Efficient Language-Guided Reinforcement Learning for Resource Constrained Autonomous Systems

Aidin Shiri
University of Maryland Baltimore County

Mozhgan Navardi
University of Maryland Baltimore County

Tejaswini Manjunath
University of Maryland Baltimore County

Nicholas R. Waytowich
US Army Research Laboratory

Tinoosh Mohsenin
University of Maryland Baltimore County

***Abstract*—**
In this paper, we propose an energy-efficient architecture which is designed to receive both images and text inputs as a step towards designing reinforcement learning agents that can understand human language and act in real-world environments. We evaluate our proposed method on three different software environments and a low power drone named Crazyflie to navigate towards specified goals and avoid obstacles successfully. To find the most efficient language-guided RL model, we implemented the model with various configurations of image input sizes and text instruction sizes on the Crazyflie drone GAP8 which consists of 8 RISC-V cores. The task completion success rate and onboard power consumption, latency, and memory usage of GAP8 are measured and compared with Jetson TX2 ARM CPU and Raspberry Pi 4. The results show that by decreasing 20% of input image size we achieve up to 78% energy improvement while achieving an 82% task completion success rate.

REINFORCEMENT LEARNING (RL) is a goal-oriented paradigm of machine learning in which the RL agent tries to learn a policy to perform tasks by trial and error. RL is the best suited for applications that involve sequential decision making like autonomous drone navigation [1] and robot control [2], where the agent needs to take several actions in an environment to perform the desired task. In RL, the agent learns the policy using the reward function. But human language can also be used to specify goals as shown in [3]. It is beneficial to train the agent in a way that easily abides the human instructions. One scalable way to do this is by using language instructions. Recent works in using

This article has been accepted for publication in IEEE Micro. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MM.2022.3199686

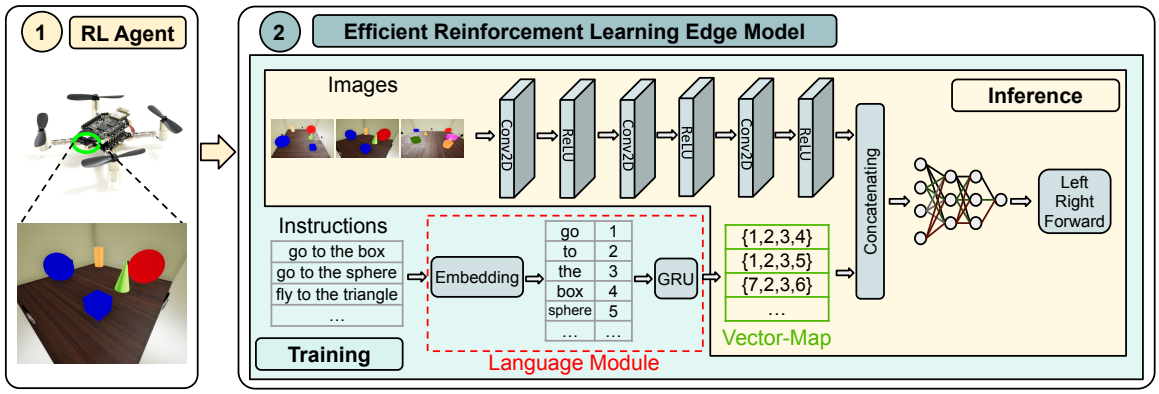

Figure 1. Proposed architecture for the RL agent training and inference. The agent learns the desired policy by receiving the instructions in the form of text from user and images from the environment. During the inference, the hardware-intensive language module is replaced by a vector-map to reduce the power and area and increase the efficiency of the design. Efficient reinforcement learning model is suitable for deploying on resource constraint autonomous systems such as Crazyflie.

language to guide RL agents have gained a great interest among AI researchers [4]. These methods have shown improvement in making agents autonomous and able to understand human language. Instructions are processed using language processing techniques to generate text embedding to feed the agent. Besides making it easy to specify goals and rewards, language can be used to convey constraints and improve the safety of RL agents. Hardware implementation of Neural Networks (NN) has gained a lot of attention in recent years. But most of the previous research in the NN-based RL domain has only focused on the algorithm and theoretical aspects, and very few works have considered hardware implementation for RL [5]. In this work, we propose an approach to implement a hardware-friendly NN-based RL for resource constrained autonomous agents to make them able to interpret the natural language instructions [6] and to perform instructed tasks in real-world scenarios. To achieve this, we used several RL environments and integrated a language instruction module into them. Then, to have an efficient language-guided RL, we remove hardware-intensive language modules. Moreover, we address the energy efficiency trade-off of the RL model on different instruction and image input sizes. Finally, we implemented the efficient model on a tiny drone to measure the power consumption and performance of the model.

Efficient Guiding Reinforcement Learning Agent with Language

In traditional RL, agents interact with the environment through a set of actions A to maximize the reward R. We build off of the traditional method which considers Markov Decision Process (MDP), defined by the tuple (S, A, P, R, γ) where S and A are the state and action spaces, respectively. $\gamma \in (0, 1)$ is the discount factor and $T\ (s^{'}|s, a)$ represents the transition distribution. The goal of the RL agent is to find the optimal policy P that maximizes the reward function R. The trajectory is a sequence of state actions.

Goals can be specified using various ways. One popular way is by using language instructions. Language is a set of structured and fixed instructions that have been given to the RL agent to accomplish desired tasks. Figure 1 illustrates the system overview of the proposed approach. The proposed system architecture consists of two main parts: agent and efficient RL edge model. RL agent explores an environment and takes feedback to reach a goal given in the language module as an instruction. The language module consists of a Gated Recurrent Unit (GRU), which translates the instruction in the form of a text to a vector of text embedding. For instance, if we have a set of words {“go”, “to”, “the”, “sphere”, “box”, “triangle”} a combination of them creates a command such as “go to the

sphere". Each word corresponds to a number, and in this way, a vector of numbers means a command. RL agent also receives states in the form of an image and produces a vector of image embedding. Later, the output of the language module is concatenated with the output of the Convolutional Neural Network (CNN) and passed through Fully-Connected (FC) layers.

In this paper, we used several instructions to guide the agent. Therefore, the language module is used for the training phase to process the human language-based commands and to generate a set of vectors corresponding to a specific instruction using GRU. We extract the encoded vector of that command during the training and store them in LUT as shown in Table 1 because, despite showing great performance in tasks such as processing the natural human language, GRUs are hard to port on the hardware and are also computing-intensive. One scalable way to address this problem is to replace the language module during the inference with a LUT since it can significantly speed up the computation during the inference. Therefore, to have an efficient deployment of language-guided RL on edge devices during the inference phase, the bulky language module is replaced with a simple vector-map. The vector-map can feed the saved instruction vectors to the agent to increase the efficiency of the design. In this way we can significantly reduce the power consumption and make the RL model suitable to be deployed on resource constrained edge devices. Moreover, for experiments with n goals, n separate vectors will be generated making this architecture appropriate for multiple goals, as opposed to binary encoding.

Experimental Setup

In this section, we explain the environments and setup used for evaluating the performance of the proposed approach. Figure 2 shows three different RL environments that are chosen in a way that each environment represents variant level of policy complexity: Gym Mini-Grid, Gym Mini-World and Air-Learning environments. For each environment, we defined the reward function for the agent as $R = 1 - 0.2 \times (current\_step/max\_step)$ if the agent performs the instructed task, and 0 otherwise. All of the models consist of GRU, convolutional, and fully-connected layers. The agent is trained using Proximal Policy Optimization (PPO) [8] in all three environments.

Table 1. Example of language instruction constraints used in three different environments to instruct the desired policy to the agent.

#	Instructions	Vector-Map	Environments
1	Use the red door	{1,4,6,9}	Mini-Grid
2	Go through the blue door	{2,5,7,9}	Mini-Grid
3	Open the gray door	{3,4,8,9}	Mini-Grid
4	Pick up the red box	{1,4,5,8}	Mini-World
5	Go to the blue box	{2,4,6,8}	Mini-World
6	Lift to the blue box	{3,4,7,8}	Mini-World
7	Fly to the red sphere	{1,4,5,7}	Air-Learning
8	Aviate towards the red sphere	{2,4,5,7}	Air-Learning
9	Fly to the blue sphere	{3,4,6,7}	Air-Learning

Gym Mini-Grid [9]

Mini-Grid Go-to-Door is a minimalistic 2D environment for RL research which is illustrated in Figure 2 (a). In this environment, there are multiple doors of different colors. The goal of the agent is to navigate towards the instructed door with a specific color and exit the room by opening the door. We designed a scenario in which the agent starts in a room with four doors, one on each wall. The agent receives an instruction vector as an input, commanding it to go towards a door: "go to the red door". Episodes terminate at 128 steps if the agent can not finish the instructed task, or violate it and receive a negative reward.

Gym Mini-World [10]

To observe the performance of the proposed method in an environment with higher complexity, we evaluated the models performance in the Mini-World environment, a 3D environment designed for RL and robotics research. The agent can navigate in different rooms and manipulate objects using first-person, as shown in Figure 2 (b). In our experiments, we set up a scenario where the agent is in a room with two boxes: red and blue. Depending on the instruction, the agent has to reach one of the boxes.

Air-Learning [11]

Air-Learning is designed specifically targeted for aerial robotics and autonomous Unmanned

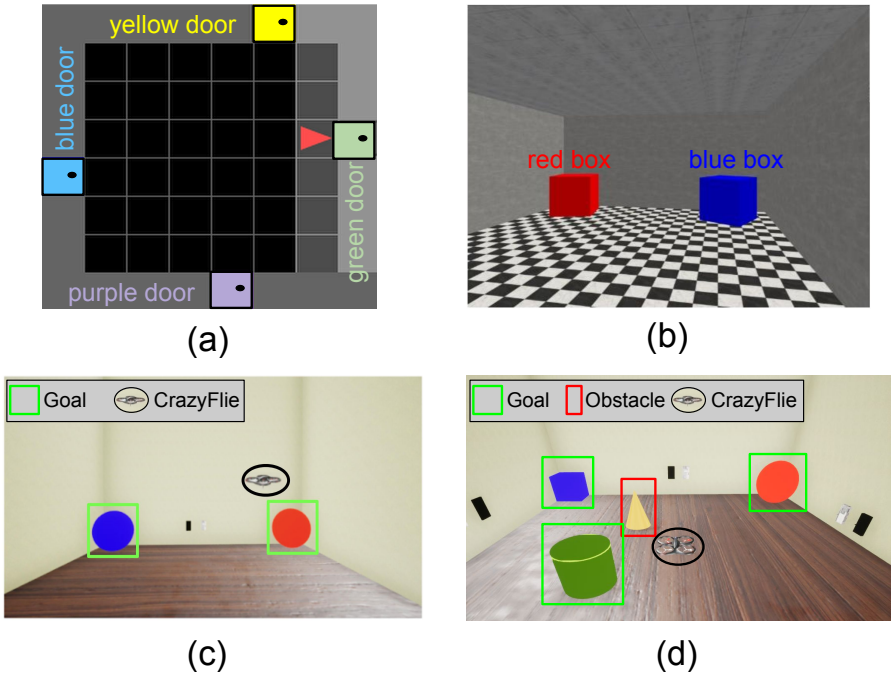

Figure 2. Snapshots of the three software environments: (a) Mini-Grid, (b) Mini-World, (c) Simple Air-Learning, and (d) Complex Air-Learning in which the agent is instructed to learn the desired policy, while the environment complexity increases in the order. In the Mini-Grid environment, the agent is instructed to open a specific door and exit the room. In the Mini-World environment, the agent is requested to navigate toward a colored box in a 3-D environment. In Air-Learning, the agent is a drone flying in a realistic environment and is instructed to find and hover over a certain object, while avoiding the obstacles during navigation. All environments back-end is modified so that the agent could be instructed using the proposed language-based RL architecture, the software codes can be found in UMBC-EEHPC GitHub.

Aerial Vehicles (UAVs) in 3D environment and offers the most complex setting for our drone experiments. It generates high-fidelity photo-realistic environments with domain randomization and flight physics model for the UAVs to fly in. We designed a scenario in Air-Learning in which a drone starts at a random location in a 10m$\times$10m room and receives an instruction for navigating toward an object. The model receives 324$\times$244 RGB images from the drone's camera as its input and generates actions to navigate toward an object in the environment. Each episode is terminated after 200 steps if the agent does not perform the instructed task. Figure 2 (c) shows an overview of the Air-Learning environment. As illustrated in Figure 2 (d), we increase the complexity of the environment by adding several obstacles for the drone to avoid while navigating towards the goals.

Simulation Results and Evaluation

We evaluated and measured the performance of each model in the related environments. The neural network architecture of all the agents for these environments consists of text embedding for the instruction, and three convolutional layers that are followed by two fully connected layers, which eventually generate the agent's action. To evaluate the performance, we measured the success rate which is the number of times the agent reaches the given goal over the total number of experiments during training. Figure 3 (top row) shows that the success rate has an increasing trend and the value loss has a decreasing trend. The final success rate for the Mini-Grid and Mini-World environments were 98% after 2 Million steps of training, and 95% after 10 Million steps of training, respectively. The value loss is also shown for these two environments. As shown

This article has been accepted for publication in IEEE Micro. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MM.2022.3199686

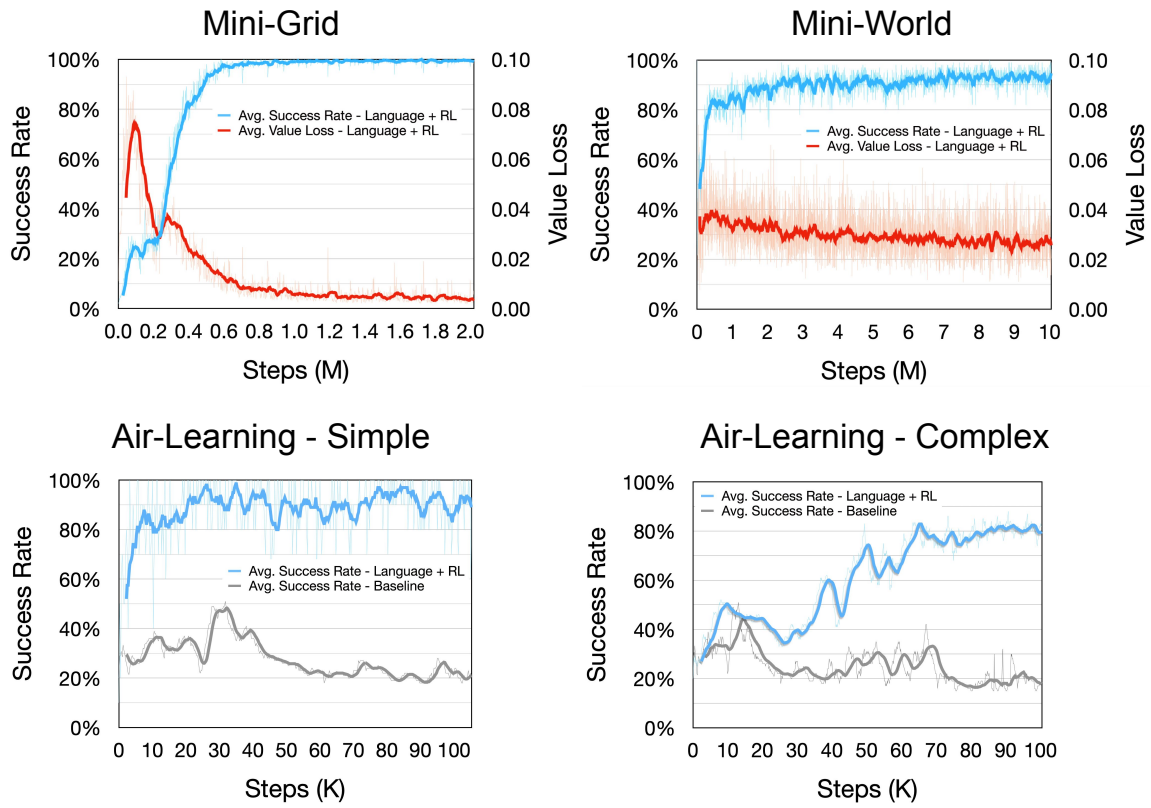

Figure 3. Success rate achieved by the proposed language-guided RL agent in the Mini-Grid, Mini-World, and Air-Learning environments with respect to the number of episode steps. The Air-Learning experiments consist of simple and complex configurations with obstacles as well as a comparison with baseline RL. The proposed technique in the Air-Learning environment achieves near 100% success rate after 25K steps whereas the baseline does not improve beyond 30% even after 100K steps.

in Figure 3 (bottom row), for the model trained in the Air-Learning environment, the RL agent achieved reasonable task completion rate of 90% and 86% after 100K steps for the simple and complex environment, respectively. As a baseline, we consider the same architecture without the language module. However, after 35 hours of training, the success rate is under 20% and the agent failed to converge to one that could successfully reach a given object. We also observe that with addition of more objects in the environment, the performance of the baseline model reduces significantly as can be seen in Figure 3. In the test phase we gave the agent some new instructions with minor changes. For instance, instead of "go to the red sphere" we gave "fly to the red sphere". The agent successfully handled these minor changes. But, when we tested with instructions that are completely different from the train set, we got a success rate of 40% over 250 experiments. To conclude, our model has no overfitting as long as the changes are minor.

Hardware Implementation and Results

To find the most efficient language-guided RL model, we implemented the model with various configurations on the Crazyflie nano drone [12]. We considered five different image input sizes: 324×244, 259×195, 194×146, 129×97, 64×48, and four different input instruction text sizes: 256KB, 128KB, 64KB, and 32KB. As shown in Figure 4 (a), Crazyflie consists of GAP8 processor which has a RISC-V based PULP platform with two compute domains: (1) a fabric controller for controlling tasks and 512KB L2 memory expandable and (2) a cluster domain with 8 cores

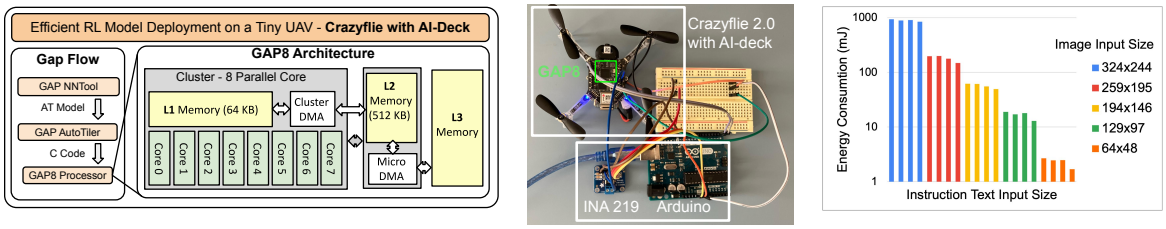

Figure 4. (a) GAP8 architecture includes 8 parallel cores, L1, L2 and L3 memory. (b) GAP8 processor on the AI-deck and Crazyflie with power measurement setup. (c) Energy consumption analysis on different instruction text input and image input size. For each image size category the text size from left is equal to 256KB, 128KB, 64KB and 32KB, respectively. It shows decreasing input image size from 324x244 to 259x195 leads to achieve up to 78% energy improvement.

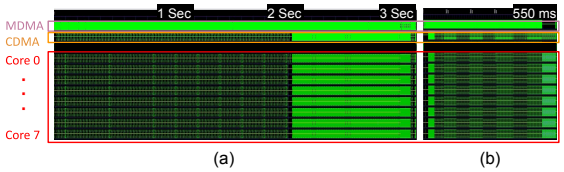

Image Size	Core Utilization	MDMA Utilization	Execution Time (ms)
324x244	68%	**100%**	3000
259x195	57%	89%	550

(c)

Figure 5. Snapshots of GAP8-based AI-deck platform that is used to get VCD traces for two different image input sizes: (a) 324x244 and (b) 259x195. (c) Core and Micro-DMA (MDMA) utilization is extracted from VCD traces.

for parallel computation of highly demanding workloads and 64KB directly accessible L1 memory. We quantized our network using TFLite and deployed it using GAPflow on the Crazyflie. GAPflow includes NNTool and AutoTiler, which NNTool parses and prepares TFLite models. AutoTiler converts the parsed model into optimized C code that can be executed efficiently in parallel by the 8 cores in the cluster.

Figure 4 (b) shows the Crazyflie drone, which is used for the real-world implementation, along with power measurement setup, using INA 219 sensor and Arduino board. The bar plots in Figure 4 (c) show the energy consumption that is calculated based on collected latency and power consumption results using the GAP8-based AI-deck platform. For all 20 models, the amount of time each layer took to execute in both cycles and microseconds was summed to get a total time for the entire model. Afterwards, the model was looped over and over again to collect power data using the attached Arduino and INA 219. From the collected results, it was found that although the amount of instruction being input to the neural network did have an effect on the time-to-completion of each inference, the image size had a much larger effect. Also, it was found that certain image sizes, especially 324×244 took what effectively seemed to be a disproportionately large amount of time for the RL model to process compared to image sizes that were not very much smaller, such as the 259×195. The result in Figure 4 (c) shows that by decreasing 20% of input image size we achieve up to 78% energy improvement with 128KB text size while achieving an 82% task completion success rate.

From the VCD traces shown in Figure 5, it appears that this is primarily coming down to usage patterns of both the Cluster-DMA (CDMA) and Micro-DMA (MDMA) units. The CDMA engine is responsible for moving data between L1 and L2 memory, L1 memory being an application-managed scratchpad. The MDMA is responsible for transferring data between the different peripherals attached to the GAP8. For Neural Networks (NN), it is used to move weights from what is known as L3 memory, essentially extra RAM attached to the HyperBus. This RAM is much slower and more power-intensive than the L1/L2 memory but stores significantly more data. Figure 5 (a) shows within the 324×244 image

Table 2. Hardware implementation results of the model extracted from Air-Learning environment on the commercial off-the-shelf devices. We use same input image and text sizes in all platforms to extract results which are 259x195 and 128KB, respectively.

Metric/Platform	Macbook Pro	Nvidia Jetson TX2	Raspberry Pi	Crazyflie GAP8
Latency (ms)	23	70	103	550
Throughput (Labels/Sec)	43.5	14.3	9.7	1.8
Power (W)	N/A	3.9	4.8	**0.36**
Performance (GOPS)	8.4	2.8	1.9	0.4
Energy/Inference (mJ)	N/A	273	495	**198**

input size, both DMA units are active nearly the entire execution time of the NN, thus causing the processor to have to waste cycles waiting for data. Figure 5 (b) shows, model with image input size 259×195 used both DMA engines much less frequently. Also, near the end of the NN inference, the MDMA unit stops being used entirely, while the CDMA unit is accessed much less frequently. Based on power usage by each NN, it was found that the smaller NN used less energy at any given time. However, the total execution time for inference was similar in each case.

We also implemented the most efficient model on several commercial off-the-shelf edge platforms including Raspberry Pi, Nvidia Jetson TX2, and Crazyflie GAP8 to measure throughput, power consumption, and energy efficiency. All platforms were configured to perform at their peak performances. The implementation result is summarized in Table 2. As is illustrated, the server-based implementation achieved the best throughput by producing 43.5 inferences per second when running the model on a MacBook Pro with ARM-based architecture which has 10 Core CPU running at maximum frequency of 3200 MHz and can achieve up to a peak performance of 65 GFlops/Sec. Nvidia Jetson TX2 and Raspberry Pi implementation also achieved 14.3 and 9.7 inferences per second throughput by consuming 273 mJ and 495 mJ energy per inference, respectively. Implementation on GAP8 and the extension board of the Crazyflie resulted in 1.8 inferences per second by consuming only 0.36 Watts of power and 198 mJ energy.

To evaluate the performance of the model in real world, we created a room similar to the Air-Learning environment and deployed the model on the Crazyflie nano drone [12]. In this scenario, the UAV starts at a random location in a room and is instructed to find and navigate to the goal. In addition to the difficulties in implementing GRUs on resource constrained devices, another significant challenge is in transferring trained models from simulation to real world. Although, the simulation environment is created to resemble the real world, some visual differences affect the real world performance leading to a low success rate. Therefore, we had to find controls that work well with Crazyflie in the real world. Additionally, due to the power constraints on Crazyflie, there is a very limited time for the UAV to reach the goal, which led us to consider 200 steps as the terminating point.

Conclusion

In this article, we presented an approach for efficient deployment of language-guided Reinforcement Learning (RL) on resource constrained autonomous systems. We evaluated the performance of the proposed approach by testing it in three RL environments with different levels of complexity, including a realistic environment for autonomous Unmanned Aerial Vehicles (UAVs). By training the model in Air-Learning, a photo-realistic environment and creating a real-world replicate of the environment, we were able to deploy the proposed efficient language-guided RL on a tiny drone named Crazyflie.

Furthermore, we found the most energy-efficient option for real-time deployment of the language-guided RL model. To achieve this goal, we measured power consumption and latency for different image and instruction input sizes on several off-the-shelf devices. In this work, we explored several aspects, including algorithmic design of the language-guided RL model, real-world deployment of the proposed work, and energy-efficient implementation of the model on edge devices. To the best of our knowledge, there

are no similar works that take all these matters into account while implementing an energy-efficient language-guided RL model. Our proposed method is also scalable to multi-agent use cases where several agents can communicate and share information such as location and task completion. We believe that the new wave of deploying RL algorithms on the edge devices, such as autonomous UAVs or robotic applications, along with the increase of open source RL environments and frameworks, will result in great demand for efficient implementation of RL applications and facilitate future research in this direction.

Acknowledgment

We thank Bharat Prakash, Edward Humes and Prakhar Dixit at EEHPC lab for discussion and initial results. We also thank Dr. Vijay Reddi and his team at the Edge Computing Lab, Harvard University for their initial help with the Air-Learning environment. This project was sponsored by the U.S. Army Research Laboratory under Cooperative Agreement Number W911NF2120076.

REFERENCES

1. Pham, H., La, H., Feil-Seifer, D. & Nguyen, L. Autonomous uav navigation using reinforcement learning. *ArXiv Preprint ArXiv:1801.05086*. (2018)
2. Johannink, T., Bahl, S., Nair, A., Luo, J., Kumar, A., Loskyll, M., Ojea, J., Solowjow, E. & Levine, S. Residual reinforcement learning for robot control. *2019 International Conference On Robotics And Automation (ICRA)*. pp. 6023-6029 (2019)
3. Christiano, P., Leike, J., Brown, T., Martic, M., Legg, S. & Amodei, D. Deep reinforcement learning from human preferences. *Advances In Neural Information Processing Systems*. **30** (2017)
4. Hermann, K., Hill, F., Green, S., Wang, F., Faulkner, R., Soyer, H., Szepesvari, D., Czarnecki, W., Jaderberg, M., Teplyashin, D. & Others Grounded language learning in a simulated 3d world. *ArXiv Preprint ArXiv:1706.06551*. (2017)
5. Shiri, A., Mazumder, A., Prakash, B., Manjunath, N., Homayoun, H., Sasan, A., Waytowich, N. & Mohsenin, T. Energy-efficient hardware for language-guided reinforcement learning. *Proceedings Of The 2020 On Great Lakes Symposium On VLSI*. pp. 131-136 (2020)
6. Blukis, V., Terme, Y., Niklasson, E., Knepper, R.A. and Artzi, Y., 2019. Learning to map natural language instructions to physical quadcopter control using simulated flight. arXiv preprint arXiv:1910.09664.(2019)
7. Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves et al. "Human-level control through deep reinforcement learning." nature 518, no. 7540 (2015): 529-533.(2015)
8. Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. Proximal policy optimization algorithms. *ArXiv Preprint ArXiv:1707.06347*. (2017)
9. Chevalier-Boisvert, M., Willems, L. & Pal, S. Minimalistic Gridworld Environment for OpenAI Gym. *GitHub Repository*. (2018), https://github.com/maximecb/gym-minigrid
10. Chevalier-Boisvert, M. gym-miniworld environment for OpenAI Gym. *GitHub Repository*. (2018), https://github.com/maximecb/gym-miniworld
11. Krishnan, S., Boroujerdian, B., Fu, W., Faust, A. & Reddi, V. Air-Learning: a deep reinforcement learning gym for autonomous aerial robot visual navigation. *Machine Learning*. pp. 1-40 (2021)
12. Palossi, D., Loquercio, A., Conti, F., Flamand, E., Scaramuzza, D. & Benini, L. A 64-mw dnn-based visual navigation engine for autonomous nano-drones. *IEEE Internet Of Things Journal*. **6**, 8357-8371 (2019)

Aidin Shiri is currently pursuing the Ph.D. degree with the Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, MD, USA. His research interests mainly focus on energy-efficient digital application-specific integrated circuits and field-programmable gate array hardware accelerator design for deep neural networks and machine learning algorithm for low-power embedded devices.

Mozhgan Navardi is currently a Ph.D. student at the Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, MD, USA. She received the MSc degree in computer engineering from Sharif University of Technology, Tehran, Iran, in 2019. Her research interests include low power embedded systems design and lifetime reliability management.

Tejaswini Manjunath is currently pursuing master's in Computer Science and Engineering at University of Maryland, Baltimore County, MD, USA. She received her bachelor's degree in Computer Science from PES Institute of Technology, Bangalore, India. Her research interests are mainly focused on machine learning, deep reinforcement learning and Computer architecture.

Nicholas Waytowich is a machine learning research scientist at the Human Research and Engineering Directorate at the US Army Research Laboratory. His research interests include human-in-the-loop machine learning, human-autonomy integration, and deep reinforcement learning. Waytowich received a PhD in biomedical engineering from Old Dominion University. He is a member of IEEE and the IEEE Systems, Man, and Cybernetics Society.

Tinoosh Mohsenin is currently an Associate Professor with the Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, where she is also the Director of the Energy Efficient High Performance Computing Lab. She has authored or co-authored over 130 peer-reviewed journal and conference publications. Her research focus is on designing energy efficient embedded processors for machine learning and signal processing, knowledge extraction techniques for autonomous systems, wearable smart health monitoring, and embedded big data computing.