

FOOL: Addressing the Downlink Bottleneck in Satellite Computing with Neural Feature Compression

Alireza Furutanpey, Qiyang Zhang, Philipp Raith, Tobias Pfandzelter,
Shangguang Wang, *Senior Member, IEEE*, Schahram Dustdar, *Fellow, IEEE*

Abstract—Nanosatellite constellations equipped with sensors capturing large geographic regions provide unprecedented opportunities for Earth observation. As constellation sizes increase, network contention poses a downlink bottleneck. Orbital Edge Computing (OEC) leverages limited onboard compute resources to reduce transfer costs by processing the raw captures at the source. However, current solutions have limited practicability due to reliance on crude filtering methods or over-prioritizing particular downstream tasks. This work presents FOOL, an OEC-native and task-agnostic feature compression method that preserves prediction performance. FOOL partitions high-resolution satellite imagery to maximize throughput. Further, it embeds context and leverages inter-tile dependencies to lower transfer costs with negligible overhead. While FOOL is a feature compressor, it can recover images with competitive scores on quality measures at lower bitrates. We extensively evaluate transfer cost reduction by including the peculiarity of intermittently available network connections in low earth orbit. Lastly, we test the feasibility of our system for standardized nanosatellite form factors. We demonstrate that FOOL permits downlinking over $100\times$ the data volume without relying on prior information on the downstream tasks.

Index Terms—Edge Computing, Edge Intelligence, Orbital Edge Computing, Low Earth Orbit, Satellite Inference, Data Compression, Learned Image Compression, Neural Feature Compression



INTRODUCTION

THE development of commercial ground stations [1] and the advancement in aerospace technology has enabled the emergence of nanosatellite constellations [2] in low earth orbit (LEO) as a novel mobile platform. The standardization of small form factors, such as CubeSat [3], reduces launch costs, allowing for frequent updates and deployments. Manufacturers typically equip satellites with sensors to capture large geographic regions. The downlinked satellite imagery enables Earth observation (EO) services with socially beneficial applications, such as agriculture [4] and disaster warning [5]. Nonetheless, most constellations follow a “bent pipe” architecture where satellites downlink raw sensor data for processing in terrestrial data centers. Notably, given the constraints of orbital dynamics, satellites may only establish a connection for a few minutes. For example, the Dove High-Speed Downlink (HSD) system [6] provides segments with volumes as low as 12 GB during a single ground station pass.

As constellation sizes and sensor resolutions increase, downlink bandwidth cannot keep up with the accumulating data volume [7], [8]. Additional ground station equipment may prevent link saturation, building and maintaining them, including licensing the necessary frequencies, is a significant cost factor for satellite operation. As an alternative, Orbital Edge Computing (OEC) proposes processing data at the source [9]–[13]. Recent work on reducing bandwidth requirements in OEC is roughly categorizable in aggressive (task-oriented) filtering and compression [14]. The former relies on subjective value measures that restrict their prac-

tability to coarse-grained tasks, such as de-duplication or cloud filtering. The latter constrains entire missions to particular tasks or prediction models. We argue that the limitations of existing compression or other data reduction approaches are particularly adverse to OEC.

First, the CubeSat design is intended for short-duration missions [3] (typically up to 3-5 years), and despite waning prices, launching sensor networks in space is still associated with substantial logistical, administrative, and monetary costs. Therefore, it seems undesirable to designate entire constellations to a small subset of tasks and Deep Neural Network (DNN) architectures. More pressingly, irrespective of whether current codecs can prevent bottlenecks, they may undermine the effectiveness of entire missions. Precisely, the assumption of prediction models only requiring a subset of information for image reconstruction may lead to false confidence in a codec to reliably discern the salient signals. We argue the opposite holds, i.e., when the objective is to accommodate *arbitrary* downstream tasks with prediction models instead of human experts, there is *less* potential for rate reductions. Intuitively, two seemingly visually identical images may have subtle differences in pixel intensities, which a prediction model could leverage to overcome physiological restrictions.

In summary, three conflicting objectives aggravate the challenges for OEC: (i) maximizing downlinking captures, (ii) ensuring the value of the captures by relying on as few assumptions on downstream tasks as possible, and (iii) minimizing the risk from unpredictable adverse effects on current and future prediction models. To this end, we propose drawing from recent work on neural feature compression

with Shallow Variational Bottleneck Injection (SVBI) [15]–[17]. The idea of SVBI is to reduce discarding information necessary for arbitrary, practically relevant tasks by targeting the shallow representation of foundational models as a reconstruction target in the rate-distortion objective. In other words, rate reductions come from constraining the solution space with abstract high-level criteria rather than reifying target tasks with an explicit definition of value or expert-crafted labels. We investigate whether the SVBI framework is suitable for EO from a compression perspective and identify lower-level system considerations given the oppressive constraints of OEC. Then, we apply our insights to introduce a **T**ile **H**olistic **E**fficient **F**eatured **O**riented **O**rbital **L**earned (THE FOOL) compression method, which we will refer to as FOOL for short. FOOL alleviates the challenges of OEC by generalizing SVBI to improve compression performance while introducing more specific methods that aid in meeting the requirements of OEC and EO tasks. FOOL comprises a simple pipeline, a profiler, and a neural feature codec with a separate reconstruction model. The pipeline minimizes overhead by CPU-bound pre- and post-processing with concurrent task execution. The profiler identifies configurations that maximize data size reduction, factoring in intermittently available downlinks and the trade-off between processing throughput and lowering bitrate from more powerful but costlier transforms. The neural codec’s architecture includes task-agnostic context and synergizes with the profiler’s objective to maximize throughput with batch parallelization by exploiting inter-tile spatial dependencies.

We perform in-depth experiments to scrutinize our approach with a wide range of evaluation measures by emulating conditions on a testbed with several edge devices. Our results show that FOOL is viable on CubeSat nanosatellites and increases the downlinkable data volume by two orders of magnitude relative to bent pipes at no loss on performance for EO. Unlike a typical task-oriented compression method, it does not rely on prior information on the tasks. Additionally, FOOL exceeds existing SVBI methods with an up to $2.1\times$ bitrate reduction. Lastly, the reconstruction model can map features from the compressed shallow feature space to the human interpretable input space. The resulting images compete with state-of-the-art learned image compression (LIC) models using mid-to-high quality configurations on PSNR, MS-SSIM, and LPIPS [18] with up to 77% lower bitrates. We open-source the core compression algorithm¹ as an addition to the community. In summary, our main contributions are:

- Demonstrating the inadequacy of image codecs for EO with satellite imagery and that the general SVBI framework [17] can address these limitations.
- Significantly improving compression performance of existing methods with novel components that embed additional modality and capture inter-tile dependencies of partitioned images.
- Introducing a reconstruction component that can recover high-quality human interpretable images from the compressed latent space of shallow features.

To the best of our knowledge, this work is the first to elaborate on the risk of image codecs on EO that distinctly rely on fine-grained details. Crucially, it proposes a solution approach that assumes reconstruction for human interpretability as a subset of objectives that prioritize maintaining the integrity of model predictions.

Section 2 compares relevant work addressing the downlink bottleneck. Section 3 motivates our approach by describing current challenges. Section 4 describes the profiling strategy and compression pipelines. Section 5 introduces the FOOL’s compression method. Section 6 details the methodology and evaluates FOOL against numerous baselines. Section 7 transparently discusses limitations to shape directions for future research. Lastly, Section 8 concludes the work.

2 RELATED WORK

2.1 Collaborative Inference and Data Compression

The Deep Learning aspect of our method draws from recent advancements in collaborative inference [19] and data compression [20]. The underlying compression algorithm and objective functions are derived and extended from previous publications such as Entropic Student (ES) [15], [16] and FrankenSplit (FS) [17]. Both works re-formulate the distortion term from lossy compression methods [21] and deploy lightweight models suitable for resource-constrained mobile devices. Besides introducing novel components to further lower transfer costs, FOOL considers the diverging requirements due to intrinsic differences between terrestrial and orbital remote sensing.

2.2 Preventing Link Saturation with Orbital Inference

The system aspect of our method aligns best with work focusing on getting the data to the ground for further processing instead of performing inference on board [9], [12], [22], [23]. We emphasize the high variability among fundamental design principles for OEC, as it is an emerging field, and a comprehensive literature review is not within the scope of this work. Instead, we discuss the approaches we found most promising and share some design elements with FOOL.

Gradre et al. introduce *Vista* [24], a Joint Source Channel Coding (JSCC) system for LoRa-enabled CubeSats, designed to enhance low-latency downlink communication of satellite imagery and DNN inference. It shows significant improvements in image quality and classification performance through LoRa-channel-aware image encoding. Moreover, the evaluation assumes simple tasks not representative of practical EO. In contrast, FOOL decouples image recovery from the initial compression objective and ensures task-agnostic preservation of information.

Lu et al. introduce *STCOD* [25], a JSCC system for efficient data transmission and object detection in optical remote sensing. STCOD integrates satellite computing to process images in space, distinguishing between regions of interest (ROIs) and backgrounds. It shows promising results with a block-based adaptive sampling method, prioritizing transmitting valuable image blocks using fountain code [26]. The caveat is that ROI detectors that can reliably prevent predictive loss for downstream tasks require strong biases

1. <https://github.com/rezafuru/the-fool>

regarding sensor properties and task granularity. FOOL includes task-agnostic context, with significantly less overhead and more robustness towards varying conditions than an ROI detector. Furthermore, it is end-to-end optimized with the other compression model components without relying on the same biases or expert-crafted labels.

Thematically, our work resembles Kodan by Denby et al. [27] the closest. Like FOOL, Kodan treats channel conditions as an orthogonal problem and primarily focuses on source coding to address the downlink and computational bottlenecks. Kodan uses a reference application for satellite data analysis and a representative dataset to create specialized small models. Once in orbit, it dynamically selects the best models for each data sample to maximize the value of data transmitted within computational limitations. Kodan’s excellent system design is promising but relies on assumptions that hinder practicability and the potential for meaningful rate reductions. Conversely, FOOL follows a different design philosophy by treating the downlink bottleneck primarily as a compression problem. We do not treat the computational deadline as a hard temporal constraint to decouple the method to a particular system design. FOOL’s profiler measures key performance indicators on the *pixel level*. Given hardware limitations, the aim is to reduce transfer cost by balancing the lower bitrate of more powerful encoders and the gain in processing throughput of more lightweight encoders.

3 BACKGROUND & PROBLEM FORMULATION

3.1 The Downlink Bottleneck

Downlink bottlenecks occur when the data volume exceeds the bandwidth within a downlink segment during a single pass. We formalize a model sufficient for our purposes by considering link conditions and sensor properties of satellites belonging to a constellation. A constellation is defined as $\mathcal{C} = (L, \mathcal{S}, I, f)$ where L is a link to communicate with a ground station and \mathcal{S} is a set of satellites. The link is determined by its *expected* downlink rate, measured in megabits per second (Mbps). The function $f : \mathcal{S} \rightarrow I$ maps each satellite $s \in \mathcal{S}$ to an interval where it passes the downlink segment into disjoint subsets $\mathcal{G} = \{\mathcal{G}_i | \mathcal{G}_i = \{s \in \mathcal{S} | f(s) = i\}, i \in I\}$, such that $\bigcup \mathcal{G}_i = \mathcal{S}$ and $\bigcap \mathcal{G}_i = \emptyset$. The link capacity V_{link} is the bandwidth available per pass and is determined by the link rate and the interval range. Satellites $S = (R_{\text{orbit}}, S_{\text{rate}}, S_{\text{spatial}}, S_{\text{bands}}, S_{\text{radio}}, S_{\text{fov}})$ are equipped with a sensor, and its properties determine the volume per capture.

$$V_{\text{capture}} = \frac{\overbrace{R_{\text{orbit}}^2 \cdot \tan^2(S_{\text{fov}})}^{\text{Total Pixels}}}{S_w \cdot S_h} \cdot \underbrace{S_{\text{bands}} \cdot S_{\text{radio}}}_{\text{Bits per Pixel}} \quad (1)$$

The radiometric resolution S_{radio} and number of bands S_{bands} determine the downlink cost per pixel in bits. The orbit R_{orbit} , sensor spatial resolution $S_{\text{spatial}} = S_h \times S_w$, and field of view S_{fov} determine the number of pixels per capture. The number of captures depends on the time to complete an orbit

$$T_{\text{orbit}} = 2\pi \sqrt{\frac{(R_{\text{orbit}} + R_{\text{earth}})^3}{GM}} \quad (2)$$

and on the capture rate S_{rate} . G is the gravitational constant and M is the earth’s mass. The orbit R_{orbit} is usually around 160 to 800 kilometers for LEO satellites. For reference, R_{orbit} is 786 kilometers for Sentinel-2 [28]. Finally, the number of captures from all satellites within the segmentation group determines the total volume per pass.

$$V_{\text{pass}} = \sum_{s^{(i)} \in \mathcal{G}_j} T_{\text{orbit}} \cdot S_{\text{rate}}^{(i)} \cdot V_{\text{capture}}^{(i)} \quad (3)$$

The superscript (i) denotes the costs associated with a satellite (i) . For constellations with homogenous sensors V_{capture} is a static value. Notice that V_{pass} scales linearly by the constellation size and a constant factor c for overlap occurrences, i.e., $|\mathcal{G}_j| = \frac{|\mathcal{S}|}{c}$. To determine c for a constellation, we must calculate the minimum angle between satellites β^* . Assuming a single ground station at the Earth’s North Pole and given the minimum communication elevation θ

$$\beta^* = 2 \times (180^\circ - (\theta + 90^\circ)) - \arcsin\left(\frac{R_{\text{earth}} \cdot \sin(90^\circ + \theta)}{R_{\text{orbit}} + R_{\text{earth}}}\right) \quad (4)$$

For example, consider a constellation at $R_{\text{orbit}} = 790,000$ meters altitude with a minimum elevation $\theta = 25^\circ$ such that $\beta^* \approx 22.52^\circ$. Then, $c = \frac{360^\circ}{22.52^\circ} \approx 16$, i.e., to prevent any interval sharing, the constellation size may not exceed 16 satellites.

In short, the aim is to facilitate cost-efficient scaling of constellations by increasing bandwidth value and substantially reducing reliance on building additional infrastructure. That is, we require an encoding scheme enc , such that $V_{\text{enc}} < V_{\text{link}}$. Note that a single satellite may experience a bottleneck even if the constellation is sparse enough to prevent interval sharing [27]. Say, each $s \in \mathcal{S}$ is equipped with a sensor using approximate Sentinel-2 configurations [28] by setting a multispectral sensor for (near-) visible light to $S_{\text{bands}} = 4$, $S_{\text{radio}} = 12$, $S_{\text{fov}} = 21^\circ$, $S_{h \times w} = 10 \times 10$, and five captures per pass. With $|\mathcal{S}| \leq 16$, the volume for each pass is $\frac{790,000^2 \cdot \tan^2(10.5^\circ)}{100} \cdot 4 \cdot 12 \cdot 5 \approx 410$ GB. To prevent a bottleneck even without sharing an interval and using a higher-end link, such as WorldView-3 [29] where $V_{\text{link}} = 90$ GB per pass, the enc needs to decrease the data volume by a factor of 4.5.

There are two overarching objectives for a codec and the system in which we deploy its encoder. The system’s objective is to process and encode large volumes of high-dimensional data, given the physical limitations of LEO (nano-) satellites. A 3U nanosatellite following the CubeSat standard is limited to $10\text{cm} \times 10\text{cm} \times 30\text{cm}$ and 4kg [30] with restricted power supply by using solar harvesting [31]. The compression objective is to achieve a sufficiently low bitrate while maintaining the data’s integrity. The following elaborates on the challenges of conceiving a method that fulfills our criteria and the limitations of applying existing codecs.

3.2 Limitations of Codecs

Given remote image captures and a set of unknown associated object detection tasks, we seek a transformation of the captures into representations that minimize transfer costs and loss of information that may impact any detection tasks. We refer to generalizability as a measure

of how well a method can minimize the predictive loss on unknown detection tasks. For example, a purely task-oriented encoding (e.g., [32]) can retain information for a set of explicitly defined tasks. Still, it does not generalize as the transformed data is unusable for non-overlapping tasks. Besides bent pipes, textitlossless codecs are the only approach with easily understood guarantees on generalization. Nevertheless, lossless compression cannot adequately address the downlink bottleneck due to theoretical lower bounds. Promising alternatives are lossy methods that relax the requirement of relying on identical reconstruction for generalization. More formally, given a distortion measure D , a constraint D_c bounds the minimal bitrate to [33]:

$$\min_{P_{Y|X}} I(X; Y) \text{ s.t. } \mathcal{D}(X, Y) \leq D_c, \quad (5)$$

where $I(X; Y)$ is the mutual information and is defined as:

$$I(X; Y) = \int \int p(x, y) \log \left(\frac{p(x, y)}{p(x)p(y)} \right) dx dy. \quad (6)$$

Learned Image Compression (LIC) replaces the typically linear transformation of handcrafted codecs with nonlinear ones to reduce dependencies from sources that are not jointly Gaussian [21]. The sender applies a parametric analysis transform $g_a(\mathbf{x}; \theta)$ into a latent \mathbf{y} , which is quantized to a latent with discrete values $\hat{\mathbf{y}}$. Then, an entropy coder losslessly compresses $\hat{\mathbf{y}}$ using a shared entropy model $p_{\hat{\mathbf{y}}}$. The receiver decompresses $\hat{\mathbf{y}}$ and passes it to a parametric synthesis transform $g_s(\hat{\mathbf{y}}; \phi)$ to recover a distorted approximation to the input. To capture leftover spatial dependencies of $\hat{\mathbf{y}}$, more recent work adds *side information* with a hyperprior \mathbf{z} [34] and a context model [35]. Including side information requires two additional parametric transforms $h_a(\hat{\mathbf{y}}; \theta_h)$ and $h_s(\hat{\mathbf{z}}; \psi_h)$. Despite efficient LIC methods [36] consistently outperforming handcrafted codecs on standardized benchmarks, the results are deceptive when assessing the impact on downstream tasks. To provide further explanation, we perform a preliminary experiment that contrasts the predictive loss with additive noise and codec distortion and summarize results in Figure 1.

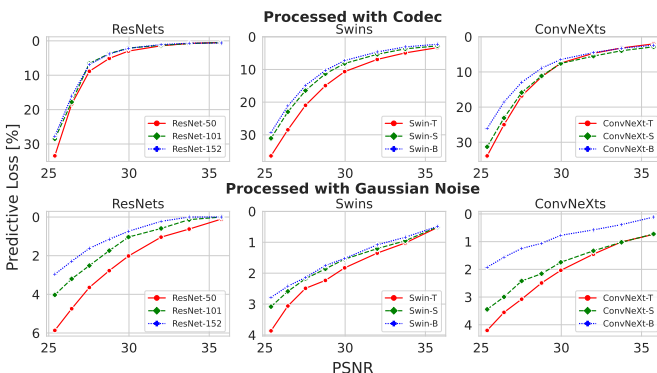


Fig. 1. Comparing Effect of Codec and Additive Noise

We download pre-trained weights [37] for the ImageNet [38] classification task of three popular architectures [39]–[41]. First, we compute the expected Peak Signal-To-Noise Ratio (PSNR) of a popular LIC model [42] for each quality level on the validation set. Then, we apply Additive

White Gaussian Noise (AWGN) on input to match the PSNR of a codec for each quality level separately. Lastly, we measure the *predictive loss* as the average difference between the accuracy of the original and processed samples. Notice that the predictive loss on the distorted input is significantly worse than the noisy input. Additive noise does not remove information; rather, it superimposes unwanted information. Conversely, lossy compression intentionally discards information from signals, and two codecs may achieve comparable rate-distortion performance despite emphasizing different information to retain. Re-training model weights on reconstructed samples may mitigate some predictive loss, but only due to adjusting prediction to input perturbations and error-prone extrapolation of lost information.

Particularly, for EO with satellite imagery that spans large geographic areas, we stress the unsuspecting danger of lossy compression, which is compounded with learned transforms [21], where it is challenging to understand behavior. The ability to differentiate between intensities beyond the capability of humans may explain why detection models can outperform domain experts. Accordingly, we should assume that lossy codecs may discard information where even experts cannot reliably verify the impact on machine interpretability. For example, suppose a codec that reduces the rate by focusing on preserving coarser-grained structures. Then, tasks that rely on assessing the environment for fine-grained object classes will lack background information (e.g., inferring region by tree species with subtle color variations).

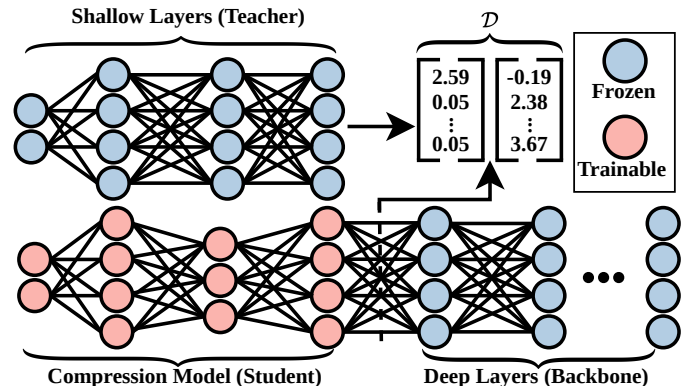


Fig. 2. Head Distillation Distortion Loss

Current limitations of image codecs put operators in a difficult position, especially for EO. The decision falls between (a combination of) lossless codecs, applying crude filtering methods, or attempting to reduce the bitrate with lossy codecs, remaining uncertain about whether the codecs retain information necessary in real conditions. As a solution, we advocate for *Shallow Variational Bottleneck Injection* (SVBI) [17], which prioritizes salient regions for (near) arbitrary high-level vision tasks.

3.3 Shallow Variational Bottleneck Injection

SVBI trains neural codecs by replacing the distortion term of the rate-distortion objective of variational image compression models [34] with head distillation (HD) [15]–[17], [43], [44]. Figure 2 illustrates an example, where the HD

distortion measure penalizes a compression model for not sufficiently approximating the shallow representation of a pre-trained foundational model. We define a foundational model as a pre-trained DNN that can accommodate multiple tasks by attaching predictors or fine-tuning the deeper layers. In Knowledge Distillation (KD) terminology, the codec is referred to as the *student* and the shallow layers of a foundational model as the *teacher*. Note that KD is not this work’s focus, as HD diverges from the typical KD objective. The intuition behind SVBI is that if a codec can reconstruct the representation of a foundational model, then the representation is sufficient for *at least* all tasks associated with that model.

3.3.1 The Effectiveness of Shallow Features

Readers may reasonably assume that using the representation for one particular network architecture instead of the input as the distortion measure is more restrictive for two reasons.

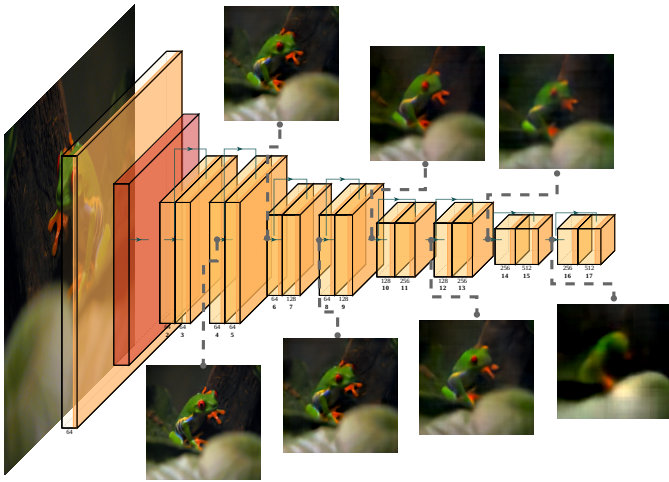


Fig. 3. Discarding Information in Discriminative Tasks

First, the features are not human-readable, i.e., we cannot overlay the bounding boxes on the images. We could infer the global coordinates to present boxed overlaid on previously captured satellite imagery. This may suffice for observing (semi-) permanent objects (e.g., landmarks) but certainly not for ephemeral or moving objects (e.g., tracking the movement of vessels). Second, even if the trend toward transfer learning with foundational models [45], [46] can accommodate various predictors, client preferences for architectures may vary.

We argue that by targeting shallow representations, both limitations can be addressed. View an n -layered feed-forward neural network as a Markov chain of successive representations R_i, R_{i+1} [47]:

$$I(X; Y) \geq I(R_1; Y) \geq \dots \geq I(R_n; Y) \geq I(\tilde{Y}; Y) \quad (7)$$

The mutual information $I(X; R_i)$ will likely decrease relative to the distance between the input and a representation. This loss stems from layers applying operations that progressively restrict the solution space for a prediction,

particularly for discriminative tasks. That is, the deeper the representation, the more information we lose regarding X :

$$I(X; X) \geq I(R_1; X) \geq \dots \geq I(R_{n-1}; X) \geq I(R_n; X) \quad (8)$$

Figure 3 visualizes the trade-off. We extract the features from a ResNet network with weights trained on the ImageNet [38] classification task and recover the original image by training separate reconstruction models for each marked location. Section 5.6 elaborates on the reconstruction. Notice that models at shallow layers can recover the input with high similarity, but the recovery progressively worsens as the path distance increases. Now assume a discriminative model (e.g., a ResNet for image classification) $\mathcal{M} = (\mathcal{H}, \mathcal{T})$ with separate shallow \mathcal{H} from deeper layers \mathcal{T} as disjoint subsets, such that $\mathcal{H}(X) = H$ (i.e., mapping input to shallow features) and $\mathcal{M}(X) = \mathcal{T}(\mathcal{H}(X))$. Further, given a codec $c = (\text{enc}, \text{dec})$ where $\text{dec}(\text{enc}(X)) = \tilde{H}$ is an approximation of $\mathcal{H}(X)$. Then, \tilde{H} is a sufficient approximation of H if $\mathcal{T}(\tilde{H})$ results in lossless prediction, i.e., no drop in prediction performance relative to $\mathcal{T}(H)$. In other words, a sufficient representation in the shallow latent space results in high similarity in the deep latent space between $\mathcal{T}(H)$ and $\mathcal{T}(\tilde{H})$. Consider that similarity in the deep latent space coincides with high similarity for human perception in the input space [18]. Therefore, the encoder output $\text{enc}(X)$ should retain sufficient information to reconstruct X with quality comparable to $\mathcal{H}(X)$, as exemplified in Figure 3. Finally, since $\text{enc}(X)$ sufficiently approximates H , it should be possible to sufficiently approximate the shallow representation of any model $\mathcal{M}' = (\mathcal{H}', \mathcal{T}')$ if $I(\mathcal{H}(X), X) \approx I(\mathcal{H}'(X), X)$.

3.3.2 Rate Reductions by Task Specificity

The idea of task-oriented communication is that messages for model prediction may require less information than human domain experts, i.e., that it should be possible to reduce bitrate by not (exclusively) using input reconstruction as the distortion measure. We argue that this assumption contradicts empirical evidence demonstrating models outperforming human experts in various image-related tasks, i.e., machines can detect signals and patterns that humans physiologically or intellectually cannot. Rather, the opposite should hold, i.e., when compressing for quality using domain experts as judges, we should see more potential for rate savings, not less. The claim is consistent with the results in Figure 1 where codecs with high reconstruction quality result in images that are deceptively similar to the input (details in Section 6.3). Conclusively, rate reductions are from *task specificity* of the distortion measure, irrespective of the input interface, whether it is a particular layer of a DNN architecture, human receptors, or textual encoding. Note that this holds, even if when limiting measures to discriminative task objectives without any image reconstruction. Besides visualizing Equation (8), the input image illustrates a practical example. The frog subset of ImageNet distinguishes between *Tree Frogs*, *Bullfrogs*, and *Tailed Frogs*. Since these frog species have distinct figures and dominant colors, the more delicate characteristics of a tree frog are redundant for ImageNet classification. The network gradually discards information regarding the fine-grained blue-yellow colored patterns, permitting only the

recovery of general shape and environment from the deep features. The deeper the features, the less structure and detail are present, which may be redundant for the task. Now, suppose training a codec where the encoder retains the minimal information necessary to reconstruct the output of the deepest layers for a classification task (e.g., similar to Vista [24]). Then, we can reduce the transfer cost to as low as $\log_2(\#\text{labels})$ without predictive loss. However, we may lack the information for other tasks, i.e., there is a trade-off between generalization and the lower bound on the bitrate. In contrast, targeting shallow features for compression may strike a balance between aiming to retain information for all possible downstream tasks and only emphasizing the salient regions for the tasks associated with a foundational model. Arguably, the limitation is negligible, as maintainers will train foundational models with useful tasks in mind.

4 THE FOOL'S SYSTEM DESIGN

4.1 Compression and Prediction Request Flow

Figure 4 illustrates a high-level view for serving requests.

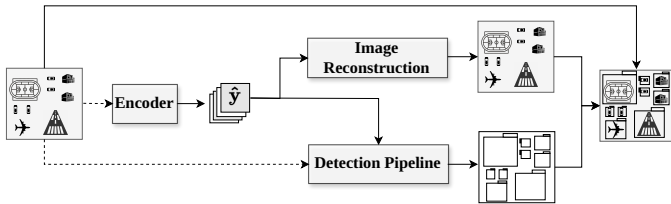


Fig. 4. High-Level Inference Request Flow

For samples processed by FOOL, there is a single encoder. The output \hat{y} is forwarded to the detection pipeline, skipping the shallow layers. A decoder transforms \hat{y} into a target representation. A predictor outputs bounding boxes for a specified task. An image reconstruction model optionally restores the latent to a human-interpretable image to overlay the bounding boxes. Samples downlinked with bent pipe or some image codec are forwarded to the shallow layers, skipping the FOOL decoder and reconstruction model. This section focuses on the pipeline, before Section 5 introduces the compression method.

4.2 Profiling Compression Pipelines for OEC

A common challenge for operators is to determine whether reported performance regarding resource usage or throughput from the latest advancements generalizes to their target hardware. This problem stems not from a lack of rigor by authors but from the sheer heterogeneity of the AI accelerator landscape [48]. Graph compilers and other vendor-specific optimizations (e.g., TensorRT², Apache TVM³) further complicate evaluation, with varying methods for operator fusion, graph rewriting, etc. Consequently, FOOL includes a simple profiling and evaluation strategy that operators may run before deployment. Notably, in contrast to existing work that partitions images to match the input size of a particular

application, the profiler regards the importance of spatial dimensions for resource efficiency. The purpose of the profiler is to determine a configuration that maximizes throughput. While throughput evaluation is straightforward, how to measure it (e.g., images/second) is not necessarily obvious, particularly for (neural) compression pipelines.

First, terrestrial and LEO remote sensing with constrained sensor networks demand resource-conscious methods, but in LEO, downlinks are only available within segments. Due to memory and storage constraints, devices must process samples according to a sensor rate, i.e., a prolonged interval between incoming samples. Hence, the objective in LEO is to maximize the number of pixels the accelerator can process before reaching a downlink segment, given a time constraint for a single sample (i.e., “frame deadline” [27]). For example, assume a cheaper and a costlier compression model where both models meet the frame deadline. Applying the latter results in half the bitrate but thrice the inference time. Using the former is beneficial in most network conditions for real-time terrestrial applications since it results in a lower end-to-end request latency. In contrast, applying the latter in LEO may be advantageous, as finishing earlier results in the needless idle time of resources. Second, satellite imagery has substantially higher resolution than captures from most terrestrial sensor networks. A standard method to improve throughput for high-dimensional images is parallel processing with tile partitioning. The distinction is that there is more control over the spatial dimensions and the batch size. Nonetheless, a caveat is the friction between a model’s size and the input size. Increasing the width (e.g., the number of feature maps output by a convolutional layer) of a neural codec’s parametric transforms may result in better compression performance but lower processing throughput. In summary, we require a measure that includes (i) the tile spatial dimensions, (ii) batch size, and (iii) the capacity-compression performance trade-off.

We can address the requirements (i) and (ii) by measuring throughput as *pixels processed per second* (PP/s). To motivate the need to expand on PP/s for (iii), we demonstrate the friction between model width, input size, and batch size using the convolutional encoder in [17] consisting of three downsampling residual blocks (Section 5.3). Figure 5 summarizes the results as the average of 100 repetitions with progressively increasing width. Notice how evaluating img/s always favors smaller spatial dimensions and disregards batch size and model width. In contrast, PP/s reveals that the optimal spatial dimension is around 500×500 but will naturally favor smaller models, as it does not consider that wider models may reduce transfer costs. To alleviate the limitations of PP/s, we measure *Transfer Cost Reduction per Second* (TCR/s) as:

$$\text{TCR/s} = \underbrace{\frac{\text{Image Dimension}}{\text{Seconds per Batch}}}_{\text{PP/s}} \times (\text{bpp}_{\text{raw}} - \text{bpp}_{\text{codec}}) \quad (9)$$

The measure now includes the compression performance as the difference between the expected bits per pixel (bpp) of compressed (bpp_{enc}) and uncompressed (bpp_{raw}) sensings. The raw bpp value refers to the bit depth, i.e., the sensor’s

2. <https://developer.nvidia.com/tensorrt>

3. <https://tvm.apache.org>

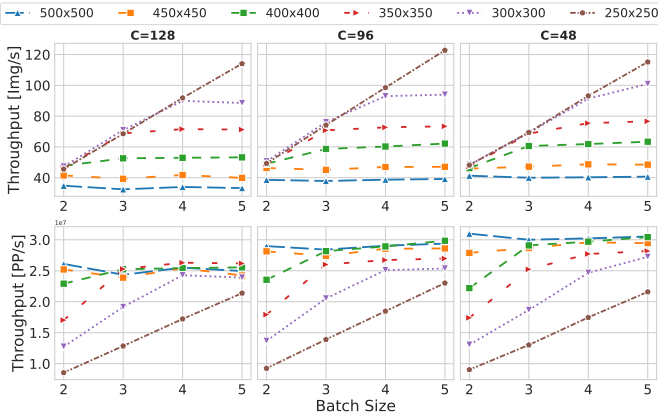


Fig. 5. Contrasting Throughput Measures

radiometric resolution and the number of bands. For example, the radiometric resolution of Sentinel-2 is 12 bits [28], so for three bands $\text{bpp}_{\text{raw}} = 3 \cdot 2^{12}$. The advantage of TCR/s is twofold. First, decoupling it from system-specific parameters, such as sensor resolution or orbital period, permits drawing generalizable insights regarding the relative trade-off between codec overhead and bitrate reduction. Second, operators can still assess the feasibility of the pipeline on target hardware and the expected downlinkable data volume by running the profiler with configurations that reflect deployment conditions.

4.3 Concurrent Task Execution

So far, this section has solely discussed the computational cost of a codec’s parametric transforms without considering pre- and post-processing. In particular, after applying the encoder transforms, it is still necessary to entropy code the output to compress the latent. Since FOOL’s entropy model is input adaptive, it requires a range coder. Although more recent range coders are efficient, they incur non-negligible runtime overhead. Therefore, given the unforgiving conditions of OEC, we argue that the entropy coder cannot be neglected in the design process and evaluation of a neural codec. FOOL virtually offsets the entire runtime overhead with simple concurrent task execution. The idea is to exploit the minimal interference of processes that draw from different resource types. For three sequentially incoming samples x_{i-1}, x_i, x_{i+1} , FOOL executes CPU-bound pre-processing of x_{i+1} , accelerator-bound inference of x_i , and CPU-bound post-processing of x_{i-1} . In this work, pre-processing corresponds to tiling the samples, and post-processing to entropy coding with rANS [49], [50]. Concurrently to inference on the accelerator, a process starts tiling x_i . After inference on x_{i-1} , $\hat{y}, \hat{z}, \hat{\sigma}, \hat{\mu}$ (Section 5.7) are persisted on the file system. Then, a separate process loads the data and losslessly compresses \hat{y}, \hat{z} with an entropy coder. We expect minimal interference between the processes, resulting in virtually no PP/s decrease.

5 THE FOOL’S COMPRESSION METHOD

We design the compression method based on three criteria. First, it should synergize with the profiling strategy (Sec-

tion 4.2). Second, it should embed context for feature compression without favoring a particular downstream task. Third, it should prioritize the integrity of downstream tasks but allow recovering human interpretable images without increasing the bitrate.

5.1 Model Building Blocks

For a focused evaluation and transparent discussion on the efficacy of our contributions in Section 6, we restrict FOOL to basic layer types and exclude methods from work on efficient neural network design (e.g., dilated convolutions to increase the receptive field). Moreover, basic layer types ensure widespread support across hardware vendors [48].

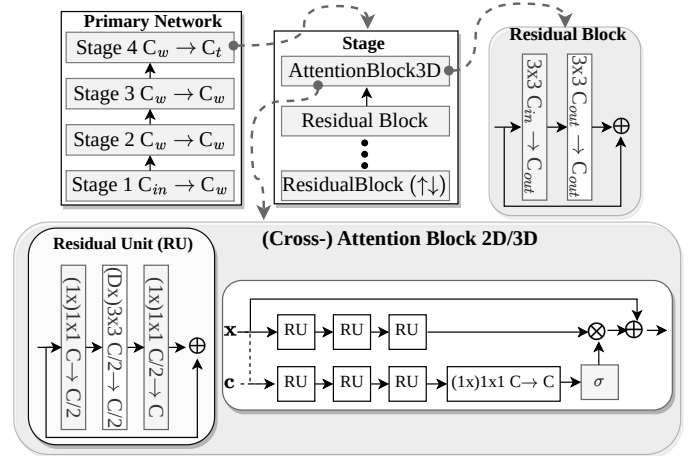


Fig. 6. Network Organization and Components

Figure 6 illustrates the architecture’s building blocks and how it organizes the primary networks for transform coding (Section 3.2). The primary networks have four stages that control the depth and width. Each stage has at most one lightweight attention block and at least one residual block. A residual block optionally up or downsamples the spatial dimensions. A stage’s width and depth parameters configure the number of channels and residual blocks within a stage.

5.2 Capturing Inter-Tile Dependencies

The input to the compression model are tiles that were partitioned to maximize processing throughput (Section 4.2), i.e., we consider an input x as a list with T separate image tensors $x_i \in \mathbb{R}^{C \times H \times W}$. To decrease transfer costs further, FOOL leverages the prior knowledge from partitioning (i.e., tiles corresponding to the same image) in two ways. The first is via weight-sharing with 2D Residual Blocks by reshaping the tensor to $T \cdot B \times C \times H \times W$. This way, we include further inductive bias during training by forward passing T similar tensors before each backpropagation. The second is with an inter-tile attention mechanism. Since self-attention from transformer architectures is prohibitively expensive for our purposes, even when applying it on downsampled representations as proposed in [51]. Therefore, we modify and extend the lightweight convolutional attention layer from [52]. The layer stacks residual units to increase the receptive field that primarily emphasizes local interactions.

FOOL partially replaces the need for global operations by assuming tiles to have “pseudo-temporal” dependencies. Intuitively, partitioning single captures that span large geographic areas may be similar to moving a video feed with large strides. In particular, tiles within the same regions or biomes have global dependencies. In the 3D version of the attention block from Figure 6, a residual unit consists of two $1 \times 1 \times 1$ convolution and a $D \times 3 \times 3$ convolution in between. The kernel size for the temporal dimension of attention layers (Section 5.1) is set as $D = 3$ for $T < 5$ and $D = 5$ for $T \geq 5$.

The advantage of 3D layers over concatenating channels and applying 2D convolutional operations (e.g., with channel attention [53]) is that it considerably reduces width. For example, given a 3×3 2D convolution with $T \times C$ in and out channels. Then for $T = 5$ image tensors, with dimensions $C = 64, H = 128, W = 128$, would require $5 \cdot 64 \cdot (3 \cdot 3 \cdot 5 \cdot 64 + 1) = 921,920$ parameters. Conversely, for a $5 \times 3 \times 3$ 3D convolution with the same in and out channels, it would result in $64 \cdot (5 \cdot 3 \cdot 3 \cdot 64 + 1) = 184,384$. Additionally, we reduce the number of multiply-and-accumulates from approximately 15 million to 9 million. Besides lowering memory requirements, this allows FOOL to scale model capacity with less friction against processing throughput.

5.3 Task-Agnostic Context for Feature Compression

The leftover spatial dependencies after encoding are commonly around high-contrast areas. Consider that high-contrast areas typically correspond to edges and other regions of interest, i.e., *keypoints*.

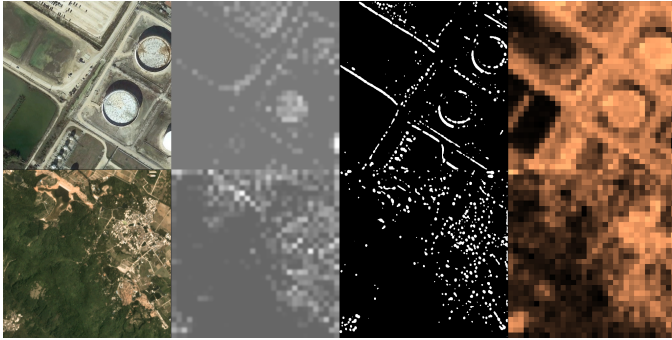


Fig. 7. Leftover Spatial Dependencies (middle left), Keypoints (middle right), Entropy Heatmap (right)

As an example, Figure 7 contrasts leftover pixel dependencies of \hat{y} from a LIC model [35] to keypoints output by a KeyNet [54] network. Therefore, we should further improve compression performance with side information by embedding keypoints as context for encoding as follows:

$$\hat{y} = Q(g_a(x; \theta)) \quad (10)$$

$$\mathbf{y}_{kp} = f_{ds}(k(\mathbf{x}) \odot \mathbf{x}; \omega_{kp}) \quad (11)$$

$$\mathbf{y}_{ca} = a_c(\mathbf{y}_{kp}, \omega_{ca}) \quad (12)$$

$$\hat{z} = Q(h_a(f_{ca}(\mathbf{y}_{kp}, \omega_{ca}); \theta_h)) \quad (13)$$

$$p_{\hat{y}|\hat{z}}(\hat{y} | \hat{z}) \leftarrow h_s(\hat{z}; \phi_h) \quad (14)$$

$$\tilde{\mathbf{h}} = g_s(\hat{y}; \phi) \quad (15)$$

where k is a keypoint extraction function $k: \mathbb{R}^{3 \times H \times W} \rightarrow \mathbb{R}^{1 \times H \times W}$, f_{ds} is a parametric downsampling function

$f_{ds}: \mathbb{R}^{1 \times H \times W} \rightarrow \mathbb{R}^{C' \times \frac{H}{2^n} \times \frac{W}{2^n}}$, and a_c is a single (2D) cross-attention block (Figure 6). The cross-attention block takes context as an additional input for weighting the latent with attention scores. For k , we use scores from a (frozen) pre-trained and simplified KeyNet [54] due to its robustness in diverse environments and low memory requirements (less than 6000 parameters). While this method should generalize to LIC, it complements feature compression exceptionally well and found quantizing and compressing \mathbf{y}_{ca} (i.e., $\tilde{\mathbf{h}} = g_s(Q(\mathbf{y}_{ca}), \psi)$) further lowers the bitrate without affecting task performance.

5.4 Compression Model Architecture

Figure 8 illustrates the compression model’s complete architecture. The dashed lines to Q indicate that we either quantize (and subsequently compress) the base latent or the cross-attention weighted latent. For the case of passing \mathbf{y}_{ca} to Q , we include an additional residual unit after attention-weighting. We skip applying the attention block to the highest input dimensions to reduce memory and computational costs. The non-linearity between the layers is ReLU to reduce vendor dependency of results from the system performance evaluation in Section 6.4. Residual blocks are two stacked 3×3 convolutions to increase the receptive field with fewer parameters and a residual connection for better gradient flow. For the remainder of the work, we

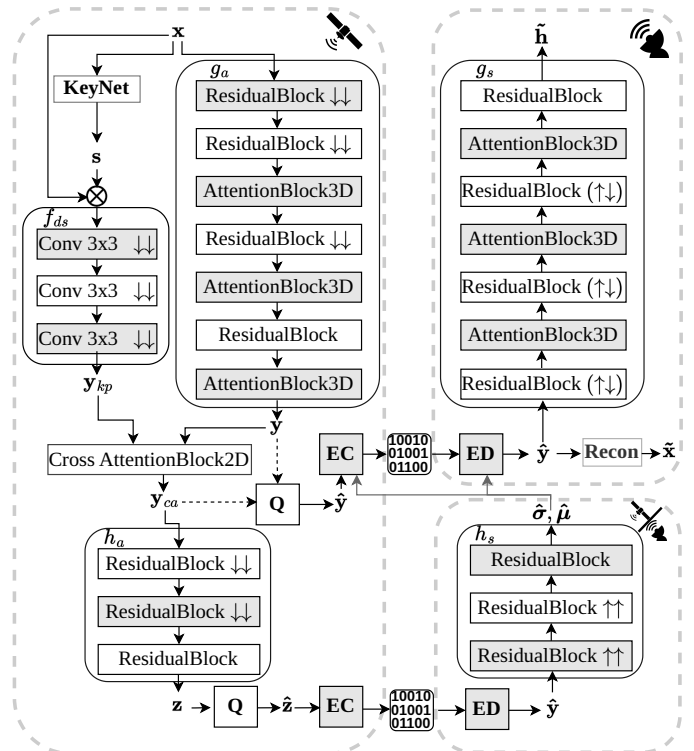


Fig. 8. The FOOL’s Compression Architecture

refer to the compression model as an encoder-decoder pair *enc, dec*. The encoder comprises g_a, h_a, k, f_{ds}, c_a , and the entropy coder. The decoder consists of the g_s and the entropy decoder. The entropy model $p_{\hat{z}}(\hat{z})$ and h_s are shared. Note that, despite deploying more components on the constrained sender, the encoder has significantly fewer

parameters than the decoder, since we increase the width of the receiver-exclusive components.

5.5 Single Encoder with Multiple Backbones and Tasks

Analogous to [17], consider a set of n shallow and deep layers pairs of backbones (i.e., foundational models):

$$\mathcal{M}_f = (\mathcal{H}_1, \mathcal{T}_1), (\mathcal{H}_2, \mathcal{T}_2) \dots (\mathcal{H}_n, \mathcal{T}_n) \quad (16)$$

The shallow layers map a sample to a shallow representation, i.e., $\mathcal{H}_i(x) = h_i$. Further, associate a separate set of m predictors $\mathcal{P} = \mathcal{P}_1, \dots, \mathcal{P}_m$ to the non-shallow (i.e., deep) layers of a backbone. Assume an encoder-decoder pair can sufficiently approximate the shallow layer’s representation of a particular backbone (i.e., $dec(enc(x)) = \tilde{h} \approx h_i$). Then, inputting \tilde{h} to \mathcal{T}_i , should result in the same predictions for all m predictors associated to \mathcal{T}_i . Since two shallow layers output different representations (i.e., $\mathcal{H}_i(x) \neq \mathcal{H}_j$), the encoder-decoder pair cannot replace the shallow layers for any \mathcal{H}_j where $i \neq j$. Accordingly, after training an initial encoder-decoder pair, FOOL instantiates $n - 1$ additional decoders, resulting in a set of separate $dec_1, dec_2 \dots dec_n$ decoders, i.e., one for each target backbone.

5.6 Image Reconstruction

After training the compression model and freezing encoder weights, FOOL separately trains a reconstruction model that maps \hat{y} to an approximation \tilde{x} of the original sample x .

5.6.1 Separate Training over Joint Optimization

We could reduce the distortion $d(x, \tilde{x})$ with a joint objective for training the reconstruction and compression model. While the resulting models would score higher on sum of error benchmarks, the added distortion term will result in higher bitrates. Instead, after optimizing the encoder with the objective of SVBI, we freeze the weights (i.e., “locking” in the rate performance). Then, we leverage the high mutual information between shallow features and the input to recover presentable approximations (Section 3.3.2). Image recovery is closely related to image restoration, such as super-resolution or denoising. The component is exchangeable with the state-of-the-art, as it is orthogonal to the compression task. For this work, we select SwinIR [55] due to its relative recency, computational efficiency, and simplicity.

5.6.2 Reconstruction Does not Replace Decoders

Approximations from decoders (Section 5.5) resulting in (near) lossless prediction, would evidence \hat{y} has sufficient information to reconstruct a sample for the input layers that result in comparable task performance. Hence, after training the encoder, we could replace all decoders with the reconstruction model to approximate the input sample. Nevertheless, sufficiency may not directly result in lossless prediction since artifacts perturb the distribution of samples derived from image recognition. We could account for the perturbation by finetuning for a relatively small number of iterations (indirectly shown in [15], [17]). The downside is that operators must maintain, store, and serve different versions of the otherwise identical backbones for each client separately. Worse, they may need to re-train the predictors

of the various downstream tasks for each backbone. Instead, we can train small decoders that directly map the low-dimensional encoder output to an adequate representation (i.e., the input to deeper layers). There are two advantages to introducing multiple decoders over multiple backbone weights. First, the number of additional weights operators must maintain only scales with supported backbones and not the number of backbone-task pairs. Second, the small decoder weights incur considerably less training and storage overhead than the weights of massive backbones.

5.7 Loss Functions

FOOL’s training algorithm starts with extracting the shallow layers of a particular detection model (teacher). Then, it freezes the encoder and trains newly initialized decoders $dec_1, dec_2, \dots, dec_n$ using the corresponding teacher models $\mathcal{H}_1, \mathcal{H}_2, \dots, \mathcal{H}_m$ (i.e., target shallow layers).

5.7.1 Rate-Distortion Loss Function for SVBI

To simplify the loss expression, we treat the components related to keypoints as part of h_a if we exclusively use it as a hint for the side-information network. Alternatively, as part of g_a , when used as an additional operation to reduce spatial dependencies before quantizing and entropy coding the latent. Analogous to the SVBI training objective [15], [17], we have a parametric analysis transform $g_a(x; \theta)$ that maps x to a latent vector z . Then, a quantizer Q discretizes z to \hat{z} for lossless entropy coding. Since we rely on HD (Figure 2) as a distortion function, the parametric synthesis transforms $g_s(\hat{x}; \phi)$ that maps \hat{y} to an approximation of a representation \tilde{h} . As introduced in [42], we apply uniform quantization Q , but replace Q with continuous relaxation by adding uniform noise $\eta \sim \mathcal{U}(-\frac{1}{2}, \frac{1}{2})$ during training for gradient computation.

Without a hyperprior, the loss function is:

$$\mathbb{E}_{x \sim p_x} D_{\text{KL}} [q || p_{\tilde{y} | x}] = \mathbb{E}_{x \sim p_x} \mathbb{E}_{\tilde{y} \sim q} - \log p(x | \tilde{y}) - \log p(\tilde{y}) \quad (17)$$

With side information, we condition on a hyperprior, such that each element \tilde{y}_i is now modeled as a Gaussian with its own mean and standard deviation:

$$p_{\tilde{y} | \tilde{z}}(\tilde{y} | \tilde{z}, \phi_h) = \prod_i \left(\mathcal{N}(\mu_i, \sigma_i^2) * \mathcal{U}\left(-\frac{1}{2}, \frac{1}{2}\right) \right)(\tilde{y}_i) \quad (18)$$

Where $\mathbf{z} = h_a(\hat{y}; \theta_h)$ and $\hat{\mu}, \hat{\sigma} = h_s(\tilde{z}; \phi_h)$. The final loss function results in the following:

$$\begin{aligned} \mathbb{E}_{x \sim p_x} D_{\text{KL}} [q || p_{\tilde{y}, \tilde{z} | x}] &= \mathbb{E}_{x \sim p_x} \mathbb{E}_{\tilde{y}, \tilde{z} \sim q} [\log q(\tilde{y}, \tilde{z} | x) \\ &\quad - \log p_{x | \tilde{y}}(x | \tilde{y}) - \log p_{\tilde{y} | \tilde{z}}(\tilde{y} | \tilde{z}) \\ &\quad - \log p_{\tilde{z}}(\tilde{z})] \end{aligned} \quad (19)$$

For the distortion term, we use the sum of squared errors between the shallow layer (teacher) representation and the compressor (student) approximation, i.e., $\text{sse}(\mathbf{h}, \hat{\mathbf{h}})$.

5.7.2 Mapping Encoder Output to Target Representations

After training the first enc, dec_1 pair, FOOL freezes enc weights, i.e., only applying the distortion term of the loss in Equation (19), for subsequent decoders $dec_2, dec_3 \dots dec_n$ (Section 5.5). Lastly, FOOL treats the reconstruction model rec as a decoder with the identity function as a teacher, i.e.,

$\mathcal{H}_{rec}(x) = x$. The difference from other decoders is that the target representation must be human-interpretable. Hence, we draw from image restoration that primarily focuses on synthesizing high-quality images and replace sse with the Charbonnier loss [56]:

$$\mathcal{L}_{rec} = \sqrt{\|\mathbf{x} - rec(enc(\mathbf{x}))\|^2 + \epsilon^2} \quad (20)$$

It is out of this work’s scope to exhaustively evaluate image restoration methods. Rather, the focus is to provide empirical evidence for the claims in Section 3.3.1. We simply found that despite performing comparable to other sums of error losses on benchmark metrics, subjective visual inspections revealed that Charbonnier resulted in smoother images.

6 EVALUATION

6.1 Experiment Design and Methodology

Our experiments reflect our aim to determine (i) the compression performance on aerial and satellite imagery without relying on prior knowledge and (ii) the feasibility of orbital inference.

6.1.1 Testbed

We benchmark [57] on an analytic and trace-driven [58] simulation based on results from a physical testbed with hardware summarized in Table 1. The power consumption is capped at 15W for the entire testbed. Our simulation

TABLE 1
Testbed Device Specifications

Device	CPU	GPU
Ground Server	16x Ryzen @ 3.4 GHz	RTX 4090
Edge (Nano Orin)	6x Cortex @ 1.5 GHz	Amp. 512 CC 16 TC
Edge (TX2)	4x Cortex @ 2 GHz	Pas. 256 CC
Edge (Xavier NX)	4x Cortex @ 2 GHz	Vol. 384 CC 48 TC

replicates a configurable CubeSat by imposing energy, memory, and bandwidth constraints. To simulate the downlink bottleneck with varying link conditions, parameterize link conditions and data volume (Section 3.1) using real-world missions [6], [29], [59], [60] as summarized in Table 2. Due to the orthogonality of compression to systems-related

TABLE 2
Constellation Link Conditions

Oper.	Constellation	Link	Rate (Mbps)	Pass (s)	Data Per Pass (GB)
Planet	Dove (3P/B13)	HSD 1	160	510	12.0
Maxar	WorldView	WorldView-3	1200	600	90
ESA	Copernicus	Sentinel 3A/B	560	600	40.0
NASA	Landsat	Landsat 8	440	120	39.6

challenges in OEC, we argue a focused simulation yields more insight results than running a full OEC simulator (e.g., [9]). Our intention is for FOOL to facilitate OEC as an auxiliary method. Therefore, we demonstrate the bitrate reduction and resource usage trade-off for various configurations representing the heterogeneity of available compute resources and nanosatellite constellations.

6.1.2 Third-Party Detection Models

FOOL derives the basic approach to accommodate multiple backbones with a single encoder (Section 5.5) from Franken-Split [17]. Foundational models (i.e., feature extractors or backbones) are interchangeable third-party components in SVBI. To complement previous work (Section 2.1) and further show the flexibility of SVBI, we focus on modern YOLO variants [61]. Figure 9 illustrates the pipeline to represent third-party detectors we prepare before evaluating codecs.

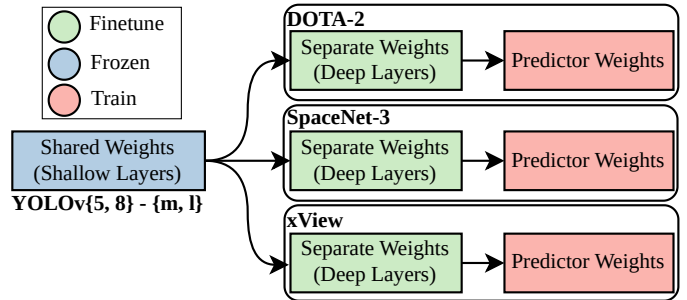


Fig. 9. Detection Pipeline for Evaluation

While the work in [17] did not explicitly evaluate object detection tasks, the support for two-stage detectors follows from the codec sufficiently approximating the representation of the feature extractor (i.e., the first stage). However, it is not apparent whether the general SVBI framework yields gains over image codecs when the targets are one-stage detectors. Therefore, to replicate a representative service for inference on aerial or satellite imagery, we apply simple transfer learning on open-source weights [61] for YOLOv5 and YOLOv8.

Image codecs pass a sample $dec(enc(\mathbf{x})) = \hat{\mathbf{x}}$ to the input layers of a target model. Feature codecs (i.e., SVBI methods) skip the shallow layers and pass $dec(enc(\mathbf{x})) = \hat{\mathbf{h}}$ to the deeper layers. Detection models with the same architecture share the frozen shallow layers (i.e., layers until the first non-residual connection).

We select detection tasks to represent varying mission conditions. DOTA-2 [62] for a more coarse-grained aerial task with comparatively lower Ground Sample Distance (GSD) and larger objects. SpaceNet-3 [63] for urban tasks (e.g., for traffic control) with high image resolutions. Lastly, xView [64] for disaster response systems where detection models rely on fine-grained details. To simplify the already intricate evaluation setup and to ease reproducibility, we deliberately refrain from more refined transfer learning methods. We merely require detection models with mAP scores that are moderately high to determine whether a codec can preserve fine-grained details for EO tasks on satellite imagery. For each architecture, we jointly finetune the deeper layers and train separate predictors that achieve around 35-65% mAP@50.

6.1.3 Training & Implementation Details

To demonstrate that FOOL can handle detection tasks without relying on prior information (Section 3.2), we do not optimize the compression model with the training set of the prediction tasks (i.e., DOTA-2, SpaceNet-3, xView). Instead, we curate other aerial and satellite datasets [65]–[72] that

cover region and sensor diversity. SVBI does not rely on labels, i.e., replacing the curation with any diverse enough dataset from satellite imagery providers (e.g., Google Earth Engine) should be possible.

We train one separate compression mode for each third-party detector using the shallow layers as teachers and verify whether the rate-distortion performance is comparable. Then, we freeze the encoder of the compression model for YOLOv5-L and discard all other encoders. Lastly, we freeze the remaining encoder’s weights and (re-)train the separate decoders to demonstrate clients may request inference on variations (YOLOv5-M) or newer models as they emerge (YOLOv8).

We fix the tile resolution to 512×512 during training. We load samples as a video sequence for FOOL by grouping tiles from the same image in partitioning order with random transformations to fill any remaining spots. After training, the tensor shape (i.e., the number of tiles and the spatial dimensions) may vary for each separate sample. We use PyTorch [73], CompressAI [74], and pre-trained detection models from Ultralytics [61]. To ensure reproducibility, we use torchdistill [75]. We use an Adam optimizer [76] with a batch size of 8 and start with an initial learning rate of $1 \cdot 10^{-3}$, then gradually lower it to $1 \cdot 10^{-6}$ with an exponential scheduler. We first seek a weight for the rate term in Equation (19) that results in lossless prediction with the lowest (best) bpp. Then, we progressively increase the term weight to evaluate trade-offs between rate and predictive loss.

6.1.4 Datasets Preparation

The train sets for third-party detectors and the train sets for the compression models are strictly separated. However, we create square tiles for all datasets by partitioning the images with a configurable spatial dimension and applying 0-padding where necessary. We extract bands from samples corresponding to RGB and convert them to 8-bit images, as to the best of our knowledge, there are no widespread open-source foundational models for detection with multi-spectral data yet. To ease direct comparisons, we convert the network detection labels of SpaceNet-3 by transforming the polygonal chains into bounding boxes. Lastly, since there are no publicly available labels for the xView and SpaceNet test sets, we create a 9:1 split on the train set.

6.1.5 Compression Performance Measures

To evaluate how codecs impact downstream task performance, we measure *Predictive Loss* as the drop in mean Average Precision (mAP) by inputting decoded samples. We regard a configuration to result in *lossless prediction* if there is less than 1% difference in expected mAP@50. We confirm the observations from [17] where the initial teacher only negligibly affects compression performance, and the predictive loss by a codec is comparable across target models (i.e., the retained information in shallow layers is similar across YOLO variations). Hence, for brevity, we aggregate the compression performance for each task separately, taking the highest predictive loss incurred on a detection model. We train the image reconstruction model using the same configurations as [55], and compare it with LIC models using common measures (PSNR, MS-SSIM, LPIPS [18]).

6.1.6 Baselines

We consider seminal work for image codecs as baselines with available open-source weights. Factorized Prior (FP) [42] as a relatively small model without side information. (Mean-)scale hyperprior (SHP, MSHP) [34], [35] for drawing comparisons to side information in LIC, and Joint autoregressive and hierarchical priors (JAHP) [35] that further improves compression performance with an autoregressive context model. Lastly, TinyLIC [36] represents recent work on efficient LIC design with state-of-the-art rate-distortion performance. Table 3 summarizes parameter

TABLE 3
Summary of Codec Parameter Distribution

Codec	Pars. Total	Pars. Enc.	Pars. Dec.	Shared
FOOL-L	4.97M	1.19M	4.06M	0.28M
FOOL-M	4.33M	0.69M	3.83M	0.16M
FOOL-S	3.91M	0.35M	3.66M	0.08M
SVBI-L	4.99M	1.25M	4.39M	0.65M
SVBI-M	4.35M	0.72M	3.91M	0.29M
SVBI-S	3.95M	0.38M	3.71M	0.16M
FP	7.03M	3.51M	3.51M	0.02M
MSHP	17.56M	14.06M	11.66M	8.15M
JAHP	25.50M	21.99M	19.60M	16.10M
TinyLIC	28.34M	21.23M	19.16M	12.05M

distributions between encoder and decoder components from LIC models.

To draw comparisons to existing work on SVBI, we combine FrankenSplit [17] and the Entropic Student [16] as a single baseline (BSVBI). We utilize FrankenSplit’s more efficient architecture design since it outperforms the latter without relying on finetuning the deeper layers. The encoder consists of stacked residual blocks (Section 5.1), and the decoder is instantiated from a YOLOv5+ blueprint (C3 blocks [61]). We scale the capacity of BSVBI by including side information from LIC (MSHP) and increasing the width and depth to match the various FOOL configurations. We train FOOL and BSVBI with the same dataset and training parameters (Section 6.1.3).

6.2 Rate Trade-off with Predictive Loss

We report the predictive loss as a percentage point difference using mAP@50 on foundational detection models.

6.2.1 Comparison to Image Codecs

Figure 10 illustrates the trade-off between bpp (left is better) and predictive loss (top is better) for LIC models on each task separately. We primarily focus on how FOOL compares to existing SVBI to draw new insights from the novel additions and confirm that our results on aerial and satellite imagery datasets with one-stage detectors are consistent with previous findings on standardized terrestrial datasets [15]–[17].

6.2.2 Comparison to Feature Codecs

Figure 11 contrasts the trade-off between bpp and predictive loss for FOOL and BSVBI with progressively increasing sizes (i.e., capacity). The efficacy of compressing shallow features is best shown by comparing BSVBI and FOOL to MSHP, as they rely on the same entropy model. The highest

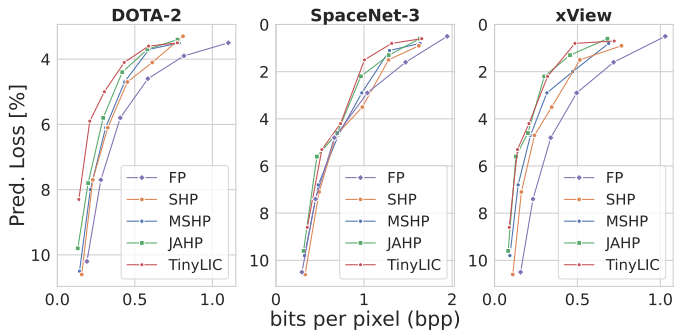


Fig. 10. Compression Performance Image Codecs

quality MSHP model results in about 3-4% predictive loss for DOTA-2. In contrast, the highest quality BSVBI-S model has 37x fewer encoder parameters but results in half the bitrate with no predictive loss. Despite BSVBI demonstrat-

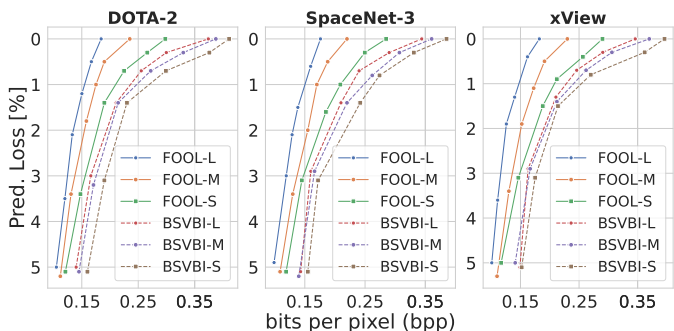


Fig. 11. Compression Performance Feature Codecs

ing strong compression performance, FOOL significantly outperforms BSVBI across all configurations. FOOL-S has a 51% lower bitrate for configurations with lossless prediction than the comparatively large BSVBI-L. Relative to the FOOL model with matching capacity (FOOL-L), BSVBI-L has twice the bitrate.

6.2.3 Ablation Study

We may consider BSVBI an ablation, as FOOL extends BSVBI’s architecture by placing 3D attention layers between the residual blocks and a cross-attention layer to include context. The auxiliary networks h_a and h_s (Section 5.3) are identical for FOOL and SVBI, i.e., three stacked residual blocks. Additionally, we perform ablation studies to assess by-component improvement and summarize the results for lossless predictions in Table 4.

The NITA models include the keypoint context without the inter-tile attention (ITA) layers. Analogous to BSVBI, we replace attention layers with residual blocks and match corresponding model sizes by increasing the depth and width of NITA models. NKPC-Ablation drops components for embedding keypoints, i.e., it only includes the IT attention layers.

The results show that relative to BSVBI, the task-agnostic context component contributes considerably more to rate reductions than the ITA layers that leverage inter-tile spatial dependencies. Still, we argue that the NTI-layers fulfill their

TABLE 4
Ablations Comparisons for Lossless Prediction

Model	DOTA-2 (bpp ↓↓)	SpaceNet-3 (bpp ↓↓)	xView (bpp ↓↓)
FOOL-L	0.1843	0.1760	0.1822
FOOL-M	0.2110	0.1993	0.2032
FOOL-S	0.2389	0.223	0.2290
BSVBI-L	0.3622	0.3440	0.3452
BSVBI-M	0.3775	0.3605	0.3699
BSVBI-S	0.3889	0.3852	0.3909
NITA-L	0.2209	0.1993	0.2032
NITA-M	0.2287	0.2193	0.2205
NITA-S	0.2433	0.2386	0.2407
NKPC-L	0.2839	0.2712	0.2768
NKPC-M	0.2926	0.2855	0.2883
NKPC-S	0.3116	0.3029	0.3112

purpose, to synergize with the partitioning strategy that maximizes processing throughput (Section 6.4).

6.3 Image Reconstruction Quality

We aim to demonstrate the feasibility of recovering presentable images from the compressed latent space of shallow features. We average results on DOTA-2, SpaceNet-3, and xView to reduce the bloat of reporting similar values summarize the results in Table 5. For transparency, we exclusively select samples from the lower quartile across all measures to qualitatively showcase the reconstruction.

TABLE 5
Comparison Between Recovery and Image Codecs

Model	PSNR↑↑	MS-SSIM↑↑	LPIPS↓↓	BPP↓↓	Pred. Loss↓↓
FOOL	36.51	15.43	0.1480	0.1808	-
FOOL-FT	35.56	14.57	0.1700	0.1808	-
FP-HQ	43.22	25.07	0.0896	1.0470	1.500
FP-MQ	35.56	16.55	0.2498	0.3200	7.508
MSHP-HQ	43.90	25.20	0.0841	1.0370	1.711
MSHP-MQ	36.45	16.83	0.2361	0.2787	1.180
JAHP-HQ	43.95	25.17	0.0818	1.0297	1.504
JAHP-MQ	36.61	16.95	0.2303	0.2641	6.397
TinyLIC-HQ	44.52	25.07	0.0683	1.0473	1.602
TinyLIC-MQ	37.42	17.27	0.2102	0.2899	5.499

HQ refers to the weights with the highest available quality, and MQ refers to mid-quality weights that roughly match FOOL in PSNR. FOOL-FT finetunes the reconstruction model for an additional $2.5 \cdot 10^5$ iterations using LPIPS [18]. Unsurprisingly, the LIC models achieve significantly better scores across all reconstruction measures (i.e., PSNR, MS-SSIM, and LPIPS). The advantage of FOOL is that it has a considerably lower bitrate with no predictive loss on tasks for which it had no prior information. Nonetheless, the results are considerably more interesting when contrasting FOOL to LIC models with mid-quality weights. Notice how FOOL matches reconstruction measures at no predictive loss and a 46-77% lower bitrate. Note that we did not find that the dataset significantly influences rate-distortion performance, except for a slight reduction in predictive loss (verified by training an FP model on the curation using the same setup as in [74]). Compression is a low-level vision task that generalizes well but may lack domain specificity when applying a standard rate-distortion reconstruction loss. In other words, the objective is the decisive difference

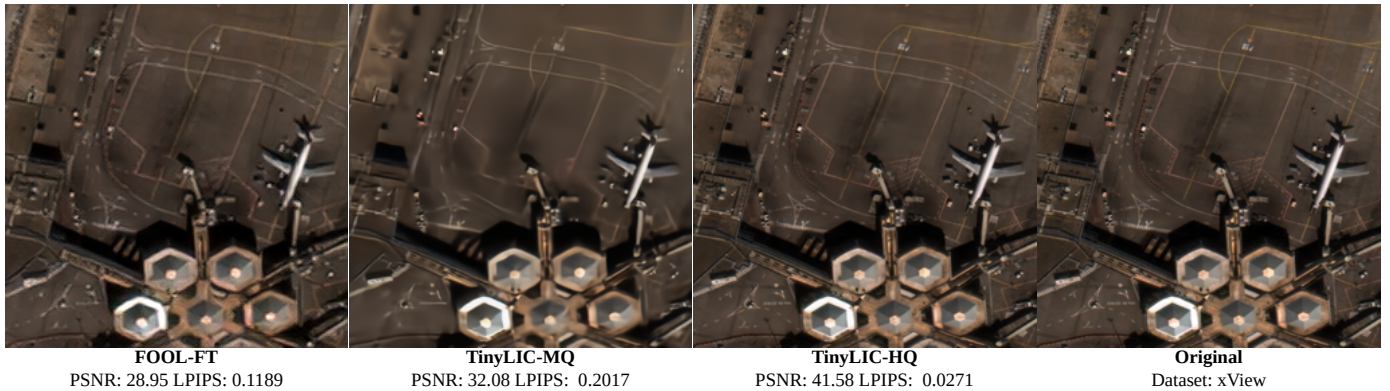


Fig. 12. Visual Comparison between FOOL Image Recovery and a State-of-the-Art LIC model

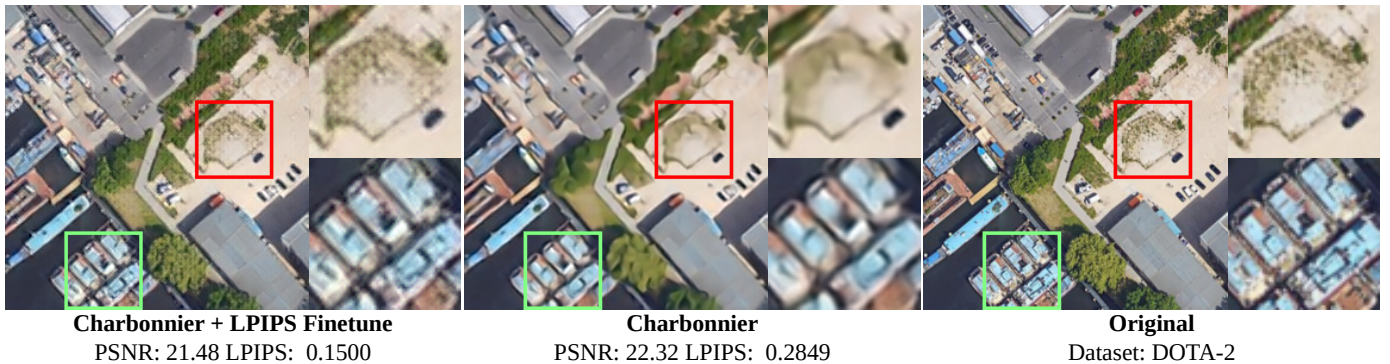


Fig. 13. Showcasing Potential of Recovery from Compressed Features with Finetuning for Perceptual Quality using LPIPS

between SVBI and LIC models. To provide some intuition to the LPIPS measure, we select an image where FOOL-FT achieves considerably lower PSNR than TinyLIC and contrast the results in Figure 12. Notice that the quality differences are most visible with fine-grained details, i.e., compared to TinyLic-HQ, TinyLic-MQ has a noticeable blur with some shadows completely missing in the bottom left. FOOL-FT preserves such details, despite lower PSNR, and this increase in perceptual quality is reflected in the LPIPS score. Figure 13 further visualizes the potential of reliably recovering fine-grained from the compressed latent space. Naturally, it should be possible to finetune the TinyLic-MQ to improve perceptual quality analogous to FOOL-FT. However, TinyLIC is still a significantly costlier model, with a worse rate and prediction performance. More pressingly, we stress that the *reliability* of a restoration model is bound by the available signals in the compressed latent space. Accordingly, we deliberately avoid generative models that prioritize realism over structural integrity. In particular, prioritizing realism over reliability defeats the primary purpose of image restoration, i.e., intervention by human experts in critical EO applications. A model outperforming experts does not imply that predictions may inexplicably be false. Specifically, where human cost is involved (e.g., disaster warning or relief [5]) it is paramount that experts can trust the codec to not include extrapolated elements to an image.

We argue that our results adequately underpin the statements in Section 3.3.1 and Section 3.3.2. In summary, if

the salient regions align, compressing for model prediction requires more information than for human observation. Task specificity determines rate savings and not an entity’s input interface. Targeting shallow features is minimally task-specific by relaxing the objective for lossless prediction on all possible tasks to those valuable for clients.

6.4 System Performance

The following evaluates FOOL’s resource usage and how well it can address the downlink bottleneck. The methodology resembles how the system aids operators in determining the correct model size for a target device and estimating the increase in data volume relative to bent pipes. We do not apply vendor-specific optimization (e.g., TensorRT) to ensure transparent evaluation and keep the results reasonably platform agnostic. Instead, we instantiate all models dynamically with half-precision in the native PyTorch environment (torch 1.14.0 with CUDA 11.4.315). Image codecs are omitted for conciseness, as even the state-of-the-art for efficient LIC design still runs considerably slower than the largest SVBI models.

6.4.1 Processing Throughput and Transfer Cost Reduction

We manually step through parts of the profiler (Section 4.2) for evaluation and to show how it estimates gains in downlinkable data volume. Consider the results from measuring the friction between model sizes and input dimensions on processing throughput in Figure 14. Notably, the processing throughput gain of FOOL-S over FOOL-M is significantly

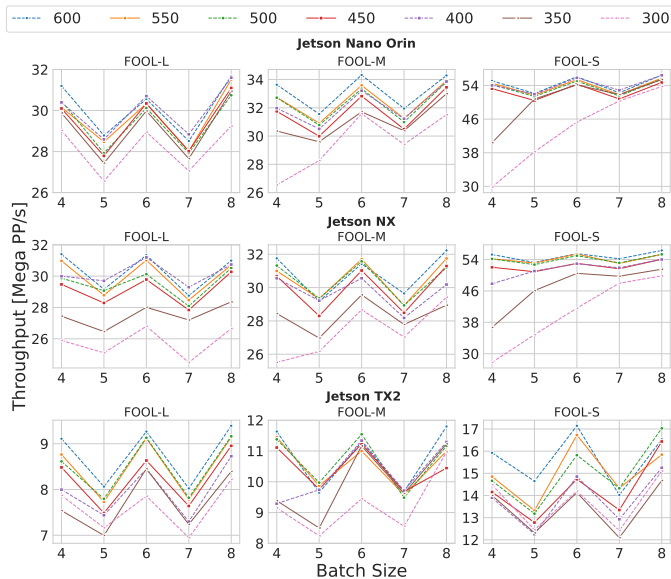


Fig. 14. Processing Throughput by Model Size

higher than FOOL-M over FOOL-L, despite FOOL-M having a comparable size difference to both models.

TABLE 6
Throughput Comparison Between Feature Codecs

Device	Model	Spatial Dim.	Batch Size	TCR/s $\uparrow\uparrow$
Orin Nano	FOOL-L	600x600	4	$7.4794 \cdot 10^8$
	FOOL-M	600x600	8	$7.6694 \cdot 10^8$
	FOOL-S	600x600	8	$1.3386 \cdot 10^9$
NX	FOOL-L	600x600	8	$7.5456 \cdot 10^8$
	FOOL-M	600x600	6	$8.1664 \cdot 10^8$
	FOOL-S	600x600	8	$1.3422 \cdot 10^9$
TX2	FOOL-L	600x600	8	$2.3696 \cdot 10^8$
	FOOL-M	600x600	8	$2.8067 \cdot 10^8$
	FOOL-S	600x600	6	$4.0751 \cdot 10^8$
Orin Nano	BSVBI-L	600x600	8	$6.1419 \cdot 10^8$
	BSVBI-M	600x600	5	$7.2504 \cdot 10^8$
	BSVBI-S	550x550	7	$6.9919 \cdot 10^9$
NX	BSVBI-L	600x600	8	$6.0007 \cdot 10^8$
	BSVBI-M	600x600	6	$7.3717 \cdot 10^8$
	BSVBI-S	600x600	6	$7.0720 \cdot 10^9$
TX2	BSVBI-L	600x600	7	$1.7861 \cdot 10^8$
	BSVBI-M	600x600	6	$2.9978 \cdot 10^8$
	BSVBI-S	600x600	7	$2.6818 \cdot 10^8$

Table 6 summarizes the configuration that maximizes profiler selection by TCR/s for all models on each device separately. Since bitrate variance is low between DOTA-2, SpaceNet-3, and xView, we average the bpp (Section 6.2) on the validation sets. The bold Model value indicates the adequate size of each model on a device, i.e., the model we will deploy to measure data volume downlinking in the following experiments. The bold TCR/s marks the highest overall value for a device, i.e., we can expect applying FOOL over BSVBI to result in considerably more downlinkable data on all devices. However, due to keypoint extraction and the ITA layers, FOOL’s processing throughput is slower than BSVBI’s. The overhead is particularly punishing for the most constrained device (i.e., the previous-generation TX2), where BSVBI-M has slightly higher TCR/s than FOOL-M,

despite the latter’s significantly better compression performance. Moreover, the profiler selects FOOL-S over FOOL-M/-L for the low-end current generation Orin Nano and high-end last-gen past generation NX. Conversely, the profiler decides on the mid-sized model for BSVBI across all devices despite FOOL’s compression performance scaling better.

Still, we argue that the results accentuate the findings from Section 6.2.2. Notice the contrast between TX2 and Nano Orin. One hardware generation was sufficient for the lowest-end device in the Jetson lineup to see a *three-fold* increase in TCR/s on FOOL-L over the last-generation midrange device. Thus, it is reasonable to claim that FOOL can (i) adequately leverage the current rapid progression of energy-efficient hardware improvement (i.e., with FOOL-M, L, and potentially larger variants) and (ii) is flexible enough to be deployed on more constrained devices using the small FOOL-S that still achieve substantial rate reduction.

6.4.2 Model Inference with Concurrent Task Execution

The following examines the claim in Section 4.3, i.e., whether FOOL’s compression pipeline can offset the runtime overhead of entropy coding. In other words, we evaluate whether interference between concurrent GPU and CPU-bound processes is negligible enough. We assume the worst case for interference, i.e., the CPU-bound processes constantly run concurrently by keeping them busy from an additional data stream when necessary. As all three devices have multicore CPUs and a dedicated GPU, we report results on the Nano Orin due to space constraints.

TABLE 7
Concurrent Entropy Coding and Effect on TCR/s

Model	TCR/s [conc]	TCR/s dec.	File Size (MB)	File/s	rANS conc. (MB/s)
FOOL-L	$7.26 \cdot 10^8$	2.94%	0.616	29	37.2
FOOL-M	$7.57 \cdot 10^8$	1.28%	0.462	31	38.3
FOOL-S	$1.32 \cdot 10^9$	1.06%	0.383	52	38.8
BSVBI-L	$5.96 \cdot 10^8$	2.88%	0.822	37	37.6
BSVBI-M	$7.15 \cdot 10^8$	1.37%	0.617	42	39.6
BSVBI-S	$6.91 \cdot 10^8$	1.28%	0.437	58	40.1

Table 7 summarizes the results from running the entire compression pipeline with concurrent task execution using the configurations that maximize TCR/s from Table 6. The bold values in the TCR/s dec. column indicates the size with the highest decrease. A file includes all model artifacts output by the neural codec’s DNN components for a single tile, i.e., the pipeline still needs to entropy code them to match the bpp in TCR/s calculations. File size refers to the storage requirements *per tile* of the encoder output tensors, i.e., the data volume the rANS process encodes. We compute file size by a worst-case upper bound by the encoder output tensor dimensionality (Section 5.4) without serialization formats that could exploit the sparsity of \hat{y} and \hat{z} . There are two essential findings from the results. First, the rANS process can consume tasks considerably faster than the inference process can produce them, i.e., there is no risk of backpressure within the pipeline. Second, there is only a minimal percentage decrease in TCR/s across all devices and models relative to sequential execution. Hence,

we argue that the pipeline successfully offsets the runtime overhead as claimed in Section 4.3 even without relying on a precomputed lookup table (e.g., tANS in ZSTD [77]).

The results are unsurprising when viewing the CPU (red) and GPU (blue) load of DNN inference without CPU-bound concurrent tasks in Figure 15. Since inference is

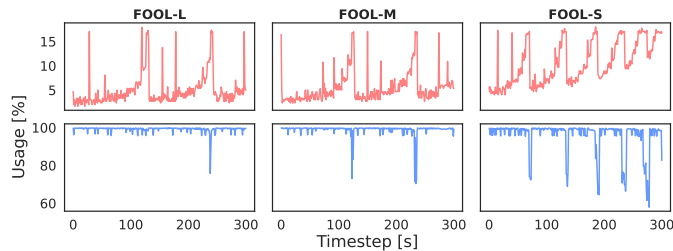


Fig. 15. CPU (red) and GPU (blue) Usage of Encoder Network

GPU-bound, CPU usage is low even when the GPU is under maximal load. Contrast this with the CPU and GPU usage in Figure 16 where we monitor [57] usage while running the entire pipeline with the two concurrent processes. If the

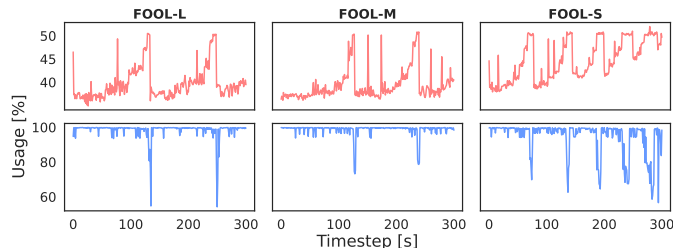


Fig. 16. CPU (red) and GPU (blue) Usage of Concurrent Pipeline

CPU-bound processing task were to interfere with the DNN execution, resource usage should reveal frequent drops in GPU load. Comparing FOOL-L to S and M reveals some dependency between DNN size and CPU usage. For FOOL-L, two discernible drops in GPU usage suggest some interference, which may explain the 2.9% decrease in TCR/s for FOOL-L and BSVBI-L. In contrast, there is no noticeable pattern difference in GPU load between Figure 16 and Figure 15 for S and M variants, explaining the negligible 1-1.5% TCR/s drop.

6.4.3 Downlinkable Data Volume

We now compare how methods can alleviate the downlink bottleneck using the traces from previous experiments. Figure 17 visualizes the transferable volume per downlink pass. Notice the logarithmic scale, i.e., FOOL improves downlinking using bent pipes by over two orders of magnitude without relying on prior information on the downstream tasks or crude filtering methods. For example, given Maxar’s WorldView-3 conditions [29], it would be possible to downlink roughly 9TB of sensor data per pass, before reaching downlink saturation. As a comparison, the state-of-the-art filtering method in [27] reports a 3× improvement, based on a definition of value. Note that to provide a realistic presentation of the opportunities SVBI provides, we assume that a nanosatellite processes tiles until reaching a

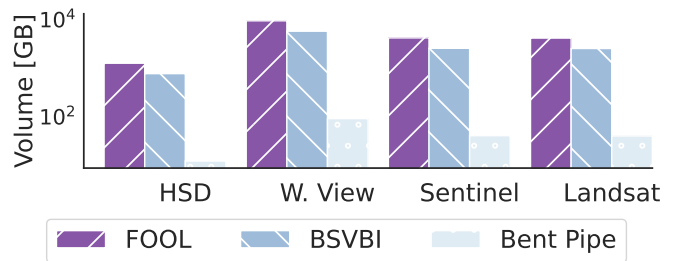


Fig. 17. Downlinkable Data Volumes by Link

downlink segment. Moreover, we disregard the “computational deadline”, i.e., it can process all the data before reaching a ground segment. This is reasonable since there should always be enough data to process. If not produced by a single onboard sensor, constellations may designate certain satellites as compression nodes using reliable, high-capacity local communication channels [78]. Further, from the throughput evaluation and the orbital period, it is infeasible that even the low-end current-generation Orin Nano would barely miss the computational deadline. We remind the reader that the presented throughput measures represent a worst-case lower bound, as no vendor-specific software or hardware optimization was applied.

6.4.4 Energy Consumption and Savings

The following investigates the energy usage of the selected model for each device. As the GPU and CPU usage patterns are highly similar, we measure by the time it takes until a method can *double* the downlinkable data. For example, if only downlinking 40 GB is possible with the unprocessed data, then we measure energy cost until the encoded size corresponds to 80 GB of raw captures. Figure 18 summarizes the results. As expected, processing on TX2 requires

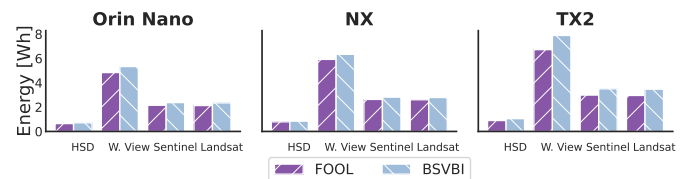


Fig. 18. Energy Cost of Compression Pipelines

more energy than on NX and Orin Nano as it is slower. Somewhat interesting is that the NX consumes more energy than the Orin Nano. As they execute the same models with comparable processing throughput, the results suggest that the newer Jetson lineup is more energy-efficient. Lastly, we measure savings from reduced transmission time, arguably an often undervalued advantage of compression. Admittedly, satellites will downlink as bandwidth permits, i.e., the transmission energy cost does not depend on the codec performance when there is saturation. Nonetheless, to intuitively show the amount of energy large volumes might require, we contrast with bent pipes in Figure 19. Given the link conditions, we measure the difference in energy cost between transmitting until saturation and transmitting the corresponding raw volume from Figure 17.

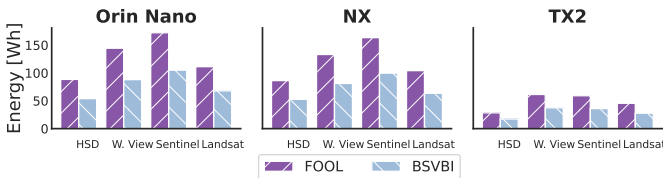


Fig. 19. Potential Energy Savings from Transmission

7 DISCUSSIONS AND LIMITATIONS

The following elaborates on limitations and highlights some research directions we find particularly promising to increase mission value. A general recommendation is to leverage recent advancements in image restoration [79] by utilizing generative models as robust priors and ensuring enough signals are available in the downlinked data to preserve structures at salient regions.

7.1 Downlink Saturation Handling

Readers may notice the omission of handling downlink saturation. A simple solution is to apply the compression model after the initial filtering. However, this contradicts FOOL's general design philosophy that it should not limit practicability due to prior knowledge of downstream tasks, excluding filtering as a preprocessing step, such as data deduplication. Therefore, we propose future work should prioritize segments according to their entropy in latent space and apply inpainting methods [80]. Intuitively, the lower the remaining uncertainty, given other tiles, the more reliable models can predict missing tiles.

7.2 Reliance on Foundational Models

Earth Observation (EO) requires considerations not included in common object detection objectives and architectural components, and widespread foundational models for satellite imagery have yet to emerge. Nonetheless, proprietary offerings already exist [81], and we argue that the community drives to open-source solutions will inevitably mitigate the limitation. We worked around not having access to an EO-native foundational model by only freezing the shallow layers to train the predictors. In other words, each predictor is complemented with a large backbone trained for a certain GSD range, i.e., not preserving the details with adequate granularity would result in predictive loss.

8 CONCLUSIONS

This work introduced a novel compression method that addresses the downlink bottleneck in LEO without relying on prior knowledge of downstream tasks. A rigorous evaluation showed that FOOL increases data volume with advancements that, to the best of our knowledge, are unprecedented. The rate reductions are primarily from the task-agnostic context. Additionally, the ITA layers further improve compression performance with an overhead that does not outweigh the processing throughput gains from batch parallelization. Lastly, we transparently listed limitations that future work should consider and identified promising future research directions for OEC based on novel insights.

ACKNOWLEDGMENT

We thank Alexander Knoll for providing us with the hardware infrastructure. Kerstin Bunte for her valuable suggestions. Florian Kowarsch for the fruitful discussions we held.

REFERENCES

- [1] R. Tubío-Pardavila and N. Kurahara, "18 - ground station networks," in *Cubesat Handbook* (C. Cappelletti, S. Battistini, and B. K. Malphrus, eds.), pp. 353–364, Academic Press, 2021.
- [2] L. Leung, V. Beukelaers, S. Chesi, H. Yoon, D. Walker, and J. Egbert, "Adcs at scale: Calibrating and monitoring the dove constellation," in *Proceedings of the AIAA/USU Conference on Small Satellites*, 2018.
- [3] S. Lee, A. Hutputanasin, A. Toorian, W. Lan, R. Munakata, J. Carnahan, D. Pignatelli, et al., "Cubesat design specification rev. 13," *California Polytechnic State University, San Luis Obispo, USA*, 2009.
- [4] R. P. Sishodia, R. L. Ray, and S. K. Singh, "Applications of remote sensing in precision agriculture: A review," *Remote Sensing*, vol. 12, no. 19, 2020.
- [5] A. Teodoro and L. Duarte, "Chapter 10 - the role of satellite remote sensing in natural disaster management," in *Nanotechnology-Based Smart Remote Sensing Networks for Disaster Prevention* (A. Denizli, M. S. Alencar, T. A. Nguyen, and D. E. Motaung, eds.), Micro and Nano Technologies, pp. 189–216, Elsevier, 2022.
- [6] K. Devaraj, R. Kingsbury, M. Ligon, J. Breu, V. Vittaldev, B. Klofas, P. Yeon, and K. Colton, "Dove high speed downlink system," in *Small Satellite Conference*, 2017.
- [7] D. Vasisht, J. Shenoy, and R. Chandra, "L2d2: Low latency distributed downlink for leo satellites," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pp. 151–164, 2021.
- [8] B. Tao, M. Masood, I. Gupta, and D. Vasisht, "Transmitting, fast and slow: Scheduling satellite traffic through space and time," in *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, MobiCom '23, (New York, NY, USA), p. Oct., Association for Computing Machinery, 2023.
- [9] B. Denby and B. Lucia, "Orbital edge computing: Nanosatellite constellations as a new class of computer system," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '20, (New York, NY, USA), p. 939–954, Association for Computing Machinery, 2020.
- [10] G. Furano, G. Meoni, A. Dunne, D. Moloney, V. Ferlet-Cavrois, A. Tavoularis, J. Byrne, L. Buckley, M. Psarakis, K.-O. Voss, et al., "Towards the use of artificial intelligence on the edge in space systems: Challenges and opportunities," *IEEE Aerospace and Electronic Systems Magazine*, vol. 35, no. 12, pp. 44–56, 2020.
- [11] C. Wu, Y. Li, M. Xu, C. Guo, Z. Yin, W. Gao, and C. Chi, "A comprehensive survey on orbital edge computing: Systems, applications, and algorithms," 2023.
- [12] G. Giuffrida, L. Fanucci, G. Meoni, M. Batič, L. Buckley, A. Dunne, C. van Dijk, M. Esposito, J. Hefele, N. Verduyssen, G. Furano, M. Pastena, and J. Aschbacher, "The Φ -sat-1 mission: The first on-board deep neural network demonstrator for satellite earth observation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–14, 2022.
- [13] S. Wang and Q. Li, "Satellite computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 10, no. 24, pp. 22514–22529, 2023.
- [14] C. Wu, Y. Li, M. Xu, C. Guo, Z. Yin, W. Gao, and C. Xi, "A comprehensive survey on orbital edge computing: Systems, applications, and algorithms," *arXiv preprint arXiv:2306.00275*, 2023.
- [15] Y. Matsubara, R. Yang, M. Levorato, and S. Mandt, "Supervised compression for resource-constrained edge computing systems," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 2685–2695, 2022.
- [16] Y. Matsubara, R. Yang, M. Levorato, and S. Mandt, "SC2 Benchmark: Supervised Compression for Split Computing," *Transactions on Machine Learning Research*, 2023.
- [17] A. Furtuanpey, P. Raith, and S. Dustdar, "Frankensplit: Efficient neural feature compression with shallow variational bottleneck injection for mobile edge computing," *IEEE Transactions on Mobile Computing*, pp. 1–17, 2024.

- [18] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Los Alamitos, CA, USA), pp. 586–595, IEEE Computer Society, jun 2018.
- [19] Y. Matsubara, M. Levorato, and F. Restuccia, "Split computing and early exiting for deep learning applications: Survey and research challenges," *ACM Comput. Surv.*, vol. 55, dec 2022.
- [20] Y. Yang, S. Mandt, and L. Theis, "An introduction to neural data compression," *Found. Trends. Comput. Graph. Vis.*, vol. 15, p. 113–200, apr 2023.
- [21] J. Ballé, P. A. Chou, D. Minnen, S. Singh, N. Johnston, E. Agustsson, S. J. Hwang, and G. Toderici, "Nonlinear transform coding," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 2, pp. 339–353, 2020.
- [22] G. Giuffrida, L. Diana, F. de Gioia, G. Benelli, G. Meoni, M. Donati, and L. Fanucci, "Cloudscout: A deep neural network for on-board cloud detection on hyperspectral images," *Remote Sensing*, vol. 12, no. 14, p. 2205, 2020.
- [23] Q. Zhang, X. Yuan, R. Xing, Y. Zhang, Z. Zheng, X. Ma, M. Xu, S. Dustdar, and S. Wang, "Resource-efficient in-orbit detection of earth objects," *arXiv preprint arXiv:2402.01675*, 2024.
- [24] A. Gadre, S. Kumar, and Z. Manchester, "Low-latency imaging and inference from lora-enabled cubesats," *arXiv preprint arXiv:2206.10703*, 2022.
- [25] A. Lu, Y. Cheng, Y. Hu, Z. Cao, Y. Chen, and Z. Li, "Satellite-terrestrial collaborative object detection via task-inspired framework," *IEEE Internet of Things Journal*, 2023.
- [26] D. MacKay, "Fountain codes," *IEE Proceedings - Communications*, vol. 152, pp. 1062–1068(6), December 2005.
- [27] B. Denby, K. Chintalapudi, R. Chandra, B. Lucia, and S. Noghabi, "Kodan: Addressing the computational bottleneck in space," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pp. 392–403, 2023.
- [28] M. Drusch, U. Del Bello, S. Carlier, O. Colin, V. Fernandez, F. Gascon, B. Hoersch, C. Isola, P. Laberinti, P. Martimort, A. Meygret, F. Spoto, O. Sy, F. Marchese, and P. Bargellini, "Sentinel-2: Esa's optical high-resolution mission for gmes operational services," *Remote Sensing of Environment*, vol. 120, pp. 25–36, 2012. The Sentinel Missions - New Opportunities for Science.
- [29] S. Cantrell, J. Christopherson, C. Anderson, G. L. Stensaas, S. N. R. Chandra, M. Kim, and S. Park, "System characterization report on the worldview-3 imager," tech. rep., US Geological Survey, 2021.
- [30] J. Carnahan, A. Hutputanasin, A. Johnstone, W. Lan, S. Lee, A. Mehrpavar, R. Munakata, D. Pignatelli, and A. Toorian, "Cube-sat design specification rev. 14.1," Tech. Rep. 141, California Polytechnic State University, San Luis Obispo, CA, USA, Feb. 2022.
- [31] B. Denby and B. Lucia, "Orbital edge computing: Machine inference in space," *IEEE Computer Architecture Letters*, vol. 18, pp. 59–62, Mar. 2019.
- [32] S. Singh, S. Abu-El-Haija, N. Johnston, J. Ballé, A. Shrivastava, and G. Toderici, "End-to-end learning of compressible features," in *2020 IEEE International Conference on Image Processing (ICIP)*, pp. 3349–3353, 2020.
- [33] C. E. Shannon, "Coding theorems for a discrete source with a fidelity criterion," in *IRE National Convention Record, 1959*, vol. 4, pp. 142–163, 1959.
- [34] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *International Conference on Learning Representations*, 2018.
- [35] D. Minnen, J. Ballé, and G. D. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.
- [36] G. Ma, Y. Chai, T. Jiang, M. Lu, and T. Chen, "Tinylic-high efficiency lossy image compression method," *arXiv preprint arXiv:2402.11164*, 2024.
- [37] R. Wightman, H. Touvron, and H. Jégou, "Resnet strikes back: An improved training procedure in timm," *arXiv preprint arXiv:2110.00476*, 2021.
- [38] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.
- [39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.
- [40] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021.
- [41] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986, 2022.
- [42] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.
- [43] Y. Matsubara, S. Baidya, D. Callegaro, M. Levorato, and S. Singh, "Distilled split deep neural networks for edge-assisted real-time systems," in *Proceedings of the 2019 Workshop on Hot Topics in Video Analytics and Intelligent Edges, HotEdgeVideo'19*, (New York, NY, USA), p. 21–26, Association for Computing Machinery, 2019.
- [44] M. Sbai, M. R. U. Saputra, N. Trigoni, and A. Markham, "Cut, distil and encode (cde): Split cloud-edge deep inference," in *2021 18th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 1–9, 2021.
- [45] M. Awais, M. Naseer, S. Khan, R. M. Anwer, H. Cholakkal, M. Shah, M.-H. Yang, and F. S. Khan, "Foundational models defining a new era in vision: A survey and outlook," *arXiv preprint arXiv:2307.13721*, 2023.
- [46] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 11, pp. 4037–4058, 2021.
- [47] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *2015 IEEE Information Theory Workshop (ITW)*, pp. 1–5, 2015.
- [48] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi, and J. Kepner, "Ai and ml accelerator survey and trends," in *2022 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–10, IEEE, 2022.
- [49] J. Duda, "Asymmetric numeral systems: entropy coding combining speed of huffman coding with compression rate of arithmetic coding," 2014.
- [50] J. Townsend, "A tutorial on the range variant of asymmetric numeral systems," *arXiv preprint arXiv:2001.09186*, 2020.
- [51] P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19–25, 2021*, pp. 12873–12883, Computer Vision Foundation / IEEE, 2021.
- [52] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Learned image compression with discretized gaussian mixture likelihoods and attention modules," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 7939–7948, 2020.
- [53] M.-H. Guo, T.-X. Xu, J.-J. Liu, Z.-N. Liu, P.-T. Jiang, T.-J. Mu, S.-H. Zhang, R. R. Martin, M.-M. Cheng, and S.-M. Hu, "Attention mechanisms in computer vision: A survey," *Computational visual media*, vol. 8, no. 3, pp. 331–368, 2022.
- [54] A. Barroso-Laguna, E. Riba, D. Ponsa, and K. Mikolajczyk, "Key.Net: Keypoint Detection by Handcrafted and Learned CNN Filters," in *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision*, 2019.
- [55] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, "Swinir: Image restoration using swin transformer," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1833–1844, 2021.
- [56] P. Charbonnier, L. Blanc-Feraud, G. Aubert, and M. Barlaud, "Two deterministic half-quadratic regularization algorithms for computed imaging," in *Proceedings of 1st International Conference on Image Processing*, vol. 2, pp. 168–172 vol.2, 1994.
- [57] P. Raith, T. Rausch, P. Prüller, A. Furutanpey, and S. Dustdar, "An end-to-end framework for benchmarking edge-cloud cluster management techniques," in *2022 IEEE International Conference on Cloud Engineering (IC2E)*, pp. 22–28, 2022.
- [58] P. Raith, T. Rausch, A. Furutanpey, and S. Dustdar, "faas-sim: A trace-driven simulation framework for serverless edge computing platforms," *Software: Practice and Experience*, vol. 53, no. 12, pp. 2327–2361, 2023.
- [59] C. Donlon, B. Berruti, A. Buongiorno, M.-H. Ferreira, P. Féménias, J. Frerick, P. Goryl, U. Klein, H. Laur, C. Mavrocordatos, J. Nieke,

- H. Rebhan, B. Seitz, J. Stroede, and R. Sciarra, "The global monitoring for environment and security (gmes) sentinel-3 mission," *Remote Sensing of Environment*, vol. 120, pp. 37–57, 2012. The Sentinel Missions - New Opportunities for Science.
- [60] National Aeronautics and Space Administration, "Landsat-8 / ldcms (landsat data continuity mission)." <https://www.eoportal.org/satellite-missions/landsat-8-ldcms#eop-quick-facts-section>, 2024. Accessed: 20 March 2024.
- [61] G. Jocher, A. Chaurasia, and J. Qiu, "YOLO by Ultralytics," Jan. 2023.
- [62] J. Ding, N. Xue, G.-S. Xia, X. Bai, W. Yang, M. Yang, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang, "Object detection in aerial images: A large-scale benchmark and challenges," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021.
- [63] A. Van Etten, D. Lindenbaum, and T. M. Bacastow, "Spacenet: A remote sensing dataset and challenge series," *arXiv preprint arXiv:1807.01232*, 2018.
- [64] D. Lam, R. Kuzma, K. McGee, S. Dooley, M. Laielli, M. Klaric, Y. Bulatov, and B. McCord, "xview: Objects in context in overhead imagery," *arXiv preprint arXiv:1802.07856*, 2018.
- [65] Class, "Airbus aircraft detection dataset." <https://universe.roboflow.com/class-dvpyb/airbus-aircraft-detection>, jan 2023. visited on 2024-01-31.
- [66] R. Bahmanyar, E. Vig, and P. Reinartz, "Mrcnet: Crowd counting and density map estimation in aerial and ground imagery," *arXiv preprint arXiv:1909.12743*, 2019.
- [67] R. Hänsch, J. Arndt, D. Lunga, M. Gibb, T. Pedelose, A. Boedihardjo, D. Petrie, and T. M. Bacastow, "Spacenet 8-the detection of flooded roads and buildings," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1472–1480, 2022.
- [68] A. Van Etten and D. Hogan, "The spacenet multi-temporal urban development challenge," *arXiv preprint arXiv:2102.11958*, 2021.
- [69] J. Shermeyer, D. Hogan, J. Brown, A. Van Etten, N. Weir, F. Pacifici, R. Hansch, A. Bastidas, S. Soenen, T. Bacastow, et al., "Spacenet 6: Multi-sensor all weather mapping dataset," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 196–197, 2020.
- [70] J. Shermeyer, T. Hossler, A. Van Etten, D. Hogan, R. Lewis, and D. Kim, "Rareplanes: Synthetic data takes flight," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 207–217, 2021.
- [71] T. BAKIRMAN and E. SERTEL, "A benchmark dataset for deep learning-based airplane detection: Hrplanes," *International Journal of Engineering and Geosciences*, vol. 8, no. 3, p. 212–223, 2023.
- [72] M. Rahnemoonfar, T. Chowdhury, A. Sarkar, D. Varshney, M. Yari, and R. Murphy, "Floodnet: A high resolution aerial imagery dataset for post flood scene understanding," *arXiv preprint arXiv:2012.02951*, 2020.
- [73] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., "Pytorch: An imperative style, high-performance deep learning library," *arXiv preprint arXiv:1912.01703*, 2019.
- [74] J. Bégin, F. Racapé, S. Feltman, and A. Pushparaja, "Compressai: a pytorch library and evaluation platform for end-to-end compression research," *arXiv preprint arXiv:2011.03029*, 2020.
- [75] Y. Matsubara, "torchdistill: A modular, configuration-driven framework for knowledge distillation," in *International Workshop on Reproducible Research in Pattern Recognition*, pp. 24–44, Springer, 2021.
- [76] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [77] Y. Collet and M. Kucherawy, "Zstandard Compression and the application/zstd Media Type." RFC 8478, Oct. 2018.
- [78] M. Mitry, "Routers in space: Kepler communications' cubesats will create an internet for other satellites," *IEEE Spectrum*, vol. 57, no. 2, pp. 38–43, 2020.
- [79] J. Su, B. Xu, and H. Yin, "A survey of deep learning approaches to image restoration," *Neurocomputing*, vol. 487, pp. 46–65, 2022.
- [80] X. Zhang, D. Zhai, T. Li, Y. Zhou, and Y. Lin, "Image inpainting based on deep learning: A review," *Information Fusion*, vol. 90, pp. 74–94, 2023.
- [81] D. Algorithms Team, "Introducing yolo-nas-sat: Small object detection at the edge," Mar 2024.