Tiny Robot Learning

Challenges and Directions for Machine Learning in Resource-Constrained Robots

Neuman, Sabrina M. ; Plancher, Brian ; Duisterhof, Bardienus P. ; Krishnan, Srivatsan ; Banbury, Colby ; Mazumder, Mark ; Prakash, Shvetank ; Jabbour, Jason ; Faust, Aleksandra ; de Croon, Guido

DOI
10.1109/AICAS54282.2022.9870000

Publication date
2022

Document Version
Final published version

Published in
Proceedings of the 2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)

Tiny Robot Learning: Challenges and Directions for Machine Learning in Resource-Constrained Robots

Sabrina M. Neuman[1], Brian Plancher[1], Bardienus P. Duisterhof[2], Srivatsan Krishnan[1], Colby Banbury[1], Mark Mazumder[1], Shvetank Prakash[1], Jason Jabbour[3], Aleksandra Faust[4], Guido C.H.E. de Croon[5], and Vijay Janapa Reddi[1]

Harvard University[1], CMU[2], University of Virginia[3], Google Brain[4], Delft University of Technology[5]

{sneuman@seas, brian_plancher@g, srivatsan@g, cbanbury@g, markmazumder@g, sprakash@g, vj@eecs}.harvard.edu, bduister@andrew.cmu.edu, jjj4se@virginia.edu, aleksandra.faust@gmail.com, g.c.h.e.decroon@tudelft.nl

2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS) | 978-1-6654-0996-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/AICAS54282.2022.9870000

***Abstract*—Machine learning (ML) has become a pervasive tool across computing systems. An emerging application that stress-tests the challenges of ML system design is *tiny robot learning*, the deployment of ML on resource-constrained low-cost autonomous robots. Tiny robot learning lies at the intersection of embedded systems, robotics, and ML, compounding the challenges of these domains. Tiny robot learning is subject to challenges from size, weight, area, and power (SWAP) constraints; sensor, actuator, and compute hardware limitations; end-to-end system tradeoffs; and a large diversity of possible deployment scenarios. Tiny robot learning requires ML models to be designed with these challenges in mind, providing a crucible that reveals the necessity of holistic ML system design and automated end-to-end design tools for agile development. This paper gives a brief survey of the tiny robot learning space, elaborates on key challenges, and proposes promising opportunities for future work in ML system design.**

I. Introduction

Machine learning (ML) has become a pervasive technology, and as it spreads beyond traditional computing platforms (e.g., servers and desktops) towards devices on the edge (e.g., mobile, embedded, IoT, AR/VR, robotics, and other cyber-physical systems), new design pressures and constraints arise that fundamentally impact the ML system design process.

An emerging application that stress-tests the challenges of designing ML for edge devices is *tiny robot learning*, the deployment of ML on resource-constrained low-cost autonomous robots. These robots are lightweight (e.g., less than a pound, or under ~ 500g) and can operate in small spaces, making them a promising solution for applications ranging from emergency search and rescue [14], [15], [35], to routine monitoring and maintenance of infrastructure and equipment [12].

Tiny robot learning dials up the challenges of edge device ML, maximizing opportunities to refine edge ML system design by putting it through the crucible of the combined challenges of tiny (i.e., embedded) systems, robotics, and machine learning, all in one system deployment (Fig. 1). Tiny robot learning is subject to challenges from size, weight, area, power (SWAP) and cost constraints; sensor, actuator, and compute hardware limitations; end-to-end system tradeoffs; and a large diversity of possible deployment scenarios.

This material is based upon work supported by the National Science Foundation under Grant 2030859. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the funding organization.

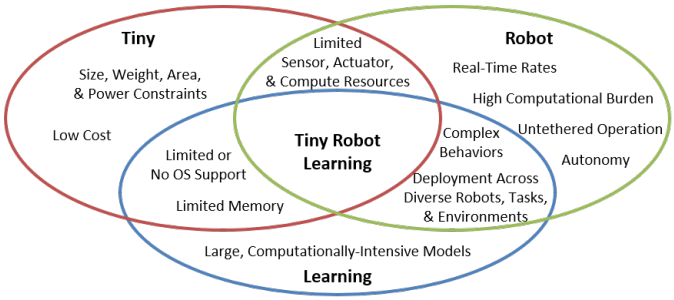

Fig. 1. Tiny robot learning combines tiny (i.e., embedded) systems, robotics, and machine learning, revealing challenges and opportunities for ML research.

Embedded *tiny* systems have severe SWAP constraints. For machine learning, this means that large, computationally-intensive models must be made to run on computing substrates, such as microcontrollers, with limited memory resources, and limited or no operating system support. Outside of robotics, SWAP constraints appear in many emerging edge device applications, and are being addressed with novel "TinyML" techniques such as model compression and distillation [5].

This challenge is compounded when tiny systems and machine learning intersect with robotics applications, and small-scale low-cost *tiny robots* impose additional constraints on both the ML model design as well as the end-to-end system surrounding the ML core. These robots have limited sensors, actuators, and compute resources, which complicate the development and implementation of TinyML models. Robotics is also a computationally-demanding application space, where ML models must learn complex and robust robot behaviors.

Adding to this difficulty, ML in untethered autonomous robots is fundamentally *embodied*, that is, the entire end-to-end robot system impacts the choices made in ML system deployment. In a tiny robot especially, critical tradeoffs must be made between the power and weight resources allocated to ML versus other parts of the system, e.g., sensor processing.

Finally, tiny robot deployments vary across robot models, system components, tasks, and environments, so it is essential to develop automatable flows to keep the design process agile.

The challenges of tiny robot learning are a call to action for the ML circuits, architecture, and systems design community. There are exciting opportunities for ML revealed by these

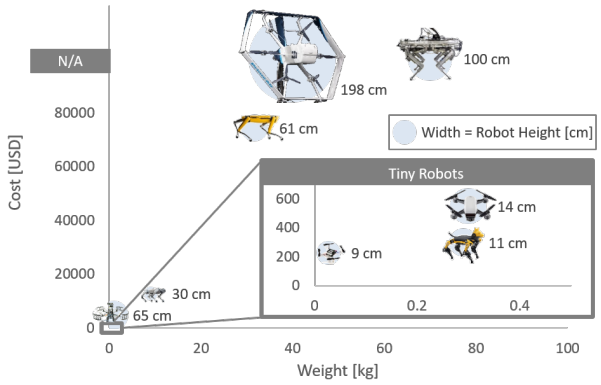

Fig. 2. Tiny robots are resource-constrained low-cost autonomous robots. Pictured are full-size [3], [7], [18], [23], [46] and tiny [6], [13], [40] robots.

challenges, including applying embedded TinyML techniques to computationally burdensome problems in robotics; using ML to compensate for the limitations of low-cost sensors and actuators; performing end-to-end co-design of ML with the surrounding cyber-physical system; and creating generalizable and automatable design flows for both software and hardware.

To explore the implications of tiny robot learning, in this work we give a brief survey of the tiny robots space, elaborate on the challenges imposed by tiny robot learning as an application for ML system design, and propose opportunities revealed by these challenges to improve ML system design.

II. Tiny Robots

We define tiny robots as resource-constrained, low-cost, low-weight autonomous robots. An example is the Petoi Bittle quadruped [40], which weighs 0.64 lbs (290 g) and costs under 300 USD (Fig. 2). By contrast, the Boston Dynamics Spot quadruped [7] weighs 69.9 lbs (31.7 kg) and costs $74,500$ USD, making it $100\times$ heavier and more expensive than Bittle. To fit within small weight and cost constraints, tiny robots are often limited in the availability and quality of onboard sensors, actuators, and compute resources, e.g., microcontrollers versus desktop-grade CPUs and GPUs.

While the same general challenges arise from ML design for a variety of different types of robots (e.g., quadrotor drones, satellites, quadrupeds, cars, submersibles), some design considerations differ between these platforms. For example, when comparing quadrotors to quadrupeds, weight is a more extreme constraint for quadrotors, while quadrupeds require more computationally expensive motion planning and control algorithms due to their increased degrees of freedom.

Tiny robots are useful in a diverse range of applications. Table I shows a brief (and incomplete) overview of emerging applications of tiny robots ranging from search and rescue, to inspection, entertainment, and STEM education. Because of their small size and ability to escape detection, we note that it is essential to exercise ethical consideration in the design and deployment of tiny robot systems [1], [32]. Concerns such as safety, privacy, and security must be factored into the engineering process as first-class constraints.

TABLE I
Emerging Tiny Robot Applications

Task	Robot	Weight [g]	Citation
Search & Rescue	Bitcraze CrazyFlie Quadcopter	33	[14], [15], [35]
Inspection	HAMR-E	1.4	[12]
Medical Robotics	Wireless Capsule Endoscope	7	[48]
Space Robotics	KickSat	5	[34]
Military Reconnaissance	Black Hornet	33	[8]
Entertainment	DelFly Nimble	28	[22], [38]
STEM Education	Mona	290	[2]
Pollination	RoboBee	0.08	[49]

The tiny robot examples surveyed in Table I demonstrate the effectiveness of co-design between sensors, compute and algorithms. These robots use sensors beyond traditional cameras: tiny lasers, optic flow sensors, light sensors, gas sensors, pressure sensors, and custom-made tiny cameras. The algorithms they use were intentionally designed with the compute constraints in mind, allowing them to run on low-power and low-cost microcontrollers. We expect novel sensors such as event cameras and optic flow sensors, as well as novel compute platforms like the Intel Lohi neuromorphic chip [11], to greatly improve the performance and capabilities of future tiny robots.

III. Challenges and Opportunities

Tiny robot learning lies at the intersection of three challenging domains (Fig. 1), making it a proving-ground for ML systems. In this section, we examine four themes arising from the challenges of tiny robot learning, and propose opportunities that they reveal for improving ML system design (Fig. 3).

Starting from the ML computation at the core of the system and moving outward, we focus on: (III-A) the onboard compute; (III-B) the sensors and actuation that represent the inputs and outputs of the robotics computation pipeline; (III-C) the entire end-to-end system including the physical robot platform; and finally, (III-D) the design tools used to re-design the tiny robot learning system for different deployment scenarios.

A. SWAP-Constrained ML Compute for Robotics Applications

In untethered robot systems, a subset of core computations must always be performed using onboard compute resources (rather than being offloaded to the cloud) due to the strict latency requirements of running at real-time rates (1kHz or more), as well as a lack of continuous communication and connectivity guarantees. Tiny robots have limited compute resources (e.g., microcontrollers) that are severely size, weight, area, and power (SWAP) constrained, yet they must handle the same heavy computational burdens and produce the same complex robust behaviors in the real world as full-size autonomous robots with access to onboard server-class CPUs and GPUs.

Outside of robotics, SWAP constraints appear in many emerging edge computing applications (e.g., IoT), and are being addressed with TinyML techniques such as quantization [20], distillation [21], and tiny model design [4].

Opportunity: Tiny robot learning reveals opportunities for the application of TinyML techniques to difficult problems in robotics. Prior work has applied ML across the stages of

the robotics computational pipeline (shown in Fig. 3): perception [33], mapping and localization [30], and motion planning and control [24], [28]. By mapping these robotics tasks to ML, one can leverage existing work to compress and accelerate ML under extreme SWAP constraints, enabling the deployment of complex tasks on resource-constrained tiny robot platforms. Learned approaches can also offer computationally cheaper alternatives to traditional robotics algorithm pipelines through end-to-end learning-based approaches (Fig. 3) [15], [31].

In addition to inference model implementation, there are implications for training in robot learning that take on a new dimension when applied to tiny robots. The mainstream approach is to learn offboard and deploy the trained solution on the robot. An obvious limitation of small robots is that the solution, typically a neural network model, has a lower complexity (fewer layers, neurons, etc.). Besides a somewhat lower performance, this also means that it is hard to train models on a large variety of environments and conditions, as the network may simply not be expressive enough.

A promising solution to this is to perform online learning, which would allow the tiny robot to learn about its immediate environment. Having a small learned model for a lower variety of environments may be acceptable, given that smaller robots will typically travel less far [27]. It would be interesting to investigate the trade-off between model complexity and practical use for tiny autonomous robots. Like small insects, tiny robots may be able to exploit less powerful learning models to function well in a specific environment [16]. Of course, online learning is already a challenge for robots in general and becomes even more challenging on tiny robots. For example, catastrophic forgetting is often solved by means of maintaining a replay buffer [36], but this will be harder to execute given very limited memory. In general, machine learning methods for tiny robots will have to more carefully assign resources to what is learned and what is forgotten.

B. Sensor and Actuator Limitations in Tiny Robot Platforms

With cost and SWAP constraints, traditional sensors are often impractical on tiny robot platforms, forcing them to rely on lower-quality sensors [15]. Tiny robots may also have fewer and less-precise actuators than full-size robots. For example, the Spot quadruped [7] has position and force sensors on its legs and 12 degrees of freedom (DoF), while Bittle [40] has no position or force sensors and only 8 DoF. ML techniques can enable complex behaviors on full-sized robots (e.g., quadruped locomotion on difficult terrain [28]), but with component limitations, tiny robot platforms face an additional hurdle.

Opportunity: Low cost proximity and light sensors can replace larger, more costly sensors like cameras or LIDAR by using additional processing, often ML, to achieve the same levels of perception as their higher-quality counterparts. For example, monocular depth estimation [33] can be substituted for wide-baseline stereo cameras. In some cases, the lower dimensional input from, e.g., light or proximity sensors, means that a simple policy can be developed to solve a complex problem. For example, system-specific sensor and algorithm selection can yield lighter computation than traditional mapping and localization using cameras or LIDAR [15].

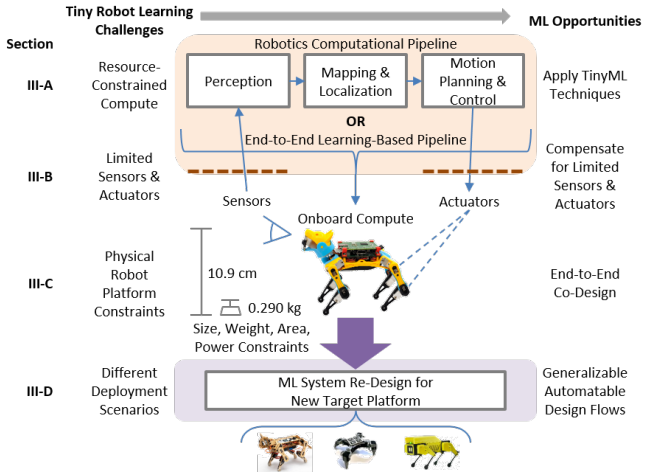

Fig. 3. Promising research directions revealed by tiny robot learning include applying embedded *TinyML* techniques to traditional robotics applications; using ML to compensate for sensor and actuator limitations; end-to-end co-design of ML with physical systems; and developing automatable design flows for agile re-deployment to new target platforms (e.g., [39], [40], [47]).

ML algorithms can also enable robots to use limited and degraded sensory input to overcome challenges such as motion control with imprecise actuators and a lack of direct position feedback. For example, recent work used a hybrid model-based and learning-based controller to enable a tiny quadruped to walk over uneven terrain [43], despite tight SWAP and cost constraints on sensing and actuation.

C. End-to-End Co-Design of ML with Physical Robot Systems

Given the severe software and hardware constraints faced by tiny robots (Sections III-A, III-B) there is a need for holistic end-to-end system co-design to develop optimized and task-specific tiny robot systems. Designers must account for physical properties (e.g., battery weight) to understand overall system metrics like maximum velocity [26] and mission time.

Opportunity: There is an opportunity to improve overall system performance by building holistic end-to-end benchmarking frameworks. These frameworks enable exploration of quantitative tradeoffs between cost, robustness, and efficiency across the entire cyber-physical stack, and help designers find efficient operating points, which are especially important for resource-constrained platforms. An example is the Air Learning [25] framework, which evaluates combinations of learning algorithms, ML models, sensing modalities, and onboard compute platforms for autonomous drones.

By leveraging end-to-end benchmarks, the physical properties of robotic systems can be co-designed with the computer hardware, algorithms, and application. Early work has explored co-design for drones [9], [26] and legged robots [17].

An end-to-end benchmarking framework also exposes design parameters that can enable the use of machine learning methods *to build machine learning systems*. Recent work has used techniques like Bayesian optimization [44], evolutionary

algorithms [10], and reinforcement learning [19] to efficiently navigate the design space of neural network model parameters. These techniques can be extended to co-design ML hardware accelerators with the physical parameters of the robot platform.

D. Generalizeable and Automatable Design Flows

An extreme challenge for tiny robot learning is navigating the large diversity of robot deployment scenarios. Deployments vary across robots, system components, tasks, and environments, creating an enormous design space to explore.

Opportunity: The diversity of robot learning deployments reveals the need for generalizeable and automatable design flows and methodologies for efficiently navigating the large design space. There is emerging work in designing such tools for non-learning-based robotics hardware accelerators [29], [37], [41], [45]. Outside of the robotics domain, there is also early work on full-stack open-source tools and generalizable design flows for the rapid deployment of TinyML accelerators [42].

Some combination of the aforementioned works, along with the holistic benchmarking frameworks from Section III-C, may be useful in designing tools for tiny robot learning systems. Automated design flows are especially appealing for tiny robot learning because they can encode domain-specific knowledge into the tools and allow for end-to-end system design without intervention from experts in multiple disparate domains: embedded systems, robotics, and ML (Fig. 1).

IV. Conclusion

In this work, we examined tiny robot learning: the deployment of ML on resource-constrained low-cost autonomous robots. Lying at the intersection of embedded systems, robotics, and ML, tiny robot learning is subject to challenges from size, weight, area, and power constraints; sensor, actuator, and compute hardware limitations; end-to-end system tradeoffs; and a large diversity of possible deployment scenarios. As such, it reveals promising opportunities for future work developing holistic ML system design techniques and automated end-to-end design tools for agile development.

References

[1] A. B. Arrieta *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Elsevier Information Fusion*, 2020.

[2] F. Arvin *et al.*, "Mona: an affordable open-source mobile robot for education and research," *Journal of Intelligent & Robotic Systems*, 2019.

[3] Ascending Technologies, "Pelican," Archived 2020, asctec.de.

[4] C. Banbury *et al.*, "Micronets: Neural network architectures for deploying tinyml applications on commodity microcontrollers," *MLSys*, 2021.

[5] C. R. Banbury *et al.*, "Benchmarking tinyml systems: Challenges and direction," *MLSys Workshop on Bench. ML Work. Emerging HW*, 2020.

[6] Bitcraze, "Crazyflie," Accessed 2022, bitcraze.io/products/crazyflie-2-1.

[7] Boston Dynamics, "Spot," Accessed 2022, bostondynamics.com/spot.

[8] G. Cai *et al.*, "A survey of small-scale unmanned aerial vehicles: Recent advances and future development trends," *Unmanned Systems*, 2014.

[9] L. Carlone and C. Pinciroli, "Robot co-design: beyond the monotone case," in *ICRA*, 2019.

[10] W. Chang *et al.*, "Hardware/software co-synthesis and co-optimization for autonomous systems," in *DAC*, 2021.

[11] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, 2018.

[12] S. D. de Rivaz *et al.*, "Inverted and vertical climbing of a quadrupedal microrobot using electroadhesion," *Science Robotics*, 2018.

[13] DJI, "Spark," Accessed 2022, dji.com/spark.

[14] B. P. Duisterhof *et al.*, "Sniffy bug: A fully autonomous swarm of gas-seeking nano quadcopters in cluttered environments," in *IROS*, 2021.

[15] ——, "Tiny robot learning (tinyrl) for source seeking on a nano quadcopter," in *ICRA*, 2021.

[16] R. Ernst and M. Heisenberg, "The memory template in drosophila pattern vision at the flight simulator," *Vision Research*, 1999.

[17] G. Fadini *et al.*, "Computational design of energy-efficient legged robots: Optimizing for size and actuators," in *ICRA*, 2021.

[18] Federal Aviation Administration, "Airworthiness criteria: Special class airworthiness criteria for the amazon logistics, inc. mk27," 2020.

[19] A. Goldie and A. Mirhoseini, "Placement optimization with deep reinforcement learning," in *ISPD*, 2020.

[20] S. Han *et al.*, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *Intl. Conf. on Learning Representations*, 2015.

[21] G. Hinton *et al.*, "Distilling the knowledge in a neural network," *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[22] M. Karásek *et al.*, "A tailless aerial robotic flapper reveals that flies use torque coupling in rapid banked turns," *Science*, 2018.

[23] B. Katz *et al.*, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *ICRA*, 2019.

[24] E. Kaufmann *et al.*, "Deep drone racing: Learning agile flight in dynamic environments," in *CoRL*, 2018.

[25] S. Krishnan *et al.*, "Air learning: a deep reinforcement learning gym for autonomous aerial robot visual navigation," *Machine Learning*, 2021.

[26] ——, "The sky is not the limit: A visual performance model for cyber-physical co-design in autonomous machines," *IEEE CAL*, 2020.

[27] K. Lamers *et al.*, "Self-supervised monocular distance learning on a lightweight micro air vehicle," in *IROS*, 2016.

[28] J. Lee *et al.*, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, 2020.

[29] W. Liu *et al.*, "Archytas: A framework for synthesizing and dynamically optimizing accelerators for robotic localization," in *MICRO*, 2021.

[30] ——, "Semi-dense visual-inertial odometry and mapping for quadrotors with swap constraints," in *ICRA*, 2018.

[31] A. Loquercio *et al.*, "Dronet: Learning to fly by driving," *RA-L*, 2018.

[32] R. Luppicini and A. So, "A technoethical review of commercial drone use in the context of governance, ethics, and privacy," *Elsevier Technology in Society*, 2016.

[33] R. Mahjourian *et al.*, "Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints," in *CVPR*, 2018.

[34] Z. Manchester *et al.*, "Kicksat: A crowd-funded mission to demonstrate the world's smallest spacecraft," in *Conf. on Small Satellites*, 2013.

[35] K. N. McGuire *et al.*, "Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment," *Science Robotics*, 2019.

[36] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, 2015.

[37] S. M. Neuman *et al.*, "Robomorphic computing: a design methodology for domain-specific accelerators parameterized by robot morphology," in *ASPLOS*, 2021.

[38] D. A. Olejnik *et al.*, "A tailless flapping wing mav performing monocular visual servoing tasks," *Unmanned Systems*, 2020.

[39] OteRobotics, "Realant," Accessed 2022, github.com/OteRobotics.

[40] Petoi, "Bittle and Nybble," Accessed 2022, petoi.com.

[41] B. Plancher *et al.*, "Accelerating robot dynamics gradients on a cpu, gpu, and fpga," *RA-L*, 2021.

[42] S. Prakash *et al.*, "Cfu playground: Full-stack open-source framework for tiny machine learning (tinyml) acceleration on fpgas," Accessed 2022, github.com/google/CFU-Playground.

[43] M. Rahme *et al.*, "Linear policies are sufficient to enable low-cost quadrupedal robots to traverse rough terrain," in *IROS*, 2021.

[44] B. Reagen *et al.*, "A case for efficient accelerator design space exploration via bayesian optimization," in *ISLPED*, 2017.

[45] J. Sacks *et al.*, "Robox: an end-to-end solution to accelerate autonomous control in robotics," in *ISCA*, 2018.

[46] C. Semini *et al.*, "Design and experimental evaluation of the hydraulically actuated prototype leg of the hyq robot," in *IROS*, 2010.

[47] Stanford, "Pupper," Accessed 2022, github.com/stanfordroboticsclub.

[48] P. Swain *et al.*, "Remote magnetic manipulation of a wireless capsule endoscope in the esophagus and stomach of humans (with videos)," *Gastrointestinal endoscopy*, 2010.

[49] R. Wood *et al.*, "Flight of the robobees," *Scientific American*, 2013.