E2EdgeAI: Energy-Efficient Edge Computing for Deployment of Vision-Based DNNs on Autonomous Tiny Drones

Mozhgan Navardi, Edward Humes, Tinoosh Mohsenin
Center for Real-time Distributed Sensing and Autonomy (CARDS)
Computer Science and Electrical Engineering Department
University of Maryland Baltimore County, USA

***Abstract*—Artificial Intelligence (AI) and Deep Neural Networks (DNNs) have attracted attention as a solution within autonomous systems fields as they enable applications such as visual perception and navigation. Although cloud-based approaches have already been highly addressed, there is a growing interest in using both AI and DNNs on the edge as this allows for lower latency and avoids the potential security concerns of transmitting data to a remote server. However, deploying DNNs on edge devices is challenging due to the limited computational power available, as well as energy efficiency being of the utmost importance. In this work, we introduce an approach named *E2EdgeAI* for *E*nergy-*E*fficient *Edge* computing that takes advantage of *AI* for autonomous tiny drones. This approach optimizes the energy efficiency of DNNs by considering the effects of memory access and core utilization on the energy consumption of tiny UAVs. To perform the experiment, we used a tiny drone named Crazyflie with the AI-deck expansion, which includes an octa-core RISC-V processor. The experimental results show the proposed approach reduces the model size by up to 14.4x, improves energy per inference by 78%, and increases energy efficiency by 5.6x. A recorded video for the proposed approach can be found here: Video.**

***Index Terms*—Edge Computing, Autonomous Systems, Obstacle Avoidance, Drone Navigation.**

I. Introduction and Related Work

Nowadays, edge devices such as tiny Unmanned Aerial Vehicles (UAVs) have been utilized in many indoor applications such as search and rescue, gas leak localization, and real-time monitoring, which can be too dangerous or time-consuming for humans to perform [1], [2], [3]. Their small sizes and ability to be equipped with various sensors increase the potential for using tiny UAVs in these applications, as tiny UAVs are safe to operate near humans and are able to access spaces humans cannot easily reach. Additionally, the aforementioned capability to have various sensors allows for easier navigation and accomplishing of mission objectives [4].

To make tiny UAVs smart and autonomous, Deep Neural Networks (DNNs) [2], [3], [5], or Reinforcement Learning (RL) [6], [7], [8], [9], [10] can be deployed to these tiny drones. DNNs help to process claimed data by the camera and other sensors of these drones in order to navigate autonomously and complete a task. However, DNNs are extremely power consuming and require intense computational resources due to the high number of parameters and computations [11]. To deal with these challenges, cloud computing and streaming data between drones and the cloud are proposed as a solution. Although cloud computing brings massive computational capacity, the amount of data that can be sent to the cloud is limited due to the bandwidth. Moreover, security concerns are another potential challenge for such approaches. Therefore, cloud computing is not suitable for real-time, low-latency, or privacy-sensitive applications due to communication latency and potential security concerns. Fortunately, recent works have shown that edge computing supports running DNN models onboard while still meeting these latency and privacy constraints [12], [13].

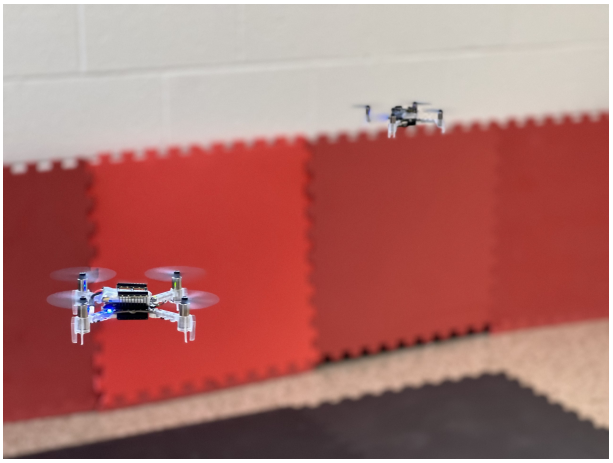

Fig. 1: A tiny UAV called Crazyflie with an AI-deck expansion. The AI-deck includes a GAP8 processor which has eight RISC-V cores able to process in parallel.

The main idea of edge computing is to process the claimed data by edge devices at the edge of the network in order to reduce data transferring and improve energy efficiency [11], [12], [13], [14], [15], [16]. While edge computing brings benefits such as lower communication latency, edge-based DNN deployment is still challenging due to the limitations on power consumption and available resources to perform the processing on the edge. Therefore, it is important to optimize DNNs for latency-aware energy-efficient deployment on the edge devices such as downsampling images [17], [18] and

DNN's sparsity [19]. This work particularly addresses state-of-the-art vision-based DNN models and optimizes them for energy-efficient edge computing.

To take advantage of UAVs in various applications to complete missions, a preliminary problem is autonomous drone navigation. Therefore, in this work, we target drone navigation and take advantage of DNN models and edge computing. We propose an approach named E2EdgeAI which delivers autonomous navigation for drones while still meeting latency and energy efficiency constraints. In summary, the main contributions of this article are:

- Discussing cloud computing challenges and bottlenecks for real-time decision-making in autonomous drone navigation applications.
- Vision-based DNN model optimization to reduce the model size and computation complexity for energy-efficient edge computing.
- End-to-end energy-efficient onboard processing and edge computing while meeting latency constraints.
- Analysing power consumption and latency in regards to DNN deployment on resource-constrained devices, memory access, and core usage.
- The hardware is optimally configured for implementation with respect to latency, throughput, power consumption, and energy efficiency.

To evaluate our approach, we deploy optimized models on an edge device named Crazyflie [5] which is shown in Figure 1. The experimental results show that E2EdgeAI reduces latency by 4x and improves energy efficiency by 5.6x in comparison with the state-of-the-art work.

II. Problem Definition and Motivation

This section defines the problem statement that is discussed in this work. Then, it shows the effectiveness of edge computing with an illustration example.

A. Problem Definition

This work deals with latency and energy efficiency in autonomous systems. The proposed approach tries to reduce latency by reducing the model size and taking advantage of edge computing. To achieve this, it optimizes the model for memory usage to improve both energy efficiency and model performance. The problem can be formulated as follows:

Inputs: Given:

- The state-of-the-art scalable DNN models.
- The multi-core embedded system which runs DNN models in parallel.
- Captured images from an environment by the edge device.

Outputs: Determining a tiny DNN model which has been optimized w.r.t the constraints.

Constraints: Latency, privacy, and energy efficiency are represented as the main constraints.

Objective: The main objective of this work is latency and energy improvement while using edge computing.

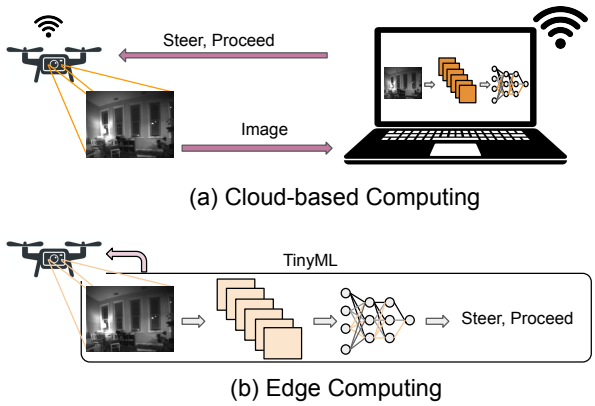

Fig. 2: An example of cloud-based computing and edge computing for drone navigation and obstacle avoidance. (a) The cloud computing approach performs the inference phase in the cloud. The inputs and outputs of the model are sent and received by the drone, respectively. Connection latency and bandwidth as well as security concerns are the two most important challenges in this approach. (b) Edge computing places the model on the edge device, and there are no communication speed or security concerns.

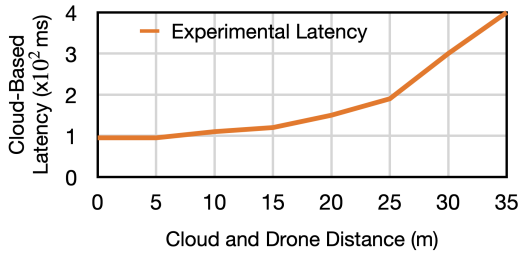

Fig. 3: Cloud-based experimental latency measurements. In this experiment, the DNN model is implemented on a laptop as a cloud and a flying drone sends captured images to the cloud and gets the command.

B. Illustration Example

Figure 2 (a) shows a system overview of cloud-based approaches. Claimed data by drones (images in this example) stream to a laptop through WiFi. DNNs are deployed on a laptop near the drone and the laptop will receive the data, perform processing, and send the result back to the drone. Communication latency in cloud-based approaches increases by increasing the distance between a drone and the WiFi network, and eventually, the drone will be disconnected from the cloud. Figure 3 shows experimental results of measuring computation and communication latency of the inference phase of the cloud-based approach. This latency reaches up to 400 ms after 35 m and there is no connection between the drone and the cloud for distances higher than 35 m. Due to these challenges, edge computing, Figure 2 (b), is proposed as a safer and more reliable approach. In this approach, DNN models have deployed onboard the drone.

III. Proposed Approach: E2EdgeAI

This section presents the proposed approach, E2EdgeAI, that optimizes a DNN model for energy-efficient edge computing. First, it introduces the model architecture and quantization approaches. Then, it discusses how memory access and core usage can affect onboard model latency and energy efficiency.

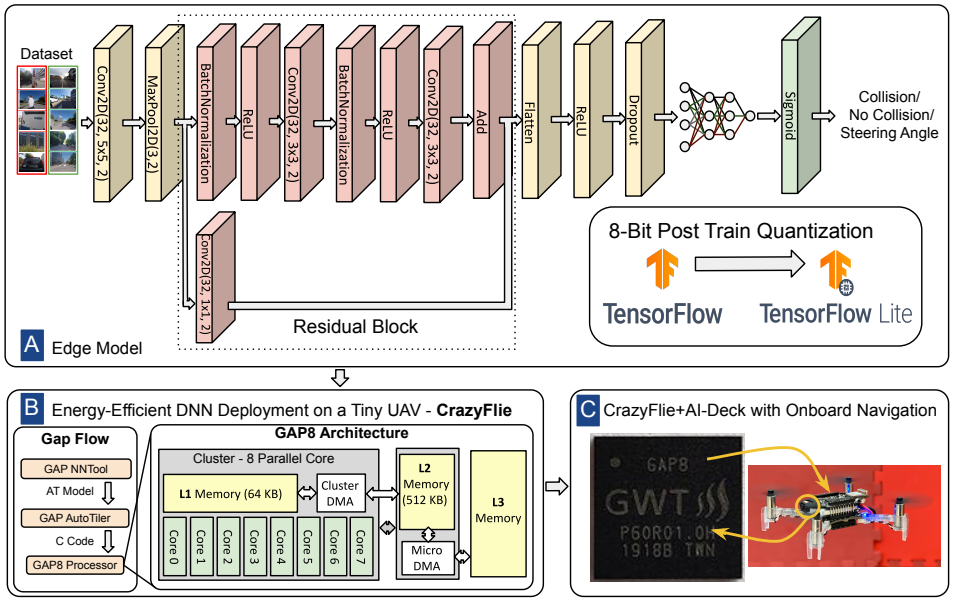

Fig. 4: The proposed approach for edge computing, E2EdgeAI: (a) optimize a DNN model, (b) model deployment on the edge and analysis of memory access and core utilization [20], [5], [21], and (c) onboard autonomous navigation.

A. Vision-Based DNNs Model Optimization

Application & Dataset. In this work, we targeted autonomous tiny-UAV navigation. To do this, we needed to train a model that enables a drone to avoid collision during flight. We applied state-of-the-art DNNs to train a model which can predict the probability of collision. Moreover, it outputs a proper steering angle to avoid obstacles while the drone is flying autonomously based on a single view grayscale camera. The drone receives two outputs from the model and generates proper navigation output accordingly. If the probability of collision generated by the first output is higher than a threshold, the drone will change its direction and steer based on the angle generated by the second output.

Two datasets are used to train a model for collision avoidance and steering from raw grayscale image inputs; (1) a dataset [22] of about 32K images is annotated with 0/1 labels as collision/no collision, respectively, (2) a dataset of Udacity's projects [23] contains 70K images, captured while driving a car by left, center, and right view cameras.

Model Architecture. Since edge devices such as IoT and embedded systems have limited computational power, edge computing demands smaller model sizes with a reduced number of operations. Therefore, we have proposed DNNs that can be scaled with respect to the model size and number of computations to meet the application and hardware implementation requirements. In this work, we trained our DNN model with four scalable models including MobileNet [25] and ResNet [26]. Similar to our proposed work in [3], we considered four different configurations and extracted accuracy, size, and the number of computations. MobileNet complexity can be modified by assigning different values to its width multiplier; 0.25, 0.5, 0.75, 1. For ResNet, we used a customized Resnet and serialized one, two, three, and four residual blocks sequentially. Figure 4 (a) illustrates the customized ResNet with one residual block. Similar to our previous work for timeseries audio input [27] and image input [28] classification and also research works in [17], [18], we consider different input image sizes as a factor for reducing the model size and the number of operations.

Edge Model. As an example of a simple point of difference that can have dramatic consequences, most neural networks are designed and trained using what are known as Floating Point (FP) numbers, which are how computers represent the set of real numbers. Due to FP numbers being incredibly common in modern software, most desktop and server processors have been optimized for FP arithmetic to the point that they can typically perform these operations at the same speed or faster than integer arithmetic operations. However, in order to save on size, weight, and power (SWaP) constraints, it is very common for embedded processors to lack an FP unit, instead only supporting integer arithmetic. Even if they do contain an FP unit, these processors are typically still slower at performing FP arithmetic than they are at performing integer arithmetic. Thus, in order to execute neural networks at any reasonable level of performance on embedded devices, one needs to convert these neural networks to only use integers rather than FP numbers in a process known as quantization. There are two common approaches for quantization: Post Training Quantization (PTQ) and Quantization Aware Training (QAT). We applied PTQ to the optimized model with the help of TensorFlow Lite to convert the model from a 32-bit FP to an 8-bit quantized model. In this way, not only is this model suitable for edge implementation but also has a smaller storage size and less memory usage. In the next section, we analyze energy efficiency and inference latency for the optimized models.

TABLE I: Average download speed, upload speed, and power model parameters of wireless networks [3], [24]

	Download/Upload Speed		Power Model		
Network Type	Download Speed (Mbps)	Upload Speed (Mbps)	α_u (mW/Mbps)	α_d (mW/Mbps)	β (mW)
WiFi	55	19	283	137	133

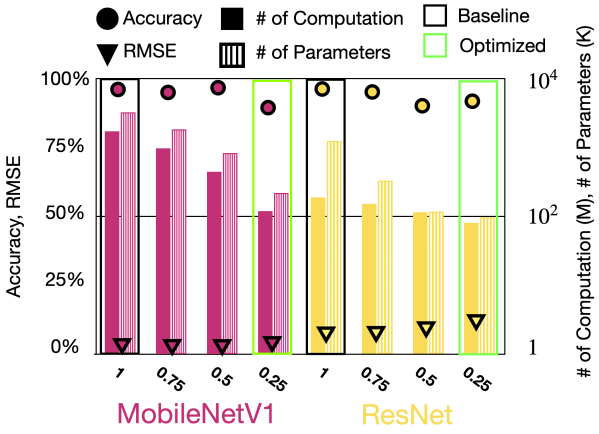

Fig. 5: Training vision-based obstacle detection DNN model with MobileNetV1[25] and ResNet [5], [22], [26] architectures for four different scale sizes with image input size 324x244. ResNet and MobilNet model sizes are optimized at about 14x and 12x in comparison with the baseline, respectively. [3]

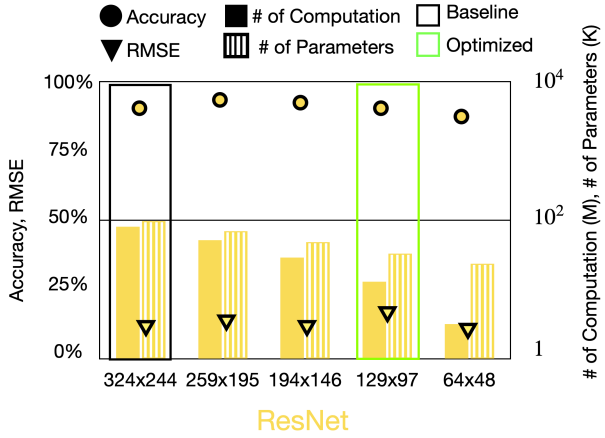

Fig. 6: ResNet with a scale size of 0.25 is optimized in terms of input size. In the ResNet model, input size 129x97 was found to have acceptable accuracy while reducing the model size by about 3x.

B. Edge Computing Resource Optimization

Latency and Power Analysis. Obstacle detection for collision avoidance plays a key role in autonomous drone navigation. Vision-based DNN models are a promising solution in this area. However, the communication latency, bandwidth, and power consumption are significant parameters in real-time decision making for drone navigation applications. For obtaining these parameters, we refer to standard download and upload speed in average and communication power consumption for wireless networks [3], [24]. With the help of Table I parameters, the latency and power consumption through WiFi can be theoretically based on the equation 1:

$$\begin{aligned} p_{uplink} &= \alpha_u \times upload\_speed + \beta, \\ p_{downlink} &= \alpha_d \times download\_speed + \beta \end{aligned} \tag{1}$$

where $upload\_speed$ and p_{uplink} are the average speed and power consumption from the drone transferring data to the cloud through WiFi. $download\_speed$ and $p_{downlink}$ represent the download speed and power consumption, respectively. α_u, α_d and β factors determine power consumption in mW/Mbps of data transfer. In order to reduce latency and power consumption, we trained DNN models which are optimized for size and the number of computations. Therefore, they can be applied to resource-constrained devices such as tiny drones.

Hardware Architecture. Figure 4 illustrates how E2EdgeAI deploy a DNN models on a resource-constrained processor named GAP8 [20], [5], [21]. Figure 4 (b) and (c) shows that Crazyflie consists of a GAP8 processor. It has a RISC-V based PULP platform with two computes domains: (1) a Fabric Controller (FC) for controlling tasks and 512KB L2 memory and (2) a cluster domain with 8 cores for parallel computation of highly demanding workloads and 64KB directly accessible L1 memory. In this work, we utilized the GAPFlow toolchain, which includes the 2 programs known as NNTOOL and AutoTiler. NNTOOL is responsible for performing adjustments to the DNN architecture, and converting it into a format AutoTiler can use, as well as converting the weights into a format that can be flashed to the GAP8. AutoTiler is responsible for algorithmically determining the best possible memory layout for the DNN, as well as converting the model operations into C code that can be compiled for the GAP8. While we were able to automate most of this process, certain DNNs will often require some level of tinkering in order to be converted correctly. For instance, adjusting the maximum stack sizes for the FC and cluster cores, as some networks allocate more data on the stack than others. Similarly, the heap space will often need to be adjusted for a specific DNN, by default AutoTiler will attempt to allocate the entire system's L1 and L2 memory for use in the DNN, however, this will inevitably result in heap overflows, data structures being corrupted in memory, and a clobbered stack, as this data is all stored in memory that would inevitably get overwritten during neural network execution. Additionally, the RTOS used by the GAP8 processor will frequently make its own heap allocations before the DNN starts up, leaving less space than expected by the DNN. We analyzed how GAP Flow and memory allocation can affect power consummation and latency on different DNN models and presented an energy-efficient model for edge computing drone navigation. In the next section, we showed model optimization results and analyzed them on the edge DNN implementation. Moreover, we discussed latency and energy efficiency improvement.

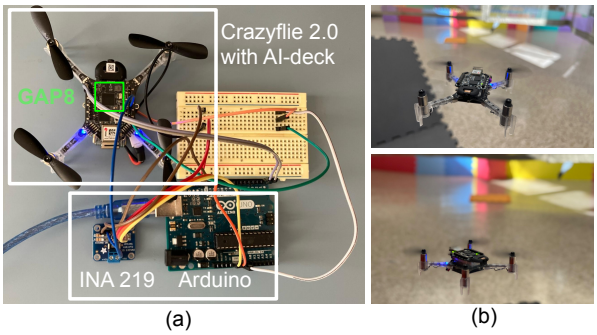

Fig. 7: (a) GAP8 processor on the AI-deck attached to the Crazyflie with power measurement setup. INA219 and Arduino measure the GAP8 power consumption. [6] (b) Crazyflie running the baseline model in action for obstacle detection and avoidance.

TABLE II: Hardware implementation results that compare E2EdgeAI with the presented work in [3]. The experiment was run with the newest NINA firmware [29] to deploy the model on the GAP8 processor and to extract latency and power consumption.

Metric/Approach	[Navardi'22] [3]	E2EdgeAI
Inference Latency (ms)	40	10
Throughput (Inference/Sec)	25	100
Power Consumption (mW)	470	400
Performance (GOPS)	1	1.3
Energy Efficiency (GOPS/W)	55	323
Energy/Inference (mJ)	18.8	4

IV. Implementation and Experimental Results

A. Software Experimental Results

This section shows the result of training an obstacle detection and steering angle DNN model. We used the ResNet and MobileNet architectures with four different scaled model sizes and six different image input sizes as we discussed in section III-A. Figure 5 illustrates that by decreasing the model size of both architectures, the number of computations and parameters is decreased. However, as an optimized ResNet model has the lowest computation complexity, being 14x lower than the baseline. Therefore, we chose this model and analyzed the effect of changing the input size on the model's accuracy. Figure 6 shows a ResNet model with one residual block and an image input size of 129x97 that has about 3x lower complexity while the accuracy is approximately the same as the baseline, being equal to 92%. We applied PTQ on the ResNet model with different image input sizes to analyze the latency and energy efficiency of the GAP8 processor.

B. GAP8 Edge-Device Implementation Results and Analysis

In order to evaluate the E2EdgeAI approach, we deployed the trained model on the GAP8 processor. Moreover, we compare the energy efficiency and latency of E2EdgeAI and a state-of-the-art work proposed in [3]. To do a fair comparison, we repeated the measurements for the edge model proposed in [3] to have results for the same configurations. Figure 7 (a) shows the power measurement setup that we used in these papers [3], [6]. Latency, power consumption, and energy efficiency of the proposed E2EdgeAI model are reported in

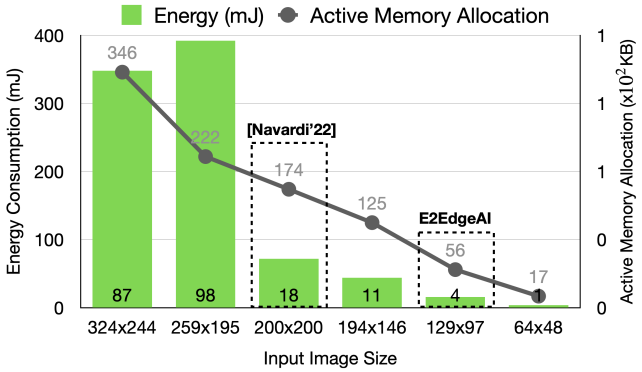

Fig. 8: Average power consumption and active memory allocation during the inference phase of the ResNet model with different input image sizes. The E2EdgeAI model is compared with the proposed work in [3]

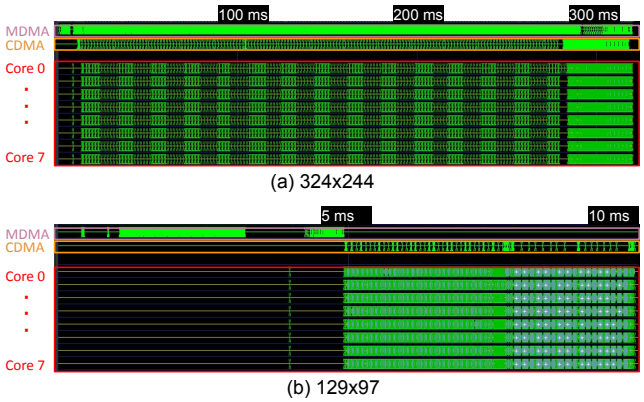

Fig. 9: The VCD trace [30] during the inference phase of the ResNet model with two different image input sizes: (a) 324x244, (b) 129x97.

Table II. The results show E2EdgeAI is about 4x faster and 5.8x more energy-efficient than the presented model in [3]. Figure 8 analyzes the effect of input size on the active memory allocation and energy consumption during the inference phase. In order to measure the active memory allocation, we extracted the maximum amount of memory used at any given time during the inference. While the result for [3] shows that it used both L1 and L2 memory as the active memory allocation size is higher than L1 memory size (64Kb), E2EdgeAI could successfully compress the model into 56 Kb which can be fitted on the local memory (L1) of the GAP8 processor. Therefore, E2EdgeAI leads to lower energy consumption equal to 4mJ. Figure 7 (b) shows running baseline model in action for obstacle detection and avoidance.

In order to show how the energy efficiency would vary with different levels of DNN complexities, we extract the VCD trace [30]. Figure 9 illustrates VCD trace includes the Cluster-DMA (CDMA), Micro-DMA (MDMA), and core utilization for the base model of ResNet (324x244) and the optimized E2EdgeAI model (129x97). 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 DNNs, 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 [6]. In Figure 9 (a) both MDMA and CDMA units are active nearly the entire inference phase, thus causing the processor to have to waste cycles waiting for data. However, results for the optimized model, Figure 9 (b), are much less frequent for both DMA units. Also, near the end of the inference, the MDMA unit stops being used entirely, while the CDMA unit is accessed much less frequently which leads to lower energy consumption.

V. Conclusion

In this paper, we proposed an approach named E2EdgeAI for energy-efficient edge computing in tiny drones which utilized Deep Neural Networks (DNNs) to navigate autonomously. We evaluated the throughput and accuracy of the proposed model, E2EdgeAI. Moreover, we implemented this model on a drone with Artificial Intelligence (AI) capabilities called Crazyflie. Then, we extracted latency, power consumption, energy per inference, performance, and energy efficiency for the proposed approach and state-of-the-art work to compare. Experimental results showed 14.4x less model complexity, 5.6x more energy efficiency, and 78% energy per inference improvement.

VI. Acknowledgment

We thank Griffin Bonner, Haoran Ren, Aidin Shiri and Tejaswini Manjunath for initial discussions and experiments. This project was sponsored by the U.S. Army Research Laboratory under Cooperative Agreement Number W911NF2120076.

References

[1] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, "Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges," *Ieee Access*, vol. 7, pp. 48 572–48 634, 2019.

[2] B. P. Duisterhof, S. Krishnan, J. J. Cruz, C. R. Banbury, W. Fu, A. Faust, G. C. de Croon, and V. J. Reddi, "Tiny robot learning (tinyrl) for source seeking on a nano quadcopter," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 7242–7248.

[3] M. Navardi, A. Shiri, E. Humes, N. R. Waytowich, and T. Mohsenin, "An optimization framework for efficient vision-based autonomous drone navigation," in *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 2022, pp. 304–307.

[4] H. Müller, V. Niculescu, T. Polonelli, M. Magno, and L. Benini, "Robust and efficient depth-based obstacle avoidance for autonomous miniaturized uavs," *arXiv preprint arXiv:2208.12624*, 2022.

[5] D. Palossi, A. Loquercio, F. Conti, E. Flamand, D. Scaramuzza, and L. Benini, "A 64-mw dnn-based visual navigation engine for autonomous nano-drones," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8357–8371, 2019.

[6] A. Shiri, M. Navardi, T. Manjunath, N. R. Waytowich, and T. Mohsenin, "Efficient language-guided reinforcement learning for resource constrained autonomous systems," *IEEE Micro*, 2022.

[7] M. Navardi, P. Dixit, T. Manjunath, N. R. Waytowich, T. Mohsenin, and T. Oates, "Toward real-world implementation of deep reinforcement learning for vision-based autonomous drone navigation with mission," *UMBC Student Collection*, 2022.

[8] B. Prakash, N. Waytowich, T. Oates, and T. Mohsenin, "Towards an interpretable hierarchical agent framework using semantic goals," *AAAI Fall Symposium 202 AI-HRI*, 2022.

[9] B. Prakash, N. Waytowich, T. Mohsenin, and T. Oates, "Automatic goal generation using dynamical distance learning," *arXiv preprint arXiv:2111.04120*, 2021.

[10] B. Prakash, N. Waytowich, T. Oates, and T. Mohsenin, "Interactive hierarchical guidance using language," *arXiv preprint arXiv:2110.04649*, 2021.

[11] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge ai: On-demand accelerating deep neural network inference via edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 447–457, 2019.

[12] Z. Chang, S. Liu, X. Xiong, Z. Cai, and G. Tu, "A survey of recent advances in edge-computing-powered artificial intelligence of things," *IEEE Internet of Things Journal*, 2021.

[13] G. Premsankar and B. Ghaddar, "Energy-efficient service placement for latency-sensitive applications in edge computing," *IEEE internet of things journal*, vol. 9, no. 18, pp. 17 926–17 937, 2022.

[14] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[15] A. N. Mazumder and T. Mohsenin, "A fast network exploration strategy to profile low energy consumption for keyword spotting," *CoRR*, vol. abs/2202.02361, 2022. [Online]. Available: https://arxiv.org/abs/2202.02361

[16] A. N. Mazumder, J. Meng, H.-A. Rashid, U. Kallakuri, X. Zhang, J.-s. Seo, and T. Mohsenin, "A survey on the optimization of neural network accelerators for micro-ai on-device inference," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2021.

[17] H. Genc *et al.*, "Flying iot: Toward low-power vision in the sky," *IEEE Micro*, vol. 37, no. 6, pp. 40–51, 2017.

[18] D. Palossi, N. Zimmerman, A. Burrello, F. Conti, H. Müller, L. M. Gambardella, L. Benini, A. Giusti, and J. Guzzi, "Fully onboard ai-powered human-drone pose estimation on ultralow-power autonomous flying nano-uavs," *IEEE Internet of Things Journal*, vol. 9, no. 3, pp. 1913–1929, 2021.

[19] L. Lamberti, V. Niculescu, M. Barciś, L. Bellone, E. Natalizio, L. Benini, and D. Palossi, "Tiny-pulp-dronets: Squeezing neural networks for faster and lighter inference on multi-tasking autonomous nano-drones," in *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 2022, pp. 287–290.

[20] E. Flamand, D. Rossi, F. Conti, I. Loi, A. Pullini, F. Rotenberg, and L. Benini, "Gap-8: A risc-v soc for ai at the edge of the iot," in *2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. IEEE, 2018, pp. 1–4.

[21] GreenWavesTechnologies, "Gap8 processor architecture. [online]," *Available: https://greenwaves-technologies.com/manuals/BUILD/HOME/html/index.html*.

[22] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza, "Dronet: Learning to fly by driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.

[23] Udacity, "An open source self-driving car," in *https://www.udacity.com/self-driving-car*, 2016.

[24] A. E. Eshratifar and M. Pedram, "Energy and performance efficient computation offloading for deep neural networks in a mobile cloud computing environment," in *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, 2018, pp. 111–116.

[25] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[27] H.Ren *et al.*, "End-to-end scalable and low power multi-modal CNN for respiratory-related symptoms detection," in *2020 IEEE 33rd International System-on-Chip Conference (SOCC 2020)*, 2020, in press.

[28] M. Khatwani, H.-A. Rashid *et al.*, "A flexible multichannel eeg artifact identification processor using depthwise-separable convolutional neural networks," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 2020.

[29] Bitcraze, "Firmware running on the esp32 nina w102 module of the ai-deck," in *https://github.com/bitcraze/aideck-esp-firmware*, 2021.

[30] GreenWavesTechnologies, "Gvsoc virtual platform for gap8 processor. [online]," *Available: https://greenwaves-technologies.com/manuals/BUILD/GVSOC/html/index.html*.