Power- and Deadline-Aware Dynamic Inference on Intermittent Computing Systems

Hengrui Zhao, Lei Xun, Jagmohan Chauhan, Geoff Merrett
School of Electronics and Computer Science, University of Southampton, UK
Email: {hz20u22, l.xun, j.chauhan}@soton.ac.uk, gvm@ecs.soton.ac.uk

***Abstract*—In energy-harvesting intermittent computing systems, balancing power constraints with the need for timely and accurate inference remains a critical challenge. Existing methods often sacrifice significant accuracy or fail to adapt effectively to fluctuating power conditions. This paper presents DualAdaptNet, a power- and deadline-aware neural network architecture that dynamically adapts both its width and depth to ensure reliable inference under variable power conditions. Additionally, a runtime scheduling method is introduced to select an appropriate subnetwork configuration based on real-time energy-harvesting conditions and system deadlines. Experimental results on the MNIST dataset demonstrate that our approach completes up to 7.0% more inference tasks within a specified deadline, while also improving average accuracy by 15.4% compared to the state-of-the-art.**

***Index Terms*—dynamic deep learning inference, intermittent computing, energy-harvesting, deadline-aware inference**

I. Introduction

Energy-harvesting systems have emerged as a promising solution for powering embedded and IoT devices in environments where traditional power sources are impractical or unavailable [1]. By converting ambient energy sources, such as solar, thermal, or radio frequency signals, into electrical energy, these systems enable devices to operate autonomously over extended periods [2]. In recent years, researchers have explored intermittent systems, where only small capacitors are used to buffer harvested energy: when the energy depletes, the system dies and remains in this state until it is recharged. [3].

Integrating deep neural networks (DNNs) into these energy-harvesting intermittent systems amplifies their utility by enabling on-device intelligence for real-time data processing and decision-making. DNNs that have proven their capacity to learn and generalize from complex datasets, are pivotal in applications such as environmental monitoring, healthcare, and smart agriculture [4]. However, conventional DNNs have been designed with fixed architectures, where the number of layers and neurons within each layer are predetermined during the training phase. When deployed in energy harvesting systems, these static networks face substantial limitations. Due to the unpredictable nature of energy availability, there may be insufficient power to complete an inference task, leading to either incomplete execution or significant delays [5]. Moreover, these traditional networks lack the flexibility to adjust their computational demands in real-time, which can result in inefficient energy usage and degraded system performance, especially under variable energy conditions. Recently, dynamic DNNs have garnered significant attention due to their ability to adapt computational complexity in response to resource constraints [6]. Unlike traditional DNNs, dynamic DNNs can adjust their width (the number of channels per layer) [7] or depth (the number of layers) [8] during execution. This flexibility makes them well-suited for deployment in energy-constrained environments, such as those powered by energy-harvesting systems, where available resources can vary significantly over time.

However, deploying dynamic neural networks in intermittent computing systems is challenging. One of the primary difficulties is the limited adaptability of current dynamic DNNs to the variability and randomness of environmental energy sources. Most existing approaches [9]–[12] rely on multi-exit networks to adjust computational load by modifying network depth. However, networks deployed in intermittent systems are typically lightweight, with only a few layers, making depth adjustment alone insufficient to provide the adaptability required. Thus, achieving deadline-aware intermittent execution demands more flexible neural network architectures capable of better handling energy fluctuations.

Another significant challenge is selecting the most appropriate neural network configuration at runtime. The system must dynamically choose a network configuration that balances the trade-off between computational complexity and energy availability. This selection process is complicated by the need to meet strict timing constraints, as inference tasks often have deadlines that must be respected to maintain system performance and reliability [12]. Consequently, an effective runtime scheduling method must be capable of accurately measuring energy availability, assessing the computational demands of different network configurations, and making informed decisions to ensure timely and correct task completion. In this paper, we address these challenges through the following contributions:

- DualAdaptNet, a novel DNN model which can adjust its structure in both depth and width dimensions, providing 8 operating points. Compared to previous adaptive models, DualAdaptNet exhibits enhanced adaptability, providing finer-grained scaling to accommodate fluctuating energy.
- A model-based runtime scheduler for sequenced task execution that utilizes real-time power measurements to select the optimal neural network configuration dynamically to meet a specific and runtime changeable deadline.
- We practically validate our approach on an MSP430 microcontroller and the MNIST dataset, demonstrating completion of up to 7.0% more inference tasks within a specified deadline while improving accuracy by 15.4% compared to the state-of-the-art.

978-3-9826741-0-0/DATE25/© 2025 EDAA

II. Related Work and Motivation

A. Intermittent Computing Systems

Many attempts have been made to ensure the correct forward progress of conventional applications on intermittent systems. Checkpointing [13], [14] is one of the earliest methods, where the system saves volatile state to non-volatile memory at each checkpoint. Another approach is just-in-time or reactive checkpoints, storing volatile states just before a power outage [15]. Task-based intermittent system frameworks [16] divide applications into tasks of appropriate size based on the energy budget, executing them atomically.

SONIC & TAILS [17] is the first to successfully deploy DNNs on intermittent systems by splitting the network into sequential, atomic tasks and automatically saving inference progress at each task's end. Despite this advancement, limitations remain: interrupted tasks must be re-executed, increasing overhead, and reliance on accurate energy budget estimates can lead to endless re-execution if estimates are incorrect. Recently, footprint-based approaches for intermittent inference, such as HAWAII [18], have been proposed. These methods save footprints alongside accelerator sub-operation results to track progress. Upon power resumption, the latest footprint determines where to resume, reducing overhead since only a limited number of sub-operations may need re-execution. In this work, we extend HAWAII [18] to dynamically adjust the network architecture while preserving inference progress.

B. Dynamic Deep Neural Networks

Dynamic neural networks, which adjust their structure during runtime, have been proposed to enhance efficiency and adaptability under resource constraints [6], [19]. Techniques include dynamically changing network depth by using early exits that allow inference to stop at intermediate layers when sufficient confidence is achieved [8], or adjusting network width by activating subsets of neurons or channels to balance performance and computational cost [7], [20].

The emergence of intermittent computing frameworks has made it possible to run Dynamic DNNs intermittently. Several works have applied dynamic neural networks to adaptively adjust the network structure in response to transient energy fluctuations. ePerceptive [10] dynamically adjusts inference quality based on instantaneous energy through multi-resolution inputs and multiple exits. HarvNet [9] utilizes multi-exit neural networks and supercapacitors for always-on DNN inference. Wu et al. [11] combine reinforcement learning and multi-exit networks for long-term energy management optimization. Zygarde [12] employs specialized scheduling and early exit strategies to ensure DNN inference completes before deadlines. While these approaches predominantly focus on adjusting network depth via multi-exit architectures, this limits the model's overall adaptability. Exploring alternative strategies, such as dynamically adjusting the network's width during runtime, offers significant potential to enhance adaptability to energy and time constraints.

C. Motivation

One of the most adaptive methods in existing work is ePerceptive [10], which adjusts network complexity by changing input resolutions and allowing early exits. To evaluate its performance, we applied ePerceptive [10] to the MNIST dataset [21] using three different input resolutions: 14×14, 20×20, and 28×28, with two exit points in the network. As a baseline, we used the same backbone network to train a two-exit network without multi-resolution inputs. Both models were implemented on the MSP430FR5994 microcontroller and evaluated for accuracy and inference latency, with the results shown in Fig. 1.

While ePerceptive increases the number of sub-networks from two to five by introducing different input resolutions, this flexibility comes at the cost of accuracy with a maximum accuracy drop of 22.76% compared to the baseline. This highlights a critical limitation: while ePerceptive improves model adaptability, it suffers from significant accuracy degradation. This motivates the need for more flexible approaches that can adapt to changing conditions without such a significant trade-off in accuracy, which is the focus of our work.

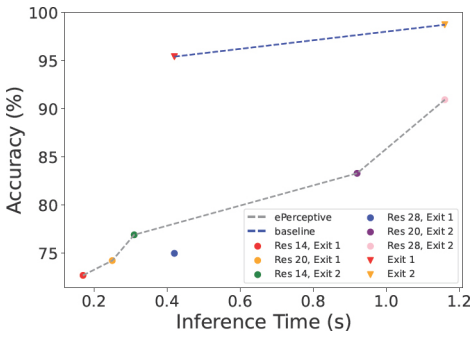


Fig. 1: Comparison of ePerceptive [10] with a baseline.

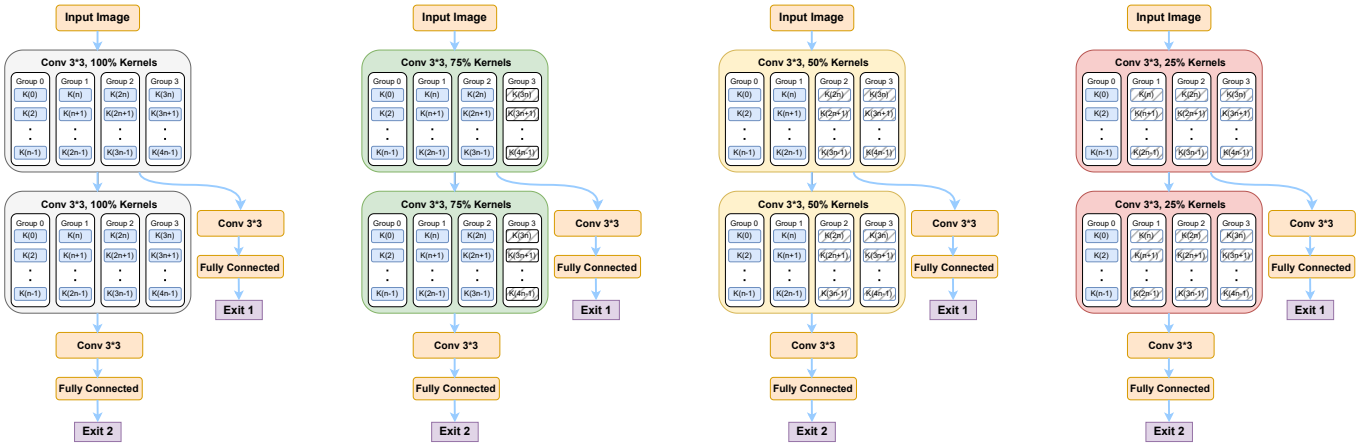
III. Methodology

This section introduces the architecture and training algorithm of DualAdaptNet, along with an extension of HAWAII [18] for deploying this network. Additionally, we present a novel runtime scheduler that dynamically selects the optimal network configuration to meet inference deadline requirements by applying real-time power measurements to a model.

A. DualAdaptNet: Flexible Model Adaptation

The objective of DualAdaptNet is to adapt DNN models to energy and time constraints on energy harvesting systems. To achieve this objective, we designed a specialized network architecture and training algorithm that enables the model to scale its width and depth simultaneously.

1) Model architecture: The inference paths and specific network structures for all sub-networks of DualAdaptNet are shown in Fig. 2. DualAdaptNet's backbone network contains three convolutional layers, with an early exit (Exit 1) placed after the first convolutional layer. To achieve a width-variable model, we have specialized the convolutional layers of the



(a) 100% width (b) 75% width (c) 50% width (d) 25% width

Fig. 2: All sub-network structures of DualAdaptNet. The four different widths network can be changed through two different exits, thus changing the depth of the network. In total, DualAdaptNet can be scaled up to eight different-sized sub-networks.

backbone network. The first two convolutional kernels are divided into four groups, which can be selectively activated during inference to change the width of the generated feature map. The width of DualAdaptNet refers to the width of these convolutional layers. For example, The minimum size that DualAdaptNet can scale to is named "25% Width, Exit 1". In this smallest sub-network, only one group of kernels in the first convolutional layer are activated and the network exits prematurely at Exit 1.

2) Training algorithm: When a static DNN model exits early or drops the convolutional kernel in the middle layer, the accuracy of inference drops substantially due to the loss of features. DualAdaptNet requires a dedicated training algorithm to enable it to have acceptable inference accuracy even after changing the depth and width of the model. The specific training algorithm is shown in Algorithm 1.

Algorithm 1 Training algorithm for DualAdaptNet

Require: Loss function weights $w1$,$w2$, termination criterion
1: Activate only group 0 kernels
2: Train the model with $L_{\text{total}} = w_1 L_1 + w_2 L_2$
3: **if** Inference accuracy $>$ Termination Criterion **then**
4: Stop training and activate group 1 kernels
5: **end if**
6: Repeat steps 2 to 5 until all four groups of kernels are activated and trained

The loss function L_{total} is used to calculate the weighted sum of the loss functions of the two exits (L_1, L_2). The weights of the loss functions for Exit 1 (w_1) and Exit 2 (w_2) are set to 0.3 and 0.7, respectively. At the beginning of training, only the kernels of group 0 are activated in DualAdaptNet. The next kernel group is activated when the model's classification accuracy on the MNIST test set meets a preset criterion (98%). This process will be repeated until all convolutional kernels are activated and trained.

3) Model Architecture Manager (MAM): Before deploying DualAdaptNet to intermittent systems, two main challenges must be addressed: 1) preserving inference progress across power intervals, and 2) efficiently scaling the network. To solve these, we extended HAWAII [18] with a Model Architecture Manager (MAM), which includes a width and depth manager. MAM uses inference breakpoints recorded by HAWAII [18] to select the appropriate network layer for processing. We designed the width manager by analyzing the effect of dropping convolutional kernels. When kernels are discarded, the output channels of the first convolutional layer are reduced, while input channels remain unchanged. For subsequent layers, both input and output channels are reduced proportionally. The width manager adjusts channels accordingly, while the depth manager decides whether to skip Exit 1 based on depth requirements.

B. The Runtime Scheduler

To enable the network to adaptively choose the optimal configuration based on energy and deadline constraints, we introduce a runtime scheduler. This scheduler dynamically selects the network configuration required to meet a specified inference deadline by modelling the device's active time based on real-time power measurements, and comparing with the latencies of different operating points.

1) Inference time modelling: We first analyzed power cycles in intermittent systems using an MSP430FR5994 microcontroller and a low-current source (simulating energy harvesting). Fig. 3 summarizes the V_{cc} voltage variation experienced during one energy cycle, which has two phases: charging and discharging. When the voltage across the capacitive energy store rises above the microcontroller's shutdown voltage (V_{off}), the system runs initialization routines, then enters low-power mode to accelerate charging. Once voltage surpasses the upper threshold (V_{on}), task execution begins, causing the voltage to

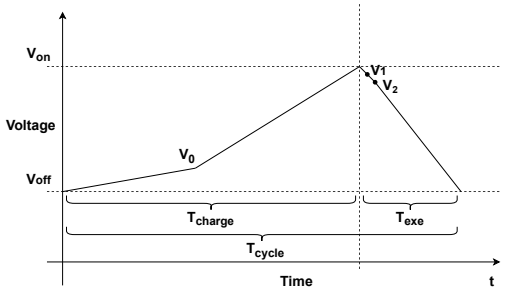


Fig. 3: A typical operating cycle in intermittent systems.

drop. Details of the voltage monitoring system are provided in the next section. During a power cycle, the total operation time consists of the charging time (T_{charge}) and the execution time (T_{exe}). To ensure that DNN inference is completed before a given deadline, the effective execution time required for DNN inference must be accomplished within the stipulated time. The system's current consumption is measured during the initialisation routine (I_{init}), in low-power mode (I_{lpm}), and during DNN inference (I_{exe}) under different voltage supplies. The results show that when the supply voltage varies between 1.8V and 3.6V, the current varies by only 2.5% during the initialisation routine, 2.1% in low-power mode, and 3.2% during DNN inference. Since these variations are minimal, they are ignored in the formulas below, and average current values are used. When current draw remains constant, the relationship between runtime, voltage change, and input current can be expressed as follows:

$$T = \frac{C \times \Delta V}{\Delta I} \tag{1}$$

Where C is the capacitance of the system capacitor, ΔI is the difference between the input current (I_{in}) and the consumption current, and ΔV is the voltage change. Let V_0 denote the voltage after the execution of the initialisation routine and T_{init} denote the running time of the routine. Through experimental measurements, T_{init} is found to be constant, hence V_0 can be expressed as:

$$V_0 = \frac{T_{init}(I_{in} - I_{init})}{C} + V_{off} \tag{2}$$

The value of V_0 is crucial. When V_0 is greater than the system's required upper voltage threshold (V_{on}), the system will skip entering low-power mode and directly start executing DNN inference tasks. Conversely, when V_0 is less than V_{on}, the time the system spends in low-power mode can be expressed as follows:

$$\begin{aligned} T_{lpm} &= \frac{C(V_{on} - V_0)}{I_{in} - I_{lpm}} \\ &= \frac{C(V_{on} - V_{off}) - T_{init}(I_{in} - I_{init})}{I_{in} - I_{lpm}} \end{aligned} \tag{3}$$

Therefore, the time for the charging phase is:

$$T_{charge} = \begin{cases} T_{init} & , V_0 > V_{on} \\ T_{init} + T_{lpm} & , V_0 < V_{on} \end{cases} \tag{4}$$

The time for the DNN inference phase, which is the discharging phase of the power cycle, can be represented as:

$$T_{exe} = \frac{C(V_{on} - V_{off})}{I_{exe} - I_{in}} \tag{5}$$

Based on equations 3, 4, and 5, the total time of a power cycle is:

$$\begin{aligned} T_{cycle} &= T_{charge} + T_{exe} \\ &= \begin{cases} T_{exe} + T_{init} & , V_0 > V_{on} \\ T_{exe} + T_{lpm} + T_{init} & , V_0 < V_{on} \end{cases} \end{aligned} \tag{6}$$

In equation 6, all other values are fixed except for the input current I_{in}. Therefore, it can be seen that the magnitude of the input current determines the charging time and execution time within a power cycle. In intermittent systems, we can infer how many power cycles are needed to complete one effective DNN inference based on the input current value.

2) Voltage Monitoring Module: To ensure proper operation and real-time current measurement in the intermittent system, we utilised a voltage monitoring module from [22] to implement two key functions: exiting low-power mode when the voltage exceeds the upper threshold and calculating the system input current by detecting voltage changes over time. As shown in Fig. 4, the module uses a voltage divider, voltage comparator, and ADC for real-time monitoring. The voltage divider consists of three 200k-ohm resistors, reducing the system voltage to one-third. As the energy harvester charges the capacitor, the system voltage (V) increases. When the divided voltage exceeds 1.2V (upper threshold set to 3.6V), the voltage comparator triggers an interrupt, exiting low-power mode.

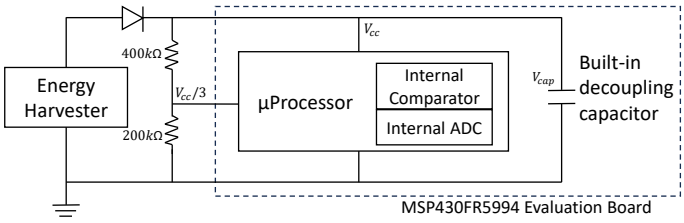


Fig. 4: The schematic of the test platform.

After the system exits low-power mode, the system reads an ADC value before and after a test computation, a finite impulse response (FIR) operation in our experiments. The input current of the system can be obtained from the difference between these two voltage readings. According to equation 1, the input current can be calculated as:

$$I_{in} = \frac{C(V_1 - V_2)}{T_{FIR}} + I_{FIR} \tag{7}$$

Where V_1 and V_2 are the voltage readings before and after the test program respectively, T_{FIR} is the execution time of

the program, and I_{FIR} is the current consumption during the execution of the program.

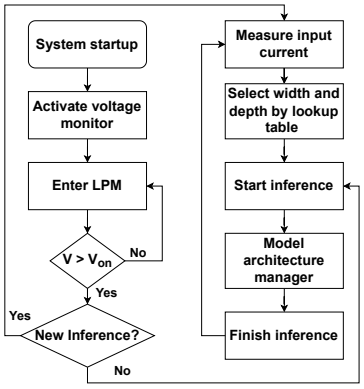


Fig. 5: The workflow of the proposed deadline-aware intermittent inference.

C. DualAdaptNet adaption based on input current

Our model establishes that inference time is directly influenced by the current supply. To evaluate the performance of our adaptive model, we conducted tests on each sub-network of DualAdaptNet under varying current supplies. Given that the system operates continuously when the current exceeds 2.3 mA and fails to start when the current is less than 0.3 mA, we focus our testing on currents between 0.3-2.3 mA, with increments of 0.1 mA. For each of the eight operating points of DualAdaptNet, we recorded the voltage difference and inference delay under different input currents. This comprehensive testing allowed us to capture a detailed performance profile for each sub-network. Based on the collected data, we established a look-up table to select the appropriate network size based on the current supply to meet an inference deadline. The table divides the voltage difference into 7 levels, each corresponding to a sub-network capable of completing DNN inference within the required time (there is a sub-model that is not on the Pareto front). As shown in Fig. 5, the system measures the voltage difference before starting a new inference and adjusts the network's width and depth according to the lookup table.

IV. Experimental Evaluation

To validate the effectiveness of DualAdaptNet and its associated runtime scheduler, we compared our approach with two state-of-art methods: ePerceptive [10] and Zygarde [12]. Our experiments were designed to assess performance under both continuous and intermittent power supply conditions. By measuring and analyzing the performance of all three methods across these scenarios, we aim to demonstrate the advantages of our approach in maintaining reliable and efficient inference in energy-constrained environments.

A. Experiment setup

The Texas Instruments MSP430FR5994 microcontroller is used as the experimental device, which contains a low-energy accelerator to provide parallel computing operations. All three models were trained using the same backbone network on the MNIST dataset [21] to ensure a fair and consistent comparison. The primary difference between our method and ePerceptive is in how each approach adjusts computational complexity. DualAdaptNet dynamically scales both the width and depth of the network, allowing it to optimize its architecture in response to the current resource availability. In contrast, ePerceptive adjusts only the network's width but compensates by varying the input resolution. Zygarde, on the other hand, employs an input-dependent early exit mechanism, where the network evaluates confidence scores at each layer to determine if it should exit early, thus reducing computation. For Zygarde, we evaluated performance under two confidence thresholds, 0.7 and 0.5, to assess the impact of changes in the confidence threshold on performance. To adapt these models to our target platform and reduce memory requirements, the model inputs and weight parameters are quantized from the 32-bit floating-point representation used during training to a 16-bit fixed-point representation (Q15 format). The network structure representations are saved in the FRAM of the microcontroller along with the quantized parameters. The Keysight 6705B power analyser is used to both measure and power the system. Fig. 6 illustrates the operating cycles of the system when running the largest sub-network of DualAdaptNet at different current inputs. The system runs continuously when the input current is $>$ 2.3 mA.

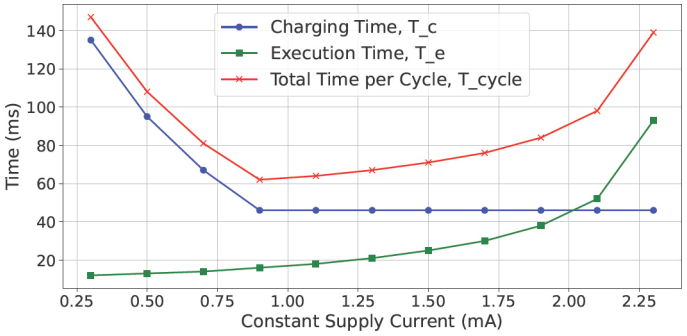


Fig. 6: System operating cycles at different input currents.

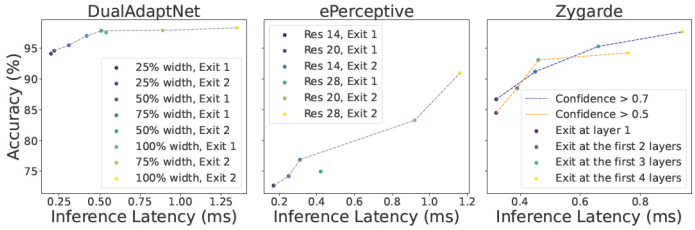


Fig. 7: Accuracy and latency tradeoffs for DualAdaptNet, ePerceptive [10], and Zygarde [12].

B. Evaluation of accuracy and latency tradeoffs

We first evaluate and compare the accuracy and latency tradeoffs of DualAdaptNet, ePerceptive, and Zygarde. All experimental data were collected from the experimental board.

As shown in Fig. 7, DualAdaptNet delivers higher accuracy at comparable latencies. This demonstrates that DualAdaptNet is able to provide more correct inferences by maintaining a balance between speed and accuracy. In addition, the ability to scale both width and depth enables DualAdaptNet to achieve a wider range of latencies than the state-of-the-art.

C. Evaluation Under intermittent power supply

In this section, we evaluate the performance of DualAdaptNet, ePerceptive, and Zygarde under intermittent power supply conditions. Both ePerceptive and Zygarde employ different strategies to handle power fluctuations: ePerceptive adjusts its network configuration based on charging time, while Zygarde relies on confidence thresholds and deadline awareness, allowing for early exits if the network reaches a sufficient confidence level or if the runtime approaches the deadline. The experiments in this section are divided into two parts, the first under a constant current supply, and the second under a varying current supply. All experiments were conducted over a 10-minute runtime the deadline for inference was set to 3 seconds. The overhead of the test computation (FIR) for measuring input current is 0.05 ms, and can therefore be considered negligible.

1) Constant current supply: In the static current experiment, we tested three methods under current supplies of 0.3 mA and 2.3 mA, respectively, to analyze their differences under low and high power supply conditions.

As shown in Fig. 8, at 0.3 mA, DualAdaptNet demonstrated its advantage by completing the highest number of correct inferences, outperforming the other methods by at least 42%. This result highlights the effectiveness of our approach in low-power environments, where efficient scaling of both network width and depth ensures maximum utilization of the available energy. At 2.3 mA, Zygarde was able to complete more total inferences due to its input-dependent early-exit strategy, which terminates inference when confidence reaches a preset threshold. However, this comes at the cost of accuracy: 9.8% of Zygarde's inferences are wrong, compared to only 1.8% for DualAdaptNet. This demonstrates that, while Zygarde may prioritize speed, our method ensures higher accuracy in higher power scenarios, providing more reliable inference.

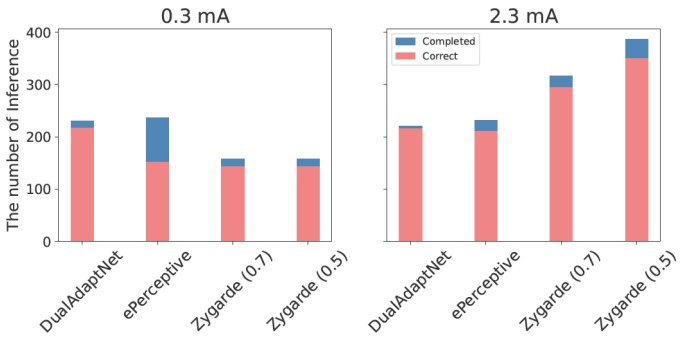


Fig. 8: Number of completed and correct inferences performed by the three models under constant current supply.

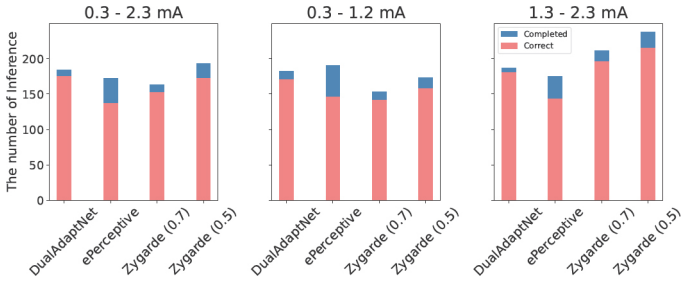


Fig. 9: Number of completed and correct inferences performed by the three models under varying current supply.

2) Varying current supply: In the dynamic current supply experiment, we tested the three approachs across three different current ranges: 0.3-2.3 mA, 0.3-1.2 mA, and 1.3-2.3 mA, to assess the impact of different power variation patterns on the experimental results. The current alternated every 10 seconds, increasing by 0.1 mA until reaching the maximum value, then decreasing back to the minimum, and repeating.

In both the 0.3-2.3 mA and 0.3-1.2 mA ranges, DualAdaptNet completed the highest number of correct inferences, demonstrating its ability to efficiently handle highly variable power conditions. In the 1.3-2.3 mA range, Zygarde again completed a higher number of total inferences, owing to its early exit mechanism, which allowed it to terminate computations before reaching the full depth of the network once confidence thresholds were met. However, similar to the static current experiment, this strategy leads to a higher error rate. Zygarde's inference error rate was 6.2% higher than DualAdaptNet. This reinforces the finding that while Zygarde can complete more inferences by exiting early, the accuracy of these inferences suffers compared to DualAdaptNet, which better balances between the number of inferences and their correctness.

V. Conclusions

This paper introduced DualAdaptNet, a dynamic, deadline-aware neural network designed for energy-harvesting intermittent computing systems. DualAdaptNet adapts both network width and depth, enabling it to respond to fluctuating power conditions while meeting inference deadlines. A runtime scheduler that dynamically selects optimal configurations based on real-time energy availability. Experiments show that DualAdaptNet consistently outperforms ePerceptive and Zygarde in accuracy, especially in low-power and dynamic scenarios. While Zygarde's early exit strategy completed more inferences, it led to higher error rates. In contrast, DualAdaptNet maintained superior accuracy and timely task completion. Its ability to balance computational complexity and accuracy makes it well-suited for real-world, energy-constrained environments.

VI. Acknowledgments

This work was supported in part by the China Scholarship Council (CSC) under 202206540012. Experimental data can be found at: https://doi.org/10.5258/SOTON/D3341.

REFERENCES

[1] M.-L. Ku, W. Li, Y. Chen, and K. R. Liu, "Advances in energy harvesting communications: past, present, and future challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1384–1412, 2015.

[2] J. Hester and J. Sorber, "The future of sensing is batteryless, intermittent, and awesome," in *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pp. 1–6, 2017.

[3] M. L. Wymore and H. Duwe, "A tale of two intermittencies," in *International Workshop on Energy Harvesting Energy-Neutral Sensing Systems (ENSsys)*, pp. 928–930, 2022.

[4] S. S. Saha, S. S. Sandha, and M. Srivastava, "Machine learning for microcontroller-class hardware: A review," *IEEE Sensors Journal*, vol. 22, no. 22, pp. 21362–21390, 2022.

[5] L. Caronti, K. Akhunov, M. Nardello, K. S. Yıldırım, and D. Brunelli, "Fine-grained hardware acceleration for efficient batteryless intermittent inference on the edge," *ACM Transactions on Embedded Computing Systems*, vol. 22, no. 5, pp. 1–19, 2023.

[6] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang, "Dynamic neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7436–7456, 2021.

[7] J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang, "Slimmable neural networks," in *International Conference on Learning Representations (ICLR)*, 2018.

[8] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *International Conference on Pattern Recognition (ICPR)*, pp. 2464–2469, 2016.

[9] S. Jeon, Y. Choi, Y. Cho, and H. Cha, "Harvnet: resource-optimized operation of multi-exit deep neural networks on energy harvesting devices," in *Proceedings of the Annual International Conference on Mobile Systems, Applications and Services (MobiSys)*, pp. 42–55, 2023.

[10] A. Montanari, M. Sharma, D. Jenkus, M. Alloulah, L. Qendro, and F. Kawsar, "eperceptive: energy reactive embedded intelligence for batteryless sensors," in *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pp. 382–394, 2020.

[11] Y. Wu, Z. Wang, Z. Jia, Y. Shi, and J. Hu, "Intermittent inference with nonuniformly compressed multi-exit neural network for energy harvesting powered devices," in *ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2020.

[12] B. Islam and S. Nirjon, "Zygarde: Time-sensitive on-device deep inference and adaptation on intermittently-powered systems," in *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, pp. 1–29, 2020.

[13] N. A. Bhatti and L. Mottola, "Harvos: Efficient code instrumentation for transiently-powered embedded sensing," in *Proceedings of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pp. 209–219, 2017.

[14] K. Maeng and B. Lucia, "Adaptive dynamic checkpointing for safe efficient intermittent computing," in *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 129–144, 2018.

[15] D. Balsamo, A. S. Weddell, A. Das, A. R. Arreola, D. Brunelli, B. M. Al-Hashimi, G. V. Merrett, and L. Benini, "Hibernus++: a self-calibrating and adaptive system for transiently-powered embedded devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 35, no. 12, pp. 1968–1980, 2016.

[16] A. Colin and B. Lucia, "Chain: tasks and channels for reliable intermittent programs," in *Proceedings of the ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pp. 514–530, 2016.

[17] G. Gobieski, B. Lucia, and N. Beckmann, "Intelligence beyond the edge: Inference on intermittent embedded systems," in *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 199–213, 2019.

[18] C.-K. Kang, H. R. Mendis, C.-H. Lin, M.-S. Chen, and P.-C. Hsiu, "Everything leaves footprints: Hardware accelerated intermittent deep inference," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 39, no. 11, pp. 3479–3491, 2020.

[19] W. Lou, L. Xun, A. Sabet, J. Bi, J. Hare, and G. V. Merrett, "Dynamic-OFA: Runtime DNN Architecture Switching for Performance Scaling on Heterogeneous Embedded Platforms," in *Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2021.

[20] J. Yu and T. S. Huang, "Universally slimmable networks and improved training techniques," in *International Conference on Computer Vision (ICCV)*, pp. 1803–1811, 2019.

[21] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[22] D. Balsamo, A. S. Weddell, G. V. Merrett, B. M. Al-Hashimi, D. Brunelli, and L. Benini, "Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems," *IEEE Embedded Systems Letters*, vol. 7, no. 1, pp. 15–18, 2014.