A Fog Robotics Approach to Deep Robot Learning: Application to Object Recognition and Grasp Planning in Surface Decluttering

Ajay Kumar Tanwani, Nitesh Mor, John Kubiatowicz, Joseph E. Gonzalez, Ken Goldberg

***Abstract*— The growing demand of industrial, automotive and service robots presents a challenge to the centralized Cloud Robotics model in terms of privacy, security, latency, bandwidth, and reliability. In this paper, we present a 'Fog Robotics' approach to deep robot learning that distributes compute, storage and networking resources between the Cloud and the Edge in a federated manner. Deep models are trained on non-private (public) synthetic images in the Cloud; the models are adapted to the private real images of the environment at the Edge within a trusted network and subsequently, deployed as a service for low-latency and secure inference/prediction for other robots in the network. We apply this approach to surface decluttering, where a mobile robot picks and sorts objects from a cluttered floor by learning a deep object recognition and a grasp planning model. Experiments suggest that Fog Robotics can improve performance by sim-to-real domain adaptation in comparison to exclusively using Cloud or Edge resources, while reducing the inference cycle time by $4\times$ to successfully declutter 86% of objects over 213 attempts.**

I. INTRODUCTION

The term 'Cloud Robotics' describes robots or automation systems that rely on either data or code from the Cloud, i.e. where not all sensing, computation, and memory is integrated into a single standalone system [1], [2]. By moving the computational and storage resources to the remote datacenters, Cloud Robotics facilitates sharing of data across applications and users, while reducing the size and the cost of the onboard hardware. Examples of Cloud Robotics platforms include RoboEarth [3], KnowRob [4], RoboBrain [5], DexNet as a Service [6], [7]. Recently, Amazon RoboMaker [8] and Google Cloud Robotics [9] released platforms to develop robotic applications in simulation with their Cloud services.

Robots are increasingly linked to the network and thus not limited by onboard resources for computation, memory, or software. Internet of Things (IoT) applications and the volume of sensory data continues to increase, leading to a higher latency, variable timing, limited bandwidth access than deemed feasible for modern robotics applications [10], [11]. Moreover, stability issues arise in handling environmental uncertainty with any loss in network connectivity. Another important factor is the security of the data sent and received from heterogeneous sources over the Internet. The correctness and reliability of information has direct impact on the performance of robots. Robots often collect sensitive information (e.g., images of home, proprietary warehouse and manufacturing data) that needs to be protected. As an example, a number of sensors and actuators using Robot Operating System (ROS) have been exposed to public access and control over the Internet [12].

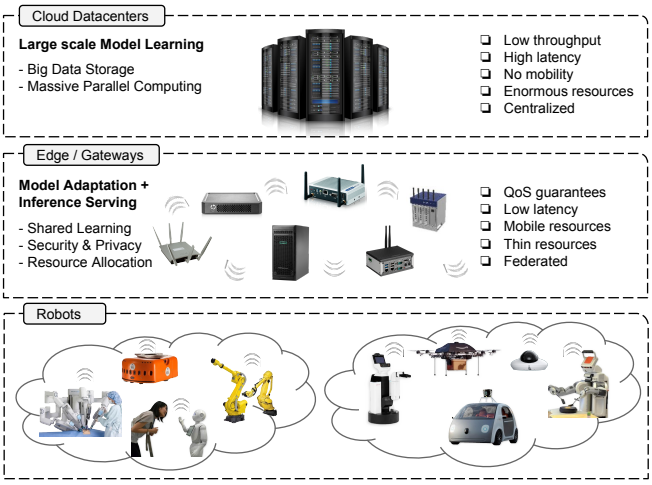

Fig. 1: A Fog Robotics approach to deep robot learning that uses resources between Cloud and Edge for training, adaptation, inference serving and updating of deep models to reduce latency and preserve privacy of the data.

Fog Robotics is *"an extension of Cloud Robotics that distributes storage, compute and networking resources between the Cloud and the Edge in a federated manner"*. The term Fog Robotics (analogous to Fog Computing[1] [16], [17], [18]) was first used by Gudi et al. [19]. In this paper, we apply Fog Robotics for robot learning and inference of deep neural networks such as object recognition, grasp planning, localization etc. over wireless networks. We address the system level challenges of network limits (high latency, limited bandwidth, variability of connectivity, etc.), security and privacy of data and infrastructure, along with resource allocation and model placement issues. Fog Robotics provides flexibility in addressing these challenges by: 1) sharing of data and distributed learning with the use of resources in close proximity instead of exclusively relying on Cloud resources, 2) security and privacy of data by restricting its access within a trusted infrastructure, and 3) resource allocation for load balancing between the Cloud and the Edge (see Fig. 1 for an overview and Sec. II for details). Shared learning reduces the burden of collecting massive training data for each robot in training deep models, while the models are personalized for each robot at the Edge of the network within a trusted infrastructure. Deploying the deep models at

The AUTOLAB at UC Berkeley (automation.berkeley.edu).
University of California, Berkeley. {ajay.tanwani, mor, kubitron, jegonzal, goldberg}@berkeley.edu

[1]The term "Fog Computing" was introduced by Cisco Systems in 2012 [13]. Other closely related concepts to Fog Computing are Cloudlets [14] and Mobile Edge Computing [15].

the Edge enables prediction serving at a low-latency of less than 100 milliseconds.

These principles are useful to efficiently train, adapt and deploy massive deep learning models by simulation to reality transfer across a fleet of robots. Surface decluttering is a promising application of service robots in a broad variety of unstructured environments such as home, office and machine shops. Some related examples include cloud-based robot grasping [20], grasping and manipulation in home environments [21], robotic butler with HERB [22] and PR2 [23], combining grasping and pushing primitives in decluttering lego blocks with PR2 [24], and robot decluttering in unstructured home environments with low cost robots [25]. In this work, we consider decluttering scenarios where a robot learns to pick common machine shop and household objects from the floor, and place them into desired bins. We learn deep object recognition and grasp planning models from synthetic images in the Cloud, adapt the model to the real images of the robot within a trusted infrastructure at the Edge, and subsequently deploy models for low-latency serving in surface decluttering.

A. Contributions

This paper makes four contributions:

1) Motivates and introduces Fog Robotics in the context of deep robot learning.
2) Presents a deep learning based surface decluttering application and demonstrates the use of Fog Robotics compared to the alternatives of exclusive Cloud or exclusive local resources.
3) Presents a domain invariant deep object recognition and grasping model by simulation to real transfer, and evaluates benchmarks for learning and inference with a mobile robot over a wireless network.
4) Surface decluttering experiments with a mobile Toyota HSR robot to grasp 185 household and machine shop objects over 213 grasp attempts.

II. Fog Robotics

While the Cloud can be viewed as a practically infinite pool of homogeneous resources in far away data centers, the Edge of the network is characterized by a limited collection of heterogeneous resources owned by various administrative entities. Resources at the Edge come in various sizes, e.g. content delivery networks, light-weight micro servers, networking devices such as gateways, routers, switches and access points. Fog Robotics explores a continuum between on-board resources on a robot to far away resources in Cloud data centers. The goal is to use the available resources, both at the Edge and in the Cloud, to satisfy the service level objectives including, but not limited to, latency, bandwidth, reliability and privacy. By harnessing the resources close by and not relying exclusively on the Cloud, Fog Robotics provides opportunities such as richer communication among robots for coordination and shared learning, better control over privacy of sensitive data with the use of locally provisioned resources, and flexible allocation of resources based on variability of workload.

Related Work: Hong et al. proposed 'Mobile Fog' to distribute IoT applications from Edge devices to the Cloud in a hierarchical manner [26]. Aazam and Huh [27] presented a resource allocation model for Fog Computing. Bonomi et al. made provision for resource constrained IoT devices in their Fog Computing platform [28]. In [29], the authors propose a framework to minimize service delays in Fog applications by load sharing. The authors in [30] use a multi-tier Fog and Cloud computing approach for a pervasive brain monitoring system that can reliably estimate brain states and adapt to track users' brain dynamics. Lee et al. in [31] and Alrawais et al. in [32] discuss the security and privacy issues and present solutions for mitigating the security threats. More details of Fog computing are in [33], [34]. Recently, several groups have also advocated the need for Fog Robotics. Katterpur et al. profile the computation times for resource allocation in a fog network of robots [35]. Gudi et al. present a Fog Robotics approach for human robot interaction [36]. Pop et al. discuss the role of Fog computing in industrial automation via time-sensitive networking [37]. For more details and updates, see [38], [39], [40].

As an example, a number of battery powered WiFi-enabled mobile robots for surface decluttering can use resources from a close-by fixed infrastructure, such as a relatively powerful smart home gateway, while relying on far away Cloud resources for non-critical tasks. Similar deployments of robots with a fixed infrastructure can be envisioned for industrial warehouses, self-driving cars, flying drones, socially aware cobots and so on. Below, we review the opportunities that Fog Robotics provides for secure and distributed robot learning:

A. Enabling Shared and Distributed Learning

Fog Robotics brings computational resources closer to mobile robots that enables access to more data via different sensors on a robot or across multiple robots. Whereas Cloud Robotics assumes the Cloud as a centralized rendezvous point of all information exchange, Fog Robotics enables new communication modalities among robots by finding other optimal paths over the network. Using a Cloud-only approach is inefficient in utilizing the network bandwidth and limits the volume of data that can be shared.

Fog Robotics enables computational resources closer to the robots to perform pre-processing, filtering, deep learning, inference, and caching of data to reduce reliance on far away data centers. For example, to support household robots, models trained in the Cloud can be periodically pushed to a smart home gateway instead of directly onto individual robots; such a smart home gateway can act as a cache of local model repository, perform adaptation of a generalized model to the specific household, provide storage of data collected from the household for model adaptation, or even run a shared inference service for local robots to support robots with very limited onboard resources. We demonstrate such

an inference service in the context of the surface decluttering application.

On a broader scale, resources at a municipal level allow for similar benefits at a geographical level. Such computational resources outside data centers are not merely a vision for the future; they already exist as a part of various projects such as EdgeX Foundry [41], CloudLab [42], EdgeNet [43], US Ignite [44], PlanetLab [45], PlanetLab Europe [46], GENI [47], G-Lab [48], among others.

B. Security, Privacy, and Control Over Data

Network connected systems significantly increase the attack surface when compared to standalone infrastructure. Deliberate disruption to wide-area communication (e.g., by targeted Denial of Service (DoS) attacks) is not uncommon [49]. The control of data collected by robots and the security of data received from a remote service is a major concern. To this end, a well designed Fog Robotics application can provide a tunable middle ground of reliability, security and privacy between a 'no information sharing' approach of standalone isolated deployments and a 'share everything' approach of Cloud Robotics.

Such control over data, however, is non-trivial as resources at the Edge are partitioned in a number of administrative domains based on resource ownership. Heterogeneity of resources further adds to the security challenge; just keeping various software to the most up-to-date versions is cumbersome. Note that merely encrypting data may not be sufficient. As an example, simple encryption only provides data confidentiality but not data integrity—a clever adversary can make a robot operate on tampered data [50]. Moreover, addressing key-management—an integral part of cryptographic solutions—is a challenge in itself [51]. Finally, managing the security of 'live' data that evolves over time is more challenging than that of a static dump.

Data-centric infrastuctures such as Global Data Plane (GDP) [52] can provide a scalable alternative to control the placement and scope of data while providing verifiable security guarantees. GDP uses cryptographically secured data containers called DataCapsules. DataCapsules are analogous to shipping containers that provide certain guarantees on data integrity and confidentiality even when they are handled by various parties during their lifetimes. The integrity and provenance of information in a DataCapsule can be verified by means of small cryptographic proofs [53]. This allows the owners of data to restrict sensitive information in a DataCapsule to, say, a home or a warehouse. In contrast, existing Cloud storage systems (say Amazon S3) do not provide provable security guarantees and rely solely on the reputation of the Cloud provider to protect the Cloud infrastructure from adversarial infiltration. Similarly, decentralized authorization systems such as WAVE [54] can protect the secrecy of data without relying on any central trusted parties. Note that secure execution of data still remains an open challenge. A wider deployment of secure hardware such as Intel's SGX (Software Guard Extensions) technology [55] has the potential to provide for an end-to-end security.

While it is important from an infrastructure viewpoint to maintain control over sensitive data and ensure that it does not leave the boundaries of infrastructure with known security properties, applications also need to be designed around such constraints. We demonstrate such an architecture for privacy preserving Fog Robotics scenario by using synthetic non-private data for training in the Cloud and use real-world private data only for local refinement of models.

C. Flexibility of Resource Placement and Allocation

The Cloud provides seemingly infinite resources for compute and storage, whereas resources at the Edge of the network are limited. Quality of service provisioning depends upon a number of factors such as communication latency, energy constraints, durability, size of the data, model placement over Cloud and/or Edge, computation times for learning and inference of the deep models, etc. This has motivated several models for appropriate resource allocation and service provisioning [34]. Chinchali et al. use a deep reinforcement learning strategy to offload robot sensing tasks over the network [56]. Nan et al. present a fog robotic system for dynamic visual servoing with an ayschronous heartbeat signal [57].

Flexibility in placement and usage of resources can give a better overall system design, e.g. offloading computation from the robot not only enables lower unit cost for individual robots but also makes it possible to have longer battery life. Consider, for example, a resource constrained network where GPUs are available on the Cloud and only CPUs are available at the Edge of the network. Even though a GPU provides superior computation capabilities compared to a CPU, the round-trip communication time of using a GPU in the Cloud–coupled with communication latency–vs a CPU locally is application and workload dependent. Note that the capital and operational expense for a CPU is far lower than that of a GPU. Simple application profiling may be used for resource placement in this context [35]. However, finding an appropriate balance for performance and cost is challenging when the application demands and the availability of resources keeps changing over time, making continuous re-evaluation necessary [58].

III. Deep Learning based Surface Decluttering

A. Problem Statement

We consider a mobile robot equipped with a robotic arm and a two-fingered gripper as the end-effector. The robot observes the state of the floor $\boldsymbol{\xi}_t$ as a RGB image $\boldsymbol{I}_t^c \in \mathbb{R}^{640\times480\times3}$ and a depth image $\boldsymbol{I}_t^d \in \mathbb{R}^{640\times480}$. The task of the robot is to recognize the objects $\{o_i\}_{i=1}^N$ as belonging to the object categories $o_i \in \{1 \dots C\}$, and subsequently plan a grasp action $\boldsymbol{u}_t \in \mathbb{R}^4$ corresponding to the 3D object position and the planar orientation of the most likely recognized object. After grasping an object, the robot places the object into appropriate bins (see Fig. 2 for an overview).

In this paper, we learn a deep object recognition and a grasp planning model for surface decluttering with a mobile robot. The object recognition model predicts the bounding

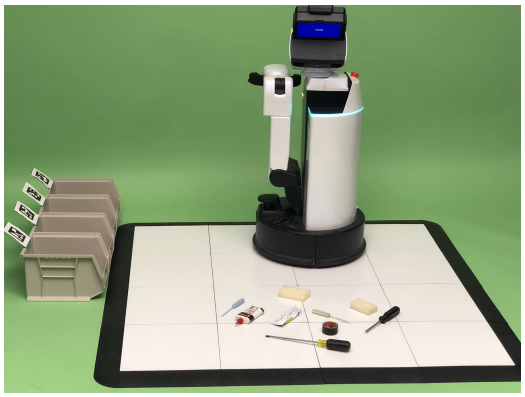

Fig. 2: Experimental setup for decluttering objects into bins with HSR.

Fig. 3: Simulation object models on *(left)* and physical objects on *(right)* for decluttering.

boxes of the objects from the RGB image, while the grasp planning model predicts the optimal grasp action from the depth image. We compare the grasp planning approach with a baseline that grasps orthogonal to the centroid of the principal axis of the isolated segmented objects, and uses the depth image to find the height of the object centroid. We are interested in offloading the training and deployment of these models by simulation to reality transfer with Fog Robotics. The deep models are trained with synthetic images in the Cloud, adapted at the Edge with real images of physical objects and then deployed for inference serving to the mobile robot over a wireless network.

B. Simulation and Real Dataset

We simulate the decluttering environment in a Pybullet simulator [59]. We collect 770 3D object meshes from TurboSquid, KIT, ShapeNet and DexNet resembling household and machine shop environments, and split them across 12 categories: screwdriver, wrench, fruit, cup, bottle, assembly part, hammer, scissors, tape, toy, tube, and utility. Camera parameters and viewpoint in the simulator is set according to the real robot facing the floor as shown in Fig. 2. We randomly drop between $5-25$ objects on the floor from a varying height of $0.2-0.7$ meters, each assigned a random color from a set of 8 predefined colors. The objects are allowed to settle down before taking the RGB and the depth image and recording the object labels. We generated $20K$ synthetic images of cluttered objects on the floor following this process.

The physical dataset includes 102 commonly used household and machine shop objects split across 12 class categories as above (see Fig. 3). We randomly load $5-25$ objects in a smaller bin without replacement and drop them on 1.2 sq. meter white tiled floor from different positions. We collected 212 RGB and depth camera images with an average number of 15.4 objects per image, and hand label the bounding box and the image categories.

C. Transfer Learning from Simulation to Reality

We train the deep object recognition model on simulated data and adapt the learned model on real data such that the feature representations of the model are invariant across the simulator and the real images [60]. The learning problem considers the synthetic images as belonging to a non-private simulated domain D_S, and real images belonging to a private real domain D_R that is not to be shared with other networks. The simulated and real domain consists of tuples of the form $D_S = \{\boldsymbol{\xi}_t^{(s)}, \boldsymbol{u}_t^{(s)}, \boldsymbol{y}_t^{(s)}\}_{t=1}^{T_S}$ and $D_R = \{\boldsymbol{\xi}_t^{(r)}, \boldsymbol{u}_t^{(r)}, \boldsymbol{y}_t^{(r)}\}_{t=1}^{T_R}$, where $\boldsymbol{y}_t^{(s)}$ and $\boldsymbol{y}_t^{(r)}$ correspond to a sequence of bounding boxes of object categories as ground-truth labels for a simulated image and a real image respectively, and $T_S \gg T_R$. The real images and the synthetic images may correspond to different but related randomized environments such as a machine shop and a household environment. For example, we randomize the colors of the 3D object models in the simulated domain, but real world objects have a fixed texture.

We use the MobileNet-Single Shot MultiBox Detector (SSD) [61], [62] algorithm with focal loss and feature pyramids as the base model for object recognition (other well-known models include YOLO, Faster R-CNN; see [63] for an overview). We modify the base model such that the output of feature representation layer of the model is invariant to the domain of the image, i.e., $\boldsymbol{\xi}_t \sim D_S \approx \boldsymbol{\xi}_t \sim D_R$, while minimizing the classification loss $\mathcal{L}_{y_c}$ and the localization loss $\mathcal{L}_{y_l}$ of the model. We add an adversarial discriminator at the output of the feature representation layer that predicts the domain of the image as synthetic or real $\boldsymbol{\xi}_t \in \{D_S, D_R\}$. The overall model parameters are optimized such that the the object classification loss $\mathcal{L}_{y_c}$ and the localization loss $\mathcal{L}_{y_l}$ is minimized, while the domain classifier loss $\mathcal{L}_d$ is maximally confused in predicting the domain of the image [64], [60], [65]. The trade-off between the loss functions governs the invariance of the model to the domain of the image and the output accuracy of the model.

We denote the augmented model as the *domain invariant object recognition* model (DIOR). The domain classification architecture is empirically selected to give better performance with 3 fully connected layers of $1024, 200$ and 100 neurons after flattening the output of logits layer. We implement two variations of DIOR: 1) **DIOR_dann** that shares parameters of the feature representation for both the sim and the real domain [60], and 2) **DIOR_adda** that has separate feature representation layers for the sim and the real domain to allow different feature maps. The parameters of the sim network are pretrained and fixed during the adaptation of the real

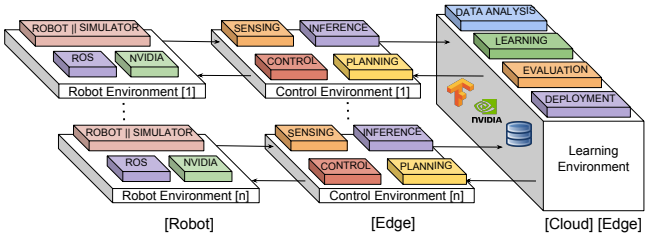

Fig. 4: Software components running in network-connected execution environments packaged and distributed via Docker images: *(left)* robot environment, *(centre)* control environment, *(right)* learning environment.

images in this variant [65].

The cropped depth image from the output bounding box of the object recognition model is fed as input to the DexNet grasp planning model [66]. The model is retrained on synthetic depth images taken as seen from the tilted head camera of the robot. Depth images are readily invariant to the simulator and the real environment. The grasp planning model samples antipodal grasps on the cropped depth image of the object and outputs the top ranked grasp for the robot to pick and place the object into its corresponding bin.

D. Networked System with Execution Environments

The overall networked system consists of three modular execution environments (see Fig. 4): 1) the robot environment or its digital twin [67] in the simulator that sends images of the environment and receives actuation commands to drive the robot; 2) the control environment responsible for sensing the images, *inferring* the objects and grasp poses from the images using the trained object recognition and grasp planning model, planning the motion of the robot for executing the grasps, and sending the actuation commands to drive the robot; and 3) the learning environment that receives images and labels from the robot or the simulator and splits the data for training and evaluation of the deep models. At the end of the training process, the best performing model on the evaluation set is deployed as an *inference graph* for secured and low-latency prediction serving at the Edge in our *robot-learning-as-a-service* platform. Note that the robot environment, the control environment and the learning environment are virtual, and their appropriate placement depends on the available storage and compute resources in the network. The software components running in network-connected execution environments are packaged and distributed via Docker images [68].

We run an instance of the learning environment to train the deep object recognition model on the Cloud with the non-private synthetic data only, while another instance runs at the Edge of the network that adapts the trained network on real data to extract invariant feature representations from the private (real) and the non-private (synthetic) data.

IV. Experiments, Results and Discussion

We now present comparative experiments of deep learning and inference for surface decluttering using: 1) Cloud resources only, 2) Edge resources only, and 3) Fog using resources on both Cloud and Edge. The Edge infrastructure includes a workstation (6-core, 2-thread Intel CPUs, 1.1 TB Hard Disk, with a Titan XP GPU) located in UC Berkeley for Edge computing and storage. We use the Amazon EC2 $\texttt{p2.8xlarge}$ instance with 8 Tesla K80 GPUs for Cloud compute and use Amazon S3 buckets for Cloud storage. We launch the EC2 instance in two regions: 1) **EC2 (West)** in Oregon (us-west-2), and 2) **EC2 (East)** in Northern Virginia (us-east-1).

A. Sim-to-Real Domain Adaptation over the Network

We divide both the simulated and the real datasets into 60% training and 40% evaluation sets: $\{\text{sim\_train}, \text{real\_train}, \text{sim\_eval}, \text{real\_eval}\}$, and estimate the model parameters described in Sec. III-C under different networks on real_eval: 1) training in the **Cloud** with only large scale non-private synthetic images $\{\text{sim\_train}\}$, 2) training at the **Edge** with only limited number of private real images $\{\text{real\_train}\}$, and 3) training in the **Fog** with both synthetic and real images on the Edge, using a pretrained model on large scale synthetic data in Cloud, under 3 baselines: a) **Sim+Real:** training on combined simulation and real data with no domain classifier, b) **DIOR_dann:** training DIOR with shared parameters for sim and real feature representation, c) **DIOR_adda:** training DIOR with separate parameters for sim and real feature representations.

Results are summarized in Table I. We observe that the models give comparable or better mean average precision (mAP) [69] and classification accuracy on the real images in the Fog in comparison to the models trained exclusively on Cloud or Edge. Naive transfer of model trained on synthetic data does not perform well on the real data with an accuracy of 24.16%. Combining sim and real data naively is also sub-optimal. The domain invariant object recognition with a few labeled real images provides a trade-off between acquiring a generalized representation versus an accurate adaptation to the real images. The **DIOR_adda** drastically improves the performance on real domain by partially aligning the feature representation with the sim domain. The **DIOR_dann** model with shared parameters gives good performance in both domains, which can further be used to update the simulator model in the Cloud [70]. We report the remainder of the results with **DIOR_dann**. Training time of each model is over 13 hours on both the Cloud and the Edge(GPU) instances suggesting that the model placement issues are less critical for training.

The cropped depth image of the closest object to the robot is fed to the grasp planning model to compute the grasp poses for robot surface decluttering (see Fig. 5 for qualitative results of the model on both synthetic and real data).

B. Communication vs Computation Cost for Inference

We deployed the trained models in the robot-learning-as-a-service platform that receives images from the robot as a client, performs inference on a server, and sends back the result to the robot. We measure the round-trip time $t^{(\text{rtt})}$, i.e., time required for communication to/from the server and

TABLE I: Comparative experiments for learning deep object recognition for simulation to reality transfer over Cloud, Edge and Fog. Metrics include mean Average Precision (mAP) on real images, classification accuracy on synthetic test images sim_eval, real test images real_eval and both synthetic and real test images mix_eval. Domain invariant object recognition with shared feature representation network parameters DIOR_dann model gives better performance in both simulation and real domain using Fog Robotics.

Training Set	mAP	sim_eval	real_eval	mix_eval
Cloud				
Sim	0.13	97.97	24.16	55.5
Edge				
Real	0.62	24.64	88.1	64.92
Fog				
Sim + Real	0.33	90.40	54.12	69.97
DIOR_dann	0.61	96.92	86.33	**95.21**
DIOR_adda	0.61	30.87	**90.64**	67.82

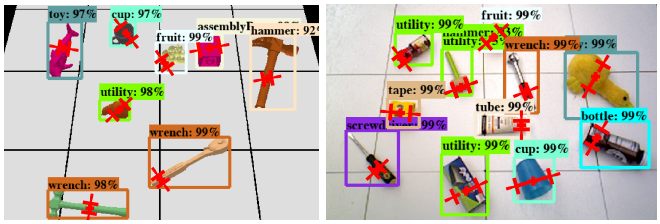

Fig. 5: Object recognition and grasp planning model output on a simulated image on *(left)* and real image on *(right)* as seen from the robot head camera.

the inference time $t^{(\text{inf})}$. We experiment with four hosts for the inference service in the order of decreasing distance to the robot: EC2 Cloud (West), EC2 Cloud (East), Edge with CPU support only, and Edge with GPU support.

Results in Table II show that the communication and not the computation time is the major component in overall cost. Deploying the inference service on the Edge significantly reduces the round-trip inference time and the timing variability in comparison to hosting the service on Cloud, with nearly $4\times$ difference between EC2 Cloud host (East) and Edge host with GPU.

C. Surface Decluttering with the Toyota HSR

We test the performance of the trained models on the mobile Toyota HSR robot for surface decluttering. We load $5-25$ objects in a smaller bin from a smaller set of 65 physical objects and drop them on the floor in front of the robot (see Fig. 2). The overall accuracy of the domain invariant object recognition and the grasping model on the robot is 90.14% and 86.85%, respectively, for a total of decluttering 185 objects across 213 grasp attempts. In comparison, grasping orthogonal to the principal axis of the segmented objected resulted in a grasping accuracy of 76.19% only. We found that the grasping results improved substantially by retraining the model with respect to the tilted camera viewpoint of the robot in comparison to the results reported in [25]. Note that we remove the pathological objects such as heavy hammers, and objects with very low ground clearance such as wrenches and scissors that the robot is not able to grasp. We observe

TABLE II: Computation time for inference $t^{(\text{inf})}$ vs round trip communication time $t^{(\text{rtt})}$ (in milliseconds) for inference over Edge (with and without GPU) and Cloud with EC2 (West) EC2 (East) instances. Results are averaged across 200 real images. Communication time dominates the computation time and increases as the distance to the server increases.

Location	$t^{(\text{inf})}$	$t^{(\text{rtt})}$
Object Recognition		
EC2(East)	31.93 ± 1.53	437.63 ± 100.02
EC2(West)	$\mathbf{31.12 \pm 1.28}$	181.61 ± 22.71
Edge(CPU)	52.34 ± 4.18	149.32 ± 21.04
Edge(GPU)	33.27 ± 3.09	$\mathbf{119.40 \pm 12.06}$
Grasp Planning		
EC2(East)	1906.59 ± 224.19	4418.34 ± 1040.59
EC2(West)	1880.28 ± 207.46	2197.76 ± 199.44
Edge(CPU)	3590.71 ± 327.57	3710.74 ± 214.08
Edge(GPU)	$\mathbf{1753.65 \pm 201.38}$	$\mathbf{1873.16 \pm 211.57}$

that the robot performs well in grasping compliant objects and objects with well-defined geometry such as cylinders, screwdrivers, tape, cups, bottles, utilities, and assembly parts (see https://sites.google.com/view/fogrobotics for video, results and supplementary details).

V. Conclusions and Future Directions

In this paper, we have introduced a Fog Robotics approach for secure and distributed deep robot learning. Secured compute and storage at the Edge of the network opens up a broad range of possibilities to meet lower-latency requirements, while providing better control over data. Standardizing robot communication with available Edge resources, nonetheless, is challenging for a wider adoption of Fog Robotics. We have presented a surface decluttering application, where non-private (public) synthetic images are used for training of deep models on the Cloud, and real images are used for adapting the learned representations to the real world in a domain invariant manner. Deploying the models on the Edge significantly reduces the round-trip communication time for inference with a mobile robot in the decluttering application. In future work, we plan to deploy various deep models for segmentation, hierarchical task planning etc, for low-latency and secure prediction in a multi-agent distributed environment with a set of robots.

Acknowledgment

This research was performed at the AUTOLAB at UC Berkeley in affiliation with the Berkeley AI Research (BAIR) Lab, Berkeley Deep Drive (BDD), the Swarm Lab, the Real-Time Intelligent Secure Execution (RISE) Lab, the CITRIS "People and Robots" (CPAR) Initiative, and by the Scalable Collaborative Human-Robot Learning (SCHooL) Project, NSF National Robotics Initiative Award 1734633, and the NSF ECDI Secure Fog Robotics Project Award 1838833. The work was supported in part by donations from Siemens, Google, Amazon Robotics, Toyota Research Institute, Autodesk, ABB, Knapp, Loccioni, Honda, Intel, Comcast, Cisco, Hewlett-Packard and by equipment grants from PhotoNeo, NVidia, and Intuitive Surgical. The authors would like to thank Flavio Bonomi, Moustafa AbdelBaky, Raghav Anand, Richard Liaw, Daniel Seita, Benno Staub, Sanjay Krishnan and Michael Laskey for their helpful feedback and suggestions.

REFERENCES

[1] J. Kuffner, "Cloud-Enabled Robots," in *IEEE-RAS International Conference on Humanoid Robots*, 2010. [Online]. Available: http://www.scribd.com/doc/47486324/Cloud-Enabled-Robots

[2] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *Automation Science and Engineering, IEEE Transactions on*, vol. 12, no. 2, pp. 398–409, 2015.

[3] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Gálvez-López, K. Häussermann, R. Janssen, J. Montiel, A. Perzylo, B. Schieß le, M. Tenorth, O. Zweigle, and R. De Molengraft, "RoboEarth," *IEEE Robotics & Automation Magazine*, vol. 18, no. 2, pp. 69–82, 2011.

[4] M. Tenorth and M. Beetz, "KnowRob – A Knowledge Processing Infrastructure for Cognition-enabled Robots," *Int. Journal of Robotics Research*, vol. 32, no. 5, pp. 566 – 590, April 2013.

[5] A. Saxena, A. Jain, O. Sener, A. Jami, D. K. Misra, and H. S. Koppula, "Robobrain: Large-scale knowledge engine for robots," *CoRR*, vol. abs/1412.0691, 2014. [Online]. Available: http://arxiv.org/abs/1412.0691

[6] N. Tian, M. Matl, J. Mahler, Y. X. Zhou, S. Staszak, C. Correa, S. Zheng, Q. Li, R. Zhang, and K. Goldberg, "A cloud robot system using the dexterity network and berkeley robotics and automation as a service (brass)," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1615–1622.

[7] L. Pusong, B. DeRose, J. Mahler, J. A. Ojea, A. K. Tanwani, and K. Golberg, "Dex-net as a service (dnaas): A cloud-based robust robot grasp planning system," in *14th International Conference on Automation Science and Engineering (CASE)*, 2018, pp. 1–8.

[8] "Amazon RoboMaker," https://aws.amazon.com/robomaker/, accessed: 2018-11-29.

[9] "Google Cloud Robotics Platform," https://cloud.google.com/cloud-robotics/, accessed: 2018-11-29.

[10] K. Goldberg, "Robots and the return to collaborative intelligence," *Nature Machine Intelligence*, pp. 2–4, 2019.

[11] J. Wan, S. Tang, H. Yan, D. Li, S. Wang, and A. V. Vasilakos, "Cloud robotics: Current status and open issues," *IEEE Access*, vol. 4, pp. 2797–2807, 2016.

[12] N. DeMarinis, S. Tellex, V. Kemerlis, G. Konidaris, and R. Fonseca, "Scanning the internet for ROS: A view of security in robotics research," *CoRR*, vol. abs/1808.03322, 2018.

[13] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[14] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Cloudlets: Bringing the cloud to the mobile user," in *Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services*, 2012, pp. 29–36.

[15] R. Roman, J. López, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *CoRR*, vol. abs/1602.00484, 2016. [Online]. Available: http://arxiv.org/abs/1602.00484

[16] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proceedings of the 2015 Workshop on Mobile Big Data*, ser. Mobidata '15. ACM, 2015, pp. 37–42.

[17] A. Dastjerdi, H. Gupta, R. Calheiros, S. aGhosh, and R. Buyya, "Chapter 4 - fog computing: principles, architectures, and applications," in *Internet of Things*, R. Buyya and A. V. Dastjerdi, Eds. Morgan Kaufmann, 2016, pp. 61 – 75.

[18] H. F. Atlam, R. J. Walters, and G. B. Wills, "Fog computing and the internet of things: A review," *Big Data and Cognitive Computing*, vol. 2, no. 2, 2018.

[19] S. L. K. Chand GUDI, S. Ojha, J. Clark, B. Johnston, and M.-A. Williams, "Fog robotics: An introduction," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 09 2017, pp. 1–2.

[20] B. Kehoe, A. Matsukawa, S. Candido, J. Kuffner, and K. Goldberg, "Cloud-based robot grasping with the google object recognition engine," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 4263–4270.

[21] M. Ciocarlie, K. Hsiao, E. G. Jones, S. Chitta, R. B. Rusu, and I. A. Şucan, *Towards Reliable Grasping and Manipulation in Household Environments*, 2014, pp. 241–252.

[22] S. S. Srinivasa, D. Ferguson, C. J. Helfrich, D. Berenson, A. Collet, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and M. V. Weghe, "Herb: a home exploring robotic butler," *Autonomous Robots*, vol. 28, no. 1, p. 5, Nov 2009.

[23] J. Bohren, R. B. Rusu, E. G. Jones, E. Marder-Eppstein, C. Pantofaru, M. Wise, L. Msenlechner, W. Meeussen, and S. Holzer, "Towards autonomous robotic butlers: Lessons learned with the pr2," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 5568–5575.

[24] M. Gupta, J. Müller, and G. S. Sukhatme, "Using manipulation primitives for object sorting in cluttered environments," *IEEE Trans. Automation Science and Engineering*, vol. 12, no. 2, pp. 608–614, 2015.

[25] A. Gupta, A. Murali, D. Gandhi, and L. Pinto, "Robot learning in homes: Improving generalization and reducing dataset bias," *CoRR*, vol. abs/1807.07049, 2018.

[26] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe, "Mobile fog: A programming model for large-scale applications on the internet of things," in *Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing*. ACM, 2013, pp. 15–20.

[27] M. Aazam and E. Huh, "Fog computing and smart gateway based communication for cloud of things," in *2014 International Conference on Future Internet of Things and Cloud*, 2014, pp. 464–470.

[28] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, *Fog Computing: A Platform for Internet of Things and Analytics*. Springer International Publishing, 2014, pp. 169–186.

[29] A. Yousefpour, G. Ishigaki, and J. P. Jue, "Fog computing: Towards minimizing delay in the internet of things," in *IEEE International Conference on Edge Computing (EDGE)*, 2017, pp. 17–24.

[30] J. K. Zao, T.-T. Gan, C.-K. You, C.-E. Chung, Y.-T. Wang, S. J. Rodrguez Mndez, T. Mullen, C. Yu, C. Kothe, C.-T. Hsiao, S.-L. Chu, C.-K. Shieh, and T.-P. Jung, "Pervasive brain monitoring and data sharing based on multi-tier distributed computing and linked data technology," *Frontiers in Human Neuroscience*, vol. 8, p. 370, 2014.

[31] K. Lee, D. Kim, D. Ha, U. Rajput, and H. Oh, "On security and privacy issues of fog computing supported internet of things environment," in *International Conference on the Network of the Future (NOF)*, 2015, pp. 1–3.

[32] A. Alrawais, A. Alhothaily, C. Hu, and X. Cheng, "Fog computing for the internet of things: Security and privacy issues," *IEEE Internet Computing*, vol. 21, no. 2, pp. 34–42, 2017.

[33] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 20, no. 1, pp. 416–464, 2018.

[34] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2018.

[35] A. Kattepur, H. K. Rath, and A. Simha, "A-priori estimation of computation times in fog networked robotics," in *2017 IEEE International Conference on Edge Computing (EDGE)*, 2017, pp. 9–16.

[36] S. L. K. C. Gudi, S. Ojha, B. Johnston, J. Clark, and M.-A. Williams, "Fog robotics for efficient, fluent and robust human-robot interaction," 2018.

[37] P. Pop, M. L. Raagaard, M. Gutierrez, and W. Steiner, "Enabling fog computing for industrial automation through time-sensitive networking (tsn)," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 55–61, 2018.

[38] "Fog Robotics," http://goldberg.berkeley.edu/fog-robotics/.

[39] "OpenFogConsortium," https://www.openfogconsortium.org/.

[40] D. Song, A. K. Tanwani, and K. Goldberg, "Networked-, cloud- and fog-robotics," in *Robotics Goes MOOC*, B. Siciliano, Ed. Springer, 2019.

[41] "EdgeX Foundry," https://www.edgexfoundry.org/.

[42] R. Ricci, E. Eide, and The CloudLab Team, "Introducing CloudLab: Scientific infrastructure for advancing cloud architectures and applications," *USENIX*, vol. 39, no. 6, 2014.

[43] "EdgeNet Project," http://edge-net.org/.

[44] "US Ignite," https://www.us-ignite.org/about/what-is-us-ignite/.

[45] D. Komosnỳ, S. Pang, J. Pružinskỳ, and P. Il'ko, "Planetlab europe as geographically-distributed testbed for software development and evaluation," 2015.

[46] "PlanetLab Europe," https://www.planet-lab.eu/.

[47] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar, "Geni: A federated testbed for innovative network experiments," *Computer Networks*, vol. 61, pp. 5–23, 2014.

[48] D. Schwerdel, B. Reuther, T. Zinner, P. Müller, and P. Tran-Gia, "Future internet research and experimentation: The g-lab approach," *Computer Networks*, vol. 61, pp. 102–117, 2014.

[49] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "Ddos in the iot: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[50] D. X. Song, D. Wagner, and X. Tian, "Timing analysis of keystrokes and timing attacks on ssh." in *USENIX Security Symposium*, vol. 2001, 2001.

[51] A. Whitten and J. D. Tygar, "Why johnny can't encrypt: A usability evaluation of pgp 5.0." in *USENIX Security Symposium*, vol. 348, 1999.

[52] N. Mor, B. Zhang, J. Kolb, D. S. Chan, N. Goyal, N. Sun, K. Lutz, E. Allman, J. Wawrzynek, E. A. Lee *et al.*, "Toward a global data infrastructure," *IEEE Internet Computing*, vol. 20, no. 3, pp. 54–62, 2016.

[53] R. Tamassia, "Authenticated data structures," in *European Symposium on Algorithms*. Springer, 2003, pp. 2–5.

[54] M. P. Andersen, J. Kolb, K. Chen, G. Fierro, D. E. Culler, and R. A. Popa, "Wave: A decentralized authorization system for iot via blockchain smart contracts," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2017-234, Dec 2017. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-234.html

[55] V. Costan and S. Devadas, "Intel sgx explained." *IACR Cryptology ePrint Archive*, vol. 2016, no. 086, pp. 1–118, 2016.

[56] S. Chanchali, 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," *CoRR*, vol. abs/1902.05703, 2019. [Online]. Available: http://arxiv.org/abs/1902.05703

[57] N. Tian, A. K. Tanwani, J. Chen, M. Ma, R. Zhang, B. Huang, K. Goldberg, and S. Sojoudi, "A fog robotic system for dynamic visual servoing," *CoRR*, vol. abs/1809.06716, 2018. [Online]. Available: http://arxiv.org/abs/1809.06716

[58] X. Xu, S. Fu, Q. Cai, W. Tian, W. Liu, W. Dou, X. Sun, and A. Liu, "Dynamic resource allocation for load balancing in fog environment," *Wireless Communications and Mobile Computing*, pp. 1–15, 2018.

[59] "E. Coumans, Bullet Physics Library," bulletphysics.org., accessed: 2018-11-29.

[60] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2096–2030, 2016.

[61] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015. [Online]. Available: http://arxiv.org/abs/1512.02325

[62] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2017.324

[63] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," *CoRR*, vol. abs/1611.10012, 2016. [Online]. Available: http://arxiv.org/abs/1611.10012

[64] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., 2014, pp. 2672–2680.

[65] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2962–2971.

[66] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," 2017.

[67] K. Borodulin, G. Radchenko, A. Shestakov, L. Sokolinsky, A. Tchernykh, and R. Prodan, "Towards digital twins cloud platform: Microservices and computational workflows to rule a smart factory," in *10th International Conference on Utility and Cloud Computing*, ser. UCC '17, 2017, pp. 209–210.

[68] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," *Linux J.*, vol. 2014, no. 239, 2014.

[69] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014.

[70] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. D. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," *CoRR*, vol. abs/1810.05687, 2018. [Online]. Available: http://arxiv.org/abs/1810.05687