Interpretable Trade-offs Between Robot Task Accuracy and Compute Efficiency

Bineet Ghosh[1], Sandeep Chinchali[2], Parasara Sridhar Duggirala[1]

Abstract— A robot can invoke heterogeneous computation resources such as CPUs, cloud GPU servers, or even human computation for achieving a high-level goal. The problem of invoking an appropriate computation model so that it will successfully complete a task while keeping its compute and energy costs within a budget is called a *model selection problem*. In this paper, we present an optimal solution to the model selection problem with two compute models, the first being fast but less accurate, and the second being slow but more accurate. The main insight behind our solution is that *a robot should invoke the slower compute model only when the benefits from the gain in accuracy outweigh the computational costs*. We show that such cost-benefit analysis can be performed by leveraging the statistical correlation between the accuracy of fast and slow compute models. We demonstrate the broad applicability of our approach to diverse problems such as perception using neural networks and safe navigation of a simulated Mars rover.

I. INTRODUCTION

Ideally, robotic computation should be highly accurate, responsive, and fast, as well as compute-and-power-efficient. Modern robots, however, face the challenge of selecting from an array of heterogeneous compute resources, each with a unique trade-off between accuracy and compute cost. For example, should a factory robot trust the perception results from an on-board deep neural network (DNN) or ask a busy human supervisor for help? Likewise, should a small drone compute its motion plan locally, or wait for a higher-fidelity plan from a remote server? At their core, these scenarios are instances of a *compute model selection* problem, where a robot must gracefully balance task-relevant accuracy with compute time, power, or network and human-processing delay.

Figure 1 illustrates the model selection problem addressed in this paper. Given the sensor observations x at each time step, a robot's model selection policy π must dynamically invoke either a fast, compute-and-power-efficient model (f_{fast}) or a slower, more accurate model (f_{slow}) based on a high-level task's required accuracy. Variants of this problem have been studied for perception tasks in cloud robotics [1], [2] and human-robot collaboration [3], [4], [5]. However, existing works either offer specialized point-solutions (e.g. for perception [6], [7]) that do not readily generalize to other domains, use hand-engineered heuristics, or employ uninterpretable, learning-based algorithms [1], [8]. *Our key contribution is to provide a unified, interpretable,*

[1]Bineet Ghosh and Parasara Sridhar Duggirala are with the Department of Computer Science, University of North Carolina at Chapel Hill, USA {bineet,psd}@cs.unc.edu
[2]Sandeep Chinchali is with the Department of Electrical and Computer Engineering, University of Texas at Austin, USA sandeepc@utexas.edu
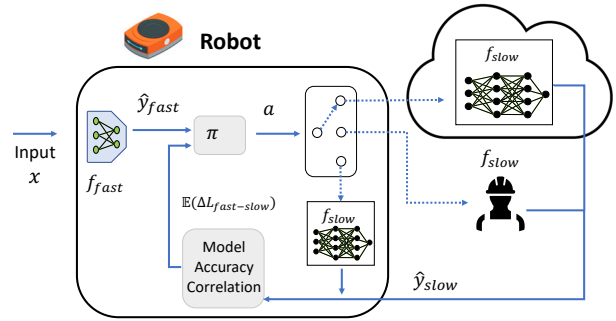
Fig. 1: **The Compute Model Selection Problem:** A robot must balance task accuracy and compute cost, such as energy or latency, when choosing between heterogeneous compute resources. Our interpretable model selection policy π leverages the statistical correlation between fast and slow compute models f_{fast} and f_{slow} to dynamically decide which model to invoke.

and theoretically-grounded framework for compute model selection in robotics.

The fundamental principle behind selecting an appropriate compute model is to perform a cost-benefit analysis. Our key insight is that a robot's model selection algorithm can leverage the statistical, and often analytical, correlation between the accuracy of the fast and the slow compute models. This correlation can enable us to perform *reliable and interpretable* cost-benefit analysis between compute cost and gain in accuracy for the different models. Crucially, such correlations between fast and slow models are now possible even for state-of-the-art DNNs, due to recent advances that compress large DNNs with provable approximation guarantees [9], [10].

Literature Review: Our work is broadly related to computational offloading in cloud robotics as well as teacher feedback for human-robot interaction. The closest work to ours is [1], which develops a deep reinforcement learning (RL) policy to select between a fast, less accurate deep neural network (DNN) or slower, more accurate DNN running at a cloud computing server. Indeed, we explicitly build upon [1] in our formulation by considering fast and slow compute models with a hierarchy of compute costs. In stark contrast to [1], however, we avoid uninterpretable, RL-based model selection policies. Instead we leverage statistical correlations between fast and slow computational models, such as compression algorithms for DNNs [9], [10]. Further, unlike [1], we introduce a theoretically-grounded cost-benefit analysis for model selection, which generalizes beyond DNNs to high-dimensional linear regression and even sampling-based reachability problems, as shown in our evaluation.

Our work is also inspired by methods to compress large

DNNs for efficient inference on compute-and-power limited robots. For example, the EfficientNet [11] suite of vision models provides 7 model variants that trade-off accuracy with model size and latency. Importantly, recent methods utilize core-set theory to train fast, compressed DNNs that provably approximate a slower DNN by pruning convolutional filters based on sensitivity analyses [9], [10].

Finally, our work is related to scenarios where a robot must selectively ask a human teacher for clarification during active learning tasks [3], [4] or remote assistance for manipulation [5]. In principle, our framework applies to such settings if a robot can accurately correlate its confidence with the marginal accuracy gain it receives from human feedback. In practice, however, such correlations can often be learned from historical interaction data but are hard to *analytically* quantify.

Contributions: Given prior literature, our contributions are three-fold. First, we design an interpretable model selection algorithm which leverages analytical correlations between fast and slow model performance to dynamically decide which model to invoke. Second, we show how our algorithm can naturally leverage recent advances that compress large DNNs with provable approximation guarantees that relate fast and slow models. Third, we show strong experimental performance of our algorithm on diverse domains ranging from robotic perception to sampling-based reachability analysis for a simulated rover navigating Martian terrain data.

Organization: This paper is organized as follows. In Section II, we introduce a general formulation for model selection to gracefully trade-off task accuracy and compute costs. In Sec. III, we provide theoretical guarantees for model selection and instantiate them for applications in perception and reachability analysis in Sec. IV. Finally, we provide our experimental results in Sec. V and conclude in Sec. VI.

II. PROBLEM STATEMENT

In this section, we formally define the problem of model selection depicted in Fig. 1 by introducing compute models, an accuracy metric, and a performance criterion.

Compute Model Input: The input to the compute model, at time t, is denoted by $x^t \in \mathbb{R}^n$. We denote the input data distribution by $\mathcal{X}$, that is, $x^t \sim \mathcal{X}$. In practice, x^t could represent a depth-camera image or laser scan.

Compute Models: The compute models are denoted $f_i : \mathbb{R}^n \to \mathbb{R}^m$, where $i \in \{\text{fast}, \text{slow}\}$. Given an input x^t, the output is denoted by $y_i^t = f_i(x^t)$. The cost associated with f_i is given by $c_i \in \mathbb{R}_+$. The cost is context-dependent, such as battery consumption, compute inference latency, or even communication latency for cloud robotics tasks. For example, the compute models could be a DNN, with image input x^t and corresponding segmentation y^t. The distribution of outputs is denoted by $\mathcal{Y}$, that is, $y_i^t \sim \mathcal{Y}$. The ground-truth output associated with input x^t is denoted by y_{oracle}^t.

Loss Function: Let $\mathcal{L}(y_1^t, y_2^t) : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}_{\geq 0}$ be the loss function, that formally quantifies the quality of the output y_1^t returned by a compute model compared to the ground-truth result of y_2^t. A lower value of $\mathcal{L}(\cdot)$ indicates a more accurate

output. In practice, the loss function is context-dependent, such as the cross-entropy loss for image classification.

Model Selection Policy: Given an input x^t, the model selection policy decides whether to use the slow compute model f_{slow}, or the fast model f_{fast}. We assume that the policy has access to the results of the fast model (without this information, the policy would be purely random). Therefore, the problem of model selection is to infer whether or not to *additionally* invoke the slow model to enhance task accuracy if the fast model results are insufficient for a robot's high-level goal. The challenge is that the robot only has access to the input x^t and the fast model output y_{fast}^t and thus must estimate the accuracy benefit of the slow model *before* invoking it.

Formally, we define the model selection policy as $\pi : \mathbb{R}^n \times \mathbb{R}^m \to \{0, 1\}$. Given input x^t and fast model prediction y_{fast}^t, we define the action as $a^t = \pi(x^t, y_{\text{fast}}^t)$. The action $a^t = 0$ indicates selecting the fast model f_{fast}, and $a^t = 1$ indicates selecting the slow model f_{slow}. We define the cost associated with each action as $\text{cost}(a^t)$:

$$\text{cost}(a^t) = \begin{cases} c_{\text{fast}} & \text{if } a^t = 0 \\ c_{\text{fast}} + c_{\text{slow}} & \text{if } a^t = 1 \end{cases}.$$

Reward: To simultaneously achieve high task accuracy while minimizing the cost of compute, we introduce a per-timestep reward. Given input x^t, the output of the fast model y_{fast}^t, and model selection a^t, the corresponding reward is:

$$R^t(a^t) = \begin{cases} -\alpha \mathcal{L}(y_{\text{fast}}^t, y_{\text{oracle}}^t) - \beta \text{cost}(0) & \text{if } a^t = 0 \\ -\alpha \mathcal{L}(y_{\text{slow}}^t, y_{\text{oracle}}^t) - \beta \text{cost}(1) & \text{if } a^t = 1 \end{cases} \tag{1}$$

where $\alpha, \beta \in \mathbb{R}_+$ are user-defined weights to balance the emphasis on accuracy and cost. These can be flexibly set by a roboticist given the unique requirements of a high-level task. For example, a fleet of low-power, compute-limited warehouse robots that rarely interact with humans might have a higher emphasis β on cost to minimize how many times they query a shared central server or remote human supervisor. Conversely, robots that operate in safety-critical scenarios will have a much higher emphasis on accuracy given by α.

A. Formal Problem Definition

Given a stream of N inputs, $\{x^1, x^2, \cdots, x^N\}$, our goal is to propose an optimal model selection policy π^*, that provably maximizes the expected cumulative reward:

$$\mathbb{E} \sum_{t=0}^{N} R^t(\pi^*(x^t, y_{\text{fast}}^t))$$

Intuitively, π^* achieves the optimal balance between the cost and accuracy over the given period of N time steps. We now formally define the model selection problem.

Problem 1 (Model Selection for Inference): Given fast model f_{fast}, slow model f_{slow}, loss function $\mathcal{L}(\cdot)$, and model selection cost $\text{cost}(\cdot)$, find the optimal model selection policy π^*, which maximizes the reward (Equation 1) over a

finite horizon N:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \, \mathbb{E} \sum_{t=0}^{N} R^t \big(a^t = \pi(x^t, y_{\text{fast}}^t)\big).$$

Section III provides our solution to Problem 1.

B. Discussion on the Problem Definition

The model selection problem is broadly applicable in robotics since it is agnostic to the nature of the compute models, the loss function, or even costs. For example, the models could represent small quantized and large, compute-intensive DNNs or even small and large databases or random forests. Further, the costs could represent battery consumption or communication delay or model inference time.

The main challenge of this problem is the limited information available to the selection policy, namely the input x^t, fast model output y_{fast}^t, and the cost function $\text{cost}(\cdot)$. The key challenge is to estimate the accuracy of the slow model, f_{slow}, before even invoking it, which motivates our key technical approach to statistically relate both models' accuracy.

III. AN ALGORITHMIC APPROACH TO MODEL SELECTION

In this section, we provide an optimal solution to the model selection problem (Problem 1). First, we make the following practically-motivated assumption.

Assumption 1 (Action and State Independence): Given a model input x^t at any time t, the model selection a^t of policy π does not affect the next robot measurement x^{t+1}.

Our assumption is practical in many robotics scenarios, since a^t is simply a choice of a compute model to process inputs, not a physical *actuation* decision. For example, a robot can run a fast perception DNN on images x^t at every timestep and its choice to optionally consult a slower DNN a^t does not affect the new image observation x^{t+1}, which is instead largely affected by its ego-motion and surroundings. Our assumption will not hold for fast-moving robots whose control decisions are heavily dependent on the perception model they invoke, which we discuss in our future work.

Theorem 1: The optimal model selection policy that solves Problem 1 is of the form:

$$\pi^*(x^t, y_{\text{fast}}^t) = \mathbb{1} \left(\frac{\beta}{\alpha} c_{\text{slow}} < \right.$$

$$\left. \underbrace{\mathbb{E} \left(\mathcal{L}(y_{\text{fast}}^t, y_{\text{oracle}}^t) - \mathcal{L}(y_{\text{slow}}^t, y_{\text{oracle}}^t) \right)}_{\text{task accuracy gain}} \right)$$

Proof: By Assumption 1, the action a^t at every time does not affect the next state x^{t+1}. Thus, given any input x^t, the actions a^t are independent, so to maximize the cumulative reward it suffices to maximize the reward at every timestep independently. Recall from Equation 1 that the reward depends on two choices of a^t, that is, $a^t \in \{0, 1\}$. Therefore, Problem 1 can be rewritten as:

$$\pi^*(x^t, y_{\text{fast}}^t) = \underset{a^t \in \{0,1\}}{\operatorname{argmax}} \mathbb{E}(R^t(a^t))$$

Substituting in the reward definition (Equation 1), we see that we should choose the slow model only when the associated

reward is higher than continuing with the fast model. Thus, we choose $a^t = 1$ only when:

$$-\alpha \mathbb{E} \big(\mathcal{L}(y_{\text{oracle}}^t, y_{\text{slow}}^t) \big) - \beta(c_{\text{slow}} + c_{\text{fast}}) >$$
$$-\alpha \mathbb{E} \big(\mathcal{L}(y_{\text{oracle}}^t, y_{\text{fast}}^t) \big) - \beta c_{\text{fast}}$$

Simplifying, we arrive at the desired result:

$$\pi^*(s^t) = \mathbb{1} \left(\frac{\beta}{\alpha} \underbrace{c_{\text{slow}}}_{\text{extra compute cost}} < \right. \tag{2}$$

$$\left. \mathbb{E} \big(\underbrace{\mathcal{L}(y_{\text{oracle}}^t, y_{\text{fast}}^t) - \mathcal{L}(y_{\text{oracle}}^t, y_{\text{slow}}^t)}_{\text{task accuracy benefit}} \big) \right).$$

∎

Theorem 1 suggests a simple model selection policy, which estimates the model accuracy gap $\Delta \mathcal{L} = \mathbb{E} \big(\mathcal{L}(y_{\text{oracle}}^t, y_{\text{fast}}^t) - \mathcal{L}(y_{\text{oracle}}^t, y_{\text{slow}}^t) \big)$, and only chooses the slow model if the gap is greater than a threshold that depends on the relative compute costs and weights of accuracy via α, β. However, the key challenge is that calculating $\Delta \mathcal{L}$ requires querying the slow model and knowledge of the ground-truth value y_{oracle}^t. We now transition to two practical approaches to directly instantiate the guarantees from Theorem 1 in practice.

First, we note that in many practical deployment scenarios, the ground-truth oracle values are not present. In such practical settings, the more accurate slow model simply serves as the ground-truth, such as when a slow human supervisor makes ground-truth decisions. In the absence of human annotations, a large, compute-intensive DNN can serve as the slow model and ground-truth. Thus, we present the following lemma of Theorem 1.

Lemma 1: The optimal model selection policy that solves Problem 1, when $y_{\text{oracle}}^t = y_{\text{slow}}^t$ at all times t is:

$$\pi^*(x^t, y_{\text{fast}}^t) = \mathbb{1} \left(\frac{\beta}{\alpha} c_{\text{slow}} < \mathbb{E} \left[\mathcal{L}(y_{\text{fast}}^t, y_{\text{slow}}^t) \right] \right)$$

Proof: The proof is the same as Theorem 1, where we note that $\mathbb{E} \big(\mathcal{L}(y_{\text{oracle}}^t, y_{\text{slow}}^t) \big) = 0$ when the oracle and slow models are identical. ∎

For our evaluation, we use Lemma 1 as our model selection policy as it best reflects practical autonomous deployments. The key challenge to directly applying Theorem 1 and Lemma 1 is to accurately estimate the loss between fast and slow models solely using predictions from the fast model. However, we now show we can indeed compute the expected accuracy benefit for a broad class of fast and slow models that are related by provable approximation guarantees. Specifically, in Subsection III-A, we instantiate the guarantees of Lemma 1 to provide a closed-form, analytic model selection policy for linear regression problems. Crucially, we then extend our analysis to DNN inference in Subsection III-B.

Lemma 1 provides a general framework for model selection. For a novel setting, it can be instantiated by selecting the (i) compute models, (ii) loss function, (iii) compute model cost, and (iv) characterizing the statistical relationship between compute models to derive the selection policy. Sections III-A and III-B instantiate Lemma 1 for specific cases of Linear Regression and DNN inference.

A. Analytical Results for Linear Regression

We now apply the guarantees from Lemma 1 to an illustrative warm-up example of high-dimensional linear regression. Recall that our challenge is to estimate the expected value of y_{slow}^t from the information available to the selection policy, namely input x^t and fast model prediction y_{fast}^t. To overcome this challenge, we apply results to approximate linear regression models using *coresets* [12], which are importance-ranked subsets of a large training dataset. Importantly, a model trained on just the coreset will provably approximate the predictions of one trained on the full dataset. For example, a fast model could be trained on only a core-set of local data on-board a robot while a large one could be trained on multiple robots' data in the cloud.

Compute Models: Let the fast and slow compute models f_i be linear regression models $f_i(x) = A_i x + b_i$, where $i \in \{\text{fast}, \text{slow}\}$, $A_i \in \mathbb{R}^{m \times n}$, and $b_i \in \mathbb{R}^{m \times 1}$. We assume the slow model f_{slow} is learned on a full set of training samples from a joint distribution on $\mathcal{X} \times \mathcal{Y}$, while the fast model is only trained on a core-set of the original data.

Loss Function: Let the loss function be the standard L_2 norm loss: $\mathcal{L}(y_1^t, y_2^t) = ||y_1^t - y_2^t||_2^2$; where $y_1^t, y_2^t \in \mathbb{R}^m$. The following coreset guarantees follow from [12]:

Property 1 (Relation between fast and slow models [12]): For all t, given input x^t, denote the compute model outputs as $y_{\text{fast}}^t = f_{\text{fast}}(x^t)$ and $y_{\text{slow}}^t = f_{\text{slow}}(x^t)$. Then, there exists an $\epsilon > 0$ such that:

$$y_{\text{fast}}^t \in \left[y_{\text{slow}}^t, (1 + \epsilon) y_{\text{slow}}^t \right], \quad (3)$$

where $\mathcal{X}$ and $\mathcal{Y}$ are the input and output distributions, meaning $x^t \sim \mathcal{X}$, and $y_{\text{slow}}^t, y_{\text{fast}}^t \sim \mathcal{Y}$. [12] provides the approximation factor ϵ based on the relative size of the core-set compared to the full training set.

Property 1 allows us to relate the fast and slow model predictions as:

$$y_{\text{slow}}^t \leq y_{\text{fast}}^t \leq (1 + \epsilon) y_{\text{slow}}^t$$
$$or, \quad y_{\text{slow}}^t \in \left[\frac{y_{\text{fast}}^t}{1 + \epsilon}, y_{\text{fast}}^t \right] \quad (4)$$

Thus, the loss function can be upper bounded as:

$$\mathcal{L}(y_{\text{fast}}^t, y_{\text{slow}}^t) \leq \frac{\epsilon^2 \cdot (y_{\text{fast}}^t)^2}{(1 + \epsilon)^2}. \quad (5)$$

Finally, we can use Equation 5 and Lemma 1 to provide a closed-form model selection policy for Problem 1 in the linear regression setting:

$$\pi(x^t, y_{\text{fast}}^t) = \mathbb{1}\left(\frac{\beta}{\alpha} c_{\text{slow}} < \frac{\epsilon^2 \cdot (y_{\text{fast}}^t)^2}{(1 + \epsilon)^2} \right). \quad (6)$$

We stress that Lemma 1 provides the *optimal* solution and the above solution is an approximation since we upper-bound the loss function between fast and slow models. However, our subsequent experiments show this is a very tight bound and implementing Eq. 6 yields very close performance to an unrealizable oracle solution that has perfect knowledge of the fast and slow model predictions.

B. Analytical Results for Deep Neural Networks (DNNs)

We now provide a similar analysis to the linear regression scenario for the important case when a robotic perception DNN has been compressed using recently developed coreset guarantees [9], [10]. Specifically, [9] compresses fully connected DNNs with ReLU activations by targetedly removing weights with low relative importance via coresets. [10] extends this work to convolutional neural networks (CNNs) by using coresets to remove convolutional filters that a prediction is least sensitive to, which enables a compressed DNN to provably approximate its original counterpart.

Compute Models: Let the models $f_i: \mathbb{R}^n \to \mathbb{R}^m$, where $i \in \{\text{fast}, \text{slow}\}$ be DNNs. Both models are trained on a set of samples drawn from a joint data distribution on $\mathcal{X} \times \mathcal{Y}$.

Loss Function: As for linear regression, the loss function is an L_2 norm loss, such as for depth estimation from a perception CNN.

We now use the following guarantees for DNNs.

Property 2 (Relation between fast and slow models [9], [10]): For all t, given x^t, $y_{\text{fast}}^t = f_{\text{fast}}(x^t)$, and $y_{\text{slow}}^t = f_{\text{slow}}(x^t)$, there exist an $\epsilon, \delta > 0$, such that the following holds:

$$\mathbb{P}\left(y_{\text{fast}}^t \in \left[(1 - \epsilon) y_{\text{slow}}^t, (1 + \epsilon) y_{\text{slow}}^t \right] \right) \geq 1 - \delta, \quad (7)$$

$$\mathbb{P}\left(y_{\text{fast}}^t \in \left[y_{\text{slow}}^t - \frac{M}{2}, y_{\text{slow}}^t + \frac{M}{2} \right] \right) \leq \delta, \quad (8)$$

where $M \geq 0$ is an upper bound on the error described below. ϵ and δ depend on the extent of DNN compression. Further, input $x^t \sim \mathcal{X}$ and outputs $y_{\text{slow}}^t, y_{\text{fast}}^t \sim \mathcal{Y}$.
Equation 8 and bound M arise from the observation that in practical engineering scenarios, the outputs of a neural network and thus the loss will be bounded since they have physical meaning. For example, for a regression loss with perception, $M \geq 0$ could be derived from the largest depth-reading a depth sensor can register. Likewise, for classification, M is naturally bounded by 1 since the outputs y are softmax scores from a cross-entropy loss.

We now use the core-set relationship to analyze Lemma 1 for DNN inference as follows:

$$y_{\text{slow}}^t \in \begin{cases} \left[\frac{y_{\text{fast}}^t}{1 + \epsilon}, \frac{y_{\text{fast}}^t}{1 - \epsilon} \right] & \text{with prob.} \geq 1 - \delta \\ \left[y_{\text{fast}}^t - \frac{M}{2}, y_{\text{fast}}^t + \frac{M}{2} \right] & \text{with prob.} \leq \delta. \end{cases} \quad (9)$$

Thus, using Equation 9, the loss can be upper bounded as:

$$\mathcal{L}(y_{\text{fast}}^t, y_{\text{slow}}^t) \leq \begin{cases} \frac{\epsilon^2 \cdot (y_{\text{fast}}^t)^2}{(1 - \epsilon)^2} & \text{with probability } (1 - \delta) \\ M & \text{with probability } \delta. \end{cases} \quad (10)$$

Therefore, the expectation of the loss function is:

$$\mathbb{E}\left[\mathcal{L}(y_{\text{fast}}^t, y_{\text{slow}}^t) \right] \leq \delta M + (1 - \delta) \left(\frac{\epsilon^2 \cdot (y_{\text{fast}}^t)^2}{(1 - \epsilon)^2} \right). \quad (11)$$

Finally, we can apply Equation 11 and Lemma 1 to provide a closed-form model selection policy for Problem 1 in the DNN setting:

$$\pi(x^t, y_{\text{fast}}^t) = \mathbb{1}\left(\frac{\beta}{\alpha} c_{\text{slow}} < \delta M + (1 - \delta) \frac{\epsilon^2 \cdot (y_{\text{fast}}^t)^2}{(1 - \epsilon)^2} \right) \quad (12)$$

As for linear regression, we emphasize Lemma 1 is optimal and Eq. 12 is an approximation since we are *bounding* the expectation using the core-set guarantee. However, our experiments show that implementing Eq. 12 as a proxy for Lemma 1 works well in practice. More broadly, we emphasize that core-set guarantees are simply one way to instantiate the general policy provided in Lemma 1. For example, a roboticist could also use other practically-relevant models such as random forests or even approximate databases if they can reliably relate fast and slow model accuracy.

IV. APPLICATION SCENARIOS

We now describe example application scenarios of high-dimensional linear regression, DNN inference, and reachable set computation for a simulated Mars Rover to demonstrate the theoretical guarantees from Section III.

A. Linear Regression

Using the analytical results from Subsection III-A, we demonstrate our model selection policy by simulating it on a toy example of linear regression. Let $f_{\text{slow}} : \mathbb{R}^4 \to \mathbb{R}^4_{[0,1]}$ be any general-purpose linear regression model. The amount of time f_{slow} takes to generate an output is 2.5 seconds. Using coresets, we compress the linear regression model f_{slow}, to a faster linear regression model f_{fast}. The f_{slow} model takes 1 second to generate an output. The compression is such that the relation in Equation 3 holds with $\epsilon = 0.1$.

We chose $\alpha = 1$ and $\beta = 0.003$ to emphasize accuracy over compute efficiency in our simulations, although α, β can be flexibly set by a user. We implement the model selection policy as in Equation 6, and demonstrate its performance in Section V against benchmark policy selection algorithms (discussed in Subsection IV-D).

B. Compute Efficient Robotic Perception

We now stress-test our algorithm on a scenario, inspired by [13], where an aircraft must autonomously track a runway center-line using a wing-mounted camera for state estimation. This scenario, henceforth referred to as the TaxiNet scenario as per [13], uses a DNN to map from camera images to an estimate of the aircraft's lateral distance from the runway center-line d and heading angle θ, which are linearly combined to create the aicraft's steering control. We chose the TaxiNet scenario since the central idea is broadly applicable to resource-constrained robotics, such as low-power drones that use efficient vision models to estimate their real-time pose relative to a landing site.

We trained a ResNet-18 DNN [14] to serve as the slow perception model f_{slow} using over 50K images from the standard X-Plane simulator [15] using a publicly-available dataset [16]. The ResNet-18 achieved a low MSE loss of 0.038 on an independent test dataset of 18,372 images, where each image took 0.17 seconds for inference on a CPU. We compressed f_{slow} to yield a quantized ResNet-18 as f_{fast}, which was 47.21% faster but had a 64% higher loss, illustrating a clear need for model selection.

For the TaxiNet scenario, we chose the model costs of $c_{\text{slow}} = 0.017$ and $c_{\text{fast}} = 0$ based on their relative inference

times and $\alpha = 1$, $\beta = 3 \times 10^{-4}$ to emphasize safety (low loss) over compute costs. Our model selection results are demonstrated in Section V along with example DNN predictions from aircraft images in Figure 4 (Right).

C. Reachable Set Computation

In this subsection, we apply our model selection policy to safety assessment for robot navigation. Consider a robot, such as a Mars rover, navigating an unexplored environment. The robot has to assess whether its maneuvers are safe while considering environment uncertainties such as the coefficient of friction, wind disturbances, etc.. This is done by computing a *reachable set*, a set that contains all the states a rover can potentially reach. The robot can make the maneuver safely if the reachable set does not overlap with any obstacles. The reachable set computed by the fast compute model has confidence that is an order of magnitude lesser than the reachable set computed by the slow model.

We assume that the closed loop dynamics of the robot is given as a nonlinear system. For computing the reachable sets, we approximate the nonlinear dynamics locally as an uncertain linear system, where the coefficients in the dynamics belong to a bounded range.

Example 1: Consider the discrete uncertain linear dynamical system $x^+ = \Lambda x$ where $\Lambda = \begin{bmatrix} 1 & \alpha \\ 4 & 6, \end{bmatrix}$ where x is the state, x^+ is the next state, and $\alpha \in [-2, 2]$ represents either the modeling uncertainty or a parameter. Given an initial state x, the reachable set of the uncertain linear system includes the set of states reached by the system for any value of α in the interval $[-2, 2]$ for a specified time horizon t.

Prior work [17], [18], [19] has shown that computing reachable sets for linear systems with uncertainties is a computationally expensive process. Recently, a statistical approximation of the reachable set has been presented in [20]. The confidence of the statistical approximation can be tuned by the user according to her performance and accuracy requirements. Leveraging the flexibility of this statistical approach, we generate a fast compute model which has medium confidence and slow model that has high confidence over the computed reachable sets. Given the various constraints on robot resources, the model selection policy should invoke the appropriate compute model to guarantee safety while minimizing the cost.

Formally, consider a linear dynamical system with uncertainties, represented as $x^+ = \Lambda x$, where $\Lambda \in \mathbb{R}^{n \times n}$ is an uncertain dynamical matrix. The reachable set of the current state x up to a time horizon t, is denoted as $\text{RS}(\Lambda, x, t)$. Though the dynamics is given as $x^+ = \Lambda x$, it can encompass the open loop behavior $x^+ = \Lambda_A x + \Lambda_B u$ (where Λ_A, Λ_B are uncertain matrices), if a control sequence u is provided. In such cases, the uncertain matrices Λ_A, Λ_B are combined together to Λ by concatenating the state and open-loop control.

A system is unsafe if the reachable set intersects with the unsafe set, such as obstacles. That is, given an unsafe set $U \subset \mathbb{R}^n$, a system is unsafe if and only if $\text{RS}(\Lambda, x, t) \cap U \neq \emptyset$. Given $\mu > 0$ and a set $S \subseteq \mathbb{R}^n$, we denote the uniform expansion (*bloating*) of set S by μ as $B_\mu(S)$.

We now formally present the model selection problem for safety assessment of robot navigation.

Computation Models: The compute models $i \in \{\text{fast}, \text{slow}\}$, denoted as $\text{RS}_i(\Lambda, x, t)$, compute approximations of the reachable set of an uncertain linear system defined by Λ [20]. The statistical guarantee $\mathcal{G}_i$ associated with RS_i is as follows:

$$\mathcal{G}_i : \text{ for any } A \in \Lambda, \ \mathbb{P}\big(\text{RS}_i(A, x, t) \subseteq \text{RS}_i(\Lambda, x, t)\big) \geq p_i$$

$\mathcal{G}_i$ has a *type I error* of δ_i. Here, the confidence $p_i \in \mathbb{R}_{[0,1]}$ and allowable type I error δ_i are user-given parameters to the models. Intuitively, $\mathcal{G}_i$ means the probability that the reachable set of any sample dynamics is contained within the reachable set $\text{RS}_i(\cdot)$ is at least probability p_i. Computing high-confidence approximations of the reachable set requires more statistical samples and therefore a higher computational time and cost. In particular, the required confidence p_i set by a user for statistical guarantee $\mathcal{G}_i$ is directly proportional to the required number of samples. Thus, we set the slow model to be a high-confidence reachable set and the fast model to be a lower-confidence approximation, so $p_{\text{slow}} > p_{\text{fast}}$, $\delta_{\text{slow}} \leq \delta_{\text{fast}}$, and therefore $c_{\text{slow}} > c_{\text{fast}}$. We denote the outputs of the fast and slow models as $y_{\text{fast}}^t = \text{RS}_{\text{fast}}(\Lambda, x, t)$ and $y_{\text{slow}}^t = \text{RS}_{\text{slow}}(\Lambda, x, t)$.

The crux of our selection policy is that we can relate the reachable sets returned by both models by a factor of ϵ:

Property 3 (Relationship between fast and slow models): Given Λ and θ, for all t, there exists an ϵ such that:

$$B_{1-\epsilon}\big(\text{RS}_{slow}(\Lambda, x, t)\big) \subseteq \text{RS}_{fast}(\Lambda, x, t) \quad (13)$$

$$\text{or,} \quad \text{RS}_{\text{slow}}(\Lambda, x, t) \subseteq B_{\frac{1}{1-\epsilon}}\big(\text{RS}_{\text{fast}}(\Lambda, x, t)\big). \quad (14)$$

In a calibration dataset, we can compute the fast and slow model reachable sets for all time steps. Then, we can set ϵ to be the minimum factor to bloat the robot's set such that the bloated version over-approximates the slow model's reachable set at all times. Thus, a robot can quickly run the fast model, bloat it by $\frac{1}{1-\epsilon}$, and continue planning if the *bloated* set does not intersect an unsafe region, as formalized below.

Loss Function: Given the safety-critical nature of navigation, the loss is 0 when the reachable set doesn't intersect an unsafe set and ∞ otherwise. Defining the reachable sets used to compute intersections with obstacles as $\Theta_{\text{fast}} = B_{\frac{1}{1-\epsilon}}(y_{\text{fast}}^t)$ and $\Theta_{\text{slow}} = y_{\text{slow}}^t = \text{RS}_{\text{slow}}(\Lambda, x, t)$, the loss for any model $i \in \{\text{fast}, \text{slow}\}$ is:

$$\mathcal{L}(\Theta_i) = \begin{cases} 0 & \Theta_i \cap U = \emptyset \\ \infty & \text{otherwise.} \end{cases} \quad (15)$$

Finally, using Equation 14 and Theorem 1, the model selection policy that solves Problem 1 for safety assessment during robot navigation is:

$$\pi^*(y_{\text{fast}}^t) = \mathbb{1}\left(B_{\frac{1}{1-\epsilon}}(y_{\text{fast}}^t) \cap U \neq \emptyset\right). \quad (16)$$

Intuitively, the above policy exploits the relationship between fast and slow models by first bloating the fast model's reachable set by a factor of $\frac{1}{1-\epsilon}$ to create a guaranteed over-approximation of the slow model's reachability computation.

If the over-approximation does not intersect obstacles, we are guaranteed safety and simply proceed. If not, we need to invoke the slow model to assess its higher-fidelity reachable set and re-plan a trajectory if it indicates unsafety. While we implemented our policy with $\alpha = 0.7, \beta = 0.3$ to prioritize safety, safety is also heavily emphasized in the loss function (Eq. 15) since the penalty is ∞ for collisions.

D. Benchmark Algorithm

We evaluate the performance of our model selection policy against the following benchmark policies:

Fast: This policy always uses the fast model with prediction y_{fast}^t for all t.

Slow: This policy always uses the slow model with prediction y_{slow}^t for all t.

Random: The robot randomly chooses between the fast and slow model with equal probability.

Our Selector: This represents our model selection policy from Equations 6, 12, and 16.

Oracle: This strategy assumes that the slow model's output is available to the model selection function at the time of inference. Thus, this strategy only selects the slow model when that decision has a better reward than using the fast model. The oracle is an upper-bound, unrealizable strategy since it assumes privileged knowledge of the slow model.

V. EVALUATION

The principal objective of our evaluation is to show that our model selection policies from Lemma 1 and Equations 6, 12, and 16 achieve a significantly higher reward than benchmark model selection policies. Further, we show how our policy achieves better accuracy with a lower cost than competing benchmarks on simulations of linear regression, aircraft taxiing with state-of-the-art DNN perception models, and rover navigation with real Martian terrain data. All our code (in Python) and models will be made publicly available at [21] after the peer-review process.

A. Linear Regression Results

We now evaluate our selection policy for linear regression, as described in Equation 6 and Subsection IV-A. The key highlight is that our policy achieves 245.4% higher reward than benchmarks in 100 trials, each of duration $N = 10^5$ timesteps with stochastic Gaussian inputs x^t. Figures 2 (Left) and 3 (Left) show the cumulative rewards and trade-off between accuracy and cost, respectively, of all algorithms.

B. Deep Neural Networks (DNN)

We now evaluate our model selection policy for the TaxiNet aircraft taxiing scenario from Subsection IV-B. Our key result on 18,372 *test* images is shown in Figure 2 (Center), where our policy (**Our Selector**) achieves 22.22% higher reward than competing benchmarks and is within 10.18% the performance of an upper-bound **Oracle**. Moreover, Figure 3 (Center) shows that our model selection policy achieves low loss with low cost unlike competing policies. This is because our policy leverages the statistical correlation between models to mostly rely on the fast model to reduce cost,
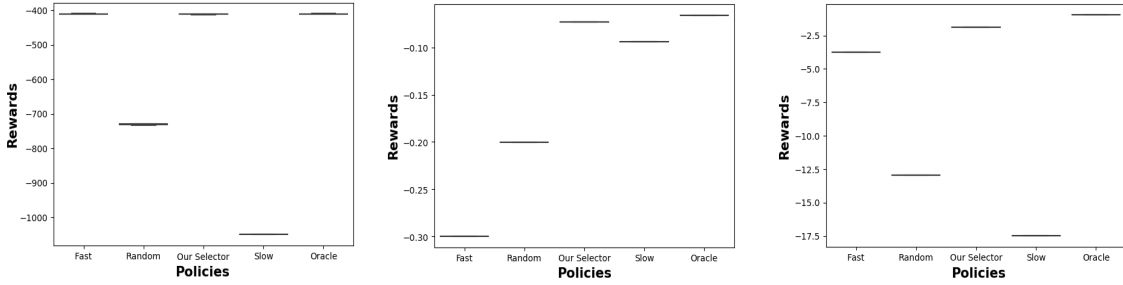
Fig. 2: **Rewards.** (*Left: Linear Regression, Center: DNN, Right: Mars Rover Reachable Set*). We illustrate the cumulative rewards (Eq. 1) gathered by various policies on the Linear Regression, DNN perception (TaxiNet), and Mars Reachable Set scenarios, respectively. Clearly, our policy (`Our Selector`) achieves the maximum reward compared to other realizable benchmarks and is close to the oracle in all cases.
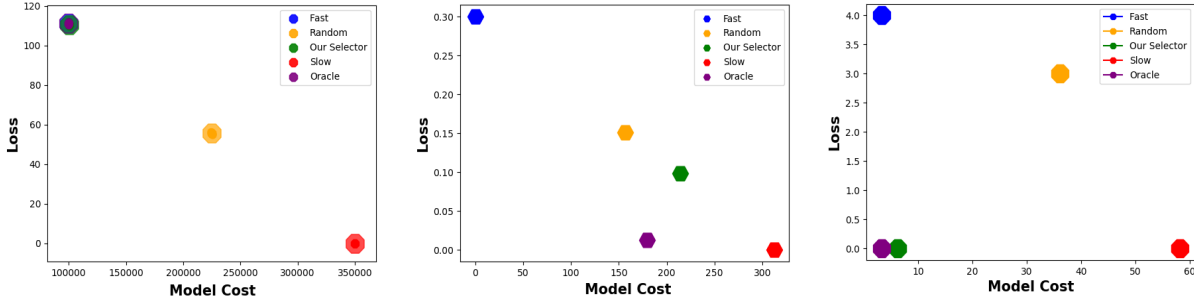

Fig. 3: **DNN Cost vs. Accuracy.** (*Left: Linear Regression, Center: DNN, Right: Mars Rover Reachable Set*). Cost vs. Loss trade-off achieved by various model selection policies on all scenarios. In all cases, we observe that the `Fast` policy has low cost but low accuracy, `Slow` has high accuracy but high cost, and `Random` lies sub-optimally in the middle (with a high variance). Only the selection policy proposed in this paper (`Our Selector`) achieves a delicate balance by exploiting the statistical relationship between models to intelligently consult the slow model.
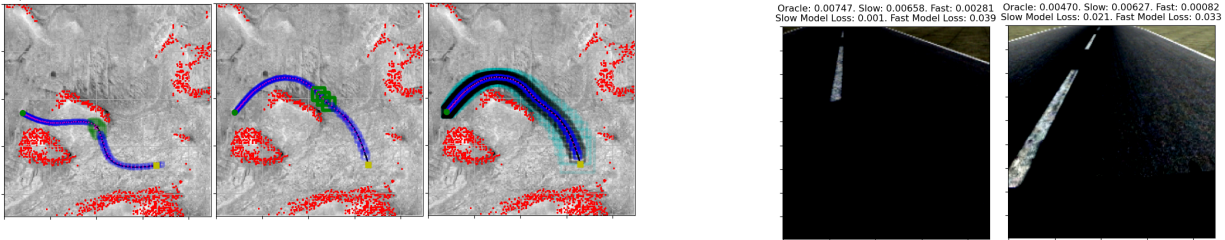

Fig. 4: **Left: (Safe Navigation of a Mars Rover).** We consider navigation of a simulated rover on real Mars HiRise terrain data [22], where the red point clouds are obstacles indicating regions of high elevation, as processed in [23]. *First two images:* We show the reachable sets of two different possible routes taken by the Mars Rover. The reachable set in green is computed by the slow model, whereas the one in blue comes from the fast model. Path safety is determined by assessing if the reachable set (accounting for uncertainties), intersects red obstacles. Clearly, our model selection policy uses the slow model only when the rover makes tricky maneuvers. Otherwise, when the rover is far from obstacles, the fast model is sufficient to determine safety, allowing us to maintain high safety with a lower compute time. *Third image:* We show the relationship beween reachable sets computed by the fast model (in blue) and the slow model (in black). Crucially, our policy uses the over-approximated reachable set of the slow model as computed by the fast model (in cyan), that is, $B_{\frac{1}{1-\epsilon}}(y^t_{\text{fast}})$. Clearly, our model selection policy only consults the slow model when it suspects a possible collision when the set represented in cyan intersects with a red obstacle. Our policy always led to safe, collision-free, efficient navigation by exploiting the relationship between fast and slow models. **Right: (Aircraft TaxiNet DNN Output).** The output of the fast and slow DNN models for a ResNet-18 TaxiNet model. Given an image, the final output shown is the rudder control.

but also opportunistically queries the slow model for higher accuracy. However, our policy is careful to only invoke the slow, accurate model when there is a substantial accuracy gain, leading it to be queried only 68.6% of the time.

C. Reachable Set Computation

We now demonstrate the performance of our model selection policy (Equation 16) to determine the safety of a simulated Mars Rover navigating steep obstacles on terrain from NASA's HiRise Dataset [22], [23]. A low-power rover must always be safe, but also fast and compute-and-power-efficient while accounting for reachable sets while planning.

The rover is assumed to follow a linearized bicycle model with bounded perturbations in the dynamics matrix for yaw angle. Given an intended path, we use our model selection policy (Equation 16) to determine safety given uncertain dynamics while minimizing compute time. Specifically, given a start set, desired goal, and a set of way-points, we compute a reference trajectory using a cubic spline planner, which is followed using Model Predictive Control (MPC). Using the planned states x and controls u at every time, our model selection policy must determine the trajectory's safety by invoking either a fast or slow reachable set computation model as described in Subsection IV-C.

Figure 4 (Left, first two images) shows how our policy (Equation 16) safely, but efficiently, follows two different paths near a red obstacle indicating an unsafe terrain gradient above 20 degrees. The key benefit of our approach is that the robot mostly uses the fast reachable set computation (blue) for high-efficiency and only *intelligently* consults the higher-fidelity slower model during tricky turns close to an obstacle. Indeed, Figure 4 (Left, third image) precisely shows how our policy (Equation 16) exploits the relationship between fast and slow models to selectively query the slow model only when required during key turns. The fast model's reachable set result is in blue, the slow model's result is in black, and the *over-approximation* from bloating the fast model's result by $B_{\frac{1}{1-\epsilon}}(y_{\text{fast}}^t)$ is in cyan.

Clearly, even the over-approximation rarely intersects unsafe obstacles and it is only necessary to consult a fine-grained result from the slow model (black) when the over-approximation is too conservative and needs to be refined. In all scenarios, we rigorously verified the simulated rover is safe and never hits an obstacle despite dynamics uncertainties. Figures 2 (Right) and 3 (Right) quantitatively illustrates the superior efficiency and accuracy (safety) of our policy, since it achieves the highest reward, never hits an obstacle, and efficiently only queries the slow model on-demand near critical obstacles.

Limitations of Our Work: In the future, we plan to account for more sophisticated nonlinear dynamics using Hamilton-Jacobi-Bellman reachability analysis. Finally, future work should address multi-step decision-making, where model selection decisions affect subsequent measurements and control decisions.

VI. CONCLUSION

To scale the deployment of low-power robotic swarms, it is increasingly important to optimize for compute energy, cost, and latency alongside standard metrics of task accuracy and resiliency. This paper presents a general algorithm for robots to flexibly trade-off task accuracy and compute cost in an interpretable manner with provable statistical guarantees. Our key insight is to leverage the statistical correlations between models to *predict* the marginal accuracy gain of a large model and balance it with additional compute costs. This general principle allows our framework to widely apply to cloud robotics, DNN perception, and reachability analysis.

In the future, we plan to address safety guarantees and investigate whether we can co-train large and small DNNs such that we can synthesize an interpretable run-time monitor that can transfer authority to a trusted controller if the DNNs are operating in uncertain regimes. Overall, we anticipate our model-selection results will become stronger with future advances in DNN verification and compression with approximation guarantees.

REFERENCES

[1] 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," in *Proceedings of Robotics: Science and Systems*, Freiburg im Breisgau, Germany, June 2019.

[2] A. Rahman, J. Jin, A. Cricenti, A. Rahman, and M. Panda, "Motion and connectivity aware offloading in cloud robotics via genetic algorithm," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 1–6.

[3] M. Cakmak and A. L. Thomaz, "Designing robot learners that ask good questions," in *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2012, pp. 17–24.

[4] D. Whitney, E. Rosen, J. MacGlashan, L. L. Wong, and S. Tellex, "Reducing errors in object-fetching interactions through social feedback," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1006–1013.

[5] K. N. Kaipa, A. S. Kankanhalli-Nagendra, N. B. Kumbla, S. Shriyam, S. S. Thevendria-Karthic, J. A. Marvel, and S. K. Gupta, "Enhancing robotic unstructured bin-picking performance by enabling remote human interventions in challenging perception scenarios," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 2016, pp. 639–645.

[6] A. E. Eshratifar and M. Pedram, "Runtime deep model multiplexing for reduced latency and energy consumption inference," in *2020 IEEE 38th International Conference on Computer Design (ICCD)*. IEEE, 2020, pp. 263–270.

[7] N. Dorka, J. Meyer, and W. Burgard, "Modality-buffet for real-time object detection," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 10 543–10 549.

[8] T. Q. Dinh, Q. D. La, T. Q. Quek, and H. Shin, "Learning for computation offloading in mobile edge computing," *IEEE Transactions on Communications*, vol. 66, no. 12, pp. 6353–6367, 2018.

[9] C. Baykal, L. Liebenwein, I. Gilitschenski, D. Feldman, and D. Rus, "Data-dependent coresets for compressing neural networks with applications to generalization bounds," in *International Conference on Learning Representations*, 2019.

[10] L. Liebenwein, C. Baykal, H. Lang, D. Feldman, and D. Rus, "Provable filter pruning for efficient neural networks," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[11] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, ser. Proceedings of Machine Learning Research, vol. 97. PMLR, 2019, pp. 6105–6114.

[12] C. Boutsidis, P. Drineas, and M. Magdon-Ismail, "Near-optimal coresets for least-squares regression," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6880–6892, 2013.

[13] K. D. Julian, R. Lee, and M. J. Kochenderfer, "Validation of image-based neural network controllers through adaptive stress testing," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–7.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[15] "X-plane 11," https://www.x-plane.com/.

[16] S. M. Katz, A. Corso, S. Chinchali, A. Elhafsi, A. Sharma, M. Pavone, and M. J. Kochenderfer, "Nasa uli aircraft taxi dataset," 2021, stanford Research Data, https://purl.stanford.edu/zz143mb4347.

[17] M. Althoff, C. Guernic, and B. Krogh, "Reachable set computation for uncertain time-varying linear systems," 01 2011, pp. 93–102.

[18] R. Lal and P. Prabhakar, "Bounded error flowpipe computation of parameterized linear systems," in *2015 International Conference on Embedded Software (EMSOFT)*, 2015, pp. 237–246.

[19] B. Ghosh and P. S. Duggirala, "Robust reachable set: Accounting for uncertainties in linear dynamical systems," *ACM Trans. Embed. Comput. Syst.*, vol. 18, no. 5s, Oct. 2019.

[20] B. Ghosh and P. S. Duggirala, "Reachability of linear uncertain systems: Sampling based approaches," University of North Carolina at Chapel Hill, Tech. Rep., 2020. [Online]. Available: https://drive.google.com/file/d/18qhQh1rZZEMRzjhPCknRsdUy567sqXRZ/view?usp=sharing

[21] B. Ghosh, S. Chinchali, and P. S. Duggirala, "Interpretable trade-offs between robot task accuracy and compute efficiency," University of North Carolina at Chapel Hill, Tech. Rep., 2021. [Online]. Available: https://sites.google.com/view/modelselection

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

[23] M. Nakanoya, S. Chinchali, A. Anemogiannis, A. Datta, S. Katti, and M. Pavone, "Task-relevant representation learning for networked robotic perception," *CoRR*, vol. abs/2011.03216, 2020.