Task-relevant Representation Learning for Networked Robotic Perception

Manabu Nakanoya*,1, Sandeep Chinchali*,2, Alex Anemogiannis3, Akul Datta3, Sachin Katti2, Marco Pavone2

Abstract—Today, even the most compute-and-power-constrained robots can measure complex, high data-rate video and LIDAR sensory streams. Often, such robots, ranging from low-power drones to space and subterranean rovers, need to transmit high-bitrate sensory data to a remote compute server if they are uncertain or cannot scalably run complex perception or mapping tasks locally. However, today’s representations for sensory data are mostly designed for human, not robotic, perception and thus often waste precious compute or wireless network resources to transmit unimportant parts of a scene that are unnecessary for a high-level robotic task. This paper presents an algorithm to learn task-relevant representations of sensory data that are co-designed with a pre-trained robotic perception model’s ultimate objective. Our algorithm aggressively compresses robotic sensory data by up to $11 \times$ more than competing methods. Further, it achieves high accuracy and robust generalization on diverse tasks including Mars terrain classification with low-power deep learning accelerators, neural motion planning, and environmental timeseries classification.

I. INTRODUCTION

Imagine a future Mars or subterranean rover that captures high-bitrate video and LIDAR sensory streams as it charts uncertain terrain, some of which it cannot classify locally. How should these robots represent, compress, and transmit their rich sensory data over bandwidth-limited wireless networks, especially if the intended audience is often a compute-intensive, remote machine learning model, not necessarily a human viewer? Indeed, even a single RGB-D (depth) camera stream produces upwards of 45 Megabytes/second of data [21], while the deep-space network only has a communication bandwidth of 0.5-4 Megabits/second [2].

More broadly, today’s representations for sensory data mostly optimize for human, not robotic, perception and thus try to faithfully represent every pixel or point-cloud in a scene. Ideally, however, resource-constrained robots should represent only salient parts of sensory streams for remote perception and planning tasks to reduce the computational cost of encoding, storing, and transmitting sensory data. This paper presents a general algorithmic framework to learn such concise, task-relevant representations. While recent work has used specialized deep neural networks (DNNs) to improve LIDAR or JPEG image compression [29], [6], [17], [30], [3], our key novelty is to use a general-purpose, pre-trained task module to guide representation learning, which allows us to generalize to multiple sensor modalities.

Specifically, as shown in Fig. 1, our design provisions a pre-trained, differentiable task module at a central server, which allows multiple robots to share an upfront cost of training the model and benefit from centralized model updates. Our key technical insight is to co-design a minimal sensory representation that is tightly coupled with the task module’s objective. Henceforth, co-design means that the pre-trained task network parameters are fixed and the task objective guides what salient parts of a sensory stream to encode. By also learning a task-relevant decoder, we enable robots to leverage a variety of pre-trained, publicly-available task modules without modifying their required input dimensions. Our approach is complementary to advances in on-robot computation and instead pertains to compute-intensive, potentially collaborative, tasks that require remote assistance.

Literature Review: Recent work has applied information bottleneck theory [28] to build controllers that focus on actionable, task-relevant visual inputs for robust, generalizable navigation and grasping policies [23], [22], [25]. In contrast, we introduce a novel algorithm for co-designing communication and machine perception, which uses pre-trained task modules to learn salient, efficiently-computable representations. Our co-design approach differs from specialized solutions that use DNNs for JPEG image compression [17], [30] and instead uses diverse, off-the-shelf task modules to learn representations for images, obstacle point clouds, and environmental sensor timeseries.

Our work is also related to cloud robotics [12], [14], where robots use remote servers to augment their grasping [16],
object recognition [11], and mapping capabilities [19]. Prior work balances cloud computing accuracy with communication delay by learning to query the cloud [4], [27]. Rather than address when to communicate, we instead address how to represent data for task-centric communication. Finally, our work is related to variational and multi-task autoencoders [13], [9], which compress inputs to minimize regularized reconstruction loss, often with domain-specific output layers in the multi-task case. In contrast, by instead focusing on task loss, we compress sensory data 11 × more than standard autoencoders.

**Statement of Contributions:** In light of prior work, our contributions are three-fold. First, we introduce a novel formulation that co-designs a distributed encoder/decoder with a fixed task network. Second, we develop a novel algorithm that aggressively compresses sensory data for diverse robotic tasks. Third, we show how to flexibly allocate computation between a compute-limited robot and server, which renders our approach compatible with low-power DNN accelerators like the Google Edge Tensor Processing Unit (TPU) [1].

**Organization:** This paper is organized as follows. In Sec. II, we introduce a novel, general problem of co-designing sensory representations for robotic perception. To address this problem, Sec. III presents our co-design algorithm and highlights significant compression gains for an illustrative example of linear systems. Then, Sec. IV evaluates our algorithm on diverse perception tasks with compute-efficient DNNs. Finally, Sec. V concludes with future directions.

## II. Problem Statement

We now introduce the core compute modules for information flow between a robot and a central server (Fig. 1), and then formalize our problem statement. First, a robot measures a sensory input \( x \in \mathbb{R}^n \), such as an image or LIDAR point cloud. Without loss of generality, \( x \) could represent a single sensory sample a robot wishes to transmit or a window \( w \) of correlated samples measured from time \( t - w \) to \( t \), such as a segment of video, denoted by \( x = x^{t-w:t} \).

**Robot Encoder:** The encoder maps raw input \( x \) to a concise representation \( z \in \mathbb{R}^d \), denoted by \( z \sim p(z|x; \theta_{enc}) \), where \( \theta_{enc} \) are encoder model parameters, such as learned DNN weights. Henceforth, \( z \) is referred to as a bottleneck representation, since it is compressed via an information bottleneck, such as a DNN hidden layer, to a size \( Z \ll n \).

**Server Decoder:** The encoder transmits bottleneck representation \( z \) over a wireless link to a differentiable decoder at a central server, which generates a reconstructed estimate of the raw sensory input \( \hat{x} \sim q(x|z; \theta_{dec}) \). The parameters of a decoder model, such as a DNN, are denoted by \( \theta_{dec} \).

**Server Task Network:** Finally, the decoded input \( \hat{x} \) is directly passed through a pre-trained, differentiable task module, which serves as the key distinction of our work from a standard autoencoder. An example of a task module could be a DNN object detector that predicts object locations and classes \( \hat{y} = f(\hat{x}; \theta_{task}) \) using model parameters \( \theta_{task} \). Crucially, by decoding to the original input dimension \( n \), our approach enables robots to directly utilize a plethora of pre-trained, publicly-available task modules that expect an input dimension \( n \) without re-training on custom bottleneck representations \( z \). In practice, the encoder, decoder, and task module can all be DNNs. We assume sensory input \( x \) and its label \( y \) are drawn from a domain-specific distribution \( D \), such as the space of all Mars terrain images and labels.

### A. Task and Reconstruction Optimization Objectives

Our principal objective is to minimize the task loss \( L_{task}(y, \hat{y}; \theta_{task}) \), which compares the resultant task outputs \( y = f(x; \theta_{task}) \) using original input \( x \) and predicted outputs \( \hat{y} = f(\hat{x}; \theta_{task}) \) using a decoded input \( \hat{x} \). Our key technical insight is that robotic perception tasks can often tolerate considerable distortion in the input estimate \( \hat{x} \), such as omitting irrelevant parts of an image/map for classification/planning, as long as the downstream task module \( f(\cdot; \theta_{task}) \) can still achieve its goal. Thus, decoded input \( \hat{x} \) should only represent task-relevant features, which enables highly-compressed representations \( z \) to be transmitted over a wireless link.

Optionally, a roboticist might want to also minimize reconstruction loss, in cases where decoded inputs \( \hat{x} \) should be moderately human-interpretable to view or debug portions of an image. For such scenarios, loss function \( L_{recon}(x, \hat{x}; \theta_{enc}, \theta_{dec}) \) incentivizes faithful reconstruction of a scene, which, in practice, could be the standard variational autoencoder regularized loss [13]. We provide a general, flexible co-design framework that allows a roboticist to optimize a weighted combination of task and reconstruction loss, with reconstruction loss weight \( \lambda \geq 0 \):

\[
L_{weight}(x, \hat{x}, y; \hat{y}; \theta_{task}, \theta_{enc}, \theta_{dec}) = L_{task}(y, \hat{y}; \theta_{task}) + \lambda L_{recon}(x, \hat{x}; \theta_{enc}, \theta_{dec}).
\]  

Our experiments evaluate the scenario of strictly optimizing for task loss (\( \lambda = 0 \)) as well as various values of \( \lambda > 0 \), which introduces a regularization term to incentivize highly-compressed representations that still yield human-interpretable reconstructions. Having defined our optimization objective, we now formalize our problem statement.

### B. Problem Statement

**Problem 1 (Sensory Co-design for Machine Perception):** Given a differentiable task module \( f(\cdot; \theta_{task}) \) with fixed, pre-trained parameters \( \theta_{task} \), fixed bottleneck dimension \( Z \), and reconstruction loss weight \( \lambda \geq 0 \), find robot encoder and server decoder parameters \( \theta_{enc} \) and \( \theta_{dec} \) that minimize weighted loss (Eq. 1) over data distribution \( D \):

\[
\theta_{enc}^*, \theta_{dec}^* = \arg\min_{\theta_{enc}, \theta_{dec}} \mathbb{E}_{(x,y) \sim D} L_{weight}(x, \hat{x}, y; \hat{y}; \theta_{task}, \theta_{enc}, \theta_{dec}),
\]

where \( \hat{x} = q(p(x; \theta_{enc}); \theta_{dec}) \) and \( \hat{y} = f(\hat{x}; \theta_{task}) \).

**Problem 2** is widely applicable to resource-limited robots, such as space and mining rovers, that can only unlock large benefits of remote computation if they send task-relevant information over a bandwidth-limited network. The key novelty of our formulation is that we leverage knowledge of a task objective, through \( f(\cdot; \theta_{task}) \), to guide task-relevant representation learning of encoder/decoder parameters \( \theta_{enc}, \theta_{dec} \). Further, we enable a roboticist to set bottleneck size \( Z \) in a principled, precise manner based on a network’s maximum allowable data-rate. For example, the data-rate could be the size of \( Z \) 32-bit floating point values sent at a certain communication frequency.
C. Illustrative Example: Linear Systems Setting

To illustrate the compression benefits of optimizing for task loss in Prob. 1, we consider a toy example where the encoder, decoder, and task module are matrices. Specifically, consider a simple robotic sensor network where a central server must estimate a function $y = Kx$ of a potentially large sensor measurement $x$, without necessarily sending the full input $x$ over a wireless link. First, robot encoder matrix $A = \theta_{\text{enc}} \in \mathbb{R}^{2 \times n}$ generates encoding $z = p(x; \theta_{\text{enc}}) = Az$. After $z$ is sent over a wireless link, it is decoded to $\hat{x} = q(z; \theta_{\text{dec}}) = Bz$ by linear decoder $B = \theta_{\text{dec}} \in \mathbb{R}^{n \times z}$. Finally, task matrix $K \in \mathbb{R}^{m \times n}$ generates the output of the linear estimation problem $y = \tilde{R}$. The task loss penalizes error in the linear estimate $y = \hat{K}x$, and the reconstruction loss optionally penalizes error in the sensor estimate $\hat{x}$ in case elements of $\hat{x}$ need to be sanity-checked. Both task and reconstruction loss are quadratic, yielding weighted loss:

$$L_{\text{weight}} = \left\| (Kx - KBAx) \right\|^2 + \lambda \left\| (x - BAx) \right\|^2. \tag{2}$$

To show the full benefits of task-based compression, we now provide an analytical solution for Prob. 1 for the special case when $\lambda = 0$ (pure task objective).

**Theorem 1 (Linear Task-Aware Compression):** Consider task matrix $K \in \mathbb{R}^{m \times n}$ with rank $r$ and compact singular value decomposition (SVD) $K = U \Sigma V^T$, where $U \in \mathbb{R}^{m \times r}$ and $V^T \in \mathbb{R}^{r \times n}$ are semi-unitary. Then, for bottleneck dimension $r$, setting $A = V^T$ and $B = V$ solves Prob. 1 with zero task loss $\left\| (Kx - KBAx) \right\|^2$ (Eq. 2 with $\lambda = 0$) for any $x \in \mathbb{R}^n$. Further, there are no encoder and decoder matrices with bottleneck $Z < r$ that achieve zero task loss.

**Proof:** The task loss is zero when, for any $x \in \mathbb{R}^n$, $y = Kx = \tilde{y} = KBAx$. This is achieved when $\tilde{B} = V$ and $A = V^T$:

$$\tilde{y} = KBAx = (U\Sigma V^T) \times \frac{V}{B} \times \frac{V^T}{A} \times x = U\Sigma V^T x = Kx = y,$$

where $V^T V = I_{r \times r}$ since $V$ is semi-unitary. We now show there are no other solutions with bottleneck $Z < r$. For the sake of contradiction, suppose there exist $\tilde{A} \in \mathbb{R}^{r \times n}, \tilde{B} \in \mathbb{R}^{m \times Z}$ with $Z < r$ that achieve zero task loss. Then:

$$\text{rank}(\tilde{B}\tilde{A}) \leq \min(\text{rank}(\tilde{A}), \text{rank}(\tilde{B})) \leq Z, \quad \text{and} \quad \text{rank}(K\tilde{B}\tilde{A}) \leq \min(\text{rank}(K), \text{rank}(\tilde{B}\tilde{A})) \leq Z.$$

However, in order for $\tilde{A}, \tilde{B}$ to achieve zero task loss, we must have $Kx = KBAx$ for any $x \in \mathbb{R}^n$, meaning we must have $K = K\tilde{B}\tilde{A}$. However, this is a contradiction since $\text{rank}(K\tilde{B}\tilde{A}) \leq Z < r = \text{rank}(K)$. Thus, we conclude there are no solutions that achieve zero task loss for $Z < r$.

**Compression benefits:** Our key insight is that, by co-designing with fixed task matrix $K$, we achieve zero task loss with a compression gain of $\text{rank}(K) \approx n$ compared to sending full input $x$. However, we do not necessarily have zero reconstruction loss since matrix $V^T$ is only semi-unitary, meaning it is not required for $V^TV = BA$ to be an identity matrix. In fact, if we want zero reconstruction loss, then we have $\hat{x} = BAx = x$ only when $B = A^{-1}$, requiring $B$ and $A$ to be square $n \times n$ matrices, yielding no compression gain. Crucially, our solution differs from naively just computing $y = Kx$ directly at the robot, since we only transmit a small representation $z = V^T x$ and use the modular task matrix $K$ at the server. Finally, we note that the choice of $A = V^T$ and $B = V$ is not unique for bottleneck $r$. For example, we can scale the encoder to $aA$ and decoder to $bB$ where $ab = 1$.

**Weighted linear case:** For the weighted $\lambda > 0$ case, we analytically compute the gradient of weighted loss (Eq. 2) and solve Prob. 1 using gradient descent. Fig. 2 shows results for a toy linear problem with input and output dimensions $n = 6$ and $m = 3$, where we achieve optimal task performance with a $2 \times$ compression gain. Due to space constraints, we provide further details online\textsuperscript{2}. Our toy example shows that, even for linear systems, we achieve large compression gains by co-designing with a task objective, which we extend to complex DNNs in the next section.

III. TASK-AWARE CO-DESIGN ALGORITHM

In general, analytically solving Problem 1 is challenging, especially when the task network and encoder/decoder are DNNs. We now present Algorithm 1, which learns an encoder/decoder that approximately solves Problem 1. Importantly, our algorithm yields experimental results that match analytical calculations for simple linear settings and also scales to deep learning tasks, presented in Sec. 1V.

Alg. 1 takes as input a bottleneck dimension $Z$, set based on communication data-rate limits, and user-desired reconstruction weight $\lambda$, on line 1. Then, it randomly initializes the encoder/decoder parameters on line 2, which are learned during training unlike task parameters $\theta_{\text{task}}$ (line 3). The main loop repeats for $T$ learning rounds, where we sample inputs from labeled training dataset $D = \{x^i, y^i\}$ where each $(x^i, y^i) \sim D$ (line 5), encode/decode sensory data (lines 6–7), and finally invoke the task module on line 8. The key step for co-design is line 9, where we backpropagate weighted loss $L_{\text{weight}}$ (Eq. 1) to update parameters $\theta_{\text{enc}}, \theta_{\text{dec}},$ while keeping the task parameters $\theta_{\text{task}}$ fixed. Importantly, Alg. 1 can be run offline given a training dataset and desired bottleneck $Z$ to avoid passing large gradients over a wireless

\textsuperscript{2}Our technical report [20] and Edge TPU software are available at https://sites.google.com/view/tasknet

Fig. 2: Task-aware compression for linear systems. When the encoder, decoder, and task module $K$ are matrices, we achieve zero task loss by only sending a representation of size $\text{rank}(K)$ compared to a measurement of size $n$. Our co-design scheme (orange, blue) yields lower task loss for much smaller $z$ compared to task-agnostic methods (red). Further, we achieve better reconstruction loss and more stable training than an end-to-end method (black), which does not guide learning by fixing task module $K$.
link. Then, a robot can deploy the trained encoder and decoder, and periodically improve them with more field data.

**Algorithm 1: Task-Relevant Compression Co-design**

1. Set bottleneck dimension $Z$, reconstruction weight $\lambda$
2. Randomly initialize encoder/decoder params. $\theta_{\text{enc}}^0$, $\theta_{\text{dec}}^0$.
3. Clamp pre-trained model $f(\theta_{\text{task}})$ params. $\theta_{\text{task}}$
4. **for** $\tau \leftarrow 0$ **to** $T$ **do**
5. Sample $\{x, y\}$ from dataset $D$ with $y = f(x; \theta_{\text{task}})$
6. Encode $z \sim p(z \mid x; \theta_{\text{enc}}^\tau)$
7. Decode $\hat{x} \sim q(x \mid z; \theta_{\text{dec}}^\tau)$
8. Compute Predictions $\hat{y} = f(\hat{x}; \theta_{\text{task}})$
9. $\theta_{\text{enc}}^{\tau+1}, \theta_{\text{dec}}^{\tau+1} \leftarrow \text{BackProp}[L_{\text{weight}}(x, \hat{x}, \hat{y}; \theta_{\text{task}}, \theta_{\text{enc}}^\tau, \theta_{\text{dec}}^\tau)]$
10. **end**

**Result:** Return learned parameters $\theta_{\text{enc}}^T$, $\theta_{\text{dec}}^T$.

A. **Wide applicability of our Co-design Algorithm**

While simple, Alg. 1 is a powerful, general solution for task-relevant communication. A principal benefit is that we can either invoke a pre-trained task module fully remotely, or also flexibly adjust the split of computation between a resource-constrained robot and server. In the latter scenario, we can take a large, pre-trained DNN and only execute a fraction of layers locally on the robot to flexibly adjust for resource constraints. Then, we can insert a minimal, learned encoder $p$ to compress an intermediate result, transmit bottleneck $z$, and then decode with $q$ to continue computation with the bulk of the pre-trained model at a server. Importantly, only the distributed encoder and decoder are trained, unlike the fixed task module. Further, by adjusting reconstruction weight $\lambda$, which serves as a regularizer, we can significantly reduce reconstruction error with only a marginal gain in bottleneck size $z$. Our ability to flexibly set $\lambda$ leads to the following variants of Alg. 1.

1. **Task-Aware-Weighted (TASKNET):** Our core contribution is the weighted-task-aware training scheme, henceforth referred to as TASKNET. By accounting for the pre-trained task network $f(\theta_{\text{task}})$ when encoding sensory inputs, TASKNET achieves low task loss for much less transmitted data $z$. TASKNET’s reconstruction loss weight $\lambda$ (Eq. 1) can be flexibly set by a roboticist per application. All our experiments performed a scan over evenly-spaced values of $\lambda$ and only plotted one representative curve, in orange, for visual clarity.

2. **Fully Task-Aware:** To quantify the full compression benefits of optimizing only for an end-task, we consider the special case of TASKNET when $\lambda = 0$, which is suitable for scenarios when video inputs are automatically classified in real-time by a DNN and not intended for a human viewer. This scheme, always colored in blue, achieves the lowest task loss for the smallest representation size $Z$, which is ideal for scenarios with strict network bandwidth limits. We now evaluate both variants of Alg. 1 as well as benchmark schemes, on large-scale deep learning experiments.

IV. **Experimental Results**

To evaluate TASKNET, we first introduce two benchmark algorithms and then describe common evaluation metrics for all schemes. Then, we highlight the wide utility of TASKNET across various sensing modalities and task DNN modules, for tasks ranging from Martian terrain classification to motion planning and environmental timeseries classification. Further, we benchmark performance on the standard MNIST dataset.

A. **Evaluation Metrics and Benchmark Algorithms**

The principal objective of Prob. 1 is to minimize task loss, and optionally minimize weighted reconstruction loss, in scenarios where decoded inputs $\hat{x}$ need to be inspected. As such, we show that TASKNET achieves (A) low task loss (i.e. high task accuracy) and (B) flexibly achieves low reconstruction loss for small bottleneck sizes $z$. To assess the effects of compression on task loss, we quantify the task loss achieved by passing uncompressed, original sensory inputs $x$, without an information bottleneck, into the task network $f$. This metric, henceforth referred to as the *uncompressed input task loss*, represents the best task accuracy we can achieve without network constraints, and is pictured by the green dashed line in all task loss figures (e.g. Fig. 3). Further, all figures show results on each domain’s test dataset. We test the above metrics on the following two benchmarks that represent conventional paradigms for networked perception:

1. **Task-Agnostic:** For each bottleneck dimension $Z$, this scheme trains a standard variational autoencoder to minimize reconstruction loss, and simply passes the decoded input $\hat{x}$ through the pre-trained task module $f(\theta_{\text{task}})$. The task-agnostic scheme is always colored in red in all figures.

2. **End-to-End:** This scheme has the exact same encoder, decoder, and task module architectures as the others, but allows the task parameters $\theta_{\text{task}}$ to be updated. It takes longer to train and performs worse than TASKNET for small bottlenecks $z$ when the networks get stuck in local minima during training, such as in Fig. 3. In essence, it does not exploit the structure of task network $f$ for faster, more stable representation learning. We now evaluate all schemes.

B. **Digit Classification with an Information Bottleneck**

We first benchmark TASKNET on the standard MNIST [15] dataset, where task network $f(\theta_{\text{task}})$ was a publicly-available ResNet-50 DNN for digit classification [7]. Fig. 3a illustrates that TASKNET (orange) achieves the same lower-bound classification error as passing original, unperturbed images to the task DNN, even for images that are compressed with an extremely small bottleneck of size $Z = 3$. In contrast, the task-agnostic scheme (red) is still not able to achieve the lower bound classification error even for a much larger bottleneck of $Z = 32$, showing that TASKNET improves the compression ratio by $10 \times$ over a standard autoencoder with better task accuracy. Fig. 3b shows that TASKNET decodes images $\hat{x}$ with low reconstruction loss, which yields human-interpretable, but highly compressible, decoded samples that are visualized in our online report [20]. In the end-to-end scheme, the task network is also learned during training.
Fig. 3: MNIST test dataset results: TASKNET achieves the same lower-bound classification error as passing original images through the task network (green) for a small bottleneck $Z = 3$. In contrast, a task-agnostic method, in red, obtains worse classification error for even larger $Z = 32$ since it does not emphasize salient features for classification.

C. Compute-Efficient Mars Terrain Classification

To test TASKNET with complex vision DNNs, we consider a scenario where a low-power Mars rover encodes terrain images for remote classification at a powerful orbiting compute server, a scenario inspired by recent proposals [8]. We used Martian terrain images from NASA’s HiRise Dataset [5], consisting of 8 classes including dunes, craters, impact ejecta, and other formations. Task network $f(\theta_{\text{task}})$ is an EfficientNet-B0 DNN, which is the smallest of a family of DNNs that trade off accuracy and model complexity [26]. Our EfficientNet has 5.36 million parameters and classification accuracy of 95.7% on 7303 HiRise test images.

Given the relatively large size of the vision DNN, we want to flexibly split computation between a compute-limited robot and server. Notably, we can exploit a common architecture in many modern DNNs, which are composed of sequential blocks of several convolutional layers, to adjust the extent of on-robot compute. For example, EfficientNet block-4c consists of all parameters from the input to a specific layer $l$ and block-5c has all successive parameters until layer $l' > l$. As per Alg. 1, we first train the full, modular EfficientNet. Then, we constrain the robot to only run part of the large DNN until a specific block $b$, which yields an intermediate convolutional feature map, and then insert our minimal task encoder $p(\theta_{\text{enc.}})$ to compress the feature map to encoding $z$. Encoding $z$ is transmitted to a remote server, which uses our task decoder $q(\theta_{\text{dec}})$ to reconstruct the original feature map generated by block $b$ and resumes computation for a final classification. Importantly, only the encoder/decoder are learned with the fixed EfficientNet.

Our extended report [20] details several configurations where the robot only encodes inputs up to EfficientNet block $b$, but still achieves within 1.5% of the uncompressed input task loss. For example, Fig. 4 shows by computing until block-5c, TASKNET achieves 3.87 $\times$ lower classification error than conventional task-agnostic methods, with a bottleneck as small as $Z = 16$. This enables our scheme to send only $<0.34\%$ of data compared to the original image.

Robot compute savings: Notably, by only devoting computational effort to encode task-relevant information, TASKNET reduces robot computation to only 8.5% of overall compute, compared to the 91.5% delegated to a remote server (Fig. 5a). As such, TASKNET is significantly more

Fig. 4: Terrain classification. TASKNET achieves much lower classification error than task-agnostic methods (red), where even a bottleneck dimension of $z = 128$ corresponds to sending only 0.34% of data compared to original images.

(a) Martian Terrain Classification

Fig. 5: (a) By only encoding task-relevant information, TASKNET reduces on-robot computation compared to what is delegated to a remote server (purple). (b) TASKNET generalizes better than the original MPNET on out-of-domain Mars terrain data to find more feasible paths (y-axis).

D. Robotic Motion Planning with Neural Networks

To demonstrate TASKNET’s utility for motion planning, we consider a scenario where a low-power drone takes an aerial terrain image and sends a compressed representation to a remote server, which generates a collision-free motion plan for a swarm of ground robots. Our task network $f(\theta_{\text{task}})$ is a Motion Planning Network (MPNET), a recently published planner [24] that generates collision-free paths an order of magnitude faster than conventional sampling-based planners such as RRT* [10], but with virtually the same path cost. MPNET consists of an obstacle encoder network, such as a contractive autoencoder, which maps an obstacle point-cloud to an embedding of dimension up to $Z = 28$. Then, MPNET’s planning network maps the obstacle embedding $z$ as well as current and goal robot configurations to predict the next navigation waypoint. The original MPNET DNN planners for two and three dimensional environments are up to 41 MB and 69 MB in size. While of moderate size compared to large perception DNNs, they are too large to run on low-power accelerators like the Edge TPU, which runs only compressed DNNs that fit in 8MB of on-chip memory.

In our experiments, the MPNET encoder runs on the robot, but the pre-trained planning network runs on a remote server to potentially plan paths for a robotic swarm. Crucially, rather than transmitting a large obstacle embedding $z_0$ as in the
original MPNet, we insert a small encoder at the robot which generates a minimal representation $z \ll z_0$, which is sent across a wireless link, decoded to $z_0$, and passed into the planning network to generate a motion plan. As described in Alg. 1, we only learn the encoder and decoder to generate compressed environment encoding $z$ and use the original MPNet planning network.

Fig. 5b shows the percentage of collision-free, feasible paths (y-axis) generated by our TASKNet co-designed planner and the original MPNet on the same 110 test environments and 40K test paths provided by the MPNet authors. In all cases, if a feasible, collision-free path is found, its path length is within 1% of the path generated by the RRT* planner. Our key result is that TASKNet achieves collision-free paths with costs and failure rate within 1% of the original MPNet, but with much smaller encoder encodings $z$. For example, TASKNet achieves good performance for $z = 4$ as opposed to $z = 28$ for the original MPNet, leading to a compression gain of $7 \times$.

Generalization of motion planner to unseen domains: We further tested the TASKNet motion planner on obstacle point clouds from the Mars HiRise dataset, where the red point clouds in Fig. 6 represent regions above 30 degrees of elevation. Even though MPNet and TASKNet’s train and test environments had point clouds representing polygonal obstacles, TASKNet adapts to curved contours of never-before-seen Mars terrain with small bottlenecks $z$.

Fig. 5b validates the above observation on over 40.5K paths, where both MPNet and TASKNet performed similarly on the first two scenarios of seen and unseen environments in MPNet’s original test data. Specifically, a seen test environment was observed by the planner during training, albeit with different start and end test configurations [24]. Our key experimental result is shown in the third column of Fig. 5b where TASKNet significantly outperforms the original MPNet for small bottlenecks $z = 4,8$ on out-of-domain Mars terrain data. We hypothesize that TASKNet generalizes better for small $z$ since it focuses the limited information capacity of the bottleneck to represent salient features for planning, and sacrifices reconstruction loss, which is irrelevant to the task goal. Since we only had access to a large, out-of-domain dataset for the Mars planning example, we plan to further stress test TASKNet’s generalization capabilities in future work. We plot all benchmarks’ performance in our online report [20], since the trends match Figs. 2, 3, and 7.

E. Environmental sensor timeseries anomaly detection

To highlight how TASKNet generalizes to multiple sensor modalities, we emulated the scenario of a micro-robot, potentially part of a swarm, that streams environmental sensory data to a remote server for anomaly detection. We used an environmental sensor that connects to the Edge TPU to measure light, temperature, pressure, and humidity timeseries, which are shown in our online report [20]. The task network $f(\cdot; \theta_{task})$ is a compact neural network that classifies each timeseries window of length $w$, $x^{t:w}$, into three sensor conditions $y^t$. The first class $y^t = 0$ represents if the sensor is being tightly clamped and tampered with, which leads to a rapid fluctuation in the light and humidity measurements. The second class $y^t = 1$ represents when the sensor reads a temperature spike, for which we generate training data by placing a hot hair-dryer over the sensor. The third class $y^t = 2$ represents natural environmental variation without anomalies. We collected two weeks of sensor data for a total of 30 training and 30 test traces of 5 minutes each, equally balanced across all three classes. The task network $f(\cdot; \theta_{task})$, which was trained on stochastic, diverse timeseries, achieved a promising 90% test accuracy. Fig. 7 shows that TASKNet aggressively compresses the timeseries measurements and outperforms the task-agnostic benchmark. Overall, our diverse results show the promise of provisioning publicly-available task modules at a central server and co-designing minimal representations for networked perception.

Limitations of our work: Our current method requires a constraint on the maximum bottleneck $Z^*$, based on wireless network capacity, and performs a search over increasing sizes $z$. A promising future direction is to automatically determine the optimal representation dimension $Z^*$ that optimizes weighted loss. Further, we could extend our experiments to compress LIDAR point clouds and long segments of video.

V. DISCUSSION AND CONCLUSIONS

This paper presents a novel framework to aggressively compress rich robotic sensory data for the ultimate needs of a machine sensing task, which allows robots to reduce on-board computation and communication bandwidth by up to 11 ×. We envision that our co-design algorithm can be applied to a variety of resource-constrained robots that need to distill sensory data for remote inference, such as future Mars Rovers, microrobots, and even low-power drones that collaborate by communicating over 5G wireless networks.

In future work, we plan to extend our algorithm to create minimal, task-relevant representations for cooperative control.
and develop representations that generalize across several tasks, leveraging ideas from multi-task and meta-learning. While our approach reduces on-board computation, we plan to also quantify the power consumption of communicating minimal representations. Finally, given our promising results with EfficientNets and deep learning accelerators, we plan to deploy our algorithm on a networked robotic autonomy stack which supports remote inference and tele-operation.

REFERENCES

[1] Edge tpu.  https://cloud.google.com/edge-tpu/  2019. [Online; accessed 01-Sep-2019].

[2] Mars reconnaissance orbiter: Communications with earth.  https://mars.nasa.gov/mro/communications/  2020. [Online; accessed 18-Oct-2020].

[3] S. Chinchali*, E. Cidon*, E. Pergament*, T. Chu, and S. Katti. Neural networks meet physical networks: Distributed inference between edge devices and the cloud. In Proceedings of the 17th ACM Workshop on Hot Topics in Networks, HotNets ’18, pages 50–56, 2018. * Indicates equal contribution.

[4] S. Chinchali, A. Sharma, J. Harrison, A. Elhafsi, D. Kang, E. Pergament, E. Cidon, S. Katti, and M. Pavone. Network offloading policies for cloud robotics: a learning-based approach. arXiv preprint arXiv:1902.05703, 2019.

[5] G. Doran, S. Lu, L. Mandrake, and K. Wagstaff. Mars orbital image (HiRISE) labeled data set version 3, Jan. 2019.

[6] J. Emmons, S. Fouladi, G. Ananthanarayanan, S. Venkataraman, G. Doran, S. Lu, L. Mandrake, and K. Wagstaff. Mars orbital image (HiRISE) labeled data set version 3, Jan. 2019.

[7] L. Engstrom, A. Ilyas, H. Salman, S. Santurkar, and D. Tsipras. Robustness (python library), 2019.

[8] K. I. for Space Studies. Virtual workshop: Nebulae - deep-space computing clouds - part 2, 2020.

[9] M. Ghifary, W. Bastiaan Kleijn, M. Zhang, and D. Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In Proceedings of the IEEE international conference on computer vision, pages 2551–2559, 2015.

[10] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller. Anytime motion planning using the rrt. In 2011 IEEE International Conference on Robotics and Automation, pages 1478–1483. IEEE, 2011.

[11] B. Kehoe, A. Matsukawa, S. Candido, K. Goldberg. Cloud-based robot grasping with the google object recognition engine. In Robotics and Automation (ICRA), 2013 IEEE International Conference on, pages 4263–4270. IEEE, 2013.

[12] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg. A survey of research on cloud robotics and automation. IEEE Trans. Automation Science and Engineering, 12(2):398–409, 2015.

[13] D. P. Kingma and M. Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

[14] I. Kuffner. Cloud-enabled robots in: Ieee-ras international conference on humanoid robots. Piscataway, NJ: IEEE, 2010.

[15] Y. LeCun. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

[16] P. Li, B. DeRose, J. Mahler, J. A. Ojea, A. K. Tanwani, and K. Goldberg. Dnx-net as a service (dnas): A cloud-based robust robot grasp planning system. In 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE), pages 1420–1427. IEEE, 2018.

[17] Z. Liu, T. Liu, W. Wen, L. Jiang, J. Xu, Y. Wang, and G. Quan. Deepnjpg: A deep neural network favorable jpg-based image compression framework. In Proceedings of the 55th Annual Design Automation Conference, pages 1–6, 2018.

[18] Z. Manchester, M. Peck, and A. Filo. Kicksat: A crowd-funded mission to demonstrate the world’s smallest spacecraft, 2013.

[19] G. Mohanarajah, V. Usenko, M. Singh, R. D’Andrea, and M. Waibel. Cloud-based collaborative 3d mapping in real-time with low-cost robots. IEEE Transactions on Automation Science and Engineering, 2015.

[20] M. Nakano, Y. Chinchali, A. Anemogiannis, A. Datta, S. Katti, and M. Pavone. Task-relevant representation learning for networked robotic perception. Technical report: Task-relevant representation learning for networked robotic perception. https://sites.google.com/view/tasknet 2020.

[21] F. Nenci, L. Spinello, and C. Stachniss. Effective compression of range data streams for remote robot operations using h. 264. In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3794–3799. IEEE, 2014.

[22] V. Pacelli and A. Majumdar. Task-driven estimation and control via information bottlenecks. In 2019 International Conference on Robotics and Automation (ICRA), pages 2061–2067. IEEE, 2019.

[23] V. Pacelli and A. Majumdar. Learning task-driven control policies via information bottlenecks. arXiv preprint arXiv:2002.01428, 2020.

[24] A. H. Qureshi, A. Simeonov, M. J. Benzy, and M. C. Yip. Motion planning networks. In 2019 International Conference on Robotics and Automation (ICRA), pages 2118–2124. IEEE, 2019.

[25] A. Sonar, V. Pacelli, and A. Majumdar. Invariant policy optimization: Towards stronger generalization in reinforcement learning. arXiv preprint arXiv:1905.11946, 2019.

[26] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946, 2019.

[27] A. K. Tanwani, R. Anand, J. E. Gonzalez, and K. Goldberg. RilaaS: Robot inference and learning as a service. IEEE Robotics and Automation Letters, 2020.

[28] N. Tishby and N. Zaslavsky. Deep learning and the information bottleneck principle. In 2015 IEEE Information Theory Workshop (ITW), pages 1–5, IEEE, 2015.

[29] C. Ts, E. Takeuchi, A. Carballo, and K. Takeda. Point cloud compression for 3d lidar sensor using recurrent neural network with residual blocks. In 2019 International Conference on Robotics and Automation (ICRA), pages 3274–3280. IEEE, 2019.

[30] M. Weber, C. Renggli, H. Grabner, and C. Zhang. Lossy image compression with recurrent neural networks: from human perceived visual quality to classification accuracy. arXiv preprint arXiv:1910.03472, 2019.
APPENDIX

In this section, we provide further experimental results to illustrate the benefits of co-designing sensory representations with a pre-trained task module’s objective using our TASKNET algorithm. Specifically, we elaborate on results for:

1) linear task-aware compression (Sec. A)
2) martian terrain classification with EfficientNets (Sec. B)
3) neural motion planning (Sec. C)
4) anomaly detection of environmental sensor timeseries (Sec. D), and
5) decoding human-interpretable reconstructed sensory inputs (Sec. E).

A. Further analysis of Linear Task-aware compression

Consider the linear setting described in Sec. II-C where the encoder is a matrix $A \in \mathbb{R}^{Z \times n}$ and the decoder is also a matrix $B \in \mathbb{R}^{n \times Z}$. Recall that the original sensory input is given by $x \in \mathbb{R}^{n}$ and reconstructed sensory input is given by $\hat{x} = B A x$, while the task output is denoted by $y = K x$.

We now consider the weighted loss function with $\lambda > 0$, where

$$L_{\text{weight}} = \| (K x - K B A x) \|_2^2 + \lambda \| (x - B A x) \|_2^2.$$  

In the linear setting, solving Problem 1 for a fixed task matrix $K$ amounts to setting the gradient of the weighted task loss with respect to the encoder and decoder matrices to zero. Thus, the optimal encoder/decoder matrices that minimize weighted task loss must satisfy:

$$\nabla_A L_{\text{weight}} = -2 B^T K^T (K x - K B A x) x^T - 2 \lambda B^T (x - B A x) x^T$$

$$\nabla_B L_{\text{weight}} = -2 K (K x - K B A x) x^T A^T - 2 \lambda (x - B A x) x^T A^T$$

If we run our co-design algorithm Alg. 1 and use the analytical gradients computed above in the backpropagation step, we arrive at the results in Figs. 2 and 3. Specifically, we see the task-aware weighted scheme achieves the upper bound task accuracy with low reconstruction loss for $Z \geq \text{rank}(K)$. Further, the relative merits of our weighted task-aware scheme mirror the trends shown in the large-scale deep learning experiments.

B. Further Description of Mars Terrain Classification with EfficientNet

In Sec. IV-C we showed how to flexibly allocate compute resources between a resource-constrained robot and server for terrain classification with EfficientNets. Fig. 9 illustrates the training data from the HiRise dataset [5]. Then, Fig. 10 illustrates how we can flexibly allocate compute between a small robot and server with the TASKNET architecture. However, we emphasize that our design also allows the full, modular EfficientNet task network to be run remotely, as depicted in the complementary configuration in Fig. 11.

Finally, Fig. 12 illustrates the resultant classification task accuracy for various configurations of computation allocated between a robot and server.

C. Generalization of Neural Motion Planning to out-of-domain Mars Data

In Section IV-D and Figs. 5b and 6, we showed that TASKNET generalizes to never-before-seen Mars terrain data, where obstacle point clouds have significantly different shapes than those of the MPN training and test data. Further examples of the TASKNET co-designed motion planner are provided in Fig. 13.

D. Further Description of Environmental Sensor Experiments

Fig. 14 shows example timeseries that we collected from an environmental sensor board connected to the Edge TPU DNN accelerator. Our comprehensive dataset spans two weeks of diverse, stochastic timeseries described in Sec. IV-C.

E. Effects of reconstruction loss on decoded images

In Eq. 1 we allow a roboticist to optimize for task loss, but also optionally for reconstruction loss, in scenarios where decoded inputs $\hat{x}$ need to be sanity-checked or viewed...
Fig. 10: **Flexible allocations of robot and server compute:** An example configuration for flexibly allocating compute between a robot and server, used for the EfficientNet Mars classification examples in Sec. IV-C. Specifically, we can take the *pre-trained* EfficientNet and insert a learned, task-relevant encoder to map an intermediate feature map to a bottleneck representation $z$. Once $z$ is transmitted over a wireless link, the learned decoder uncompresses the intermediate feature map so that computation can continue at a remote server with the rest of the pre-trained EfficientNet. Our flexible co-design algorithm also allows the modular EfficientNet to fully run at a remote server.

remotely to debug. For the MNIST experiments in Sec. IV-B, we show how introducing a reconstruction loss weight $\lambda > 0$ yields highly-compressible, but human-interpretable results in Figs. 15 and 17. We contrast these with the case where we optimize solely for a machine’s task loss in Figs. 16 and 18 with $\lambda = 0$.

Fig. 11: **Remote perception configuration:** This figure illustrates the conventional configuration of TASKNET, where the robot’s learned encoder maps sensory input $x$ to compressed representation $z$. After $z$ is transmitted over a network link, it is decoded to $\hat{x}$ at a server and directly passed to a pre-trained, modular task network $f(\cdot; \theta_{\text{task}})$ that fully runs at a remote server. This configuration corresponds to Fig. 1, which complements the alternative architecture in Fig. 10 where TASKNET flexibly allocates computation that is split between a robot and server.

Fig. 12: **Mars terrain classification:** This plot illustrates our ability to flexibly allocate compute between a resource-constrained robot and cloud server. Specifically, as described in Sec. IV-C we can run layers until EfficientNet block $b$, co-design task relevant encoder/decoder to transmit intermediate computations, and finally continue computation with the rest of the *pre-trained* EfficientNet at a server. We plot TASKNET with $\lambda = 0$ for scenarios where the robot only runs up to EfficientNet block $b$ locally and achieve within 1% of the uncompressed input error rate (green dashed line) at block 4c and beyond for $Z = 128$. 
Fig. 13: **Motion planning on Mars Terrain Data**: A further example of how TASKNET can plan feasible, collision-free motion plans on out-of-domain Mars data, as illustrated in Fig. 5b and 6.

Fig. 14: (Left) Environmental sensor on the Google Edge Tensor Processing Unit (TPU) and example stochastic time-series measurements (right). Our experiments in Sec. IV illustrate how TASKNET can compress these sensory measurements for anomaly detection using a neural network task module.

Fig. 15: **MNIST with reconstruction loss**: Our general co-design framework of TASKNET allows highly-compressible, but also human-interpretable decoded images. Specifically, when we optimize for classification task loss and weighted reconstruction loss, with weight $\lambda = 0.01$ (Eq. 1), decoded images $\hat{x}$ (bottom row) are human-interpretable and resemble original images $x$ (top row). However, the key benefit of TASKNET is that we have a heavily-compressed bottleneck representation of $Z = 4$, which is much smaller than task-agnostic representations, as shown in Fig. 3. We note that the decoded samples emphasize salient features for machine classification and do not produce the sharpest reconstructions since reconstruction is not the primary objective. However, these blurry decoded samples $\hat{x}$ are sufficient to achieve high machine classification accuracy.

Fig. 16: **MNIST classification without reconstruction loss**: Our general co-design framework also allows us to create highly-compressible representations that are purely optimized for a machine’s task loss with reconstruction weight $\lambda = 0.0$ (Eq. 1). Though we have a heavily-compressed representation with a size of $Z = 4$, decoded images $\hat{x}$ (bottom row) are not human-interpretable and don’t resemble original images $x$ (top row), unlike the scenario of Fig. 15. This motivates our addition of a regularization term with reconstruction weight $\lambda > 0$ to develop representations that are highly-compressible, but yield interpretable decoded samples $\hat{x}$ as in Fig. 15.

Fig. 17: **MNIST with reconstruction loss**: A further example of how our general co-design framework of TASKNET allows highly-compressible, but also human-interpretable decoded images. This example mirrors Fig. 15 but for another set of data.

Fig. 18: **MNIST without reconstruction loss**: A further example of how optimizing for purely a machine’s task loss, without reconstruction weight $\lambda$, yields highly-compressible, but uninterpretable decoded images $\hat{x}$. This example mirrors Fig. 16 but for another set of data.