Robotics: Science and Systems 2021
Held Virtually, July 12–16, 2021

Co-Design of Communication and Machine Inference for Cloud Robotics

Manabu Nakanoya*, Sandeep Chinchali†‡, Alexandros Anemogiannis, Akul Datta, Sachin Katti‡, Marco Pavone§
NEC Corporation*, Kanagawa, Japan
Departments of Computer Science‡ and Aeronautics and Astronautics§, Stanford University, Stanford, CA
Department of Electrical and Computer Engineering†, The University of Texas at Austin, Austin, TX
E-mail: nakanoya@nec.com, {csandeep,skatti,pavone}@stanford.edu

***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 × 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 [25], while the deep-space network only has a communication bandwidth of 0.5-4 Megabits/second [2]. [1]

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 [4, 9]. 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. We note that several recent works have used specialized deep neural networks (DNNs) to improve LIDAR or JPEG image compression [33, 22, 35], motivated by the observation that standard compression schemes do not emphasize features that are most sensitive for accurate *machine* perception [9, 4]. However, the key novelty of our work is to use a general-purpose, pre-trained task module to *guide* representation learning, which allows us to easily generalize to multiple sensor modalities unlike prior specialized point-solutions.

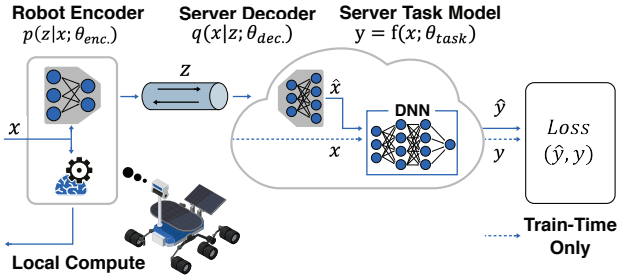


Fig. 1: **Task-Relevant Communication for Perception**: A compute-limited robot *learns* how to compress sensory input x, transmit salient features z, and decode the input $\hat{x}$ so that it can directly leverage a pre-trained, potentially "off-the-shelf" task module $f(;\theta_{\text{task}})$ at a central server. By learning an encoder and decoder (gray) within the context of a pre-trained task module's goal, we only transmit minimal, salient information. In the figure, only components in gray (the encoder and decoder) are learned, while the task network f has pre-trained, fixed parameters. Original sensory input x and task output y (dashed lines) are only used for training.

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. Indeed, our work is partly inspired by recent proposals to place powerful servers in deep space to assist future rovers and space telescopes with compute-intensive machine learning tasks, data processing, and storage before relaying selected data back to earth [11, 34].

[1] NASA estimate for Earth to Mars Reconnaissance Orbiter link [2].

Literature Review: Recent work has applied information bottleneck theory [32] to build controllers that focus on actionable, task-relevant visual inputs for robust, generalizable navigation and grasping policies [27, 26, 29]. 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 [22, 35] 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 [17, 19], where robots use remote servers to augment their grasping [21], object recognition [16], and mapping capabilities [24]. Prior work balances cloud computing accuracy with communication delay by learning when to query the cloud [5, 31]. Rather than address *when* to communicate, we instead address how to represent data for task-centric communication. Our work is also related to variational and multi-task autoencoders [18, 12], 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 $11\times$ more than standard autoencoders.

Finally, our work is complementary to methods that compress large DNNs to run on compute-limited robots, using methods like weight pruning or knowledge distillation [3]. Despite such advances, state-of-the-art DNNs, such as transformers for NLP (e.g., BERT [7]) or vision are still infeasible to run on low-power, compute-limited robots. Thus, our work addresses a viable alternative to transmit salient features for remote, compute-intensive machine learning tasks.

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 $\mathbf{x} = x^{t-w:t}$.

Robot Encoder: The encoder maps raw input x to a concise representation $z \in \mathbb{R}^Z$, denoted by $z = p(z|x;\theta_{\text{enc.}})$, where $\theta_{\text{enc.}}$ 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 $\hat{x} = q(x|z;\theta_{\text{dec.}})$. The parameters of a decoder model, such as a DNN, are denoted by $\theta_{\text{dec.}}$.

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 $\hat{y} = f(\hat{x};\theta_{\text{task}})$ using model parameters θ_{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 $\mathcal{D}$, 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* $\mathcal{L}_{\text{task}}(y,\hat{y};\theta_{\text{task}})$, which compares the resultant task outputs $y = f(x;\theta_{\text{task}})$ using original input x and predicted outputs $\hat{y} = f(\hat{x};\theta_{\text{task}})$ 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 $f(;\theta_{\text{task}})$ 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 $\mathcal{L}_{\text{recon.}}(x,\hat{x};\theta_{\text{enc.}},\theta_{\text{dec.}})$ incentivizes faithful reconstruction of a scene, which, in practice, could be the standard variational autoencoder regularized loss [18]. 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 $\lambda \geq 0$:

$$\mathcal{L}_{\text{weight}}(x,\hat{x},y,\hat{y};\theta_{\text{task}},\theta_{\text{enc.}},\theta_{\text{dec.}}) = \mathcal{L}_{\text{task}}(y,\hat{y};\theta_{\text{task}}) + \lambda\mathcal{L}_{\text{recon.}}(x,\hat{x};\theta_{\text{enc.}},\theta_{\text{dec.}}). \quad (1)$$

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 $f(;\theta_{\text{task}})$ with fixed, pre-trained parameters θ_{task}, fixed bottleneck dimension Z, and reconstruction loss weight $\lambda \geq 0$, find robot encoder and server decoder parameters $\theta_{\text{enc.}}$ and $\theta_{\text{dec.}}$ that minimize weighted loss (Eq. 1) over data distribution $\mathcal{D}$:*

$$\theta^*_{\text{enc.}}, \theta^*_{\text{dec.}} = \underset{\theta_{\text{enc.}},\theta_{\text{dec.}}}{\operatorname{argmin}} \underset{(x,y)\sim\mathcal{D}}{\mathbb{E}} \mathcal{L}_{\text{weight}}(x, \hat{x}, y, \hat{y}; \theta_{\text{task}}, \theta_{\text{enc.}}, \theta_{\text{dec.}}),$$

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

Prob. 1 is widely applicable to resource-limited robots, such as space and mining rovers, that can 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 $f(;\theta_{\text{task}})$, to guide task-relevant representation learning of encoder/decoder parameters $\theta_{\text{enc.}}, \theta_{\text{dec.}}$. Our formulation is related to multi-task learning for vision, which often considers semantically-similar tasks, such as segmentation and object detection [6]. In contrast, however, our weighted objective has two semantically-different (and often competing) goals of learning a sparse set of features for machine perception while also regularizing for pixel-wise reconstruction for human-interpretability. Further, we enable a roboticist to set bottleneck size Z in a 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 $A = \theta_{\text{enc.}} \in \mathbb{R}^{Z\times n}$ generates encoding $z = p(x;\theta_{\text{enc.}}) = Ax$. 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 $y \in \mathbb{R}^m$. The task loss penalizes error in the linear estimate $y = Kx$, 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 the task and reconstruction loss are quadratic, yielding weighted loss:

$$\mathcal{L}_{\text{weight}} = \underbrace{\|(Kx - KBAx)\|_2^2}_{\text{task loss}} + \lambda \underbrace{\|(x - BAx)\|_2^2}_{\text{recon.loss}}. \quad (2)$$

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 loss 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) $K = U\Sigma V^\top$, where $U \in \mathbb{R}^{m\times r}$ and $V^\top \in \mathbb{R}^{r\times n}$ are semi-unitary. Then, for bottleneck dimension r, setting $A = V^\top$ and $B = V$ solves Prob. 1 with zero task loss $\|(Kx - KBAx)\|_2^2$ (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 $x \in \mathbb{R}^n$, $y = Kx = \hat{y} = KBAx$. This is achieved when $B = V$ and $A = V^\top$:

$$\begin{aligned}\hat{y} = KBAx &= \underbrace{(U\Sigma V^\top)}_{SVD(K)} \times \underbrace{V}_{B} \times \underbrace{V^\top}_{A} \times x \\ &= U\Sigma V^\top x = Kx = y,\end{aligned}$$

where $V^\top V = I_{r\times r}$ 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 $\tilde{A} \in \mathbb{R}^{Z\times n}, \tilde{B} \in \mathbb{R}^{n\times Z}$ with $Z < r$ that achieve zero task loss. Then:

$$\begin{aligned}\text{rank}(\tilde{B}\tilde{A}) &\leq \min(\text{rank}(\tilde{A}), \text{rank}(\tilde{B})) \leq Z, \text{and} \\ \text{rank}(K\tilde{B}\tilde{A}) &\leq \min(\text{rank}(K), \text{rank}(\tilde{B}\tilde{A})) \leq Z.\end{aligned}$$

However, in order for $\tilde{A}, \tilde{B}$ to achieve zero task loss, we must have $Kx = K\tilde{B}\tilde{A}x$ 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 $\frac{n}{\text{rank}(K)}$ compared to sending full input x. However, we do not necessarily have zero *reconstruction loss* since matrix $V^\top$ is only semi-unitary, meaning it is not required for $VV^\top = BA$ 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 naïvely just computing $y = Kx$ directly at the robot, since we only transmit a small representation $z = V^\top x$ and use the modular task matrix K at the server. Finally, we note that the choice of $A = V^\top$ and $B = V$ is not unique for bottleneck r. For example, we can scale the encoder to aA and decoder to bB where $ab = 1$.

D. Illustrative Example: Weighted Linear Setting

We now generalize our previous result for the *weighted* linear setting with $\lambda > 0$. The key highlight of our solution is that the optimal linear encoder/decoder to minimize weighted task loss are simply a solution to a low-rank approximation problem. To establish this result, we first concatenate N examples drawn from training dataset $(x, y) \sim \mathcal{D}$ into a sample matrix $\mathbf{X} \in \mathbb{R}^{n\times N}$. Given the dataset $\mathbf{X}$ and a

bottleneck constraint Z, minimizing the weighted task loss (Eq. 2) in the linear setting amounts to solving:

$$\underset{A,B}{\operatorname{argmin}} \sum_{i=1}^{N} (\hat{\mathbf{X}}_i - \mathbf{X}_i)^\top (K^\top K + \lambda I)(\hat{\mathbf{X}}_i - \mathbf{X}_i) \quad \text{where}$$
$$\hat{\mathbf{X}} = BA\mathbf{X}, \ \operatorname{rank}(B) \leq Z \text{ and } \operatorname{rank}(A) \leq Z. \tag{3}$$

In the above problem, the i-th column of $\mathbf{X}$ and $\hat{\mathbf{X}}$ are given by $\mathbf{X}_i$ and $\hat{\mathbf{X}}_i$ respectively. The following theorem solves Problem 1 for the weighted linear setting (Eq. 2) by reducing it to a canonical low-rank approximation problem.

Theorem 2 (Linear Weighted Compression)**.** *The linear setting of Problem 1 is a canonical low-rank approximation problem with an analytical solution for an optimal encoder matrix A and decoder matrix B.*

Proof. (Sketch) We first note that the real matrix $K^\top K + \lambda I$ is positive semi-definite, which admits an eigen-decomposition denoted by $Y \Lambda Y^\top$, where Y is orthogonal. Denoting the Frobenius norm of a matrix as $||\cdot||_F$, we can write the weighted task loss for Problem 1 in the linear setting as:

$$\begin{aligned}
&\sum_{i=1}^{N} (\hat{\mathbf{X}}_i - \mathbf{X}_i)^\top (K^\top K + \lambda I)(\hat{\mathbf{X}}_i - \mathbf{X}_i) \\
=&\sum_{i=1}^{N} (\hat{\mathbf{X}}_i - \mathbf{X}_i)^\top (Y \Lambda Y^\top)(\hat{\mathbf{X}}_i - \mathbf{X}_i) \\
=&||\Lambda^{\frac{1}{2}} Y^\top \hat{\mathbf{X}} - \Lambda^{\frac{1}{2}} Y^\top \mathbf{X}||_F^2.
\end{aligned}$$

Thus, the solution to Problem 1 with a bottleneck Z in the linear setting can be written in the following low-rank approximation form:

$$\underset{A,B}{\operatorname{argmin}} || \underbrace{\Lambda^{\frac{1}{2}} Y^\top BA\mathbf{X}}_{\text{approximation}} - \underbrace{\Lambda^{\frac{1}{2}} Y^\top \mathbf{X}}_{\text{original}} ||_F^2 \quad \text{where}$$
$$\operatorname{rank}(B) \leq Z \text{ and } \operatorname{rank}(A) \leq Z. \tag{4}$$

Thus, as stipulated by the Eckhart-Young theorem, the solution to Eq. 4 is the rank Z truncated singular value decomposition (SVD) of matrix $\Lambda^{\frac{1}{2}} Y^\top \mathbf{X}$, denoted by $U \Sigma V^\top$. For the SVD, $U \in \mathbb{R}^{n \times Z}$ is semi-orthogonal, $V \in \mathbb{R}^{N \times Z}$ is semi-orthogonal, and $\Sigma \in \mathbb{R}^{Z \times Z}$ represents a diagonal matrix of singular values. Thus, an encoder of $A = U^\top \Lambda^{\frac{1}{2}} Y^\top$ and decoder of $B = (\Lambda^{\frac{1}{2}} Y^\top)^{-1} U$ are solutions since:

$$\begin{aligned}
\underbrace{\Lambda^{\frac{1}{2}} Y^\top BA\mathbf{X}}_{\text{approximation}} &= \Lambda^{\frac{1}{2}} Y^\top \underbrace{(\Lambda^{\frac{1}{2}} Y^\top)^{-1} U}_{B} \underbrace{U^\top \Lambda^{\frac{1}{2}} Y^\top}_{A} \mathbf{X} \\
&= U(U^\top \Lambda^{\frac{1}{2}} Y^\top \mathbf{X}) = \underbrace{U \Sigma V^\top}_{\text{optimal rank Z approximation}}.
\end{aligned}$$

Weighted Linear Results: For the weighted $\lambda > 0$ case, we solve Prob. 1 using the SVD as in Theorem 2. 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. Our toy example shows that, even for linear systems, we achieve large compression gains by co-designing with a task objective, which mirror our subsequent results with complex DNNs. While we note our theory only applies to linear systems, such a theoretical underpinning is informative to understand our results for complex DNNs.

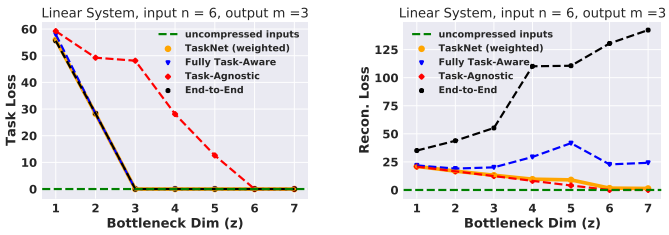


Fig. 2: **Task-aware compression for linear systems.** When the encoder, decoder, and task module K are matrices, we achieve zero task loss by only sending a representation of size $\operatorname{rank}(K)$ compared to a measurement of size n. Our co-design scheme (orange, blue) yields lower task loss 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.

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. IV.

Alg. 1 takes as input a bottleneck dimension Z, set based on communication data-rate limits, and user-desired reconstruction weight λ, on line 1. Then, it randomly initializes the encoder/decoder parameters on line 2, which are learned during training *unlike* task parameters θ_{task} (line 3). The main loop repeats for T learning rounds, where we sample inputs from labeled training dataset $\mathbf{D} = \{x^i, y^i\}$ where each $(x^i, y^i) \sim \mathcal{D}$ (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 $\mathcal{L}_{\text{weight}}$ (Eq. 1) to update parameters $\theta_{\text{enc.}}, \theta_{\text{dec.}}$, while keeping the task parameters θ_{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 link. Then, a robot can deploy the trained encoder and decoder, and periodically improve them with more field data.

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, depicted in Fig. 5a, 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

Algorithm 1: Task-Relevant Compression Co-design

Set bottleneck dimension Z, reconstruction weight λ
Randomly initialize encoder/decoder params. $\theta^0_{\text{enc.}}$, $\theta^0_{\text{dec.}}$
Clamp pre-trained model $f(;\theta_{\text{task}})$ params. θ_{task}
for $\tau \leftarrow 0$ **to** T **do**
  Sample $\{x, y\}$ from dataset $\mathbf{D}$ with $y = f(x; \theta_{\text{task}})$
  Encode $z = p(z \mid x; \theta^\tau_{\text{enc.}})$
  Decode $\hat{x} = q(x \mid z; \theta^\tau_{\text{dec.}})$
  Compute Predictions $\hat{y} = f(\hat{x}; \theta_{\text{task}})$
  $\theta^{\tau+1}_{\text{enc.}}, \theta^{\tau+1}_{\text{dec.}} \leftarrow$
  BACKPROP$\big[\mathcal{L}_{\text{weight}}(x, \hat{x}, y, \hat{y};\ \theta_{\text{task}}, \theta^\tau_{\text{enc.}}, \theta^\tau_{\text{dec.}})\big]$
end
Result: Return learned parameters $\theta^T_{\text{enc.}}, \theta^T_{\text{dec.}}$

λ, 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 λ 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. We emphasize that TASKNET refers to our co-design training scheme and is compatible with any domain-specific, pre-trained task module f. TASKNET's reconstruction loss weight λ (Eq. 1) can be flexibly set by a roboticist per application. All our experiments performed a scan over evenly-spaced values of λ 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. To recreate our work, we provide all software, data, and pre-trained DNNs for the Edge Tensor Processing Unit (TPU) and servers at https://sites.google.com/view/tasknet.

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 θ_{task} to be updated. In principle, an end-to-end approach can achieve the same compression and accuracy as our task-aware co-design approach since all parameters can be optimized for the ultimate weighted task loss. However, in practice, all end-to-end schemes took longer to train and performed worse than TASKNET for small bottlenecks z when the networks got stuck in local minima during training, such as in Fig. 3 for $Z = 2$ and $Z = 4$. In essence, an end-to-end approach 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 [20] dataset, where task network $f(;\theta_{\text{task}})$ was a publicly-available ResNet DNN for digit classification [10]. 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 Fig. 4. While we did a sweep over reconstruction weights λ, we plot results for $\lambda = 0.01$, which was chosen to reflect similar emphasis on compressed accuracy and reconstruction loss. Further, our learned encoder p and decoder q were standard 3-layer convolutional variational autoencoders that added a small overhead of only $< 7\%$ extra compute

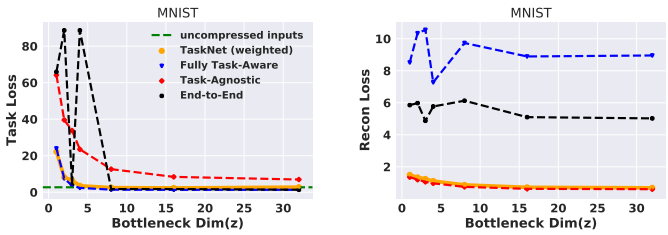


Fig. 3: **MNIST test dataset results:** TASKNET achieves the same lower-bound classification error as passing *original* images through the task network (green) for a small bottleneck $Z = 3$. In contrast, a task-agnostic method, in red, obtains worse classification error for even larger $Z = 32$ since it does not emphasize salient features for classification.

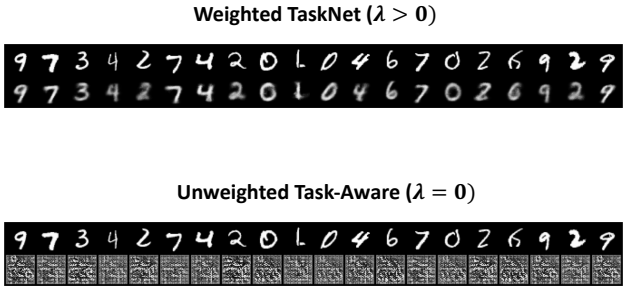


Fig. 4: **Motivation for regularization with a reconstruction loss.** The key benefit of adding a reconstruction weight with $\lambda > 0$ in our task loss (Eq. 1) is to yield decoded images that are human-interpretable, as shown for our warm-up MNIST example. In each panel, we show original images x in the top row and decoded images $\hat{x}$ in the bottom row. **Top Panel, TASKNET with $\lambda = 0.01$:** Clearly, adding a regularization term with a representative reconstruction weight $\lambda = 0.01$ in Eq. 1 learns representations that are highly-compressible with $Z = 4$, but yield interpretable decoded samples $\hat{x}$ that a human can debug. **Bottom Panel, $\lambda = 0$:** In contrast, if we purely optimize for a machine's task loss with reconstruction weight $\lambda = 0.0$, decoded images $\hat{x}$ (bottom row) are *not* human-interpretable, motivating our weighted co-design approach.

compared to solely running task network f. In the end-to-end scheme, the task network is also learned during training.

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 [11]. We used Martian terrain images from NASA's HiRise Dataset [8], consisting of 8 classes including dunes, craters, impact ejecta, and other formations. The labeled HiRise Dataset images [8] are several hundreds of megabytes large, unlike the larger HiRise panoramas we use for our subsequent motion planning experiments in Sec. IV-D. 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 [30]. 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, as shown in Fig. 5a, 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, Fig. 5a shows how 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.

Fig. 5b 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, Figs. 5b and 6 show by computing until block-4c, TASKNET achieves 3.87 $\times$ lower classification error than conventional task-agnostic methods, with a small bottleneck of $Z = 128$ 32-bit floating-point values. This enables our scheme to send only 3.27% of data compared to the full feature map of size $14 \times 14 \times 80$. However, by introducing our learned robot encoder p and server decoder q (pictured in Fig. 5a), we introduce a small overhead of $< 7.37\%$ extra compute compared to simply running the original EfficientNet, as shown in Fig. 7a. In practice, this small compute overhead is tolerable given the accuracy benefits of querying a powerful remote server with a small transmitted data representation.

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. 7a). As such, TASKNET is significantly more efficient than naïvely just splitting a compute-intensive DNN in half or running it fully on-board, which could be infeasible for micro-robots or miniature satellites like KickSats [23].

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 [28] that generates collision-free paths an order of magnitude faster than conventional sampling-based planners such as RRT* [15], but with virtually the same path cost. MPNET consists of an obstacle encoder network, such as a

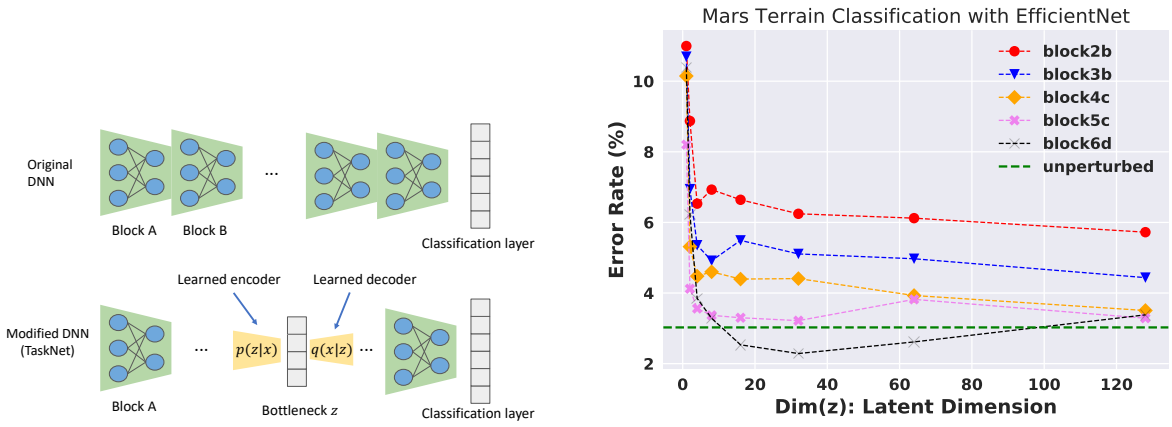


Fig. 5: **Mars terrain classification. (Left) Flexible allocations of robot and server compute:** In a nominal configuration, our co-design algorithm fully runs a modular perception model at a remote server and simply encodes task-relevant features at a robot, as shown in Fig. 1. However, a key benefit of our approach is that we can *also* flexibly divide computation for a large perception model between a robot and server. As shown above, we can take a *pre-trained* EfficientNet and insert a learned, task-relevant encoder (yellow) to map an intermediate feature map to a bottleneck representation z. Once z is transmitted, the learned decoder (yellow) uncompresses the intermediate feature map so that computation can continue at a remote server with the rest of the pre-trained EfficientNet. Unlike classical approaches that simply split large DNNs between devices [14, 13], we co-learn an encoder/decoder (yellow) along with the pre-trained parameters (green) for high accuracy with a small Z. **(Right)** We plot the accuracy for various pictured allocations between robot and server compute. Specifically, we plot TASKNET with $\lambda = 0$ for scenarios where the robot only runs up to EfficientNet block b locally and achieve within 1.5% of the uncompressed input error rate (green dashed line) at block $4c$ and beyond for $Z = 128$.

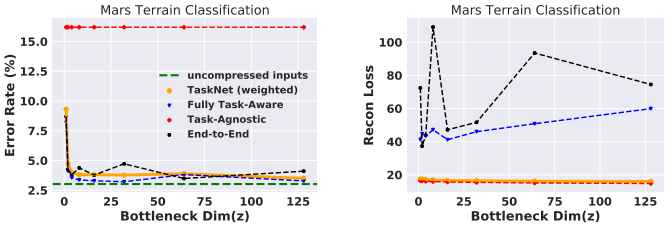
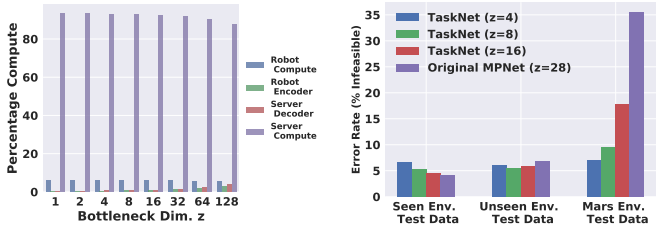


Fig. 6: **Terrain classification.** TASKNET achieves much lower classification error than task-agnostic methods (red), where even a bottleneck dimension of $z = 128$ corresponds to sending only 3.27 $\%$ of data compared to original convolutional feature maps. While our method incurs a compute overhead by adding a task-relevant encoder and decoder, it is less than 7.37% of total computation. Such overhead is worthwhile for robots to unlock the accuracy of remote computation while transmitting minimal data.



(a) Martian Terrain Classification (b) Neural Motion Planning

Fig. 7: (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).

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. Since the original embedding z_0 is not human-interpretable, we plot results for $\lambda = 0$ though we thoroughly evaluated λ uniformly. Our task-relevant encoder/decoder consist of a small fully connected neural network, leading to a compute overhead of only $< 2\%$ more parameters.

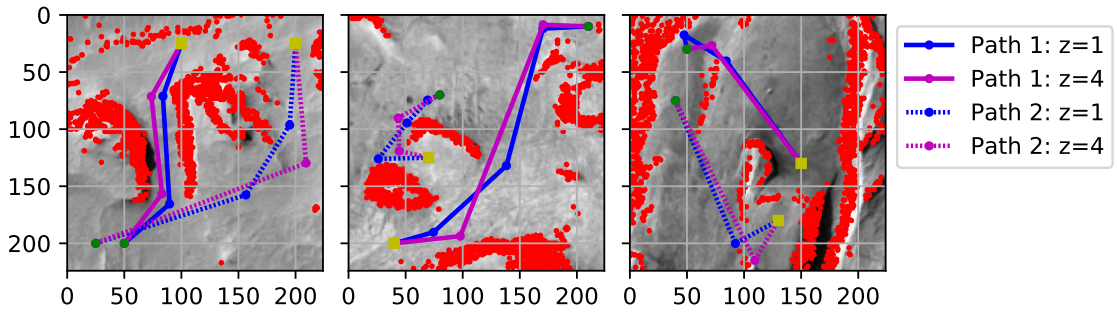


Fig. 8: **Neural Motion Planning on Mars Terrain Data:** Our task-aware motion planner generalizes to unseen Martian terrain data for much smaller point-cloud embedding sizes z compared to a publicly-available neural motion planner.

Fig. 7b 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 environment 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. 8 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. 7b 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 [28]. Our key experimental result is shown in the third column of Fig. 7b, 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 example, we plan to further stress test TASKNET's generalization capabilities in future work.

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. The task network $f(;\theta_{\text{task}})$ is a compact neural network that classifies each timeseries window of length w, $x^{t-w:t}$, 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 $f(;\theta_{\text{task}})$, which was trained on stochastic, diverse timeseries, achieved a promising 90% test accuracy. Fig. 9 shows that TASKNET aggressively compresses the timeseries measurements and outperforms the task-agnostic benchmark. Furthermore, we plot for $\lambda = 1$ which reflects similar weight between reconstruction loss and accuracy. 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 minimizes 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 $\times$. A key benefit of our co-design algorithm is that it is rooted in linear systems theory and gracefully scales to complex DNNs. We envision that our co-design algorithm can be applied to a variety of resource-constrained robots that need to distill

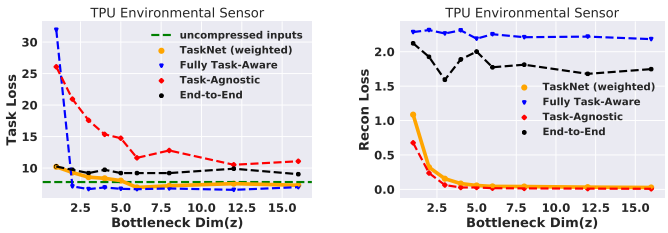


Fig. 9: **Environmental Sensing:** TASKNET achieves low reconstruction loss and matches the low classification error rate achieved when using *original* sensor data.

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.

VI. ACKNOWLEDGEMENTS

The NASA University Leadership initiative (grant #80NSSC20M0163) provided funds to assist the authors with their research, but this article solely reflects the opinions and conclusions of its authors and not any NASA entity.

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/mission/communications/, 2020. [Online; accessed 18-Oct.-2020].

[3] Y. Cheng, D. Wang, P. Zhou, and T. Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

[4] 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.

[5] 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.

[6] M. Crawshaw. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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

[9] J. Emmons, S. Fouladi, G. Ananthanarayanan, S. Venkataraman, S. Savarese, and K. Winstein. Cracking open the dnn black-box: Video analytics with dnns across the camera-cloud boundary. In *Proceedings of the 2019 Workshop on Hot Topics in Video Analytics and Intelligent Edges*, pages 27–32, 2019.

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

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

[12] 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.

[13] B. Hassibi, D. G. Stork, and G. J. Wolff. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pages 293–299. IEEE, 1993.

[14] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGPLAN Notices*, 52(4):615–629, 2017.

[15] 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.

[16] B. Kehoe, A. Matsukawa, S. Candido, J. Kuffner, and 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.

[17] 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.

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

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

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

[21] P. Li, B. DeRose, J. Mahler, J. A. Ojea, A. K. Tanwani, and K. Goldberg. Dex-net as a service (dnaas): 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.

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

[23] Z. Manchester, M. Peck, and A. Filo. Kicksat: A crowd-funded mission to demonstrate the worlds smallest spacecraft. 2013.

[24] 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.

[25] 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.

[26] 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.

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

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

[29] A. Sonar, V. Pacelli, and A. Majumdar. Invariant policy optimization: Towards stronger generalization in reinforcement learning. *arXiv preprint arXiv:2006.01096*, 2020.

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

[31] 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.

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

[33] C. Tu, 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.

[34] J. Vander Hook, J. Castillo-Rogez, R. Doyle, T. S. Vaquero, T. M. Hare, R. L. Kirk, V. Fox, D. Bekker, and A. Cocoros. Nebulae: A proposed concept of operation for deep space computing clouds. In *2020 IEEE Aerospace Conference*, pages 1–14. IEEE, 2020.

[35] 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.