5 builds on this work. The rate–distortion–perception view [5] further shows that, at low rates, perceptual realism and pixel fidelity are in tension, which motivates generative reconstruction on the ground side.

Learned compression. Variational autoencoders with learned entropy models [6] already outperform classical codecs at equal quality, but their encoders are floating-point networks sized for GPUs rather than for flight hardware.

Extreme quantization. Binary and ternary networks restrict weights to one or two bits [7, 8, 9]. BitNet introduced 1-bit transformers trained from scratch [11]. The Microsoft Research BitNet b1.58 work then showed that large language models trained with ternary weights, the so-called 1.58-bit models, can match full-precision models in perplexity and zero-shot accuracy from a given scale while replacing multiply-accumulates with additions [10]. We carry this idea from language modeling to multimodal semantic encoding for edge and space use.

Ternary multimodal models. Sundaram and Iyer built LLaVaOLMoBitnet1B, described as the first ternary multimodal LLM able to accept images and text and produce coherent text responses [12]. It combines a vision encoder and a small MLP connector with a ternary OLMo-BitNet 1B language model, which holds more than 90% of the parameters. That result shows ternary backbones can handle image features. Our setting differs in three ways: the model is an *encoder* that emits channel tokens

rather than text, it also ingests scalar telemetry, and it is co-designed with an FPGA datapath and a noisy Sub-GHz link.

Why energy matters. Arithmetic energy is dominated by multipliers and memory access [18]. Table 1 lists representative per-operation energies; Figure 1 visualizes the resulting multiply-accumulate (MAC) cost. A ternary MAC needs only an integer add, subtract, or skip, which is the central hardware opportunity exploited here.

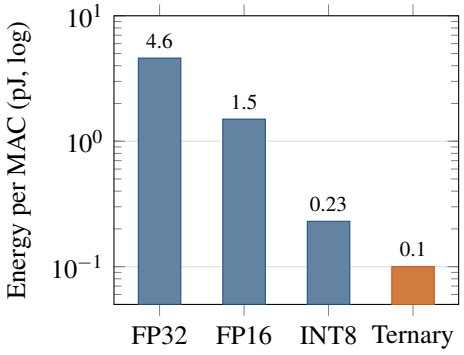


Figure 1: Energy per multiply-accumulate by data type (values from Table 1). Ternary weights remove the multiplier entirely, leaving only an integer accumulate.

3. The Bandwidth–Distortion–Energy Trade-off

Classical rate–distortion theory trades bits against fidelity. For a battery- or solar-limited node, a third axis is unavoidable: *energy*. Figure 2 places four candidate designs on the BDE triangle. Each corner is the *best* case for one axis (lowest bandwidth, lowest distortion, or lowest energy), and a point sits closer to a corner the better that design performs on that axis. Pushing toward any one corner costs the other two, and the center represents a balanced design. Section 3.4 explains how the coordinates are computed.

3.1 Formulation

Let $\mathbf{x}$ denote a multimodal observation (image, text, telemetry), f_θ the ternary encoder with weights $\theta \in \{-1,0,+1\}^N$, $\mathbf{z} = f_\theta(\mathbf{x})$ the semantic tokens, $R = |\mathbf{z}|$ the number of transmitted bits, g_ϕ the ground-side decoder, and h the channel with signal-to-noise ratio γ. The reconstruction is $\hat{\mathbf{x}} = g_\phi(h(\mathbf{z};\gamma))$. We seek

$$\min_{\theta,\phi} \ \mathbb{E}\big[d(\mathbf{x},\hat{\mathbf{x}})\big] \ + \ \lambda\, R \ + \ \mu\, E_{\text{tot}} \qquad \text{s.t.} \quad \theta \in \{-1,0,+1\}^N, \tag{1}$$

where $d(\cdot,\cdot)$ is a task-aware distortion (pixel, perceptual, or downstream-task loss), and $\lambda,\mu \geq 0$ weight bandwidth and energy. The total energy has a compute and a radio component:

$$E_{\text{tot}} \ = \ \underbrace{N_{\text{op}}\, e_{\text{op}}}_{E_{\text{comp}}} + \underbrace{P_{\text{tx}}\, T_{\text{air}}(R)}_{E_{\text{tx}}}, \qquad T_{\text{air}}(R) = \frac{R}{r_{\text{b}}}, \tag{2}$$

with N_{op} the operation count per inference, e_{op} the energy per operation (Table 1), P_{tx} the transmit power, and r_{b} the link bit rate.

3.2 Why the two terms interact

Quantizing the encoder lowers e_{op} and shrinks E_{comp}, but it can raise distortion at a given R. Compressing harder lowers E_{tx}, but demands a more capable encoder and may raise N_{op}. On a LoRa link, E_{tx} is large because the airtime is long: at spreading factor 7 with 125 kHz bandwidth the raw bit rate is roughly 5.5 kbps, and it falls to about 0.3 kbps at spreading factor 12. Cutting R therefore pays off disproportionately, and a semantic encoder is worth its compute cost whenever

$$N_{\text{op}}\, e_{\text{op}} \ < \ P_{\text{tx}}\, \frac{R_{\text{raw}} - R_{\text{sem}}}{r_{\text{b}}}. \tag{3}$$

Ternary weights shrink the left-hand side by roughly an order of magnitude relative to FP16 (Figure 1), widening the regime in which inequality (3) holds.

3.3 Illustrative airtime budget

Figure 3 compares the time on air needed to deliver one 1024×1024 RGB frame at SF7 under three strategies, ignoring protocol overhead and duty-cycle limits. Raw transmission is about 25 Mb; a typical lossy-codec payload is on the order of 1 Mb; and a semantic payload of 256 tokens at 12 bits each is about 3 kb. These are planning numbers for scale, not measurements.

Data type	Multiply (pJ)	Add (pJ)	MAC (pJ)
FP32	3.7	0.9	4.6
FP16	1.1	0.4	1.5
INT8	0.2	0.03	0.23
Ternary (INT32 accumulate)	—	0.1	≈0.1 (add/sub/skip)

Table 1: Representative 45 nm per-operation energies after Horowitz [18]. Absolute values differ on FPGA fabric and newer nodes; the ordering and ratios are what motivate the design.

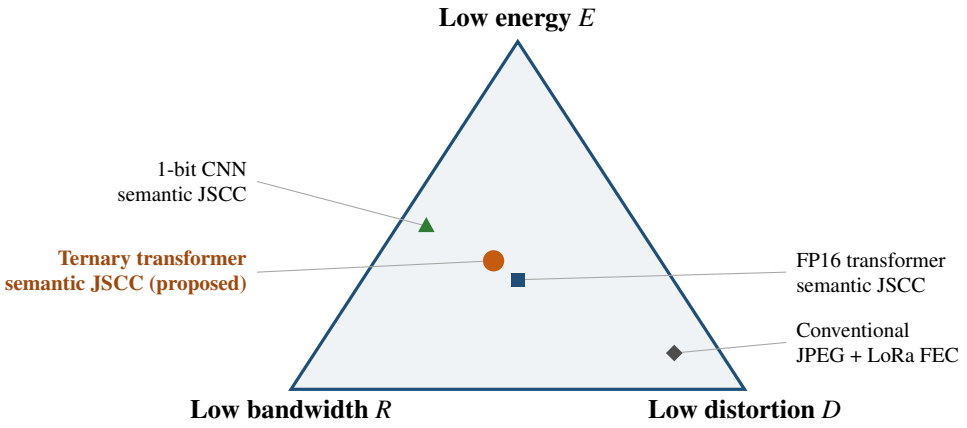


Figure 2: The BDE triangle with four design points. A point is closer to a corner when the design is better on that axis. Conventional coding hugs the distortion corner but is far from the bandwidth and energy corners. The FP16 transformer is balanced but costs compute energy. The 1-bit CNN is cheapest to compute but pays in distortion. The ternary transformer moves toward the bandwidth and energy corners while keeping distortion moderate. Positions are computed from the illustrative numbers in Table 2.

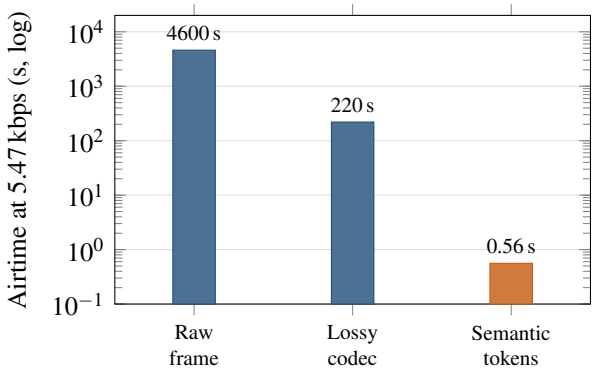


Figure 3: Illustrative time on air to deliver one 1024×1024 RGB frame over LoRa (SF7, 125 kHz). The semantic payload is about four orders of magnitude smaller in airtime than the raw frame.

3.4 Comparing design points on the triangle

To see where the proposed design sits, we compare it with three alternatives under one set of assumptions:

- **Conventional (non-semantic):** lossy image coding (JPEG-class) with LoRa forward error correction.
- **FP16 transformer semantic JSCC:** the same pipeline as ours but with a full-precision encoder.
- **1-bit CNN semantic JSCC:** a convolutional encoder with binary weights, representing a different network type at an even lower precision.

Assumptions are a 1024×1024 RGB frame sent over LoRa at SF7 (5.47 kbps, ignoring protocol overhead) with a transmit power of $P_{\text{tx}} = 0.5$ W. The transformers perform 86.4 G operations per frame (300M parameters, 288 tokens) at the per-operation energies of Table 1. Off-chip weight reads cost 0.16 nJ per byte, from roughly 1.3 nJ per 64-bit DRAM access [18]. The CNN is assumed to have 50M parameters and 20 G binary operations at 0.05 pJ each. Payload sizes and distortions fol-

low the illustrative curves of this section. Table 2 lists the resulting per-frame figures; every entry is a planning estimate, not a measurement.

Placing the points. For each axis $a \in \{R, D, E\}$ with cost x_a for a given design, let $g_a = 1 - \frac{\ln x_a - \ln x_a^{\min}}{\ln x_a^{\max} - \ln x_a^{\min}}$ be its log-scaled goodness across the four designs ($g_a = 1$ for the best, 0 for the worst). The barycentric weight is $w_a = (g_a + \varepsilon) / \sum_{a'} (g_{a'} + \varepsilon)$ with $\varepsilon = 0.15$, which keeps points off the edges. Positions are therefore relative to the designs compared, and adding another design changes the scaling.

Reading the result.

- *Conventional coding* has the best distortion at good SNR but sends over 390× more bits and spends over 100 J per frame on the radio, so it sits near the distortion corner and far from the others. Below its design SNR it also loses its distortion advantage (Figure 7).
- *FP16 semantic coding* matches the bandwidth and distortion of the best case, but its compute and weight traffic (0.23 J) rival the radio cost, and a 600 MB weight image is hard to host on a PocketQube (Section 9).
- *The 1-bit CNN* has almost no compute cost, but its higher distortion requires a larger payload to compensate, which raises radio energy. Its total is therefore higher than the ternary design despite the cheaper arithmetic.
- *The ternary transformer* gives up some distortion relative to FP16 (0.028 versus 0.020) in exchange for roughly 10× less compute and memory energy, reducing total energy per frame by about 40% (0.30 J versus 0.51 J).

At a 0.5 W radio and 5.47 kbps the radio term (0.28 J) dominates all three semantic designs, so the ternary advantage in total energy is modest per transmitted frame. It grows when the link is cheaper (higher data rate, lower transmit power) or when the node runs inference more often than it transmits, as in on-board screening and anomaly detection. This is why the evaluation plan reports compute and radio energy separately (Section 12).

3.5 Distortion versus rate: semantic, lossy, and lossless payloads

Figure 4 compares the distortion–rate behavior of the three semantic encoders with a conventional lossy codec and with lossless coding, on log–log axes so that payloads spanning five orders of magnitude and distortions spanning two can be read together. The semantic curves fall with rate, then flatten at a floor set by the encoder's representational capacity. Full-precision encoders have the lowest floor; ternary encoders trained with quantization-aware training (QAT) sit slightly above it; and ternary encoders obtained by post-training quantization sit well above, which is why we train ternary from the start [10]. The lossy codec curve keeps improving as bytes are added and approaches zero distortion near the lossless point, but it needs far more bytes to reach the same distortion: at $D = 0.03$ the ternary semantic encoder needs about 290 B, whereas the lossy codec needs about 90 kB, roughly 310× more. Lossless coding reaches $D = 0$ at about 1.6 MB per frame, which is about 4,200× the semantic payload of 384 B and corresponds to roughly 39 minutes of continuous airtime at 5.47 kbps. All curves are analytic illustrations to be replaced by measured results (Section 12).

3.6 Robustness to the channel

Because the encoder is trained end-to-end with a differentiable channel layer (additive noise, fading, and packet erasure), the learned tokens are redundant in a graceful way: lowering γ increases D smoothly rather than causing a cliff, in contrast with bit-exact codecs whose output is destroyed beyond the error-correction threshold [4]. A thin outer error-correcting code protects the small subset of tokens (headers, cryptographic tags) that must arrive intact.

4. Ternary Multimodal Semantic Architecture

Figure 5 gives the end-to-end data path. Heterogeneous inputs pass through lightweight modality front-ends into one shared transformer backbone; a semantic tokenizer then quantizes the latent into a short token sequence that is encrypted and sent over Sub-GHz LoRa.

Design	Payload	Airtime	D	Compute + memory (J)	Total (J)
Conventional + FEC	150 kB	219 s	0.020	0.010	109.7
FP16 transformer	384 B	0.56 s	0.020	0.226	0.51
1-bit CNN	512 B	0.75 s	0.055	0.002	0.38
Ternary transformer	384 B	0.56 s	0.028	0.021	**0.30**

Table 2: Illustrative per-frame comparison. Total energy is compute, memory traffic, and radio ($P_{\text{tx}}T_{\text{air}}$) combined. Conventional distortion holds only above its design SNR (Figure 7).

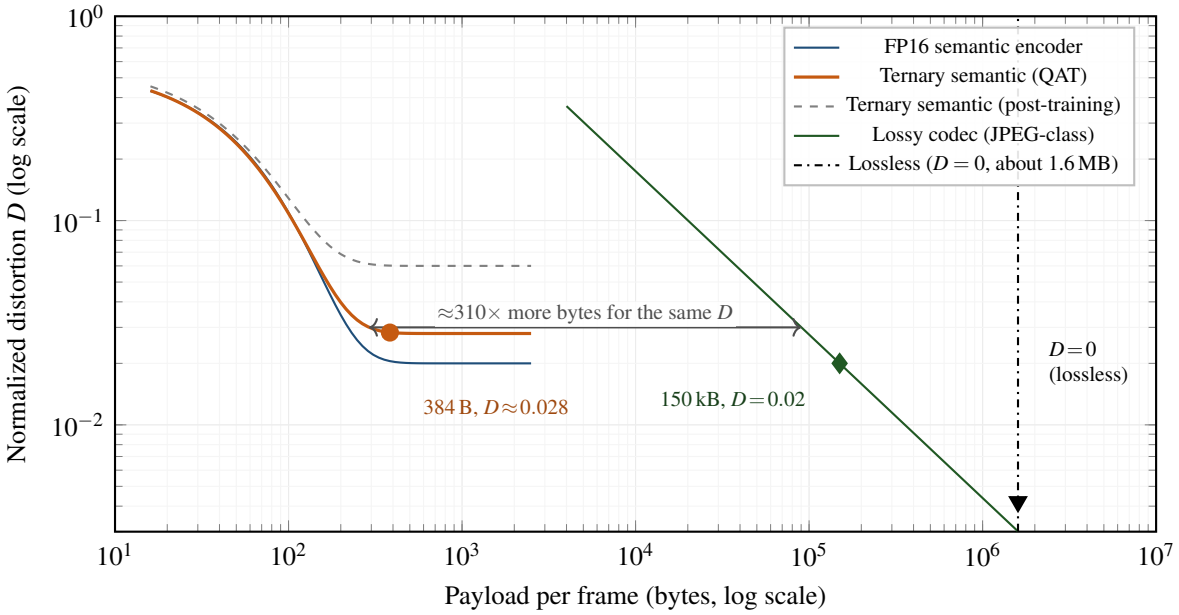


Figure 4: Distortion versus payload per frame for a 1024×1024 RGB image (raw size 3.15 MB). The three semantic encoders flatten at their capacity floors within a few hundred bytes. The lossy codec curve is anchored at 150 kB with $D = 0.02$. Lossless coding is shown at an assumed 2:1 ratio, giving about 1.6 MB. Curves are analytic illustrations, not measured data.

4.1 Modality front-ends and shared latent space

Each modality is projected to a common embedding dimension d so that the backbone can attend across them:

- **Images:** a patch-embedding layer splits the frame into $P \times P$ patches, each flattened and linearly projected to $\mathbb{R}^d$.
- **Text and commands:** a compact subword vocabulary is embedded into $\mathbb{R}^d$.
- **Scalar telemetry:** each channel (temperature, voltage, attitude, and so on) is normalized, then mapped by a small learned projection, optionally with a Fourier feature expansion to capture periodic signals.

Learned modality-type embeddings are added so the shared self-attention can distinguish sources. Because all modalities live in one latent space, a single model serves every sensor, and cross-modal context (for example, using telemetry to disambiguate an image) comes for free.

4.2 Ternary transformer backbone

Every linear layer in the backbone uses ternary weights. Following the absmean scheme [10], a full-precision latent weight matrix W is quantized

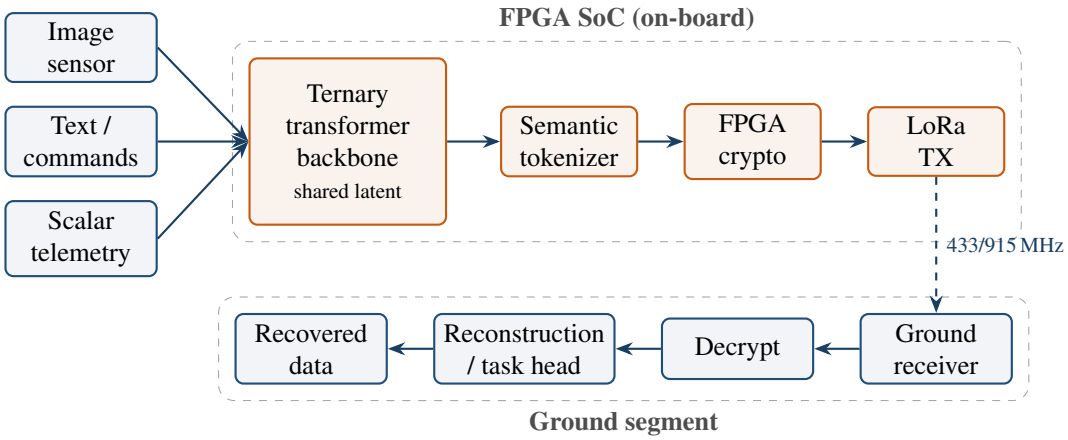


Figure 5: End-to-end data path. Orange blocks run in FPGA fabric. Three modalities share one ternary backbone; only encrypted semantic tokens cross the link.

on the forward pass as

$$\begin{aligned} \tilde{W} &= \mathrm{RoundClip}\Big(\frac{W}{\gamma_W + \varepsilon}, -1, 1\Big), \\ \gamma_W &= \frac{1}{nm}\sum_{i,j} |W_{ij}|, \end{aligned} \tag{4}$$

while gradients flow to W through a straight-through estimator. Activations are quantized to 8 bits per token. At inference time only $\tilde{W}$ and the scalar γ_W per layer are stored, so a matrix–vector product becomes

$$y_j = \gamma_W \Big(\sum_{i \in \mathscr{P}_j} x_i - \sum_{i \in \mathscr{N}_j} x_i \Big), \tag{5}$$

where $\mathscr{P}_j$ and $\mathscr{N}_j$ are the index sets of $+1$ and -1 weights in row j. Indices with $w = 0$ are simply skipped.

In pure weight storage a ternary model is 8–10× smaller than FP16: roughly 75–125 MB for 300M–500M parameters at 2 bits per weight, against 600–1,000 MB at FP16. Section 9 gives the full accounting, including mixed-precision layers and runtime RAM. Practical deployments may begin with a much smaller backbone and scale up as the platform allows.

4.3 Semantic tokenization and channel mapping

The backbone's output latent $\mathbf{h} \in \mathbb{R}^{L \times d}$ is reduced to $K \ll L$ tokens by a learned pooling layer, and each token is vector-quantized against a codebook of 2^b entries (for example $b = 12$), producing a sequence of discrete indices. Entropy coding of the indices, conditioned on a prior shared with the ground decoder, removes remaining redundancy. The result is a payload of Kb bits or fewer, packed into LoRa frames. Token importance can be ranked so that, under a poor link, only the most informative tokens are sent first (progressive transmission).

4.4 Ground-side reconstruction

The ground segment holds a larger, full-precision decoder that has no power constraint. It can either reconstruct the input modality (an image decoder, possibly generative to improve perceptual quality [5]) or feed the tokens directly into a task head (classification, detection, anomaly flagging) without reconstructing pixels at all. Task-only decoding yields the largest bandwidth savings, since the payload can be tuned to the task rather than to the picture. A ground terminal may instead run the same ternary network that flies on the satellite (Section 7), trading some reconstruction quality for a much cheaper receiver.

5. Semantic Joint Source–Channel Coding

5.1 Why code jointly

Shannon's separation theorem says that compressing and channel-coding independently loses nothing, but only in the limit of unbounded block length and delay. A small node sending a few hundred bytes per frame over a fading link is far from that limit.

Separate designs also suffer from the *cliff effect*: a channel code sized for a target SNR works until the link drops below that threshold, after which the output collapses [4]. Deep JSCC replaces both blocks with one learned mapping that degrades gracefully with SNR. It has since been extended to progressive and bandwidth-adaptive transmission [13], SNR-adaptive attention modules [14], and entropy-model-guided rate adaptation [15]. We adopt this line of work and combine it with ternary weights and multimodal inputs.

5.2 Training through simulated channels

During training, a differentiable channel layer sits between encoder and decoder (Figure 6). For latent symbols $\mathbf{z} \in \mathbb{C}^k$ with average power constraint $\frac{1}{k}\|\mathbf{z}\|^2 \le P$, the received signal is

$$\begin{aligned} \mathbf{y} &= h\mathbf{z} + \mathbf{n}, \quad \mathbf{n} \sim \mathscr{CN}(\mathbf{0}, \sigma^2\mathbf{I}), \\ \gamma &= \frac{P}{\sigma^2}, \end{aligned} \tag{6}$$

with $h = 1$ for AWGN and $h \sim \mathscr{CN}(0,1)$, constant over a block, for Rayleigh fading. The end-to-end objective averages over a range of channel conditions so that one model serves the whole operating envelope:

$$\begin{aligned} \mathscr{L}(\theta, \phi) = \mathbb{E}_{\mathbf{x}} \mathbb{E}_{\gamma \sim p(\gamma)} \mathbb{E}_{h,\mathbf{n}} \Big[\\ d\big(\mathbf{x}, g_\phi(h f_\theta(\mathbf{x}) + \mathbf{n}; \gamma)\big) \Big]. \end{aligned} \tag{7}$$

Here γ is drawn from a distribution covering the expected link range (for example 0–20 dB), and the decoder receives an estimate of γ as side information, in the spirit of SNR-adaptive attention modules [14]. Gradients pass through the ternary weight quantizers, and through any ternary latent quantizer, by the straight-through estimator described in Section 4. Because the channel is part of the training graph, the encoder learns to place redundancy where the decoder needs it, rather than relying on a separately designed error-correcting code.

5.3 Why ternary latents resist channel degradation

A ternary latent maps each symbol $s \in \{-1, 0, +1\}$ to one of three amplitude levels $\{-a, 0, +a\}$. With equiprobable symbols, average power is $P = \frac{2}{3}a^2$. A nearest-level receiver places decision thresholds at $\pm a/2$, so any noise smaller than half the level spacing is removed completely: the quantizer acts as a denoiser. With per-dimension noise standard deviation σ_r, the symbol error probability is

$$\begin{aligned} P_e &= \tfrac{1}{3} Q\Big(\tfrac{a}{2\sigma_r}\Big) + \tfrac{1}{3} Q\Big(\tfrac{a}{2\sigma_r}\Big) \\ &\quad + \tfrac{1}{3} \cdot 2Q\Big(\tfrac{a}{2\sigma_r}\Big) \\ &= \tfrac{4}{3} Q\Big(\tfrac{a}{2\sigma_r}\Big), \end{aligned} \tag{8}$$

where $Q(\cdot)$ is the Gaussian tail function. Below threshold, degradation is therefore not a gradual smearing of every value but a sparse set of discrete flips, each moving a symbol to an adjacent level. A decoder trained against exactly this error pattern, by sampling discrete flips in the channel layer, learns to tolerate it. Hard decisions do discard soft information, so a soft-input decoder fed with per-symbol log-likelihoods is a design option that trades some decoder complexity for extra robustness near the threshold. We treat the robustness of ternary latents as a hypothesis to be tested in Section 12, not an assumed result.

5.4 A LoRa-compatible channel surrogate

LoRa uses chirp spread spectrum with its own whitening and forward error correction, so arbitrary analog amplitudes cannot be placed on the air as in Equation (6). We therefore treat the radio as a digital pipe: token indices are packed into bits, bits into LoRa symbols of SF bits each, and the demodulator returns a symbol, correct or not. The training-time channel layer is then a symbol-error surrogate in which each transmitted symbol is replaced by a random wrong symbol with probability $p_s(\mathrm{SNR}, \mathrm{SF})$, taken from a characterization of the actual radio. Two design choices make this surrogate useful:

- **Semantic index assignment.** Codebook indices are assigned so that indices differing in few bits decode to embeddings that are close in meaning. A flipped bit then produces a near-miss rather than an unrelated token, which is the discrete analogue of graceful degradation.
- **Selective outer protection.** Only fields that must arrive intact (headers, authentication tags,

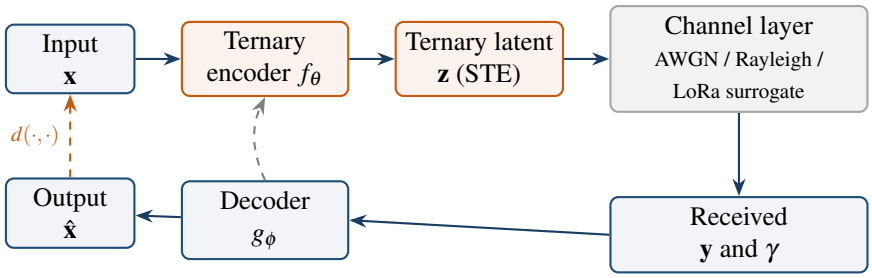


Figure 6: Joint source–channel training loop. The channel layer is differentiable, so distortion gradients (dashed) flow back through the channel to the ternary encoder.

and critical telemetry) receive an additional outer error-correcting code. Image tokens rely on the learned robustness.

Where an SDR-based radio is available, the same framework carries over to direct analog symbol mapping as in the original Deep JSCC work [4].

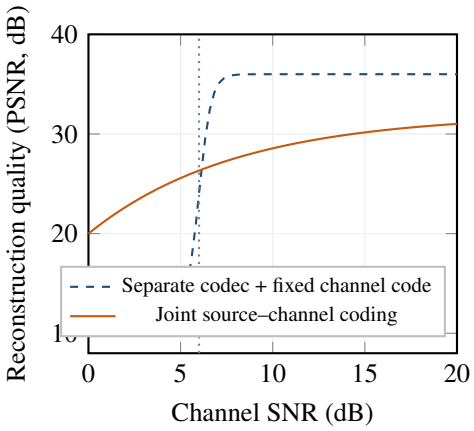


Figure 7: Illustrative cliff effect versus graceful degradation (analytic curves, not measured data). The separate design is best above its design SNR and collapses below it; the joint design degrades smoothly.

6. Dynamic Cross-Modal Bandwidth Allocation

6.1 Motivation

Modalities differ sharply in both size and urgency. Scalar telemetry is a few bytes but may signal a fault; images are large but are rarely time-critical. A static split wastes capacity in normal operation, or starves telemetry exactly when an anomaly makes it most valuable. We therefore make the split a function of the node's own observations.

6.2 Budgets and progressive layers

At each transmission window t, the link controller computes a bit budget $B(t) = r_{\mathrm{b}}\, T_{\mathrm{win}}\, \eta(t)$ from the current spreading factor, estimated SNR, and remaining duty-cycle credit $\eta(t)$. Modalities $m \in \{\mathrm{tel}, \mathrm{txt}, \mathrm{img}\}$ receive bit allocations $R_m(t)$ with $\sum_m R_m(t) \le B(t) - R_{\mathrm{hdr}}$. Image tokens are organized in progressive layers (a base layer followed by successive refinements), so that dropping later layers lowers quality gracefully instead of corrupting the frame, as in bandwidth-agile Deep JSCC [13].

6.3 Priority score and allocation

A lightweight anomaly scorer reads the telemetry tokens and produces a priority score $s_t \in (0,1)$. A simple choice is a smoothed, normalized deviation,

$$s_t = \sigma\Big(\tfrac{1}{\kappa}\Big[\max_c \tfrac{|x_c(t)-\mu_c|}{\sigma_c} - \tau_a\Big]\Big) \tag{9}$$

where μ_c, σ_c are per-channel nominal statistics, τ_a is an anomaly threshold, and a learned head can replace it. Allocation weights follow from base logits $\mathbf{u}$ and an anomaly-sensitivity vector $\mathbf{v}$ with $v_{\mathrm{tel}} > 0$ and $v_{\mathrm{img}} < 0$:

$$\begin{aligned} \mathbf{w}(t) &= \mathrm{softmax}\Big(\frac{\mathbf{u} + s_t\,\mathbf{v}}{T}\Big), \\ R_m(t) &= R_m^{\min} \\ &\quad + w_m(t)\Big(B(t) - R_{\mathrm{hdr}} - \sum_{m'} R_{m'}^{\min}\Big). \end{aligned} \tag{10}$$

The floors $R_m^{\min}$ guarantee a telemetry heartbeat at all times. The image floor may be zero in critical states.

6.4 Priority-aware attention gating

The same priority signal shapes the transformer itself. A per-modality gate $g_m = w_m / \max_{m'} w_{m'}$ adds a bias to the attention logits,

$$\alpha_{ij} = \frac{\exp\big(e_{ij} + \log g_{m(j)}\big)}{\sum_{j'} \exp\big(e_{ij'} + \log g_{m(j')}\big)}, \qquad e_{ij} = \frac{\mathbf{q}_i^\top \mathbf{k}_j}{\sqrt{d_h}}, \tag{11}$$

so that low-priority keys receive less attention mass when the priority state rises. Token importance is the attention each token receives, $\rho_j = \frac{1}{HL}\sum_h \sum_i \alpha_{ij}^{(h)}$, and each modality transmits its top $\lfloor R_m/b \rfloor$ tokens by ρ_j. Image tokens ranked lower fall into later progressive layers and are the first to be dropped. In hardware the gate is one precomputed additive term per key, so the mechanism adds almost no logic.

6.5 Control loop

Each window proceeds as follows:

(1) Read link state and compute $B(t)$.

(2) Update the smoothed anomaly score s_t with hysteresis thresholds, to avoid oscillation.

(3) Compute $\mathbf{w}(t)$ and the allocations $R_m(t)$ from Equation (10).

(4) Run the backbone with gate biases, then rank tokens by ρ_j.

(5) Select tokens per modality, assign image tokens to progressive layers, and drop layers beyond R_{img}.

(6) Packetize in priority order (telemetry, text, image base layer, image refinements), encrypt, and transmit.

(7) If $s_t \geq s_{\text{crit}}$, flush queued image refinement packets and send telemetry with outer error correction in the next slot.

When the anomaly clears, image transmission resumes from the next unsent refinement layer, since progressive coding means no earlier work is repeated.

7. Bidirectional Operation with a Shared Ternary Codec

7.1 From one-way to two-way

The architecture described so far is one-way: the satellite encodes and transmits, and a ground-side decoder reconstructs. Many real operations are two-way. A ground operator sends commands and queries up, satellites exchange tokens over inter-satellite links (ISLs), and low-cost ground terminals may need to decode downlinked tokens without a large model. Table 3 lists the three links and what each end must decode.

Running a separate decoder network for each direction would double the weight image and its protection and scrubbing overhead. We instead reuse the *same ternary backbone* for transmit and receive on the satellite, and optionally on the ground.

7.2 One backbone, two modes

Let B_θ be the ternary transformer backbone of Section 4. Transmit and receive differ only in the light layers around it:

$$E_\theta(\mathbf{x}) = \mathscr{Q}\big(B_\theta(F(\mathbf{x}) + \mathbf{e}_{\text{enc}})\big), \tag{12}$$

$$D_\theta(\mathbf{y}) = H\big(B_\theta(\mathscr{Q}^{-1}(\mathbf{y}) + \mathbf{e}_{\text{dec}})\big), \tag{13}$$

where F stands for the modality front-ends, $\mathscr{Q}$ for the semantic tokenizer (pooling and vector quantization), $\mathscr{Q}^{-1}$ for codebook lookup that turns received indices back into embeddings, H for output heads, and $\mathbf{e}_{\text{enc}}, \mathbf{e}_{\text{dec}}$ for learned mode embeddings added like the modality-type embeddings of Section 4. The backbone weights θ are stored once. Per-mode normalization scale and bias parameters are small and held at higher precision. In the notation of Section 5, $f_\theta = E_\theta$ and $g_\phi = D_\theta$ when the decoder backbone is shared.

Joint training. Each link has its own channel h_l with its own SNR distribution γ_l (the uplink typically has a stronger transmitter than the downlink, and ISLs differ again). The shared weights are trained

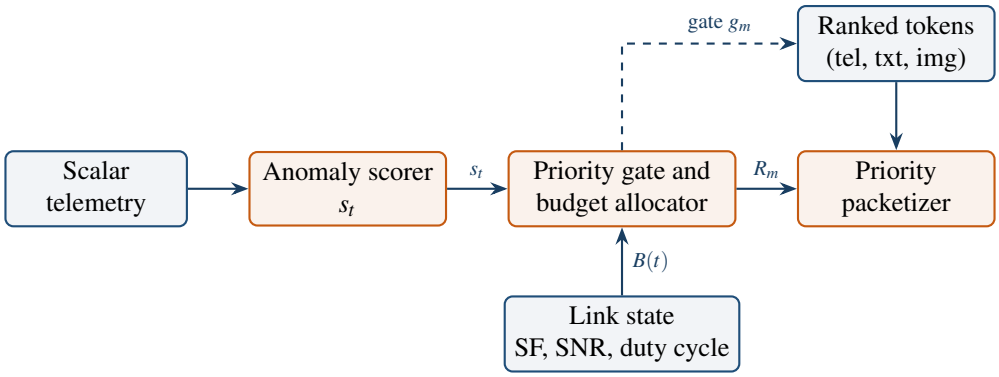


Figure 8: Bandwidth allocation controller. Telemetry drives the priority score, the link state sets the budget, and the resulting gate both biases attention and sets per-modality bit allocations.

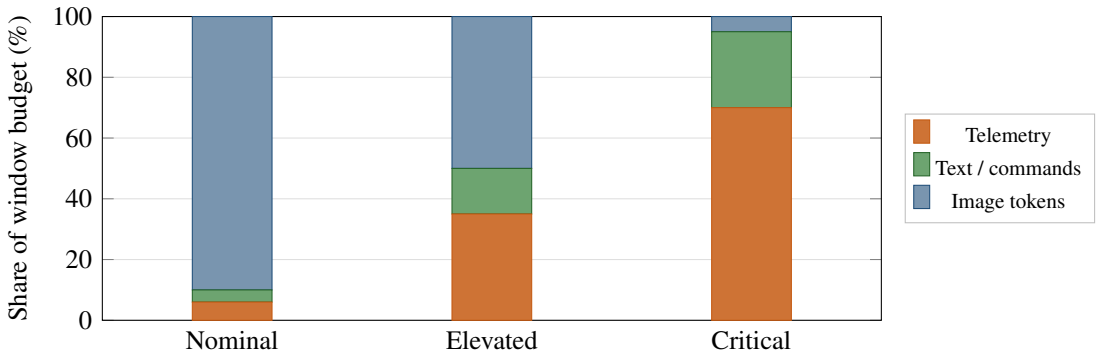


Figure 9: Illustrative reallocation of one window's bit budget as the anomaly state rises. Image refinement layers are throttled first; telemetry is never below its floor. Values are design examples, not measurements.

on all links at once:

$$\mathscr{L}(\theta) = \sum_{l \in \{\mathrm{dn}, \mathrm{up}, \mathrm{isl}\}} \alpha_l \cdot \mathbb{E}_{\mathbf{x}, \gamma_l, \mathbf{n}} \Big[d_l \big(\mathbf{x}, D_\theta \big(h_l(E_\theta(\mathbf{x}); \gamma_l) \big) \big) \Big], \tag{14}$$

with mode-specific heads and tokenizers carrying their own small parameter sets. A ground station is free to use a different, larger decoder, provided it is trained against the frozen satellite encoder E_θ; both consume the same tokens.

7.3 Design options and storage

Table 4 compares three ways to realize two-way operation for the 300M model. Mode-specific heads are assumed to hold about 5% of the parameters (15M).

Variant B saves between 30% and 48% of the weight storage of variant A, and, as importantly, keeps a single image to protect with the embedded ECC of Section 10. We treat B as the default and evaluate A and C as ablations (Section 12).

7.4 Hardware: one PE array, time-multiplexed

The ternary PE array of Section 8 executes the same layers in either mode, so a duplex node needs one array, one weight stream, and one set of tile buffers. A small mode controller selects the front-end or head, the mode embedding, and the tokenizer or detokenizer. Figure 10 shows the data paths.

Three details matter:

- **Authenticate before decoding.** On the receive path the authentication tag is verified and the payload decrypted before any token reaches the network, so unauthenticated bytes cannot steer the model or spoof a command. The outer error-correcting code of Section 5 is applied at the same stage.

Link	Encoder	Decoder	Decoded output
Downlink	Satellite	Ground (full-precision or ternary)	Pixels, text, telemetry, or task labels
Uplink	Ground	Satellite (ternary)	Commands, queries, threshold updates
ISL	Peer satellite	Satellite (ternary)	Latents for fusion, consensus, or relay

Table 3: Links and decoders in a two-way constellation.

Variant	Weights (MB)	Notes
A. Separate ternary encoder and decoder	150	Best per-direction quality; weight image, ECC, and scrubbing doubled
B. Shared backbone + mode heads (proposed)	≈79 (ternary heads), ≈105 (FP16 heads)	One protected weight image; small heads
C. Weight-tied, decoder uses transposed layers	75	The transpose of a ternary matrix is ternary, so reuse is free, but it constrains the decoder

Table 4: Weight storage for two-way operation (300M model, 2 bits per ternary weight including ECC).

- **Receive is cheaper than transmit for commands.** A command is a short token sequence, so decoding it costs a small fraction of imaging inference (Table 5). Sixteen 12-bit tokens are 24 B, or about 35 ms of airtime at 5.47 kbps.
- **Preemption.** A valid command arriving mid-inference is handled either by finishing the current frame or by preempting at a layer boundary. The 16-token command pass needs only about 115 KB of activation memory at the 300M scale, while the interrupted frame's 0.59 MB residual stream is parked in spare block RAM or external memory.

7.5 Compute cost per mode

A ternary ground terminal running the same decode costs the same 7.68 mJ per 256-token frame as the ISL case, which makes a low-power, low-cost receiver feasible; Table 3 marks this as an option, with a larger full-precision decoder available when the best reconstruction quality is needed.

7.6 Risks

Sharing capacity between compression and expansion may lower quality relative to dedicated networks, and the effect is likely strongest for pixel reconstruction, which is the heaviest decoding task. Joint training over several links with different SNR distributions is harder to tune than a single link. We therefore treat the shared design as a hypothesis, and the ablations in Section 12 compare it with separate networks (variant A) and weight tying (variant C).

8. FPGA Hardware–Software Co-Design

8.1 Multiplier-free processing elements

On an FPGA, an integer multiplier consumes either DSP slices or many LUTs, whereas an adder maps to a carry chain. With ternary weights each processing element (PE) reduces to a two-bit weight decode, a sign-controlled add/subtract, and an accumulator. Figure 11 shows the datapath. The weight is the only ternary quantity: the activations it acts on are ordinary low-bit integers, and the running sum is a wider integer, as explained below the figure.

Why integers appear in a ternary design. Ternary describes the *weights*. The values flowing between layers still have to be represented somehow, and we follow BitNet b1.58 in using 8-bit integer activations [10]. Summing n such activations produces a result up to $127n$ in magnitude, so the running sum needs a signed width of $\lceil \log_2(127n) \rceil + 1$ bits to be exact: 20 bits at $n = 4096$ (the feed-forward

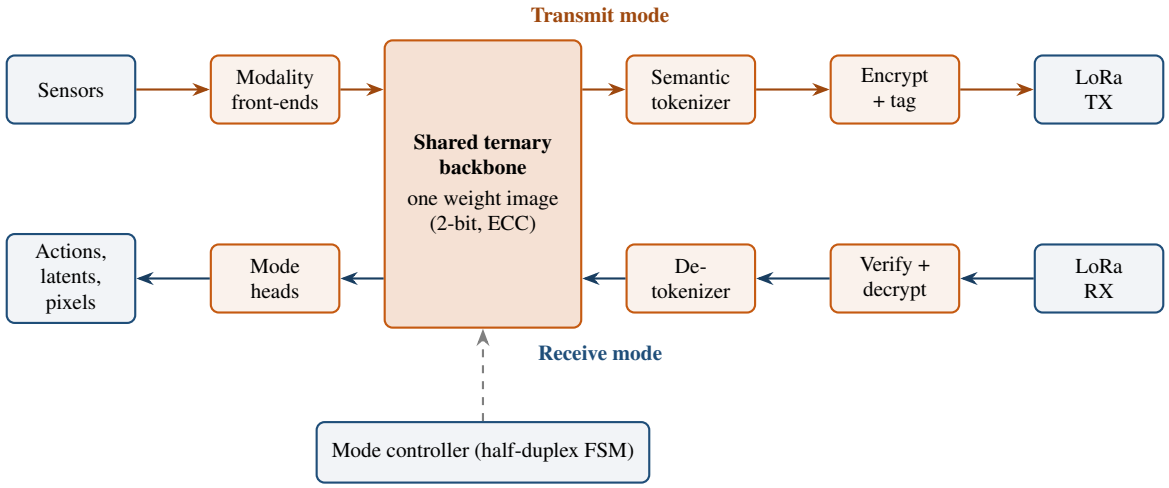


Figure 10: Duplex node. The same ternary backbone serves both paths. Orange arrows are the transmit path and blue arrows the receive path. Because LoRa transceivers are half-duplex, the two modes never run at the same time and share the PE array without loss of throughput.

Mode	Tokens	Ops (G)	Ternary (mJ)	FP16 (mJ)
Encode image frame (downlink)	288	86.4	8.64	129.6
Decode peer tokens (ISL fusion)	256	76.8	7.68	115.2
Decode command (uplink)	16	4.8	0.48	7.2

Table 5: Arithmetic energy per pass for the 300M model, using the per-operation energies of Table 1 (45 nm, one operation per parameter per token). Memory traffic and radio energy are excluded.

down-projection of the 300M model) and 21 bits at $n = 6144$ (the 500M model). Table 1 conservatively prices this accumulate as a full 32-bit integer add, so the energy figures in this paper are, if anything, pessimistic for the ternary design. These integers are not a departure from the ternary approach. What ternary weights remove is the per-weight multiplier, which dominates the cost of the floating-point or INT8 alternatives, and the PE still has none. The one multiplication that remains is the per-layer scale, applied once per output element and implementable as a shared multiplier or, if the scale is constrained to a power of two, a bit shift.

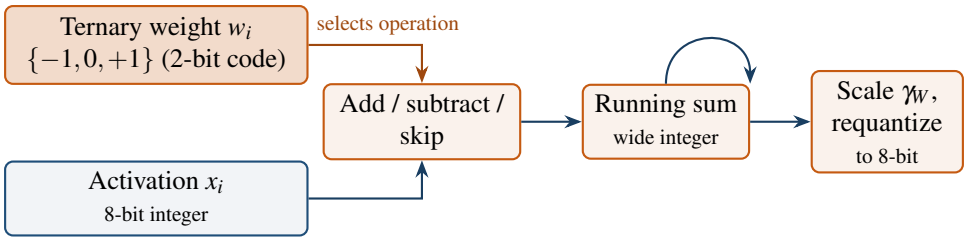
8.2 Exploiting sparsity

Because roughly one-third or more of ternary weights are zero in trained models, the accelerator stores weights in a packed format and processes only the non-zero entries when beneficial. Two complementary mechanisms apply:

- **Zero-skipping:** row-wise index lists let the controller issue cycles only for non-zero weights, raising effective throughput.
- **Clock/operand gating:** when a dense schedule is used, a zero weight gates the adder input and accumulator enable, lowering switching activity.

8.3 Dataflow and memory hierarchy

Energy on small devices is often dominated by data movement rather than arithmetic [18]. The design therefore uses a weight-streaming dataflow: packed ternary weights stream from external flash into small on-chip block RAM (BRAM) tiles, activations are held in BRAM, and partial sums stay in registers. Since weights are only 1.58–2 bits each, memory bandwidth per MAC is several times lower than for INT8 designs. Layer scheduling is static, so the control logic is a small finite-state machine.



No multiplier per weight. The layer scale γ_W is applied once per output element, not once per weight.

Figure 11: A ternary processing element. Only the weight is ternary: its 2-bit code selects add, subtract, or skip, so no per-weight multiplier is needed. Activations are 8-bit integers and the running sum is a wider integer so long dot products cannot overflow. A zero weight gates the accumulate, saving dynamic power.

8.4 Software flow

The offline flow comprises (i) ternary QAT in a high-level framework, (ii) export of packed weights and per-layer scales, (iii) hardware generation parameterized by PE count, tile size, and sparsity mode, and (iv) co-simulation against the software model to verify bit-exact agreement. The PE array size is the main knob trading throughput, resource utilization, and power.

8.5 Radiation considerations

For orbital use, configuration memory is vulnerable to single-event upsets. Mitigations include configuration scrubbing, triple modular redundancy on control logic, and error-correcting codes on weight storage. Ternary weights are expected to be comparatively tolerant to individual bit flips because each weight carries little magnitude information, and the model can be periodically re-verified against a checksum. Section 10 describes a storage encoding that adds single-error correction to the weights at no extra storage cost.

9. Memory Footprint Analysis

Memory is the binding constraint on a small spacecraft: weights must be held in non-volatile storage, and the working set must fit in on-chip block RAM (BRAM) or a small external SRAM/DDR RAM.

9.1 Weight storage

For N parameters at b bits each, storage is $Nb/8$ bytes. Real models are mixed precision: embeddings, normalization parameters, the vision front-end, and the output head are typically kept at higher precision, and only the large linear layers are ternary. LLaVaOLMoBitnet1B follows this pattern, with the ternary language model holding over 90% of its parameters while the remaining components stay in higher precision [12]. If a fraction f of parameters is ternary at b_t bits and the remainder uses b_o bits,

$$M_{\mathrm{w}} = \frac{N}{8}\left[f\,b_t + (1-f)\,b_o\right]. \tag{15}$$

Table 6 evaluates this for the 300M and 500M models, using $f = 0.9$ as a conservative planning value.

Two observations follow. First, an all-ternary model is 8–10× smaller than FP16, but a realistic mixed model is 4.7–6.2× smaller, so quantizing or compressing the remaining embeddings and heads is worth as much as a better ternary packing. This is consistent with the observation that BitNet b1.58's measured memory reductions grow with model size, since the non-ternary share shrinks as the linear layers dominate [10]. Second, the 2-bit layout costs 26% more than ideal packing (2 versus 1.585 bits), but it is trivial to decode and, as shown in Section 10, the spare capacity can be spent on error correction at no extra cost.

Consequence for updates. A full 75 MB weight image takes about 30 hours of continuous airtime at 5.47 kbps, ignoring protocol overhead and duty-cycle limits, so weights must be loaded before launch. Over the air, only small deltas are realistic: codebook updates, per-layer scale factors, or low-rank adapters. A node that both transmits and receives with one shared weight image stores the

Storage format	Bits/wt.	300M (MB)	500M (MB)	vs. FP16
FP16 baseline	16	600	1,000	1.0×
INT8	8	300	500	2.0×
Ternary, ideal entropy packing	1.585	59.4	99.1	10.1×
Ternary, 5 trits per byte	1.6	60.0	100.0	10.0×
Ternary, 2-bit packing	2	75.0	125.0	8.0×
2-bit + conventional SECDED (39,32)	2.44	91.4	152.3	6.6×
Ternary + embedded ECC (Section 10)	2	75.0	125.0	8.0×
Mixed: 90% ternary + 10% FP16	2.0 / 16	127.5	212.5	4.7×
Mixed: 90% ternary + 10% INT8	2.0 / 8	97.5	162.5	6.2×

Table 6: Weight storage for the 300M and 500M models. Upper rows assume every parameter is ternary; the last two rows use Equation (15) with $f = 0.9$ and ternary layers stored at 2 bits including ECC.

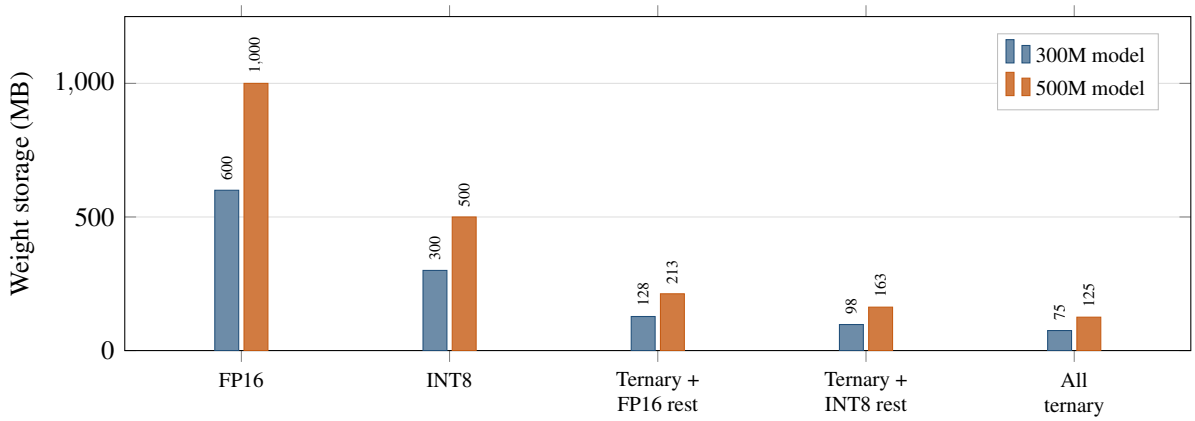


Figure 12: Weight storage by format (values from Table 6). The gap between "All ternary" and the mixed cases shows why the non-ternary fraction matters.

weights once; Section 7 quantifies the saving relative to separate encoder and decoder networks.

9.2 Runtime RAM

Weight storage is only half the picture. Runtime RAM holds the residual stream, activation buffers, and the weight tiles currently being processed. We model a 300M-class backbone with $d = 1024$ and 24 layers ($12d^2 \times 24 \approx 302$M parameters), and a 500M-class backbone with $d = 1536$ and 18 layers (≈ 510M), processing $L = 288$ tokens per frame (256 image patches, 16 telemetry tokens, 16 text tokens). The baseline holds FP16 activations and weights; the ternary design uses INT8 activations and 2-bit weight tiles, with the residual stream kept at 16 bits in both cases. Peak working memory is the residual stream plus the larger of the attention phase and the feed-forward phase, plus two weight tiles (double buffering) of $d \times 256$ weights.

The RAM reduction (2.1×) is smaller than the weight reduction (8×) because the residual stream stays at 16 bits and activations are only halved. Three techniques recover more:

- **Feed-forward tiling.** Processing the $4d$ hidden dimension in four chunks cuts the 300M ternary peak from 2.20 MB to about 1.31 MB (and the 500M peak to about 1.97 MB).
- **Per-head attention streaming.** Computing one head at a time keeps the attention score tile to L^2 entries (about 83 k, or 0.08 MB at INT8).
- **No KV cache for encoding.** The encoder pass over a fixed token set needs none. For autoregressive text or command output, an 8-bit KV cache costs about 49 KB per token for the 300M model (roughly 3.1 MB for 64 tokens),

	300M-class		500M-class	
Component (peak)	FP16	Ternary	FP16	Ternary
Residual stream ($L\times d$, 16-bit)	0.59	0.59	0.88	0.88
Input buffer + FFN hidden ($L\times 4d$)	2.95	1.47	4.42	2.21
Weight tiles, double-buffered	1.05	0.13	1.57	0.20
Total working RAM (MB)	**4.59**	**2.20**	**6.88**	**3.29**
Reduction	2.1×		2.1×	

Table 7: Estimated peak working RAM (MB) for one 288-token frame, with full-width feed-forward buffers.

half of the FP16 cost.

A working set of 1.3–2.2 MB sits at the upper end of the block RAM available on low-cost FPGAs, which offer hundreds of kilobytes to a few megabytes, so small devices will need tiling or a small external SRAM.

9.3 Memory traffic

Because off-chip access costs orders of magnitude more energy than an integer add [18], traffic matters as much as capacity. A ternary model streams 75 MB of weights per pass for the 300M case versus 600 MB for FP16, an 8× reduction in the dominant off-chip energy term, and each weight is reused across all 288 tokens while it sits in the tile buffer.

10. Radiation-Hardened Ternary Weight Encoding

10.1 The unused state

A hardware-friendly layout stores each ternary weight in two bits, but only three of the four codes are used (Figure 13, top). The unused code carries real capacity: $2-\log_2 3 \approx 0.415$ bits per weight, or 6.6 bits per 16 weights. We exploit this spare capacity to protect against single-event upsets (SEUs) and bit flips without growing the footprint.

10.2 Symbol-level sentinel

The simplest use is to treat `11` as an invalid code. Hardware decodes it as zero (a skipped weight, which is the safest default) and raises a fault flag. This costs nothing, but its coverage is limited: of the six possible single-bit flips among the valid codes, only two (`01`→`11` and `10`→`11`) land on the invalid code and are detected, and none can be corrected. A flip into another valid code is silent. The sentinel is therefore a cheap first line of defense, not a complete solution.

10.3 Block-level embedded SECDED

To obtain real correction, we pool the spare capacity across a block. Sixteen ternary weights have $3^{16} = 43{,}046{,}721 < 2^{26}$ states, so they fit in a 26-bit field. A 26-bit data word plus 6 check bits form a SECDED (single-error-correcting, double-error-detecting) Hamming code with an overall parity bit [16, 17], giving exactly 32 bits for 16 weights, the same as plain 2-bit packing:

$$\underbrace{16 \text{ weights}}_{\log_2 3^{16}\approx 25.4 \text{ b}} \longrightarrow \underbrace{26 \text{ data bits}}_{2\times 13} + \underbrace{6 \text{ check bits}}_{\text{SECDED}} = 32 \text{ bits}. \tag{16}$$

As Figure 13 (bottom) shows, each 13-bit field holds 8 trits ($3^8 = 6561 < 2^{13} = 8192$) and is expanded to eight 2-bit codes by a small 8192-entry, 16-bit read-only table (16 KB), shared across the PE array. Field values of 6561 or above are not valid trit patterns, so a corrupted field that lands there is flagged immediately, providing extra detection on top of the code.

10.4 Comparison of layouts

Table 8 shows the benefit: the embedded layout delivers the protection of a conventional SECDED code at 18% less storage (2.00 versus 2.44 bits per weight) and with no more storage than unprotected 2-bit packing. The price is a small decode stage, a 6-bit syndrome computation, correction logic, and two table lookups, which are pipelined behind the

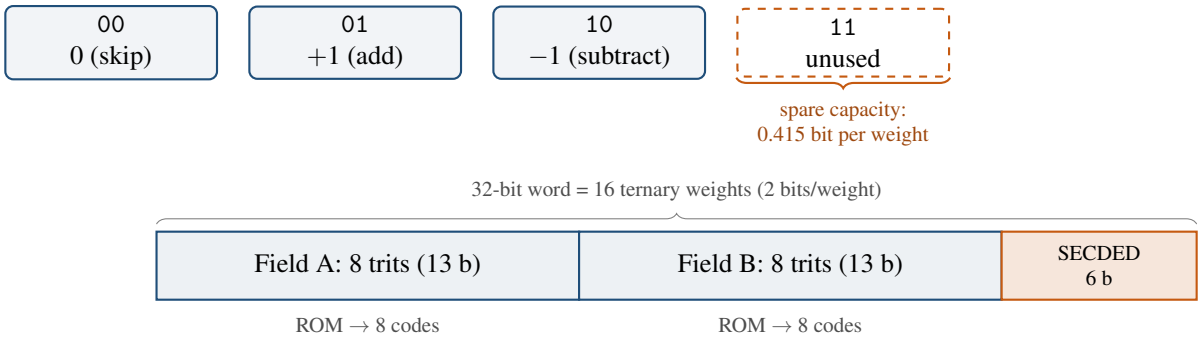


Figure 13: Top: the 2-bit ternary code and its unused fourth state. Bottom: block layout that reclaims the spare capacity for a SECDED check field at the same 2 bits per weight.

Layout	Bits/wt.	500M (MB)	Single flip	Double flip
Plain 2-bit	2.00	125.0	silent	silent
2-bit + sentinel 11	2.00	125.0	2 of 6 detected	mostly silent
2-bit + SECDED (39,32)	2.44	152.3	corrected	detected
Radix-packed + embedded SECDED	2.00	125.0	corrected	detected

Table 8: Protection versus footprint. The embedded layout matches the plain 2-bit footprint while providing full single-error correction and double-error detection per 16-weight block.

weight stream and are small compared with the PE array.

10.5 Operation, scrubbing, and limits

On a detected but uncorrectable error (a double flip, or an invalid field), the controller zeroes the affected 16-weight block, since zero weights are skipped and the damage is bounded, and requests a refresh from a golden copy. A background scrubber periodically reads, corrects, and rewrites every word so that upsets do not accumulate. If a word sees upsets as a Poisson process of rate λ and the scrub period is T_s, the probability that it accumulates two or more upsets within one period is approximately

$$P_{\geq 2} \approx \frac{(\lambda T_s)^2}{2} \quad (\lambda T_s \ll 1), \tag{17}$$

so halving T_s cuts the uncorrectable-error probability by a factor of four. Multi-bit upsets from a single particle strike are handled by interleaving the bits of neighboring words, so that adjacent cells belong to different codewords. The small high-precision parameters (embeddings, scale factors) are stored with conventional SECDED, since they are a minor fraction of memory.

This scheme protects *stored weights*. It does not protect upsets in logic or in configuration memory, which still require configuration scrubbing and triple modular redundancy on control logic as described in Section 8. The two layers of defense are complementary.

11. Deployment: PocketQube Constellation

11.1 Form factor and Sub-GHz links

PocketQubes (nominally $5 \times 5 \times 5$ cm per unit) cut manufacturing and launch cost sharply relative to larger platforms and allow incremental fleet growth. The constellation uses 433 MHz and 915 MHz LoRa. These bands offer narrow bandwidth, but highly compressed semantic tokens fit within small packet windows, and inexpensive, widely available receivers can serve as ground terminals.

11.2 On-board security

Custom cryptography is implemented in the FPGA fabric and applied to semantic tokens *before* modulation, so no plaintext payload leaves the spacecraft. Encrypting tokens rather than pixels also keeps the cipher workload tiny: a few hundred bytes per frame. Authentication tags protect uplinked commands and

model updates against spoofing, and on the receive path they are verified before any token reaches the network (Section 7).

11.3 Decentralized compute and mesh networking

Through inter-satellite links (ISLs), each satellite acts as an in-orbit compute and routing node, as shown in Figure 14. The constellation can:

- route packets between nodes toward the best-positioned downlink;
- run consensus algorithms across the fleet; and
- detect anomalies autonomously, without continuous ground control.

Because tokens are small, relaying them costs little airtime, and cross-satellite fusion (for example combining two satellites' tokens for the same target) can improve quality beyond what any single node sends. A relay can forward tokens unchanged, or decode and fuse them with the same shared network used for transmit (Section 7).

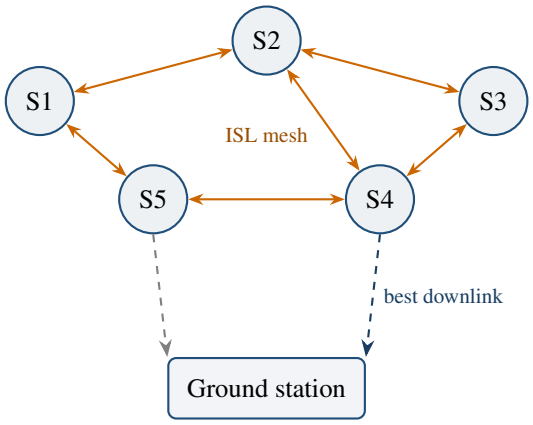


Figure 14: Inter-satellite mesh. A source node's tokens are relayed over ISLs to whichever node currently has the best downlink to the ground station.

12. Evaluation Plan

The figures above are analytic illustrations. The following plan will replace them with measured results.

Experiments proceed in three stages: (i) software simulation of the BDE sweep over model width, ternary vs. full precision, token budget, and channel SNR; (ii) FPGA implementation with power measured on a development board; and (iii) an end-to-end link test over LoRa with an emulated orbital channel (Doppler shift and fading), followed by flight-representative environmental testing.

13. Expected Benefits and Limitations

Benefit	Enabling mechanism
Bandwidth efficiency	Semantic tokenization with ground-side reconstruction or task decoding
Power efficiency	Multiplier-free ternary inference; zero-skipping; weight streaming
Generality	One shared multimodal latent space for text, images, and telemetry
Low cost	PocketQube form factor; commodity Sub-GHz ground receivers
Channel resilience	Joint source–channel training; discrete ternary latents; semantic index assignment
Adaptive priority	Attention gating reallocates bandwidth toward critical telemetry in real time
Memory efficiency	8–10× smaller weights; about 2× smaller working RAM
Two-way reuse	One ternary backbone for transmit and receive; 30–48% less weight storage than separate networks
Radiation resilience	Embedded SECDED in the unused ternary state space at no extra storage
Security	In-fabric cryptography applied before modulation
Autonomy	ISL mesh routing, consensus, and onboard anomaly detection
Scalability	Small inexpensive nodes added incrementally

Limitations. Semantic reconstruction is lossy and, with generative decoders, may hallucinate plausible but incorrect detail, which is unacceptable for some scientific uses; critical frames should fall back to higher-fidelity modes. Ternary models require training from scratch or extensive fine-tuning. A 300M–500M parameter model is a demanding fit for the smallest FPGAs and PocketQube power envelopes, so staged model sizes and duty-cycled inference are likely necessary. Semantic codecs are only as good as the distribution they were trained on, so out-of-distribution scenes need detection and fallback. The LoRa channel surrogate is only as accurate as the characterization behind it, and a mismatch between simulated and real orbital channels (Doppler, Ri-

Axis	Metrics	Baselines
Bandwidth	Payload bytes per frame; airtime; effective throughput	Raw; JPEG/JPEG2000; learned codec [6]
Distortion	PSNR, MS-SSIM, LPIPS; downstream accuracy; text and telemetry error	FP16 encoder; INT8 encoder
Energy	Joules per inference (measured on board); joules per delivered frame (compute + radio)	INT8 and FP16 FPGA accelerators
Hardware	LUTs, BRAM, DSP use; clock rate; latency; throughput	Same device, multiplier-based design
Channel robustness	D versus SNR under AWGN, Rayleigh, and the LoRa surrogate; packet-loss sweep	Separate codec with LoRa FEC; JSCC with FP16 encoder
Allocation	Telemetry latency under injected anomalies; image recovery time; budget compliance	Static split; priority queue without gating
Memory	Weight MB; peak working RAM; off-chip traffic per frame	FP16 and INT8 designs
Two-way operation	D per link for shared versus dedicated decoder; command decode accuracy and latency; mode-switch overhead	Separate ternary encoder and decoder; weight-tied variant; FP16 ground decoder
Radiation	Accuracy after injected single and double flips; uncorrectable-error rate versus scrub period	Plain 2-bit; 2-bit + SECDED (39,32)

Table 9: Proposed evaluation axes, metrics, and baselines.

cian fading) could erode the robustness gains. The anomaly gate can produce false positives that needlessly throttle imagery, so thresholds and hysteresis need tuning on real telemetry. Sharing one backbone between encoding and decoding may reduce quality relative to dedicated networks, particularly for pixel reconstruction. The embedded ECC protects stored weights only; logic and configuration upsets need separate mitigation, and the non-ternary fraction of the model limits the achievable memory savings.

14. Conclusion

We presented an ultra-low-power edge intelligence architecture that unifies ternary multimodal transformers with semantic joint source–channel coding, priority-aware bandwidth allocation, and radiation-aware weight storage on FPGA hardware. The BDE framework explains why the combination is attractive: semantic tokenization removes bits from the link, ternary arithmetic removes multipliers from the datapath, and the energy saved on each side compounds on a power-starved node. Instantiated in a PocketQube constellation with Sub-GHz links, in-fabric encryption, and an inter-satellite mesh, the approach offers a scalable, secure, and power-efficient blueprint for smart orbital and IoT infrastructure. Future work will validate the analytic trends reported here with measured results.

References

[1] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, 1948.

[2] D. Gündüz *et al.*, "Beyond transmitting bits: Context, semantics, and task-oriented communications," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 1, pp. 5–41, 2023.

[3] H. Xie, Z. Qin, G. Y. Li, and B.-H. Juang, "Deep learning enabled semantic communication systems," *IEEE Trans. Signal Process.*, vol. 69, pp. 2663–2675, 2021.

[4] E. Bourtsoulatze, D. B. Kurka, and D. Gündüz, "Deep joint source-channel coding for wireless image transmission," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 3, pp. 567–579, 2019.

[5] Y. Blau and T. Michaeli, "Rethinking lossy compression: The rate-distortion-perception tradeoff," in *Proc. ICML*, 2019.

[6] J. Ballé *et al.*, "Variational image compression with a scale hyperprior," in *Proc. ICLR*, 2018.

[7] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," in *Proc. NeurIPS*, 2015.

[8] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," arXiv:1605.04711, 2016.

[9] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," in *Proc. ICLR*, 2017.

[10] S. Ma, H. Wang, L. Ma, *et al.*, "The era of 1-bit LLMs: All large language models are in 1.58 bits," arXiv:2402.17764, 2024.

[11] H. Wang, S. Ma, L. Dong, S. Huang, H. Wang, L. Ma, F. Yang, R. Wang, Y. Wu, and F. Wei, "BitNet: Scaling 1-bit transformers for large language models," arXiv:2310.11453, 2023.

[12] J. Sundaram and R. Iyer, "LLaVaOLMoBitnet1B: Ternary LLM goes multimodal!," arXiv:2408.13402, 2024.

[13] D. B. Kurka and D. Gündüz, "Bandwidth-agile image transmission with deep joint source-channel coding," *IEEE Trans. Wireless Commun.*, vol. 20, no. 12, pp. 8081–8095, 2021.

[14] J. Xu, B. Ai, W. Chen, A. Yang, P. Sun, and M. Rodrigues, "Wireless image transmission using deep source channel coding with attention modules," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 4, pp. 2315–2328, 2022.

[15] J. Dai *et al.*, "Nonlinear transform source-channel coding for semantic communications," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 8, pp. 2300–2316, 2022.

[16] R. W. Hamming, "Error detecting and error correcting codes," *Bell Syst. Tech. J.*, vol. 29, no. 2, pp. 147–160, 1950.

[17] M. Y. Hsiao, "A class of optimal minimum odd-weight-column SEC-DED codes," *IBM J. Res. Develop.*, vol. 14, no. 4, pp. 395–401, 1970.

[18] M. Horowitz, "Computing's energy problem (and what we can do about it)," in *Proc. IEEE ISSCC*, 2014, pp. 10–14.