Energy-efficient Content-aware DNN Inference for Mobile Video via Deep Reinforcement Learning

Guangfeng Guo*†, *IEEE Student Member* and Junxing Zhang*, *IEEE Member*
*College of Computer Science, Inner Mongolia University, Hohhot, China
†Baotou Teachers' College, Inner Mongolia University of Science and Technology, Baotou, China
guoguangfeng@163.com, junxing@imu.edu.cn

ICC 2022 - IEEE International Conference on Communications | 978-1-5386-8347-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICC45855.2022.9838620

***Abstract*—Recent work has shown that wearable devices can assist users in cognitive decline through context-aware scene interpretation. These devices should function in real time with sufficient functions, performance, and usability. However, scene interpretation relies on the Deep Neural Network (DNN) inference of continuous video streams, which poses enormous challenges to wearable devices because the resulting computational tasks will quickly drain their batteries. Therefore, we propose an energy-efficient and content-aware DNN inference scheme to address these challenges for assistive devices. We first present the architecture of the assistive system. Then, we leverage the temporal correlation of video frames to save energy spent on the inference by designing a novel online planning method that performs Deep Reinforcement Learning (DRL) using only a subset of frames. Moreover, we come up with a lookup algorithm to select the Dynamic Voltage and Frequency Scaling (DVFS) gear scheme for further energy-saving. Finally, we implement a system prototype and evaluate its energy consumption. The experimental results show that our solution saves almost 40% of the energy on average with negligible impact on inference accuracy and latency compared with the existing approach.**

***Index Terms*—Cognitive Assistance, Deep Neural Network, Deep Reinforcement Learning, Energy Efficient, Content Aware, Temporal Correlation**

I. Introduction

According to Alzheimer's Disease International [1], there were more than 55 million people with dementia worldwide in 2021. Many people are affected by some form of cognitive decline, such as inability to recognize people, places, and objects, which seriously affects their ability as independent members of society. Recently, researchers at Carnegie Mellon University and Intel Labs designed and implemented a cognitive assistive system based on Google Glass devices [2] for helping users in cognitive decline. It recognizes and interprets objects, faces, activities, signage texts, etc.

Ideally, such a cognitive assistance should work at any time and place with sufficient functions, performance, and usability. For this purpose, Deep Neural Network (DNN) has been leveraged to interpret scenes on the wearable device that integrates CPUs and GPUs in the same chip. The assistance also needs to connect to a nearby cloudlet and offload its computations due to its constrained resource. When no suitable cloudlet is available, it offloads directly to a remote cloud. If the offloading delay to the cloud is intolerable, or it has no network access, the assistive device has to perform the real-time scene inference on-device to meet the requirements of users in cognitive decline.

If the DNN inference has to be performed locally on wearable devices, energy consumption will increase exponentially. For example, the real-time image classification with AlexNet executed locally consumes five times as much energy as offloaded execution, and typical smartphones cannot perform the real-time image classification task for more than an hour [3]. Therefore, it is still tricky to run state-of-the-art DNNs on battery-powered wearable devices due to their limited energy budget. Assistive devices that rely on the DNN inference for scene interpretation face great challenges in energy consumption, especially when computations cannot be offloaded. Running DNN inference tasks of continuous video streams solely on-device will quickly drain the batteries of assistive devices.

One of the promising power management strategies for battery-powered wearable devices is Dynamic Voltage and Frequency Scaling (DVFS) [4], which reduces power consumption from the circuit by adjusting CPU or GPU voltage-frequency levels at runtime. Conventional DVFS implementations mostly reside in Operating System (OS) kernels and thus become application-agnostic. To further reduce energy consumption, it is essential to tailor appropriate DVFS strategies to distribute the power budget judiciously among processors by incorporating the actual performance of applications, e.g. inference latency, for scene interpretation tasks.

In response to the aforementioned challenges, we propose an energy-efficient and content-aware DNN inference scheme for mobile videos of the cognitive assistive device. The proposed scheme adopts Deep Reinforcement Learning (DRL) and DVFS to reduce energy consumption. Our paper makes the following contributions:

- Leveraging the temporal correlation of continuous video streams, we propose a novel online planning method to select a subset of frames for DRL, in order to decrease energy consumption.

This work was partially supported by the National Natural Science Foundation of China (Grant N0. 61261019 and 61762071), the Inner Mongolia Science & Technology Plan (Grant No. 201802027), the Inner Mongolia Science and Technology Innovation Guidance and Incentive Fund (Grant No. 111-0406041701), and the Inner Mongolia Autonomous Region Natural Science Foundation (Grant No. 2016MS0614 and 2018MS06023).
Corresponding author: Junxing Zhang (junxing@imu.edu.cn)

- For the purpose of further energy conservation, we present a lookup algorithm to tailor the appropriate DVFS strategies for the DNN inference of mobile video clips.
- We implement a system prototype and evaluate its energy consumption. Our experiments reveal that compared to the current approach the proposed solution saves almost 40% of the energy consumption with negligible impact on inference accuracy and latency.

II. RELATED WORK

There exists some work on the real-time video analytics for wearable devices. Wearable devices commonly use object detection methods for the scene interpretation. On the one hand, a number of lightweight object detection methods have been proposed to improve detection speed and to meet the demand for high performance, such as YOLOv4-tiny. On the other hand, some optimized professional frameworks have also been developed. TensorRT launched by NVIDIA, for instance, is written in CUDA for optimizing the inference of deep learning models on their GPUs. In response to NVIDIA Jetson Boards, Micaela Verucchi et al. developed tkDNN [5], an open-source DNN library built with cuDNN and tensorRT primitives, whose main goal is to exploit those boards as much as possible to obtain the best inference performance.

The main technology for managing the energy consumption of wearable devices is DVFS. Many vendors prefabricate the DVFS function into their chips. For example, the Jetson TX2 consists of a 256-core Pascal GPU along with a CPU cluster. Each component, such as CPU, GPU, and Memory, includes a voltage/frequency gate that can be adjusted via software. In [6], NeuOS uses it with 11759 unique DVFS configurations. For convenience, NVIDIA has completed the calculations to figure out which of the core and clock frequencies provide the best performance within the energy budget. By configuring different CPU cores and different maximal frequencies of components, it typically runs in five different modes. Moreover, each mode has the Max and Auto options. The former represents setting each component to the maximal frequency, while the latter stands for dynamic tuning their clocks governed by the OS kernel.

In edge computing, although most researchers study minimizing the delay [7], there are still a few people concentrating on energy consumption. With the recent advances of deep learning, researchers have been focusing on DRL methods to optimize on-device energy consumption. In [8], Mengwei Xu et al. propose Elf, a runtime for an energy-constrained camera to continuously summarize video scenes as approximate object counts. It employs a learning-based planner with the DRL method (A3C), but it does not evaluate the energy consumption of the decision algorithm itself in detail. In this paper, similar as [8], we exploit a DRL method with offline training and online prediction. Different from [8], we select a more lightweight model Deep Q-network (DQN) to train an online planner by selecting a subset of frames for energy-saving, tailor it for wearable devices to meet the demand for low energy consumption and high performance, and also evaluate its energy consumption for online local prediction. In addition, while Elf [8] focuses on video streams from fixed cameras, our scheme handles streams from mobile cameras. The scenes captured by assistive devices may change from time to time, which poses greater challenges to our decision algorithm.

For people with dementia, the cognitive assistance has to function "in the wild" with sufficient functionality, performance, and usability to be valuable. In many cases, there is no available cloudlet and the delay for offloading to a cloud exceeds the tolerant time of users, or even worse the Internet is inaccessible. The assistive device must run the inference task with the DNN object detection methods locally to meet the requirements of users. However, for the battery-powered device, the DNN inference of continuous video streams brings about great challenges to its energy budget. Therefore, it is important for us to investigate the energy-efficient solution for the assistive device by integrating tailored DVFS strategies with lightweight DNN inference methods.

III. MOTIVATION

A. Energy Consumption with DVFS

To better understand the energy spent on the computation of object detection for the video stream, we run the YOLOv4-tiny DNN model for object detection of the video (1080p FHD, 25 FPS, 29 seconds) implemented in TensorRT on the Jetson TX2, and measure the inference latency and the computation energy consumption per frame and the standby power at the DVFS gears respectively. As shown in Fig. 1 and Fig. 2, a Jetson TX2 typically runs in ten different gears (each one in the five modes has the Auto and Max options), the inference latency, energy consumption per frame, and the standby power have a certain difference at diverse gears. According to the above measurement, no DVFS gear is a silver bullet with the shortest inference latency and the least energy consumption. Intuitively, we can trade off inference latency and energy consumption to find the corresponding most energy-efficient scheme for each user's tolerant latency. For example, given the tolerant latency τ = 24 ms and the video frame rate r = 25 FPS, the most energy-efficient scheme is that the device works at gear Max-N/Max for object detection in the first 23.5 ms, and works standby at gear Max-Q/Auto in the remaining 16.5 ms during each frame period (40 ms). While τ = 40 ms and r = 25 FPS, the most energy-efficient scheme is that the device works at gear Max-Q/Max for object detection in the first 35.5 ms, and works on standby at gear Max-Q/Auto in the remaining 4.5 ms during each period.

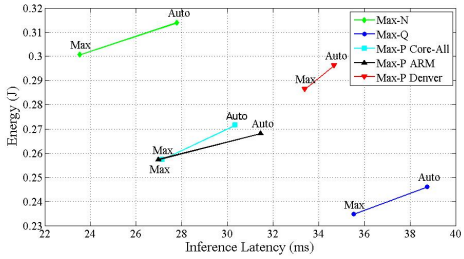


Fig. 1. Inference latency and computation energy consumption per frame

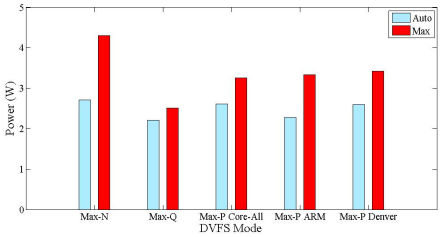


Fig. 2. Standby power at the different DVFS gears

B. Temporal Correlation for Video Streaming

A video can be divided into a series of scenes that contain several consecutive frames which have similar object content. These frames and associated objects can be analyzed in the temporal domain. A series of frames in a single scene is composed of some similar objects which may change in position and size. In order to make a quantitative analysis of this characteristic, we run the YOLOv4-tiny model for object detection of the video (as the same as Section III-A). As shown in Fig. 3, in the consecutive three frames, the position of the car identified remains almost unchanged. Using the metric Intersection over Union (IoU) to evaluate the changes, we set the car's bounding box b_1 of the first frame as the predictive box of the second and third frame, and calculate the IoU value of the last two frames $IoU(b_2, b_1) = 0.95$ and $IoU(b_3, b_1) = 0.91$ (b_2 and b_3 represent their ground truth respectively). Based on the above analysis, an intuitive idea for the energy-saving scheme is selecting some frames of the continuous frames with large changes for detection instead of detecting each frame. To quantify the sensitivity of each user for the variations in the consecutive frames, we introduce a variable named the tolerant deviation α. For example, if a user sets $\alpha = 0.05$, we select the first and third frame for detection while skipping the second frame; while setting $\alpha = 0.1$, we detect the first and fourth frame skipping the second and third frame.

(a) the 1st frame (b) the 2nd frame (c) the 3rd frame

Fig. 3. An example of the frame sequence

IV. System Design

In this section, we present the system architecture of the cognitive assistance shown in Figure 4. First, the user sets the system parameters such as the tolerant deviation, the tolerant latency, and the frame rate. According to the configured frame rate, it captures the video stream. In the light of the configured tolerant latency, the online planner (see Section V) selects some frames instead of all frames of the captured video stream for the object detection actuator. According to the tolerant latency and the video frame rate, we build a runtime DVFS profile using a lookup algorithm (see Section VI) to control its DVFS gear's switch for energy-saving. By the profile, the object detection actuator performs the DNN inference model (e.g., Yolov4-tiny) for the received frames at the reasonable DVFS gears.

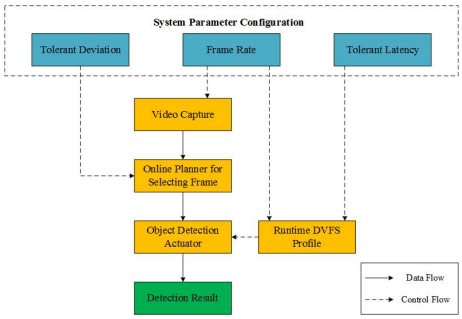


Fig. 4. System architecture of the cognitive assistance

V. Online Planner for Selecting Frames

In this section, leveraging the temporal correlation of video streams, we propose an online planning method to choose only a subset of all frames for the DNN inference in order to conserve energy.

To select some frames for object detection, we shall meet two requirements:

- **The requirement for global knowledge.** An intuitive idea for the energy-saving scheme is selecting some frames of the continuous stream with large changes for detection instead of detecting each frame. On the basis of the temporal correlation for video streams, in addition to perceiving the current frame information, we also shall perceive the information of the next few frames to decide which next frames can be skipped without detection. For a period of time of object detection, we shall perceive each frame information in the period and accurately select the frames with large changes while not affecting the overall experience of each user.
- **The requirement for on-the-go decisions.** Deferring object detection at the end of the period leaves users with an excessive amount of outdated information. First, outdated information is meaningless to address cognitive decline. Second, the large number of messages received by users at the end of each period can also lead to cognitive overload, which makes the user more confused. To acquire information about the environment timely, we need to extract the information from the captured videos and expose it to the user timely. Once an image has been captured, we should decide whether it is an important frame with large changes or not. If important, it will be detected; otherwise, it will be skipped.

We meet both requirements above with an online planner. Our key idea is to make online decisions by mimicking what an oracle planner would do. Specifically, the oracle planner works offline: with full knowledge of a video, it decides which frame should have been skipped. Trained with the oracle decisions, the online planner acquires the optional strategy and makes decisions that the oracle would make with a similar

observation of videos in the future. We assume that changes of video frames follow stationary distributions, i.e. training data (changes captured in history by a cognitive assistance) are Independent and Identically Distributed (IID) with test data (changes currently captured).

A. The Oracle Planner

The cognitive assistance captures the video stream at a fixed rate constantly. Because the oracle planner works based on an impractical assumption: it knows each frame information in the period, we shall decide which subsequent frames (that are particularly similar to the current frame) can be skipped when receiving a frame.

Given the predefined tolerant deviation α and received the m-th frame f_m, we detect p objects $O_m = \{o_{m,1}, o_{m,2}, \ldots, o_{m,p}\}$ corresponding the bounding boxes set $B_m = \{b_{m,1}, b_{m,2}, \ldots, b_{m,p}\}$. We compare the similarity of each subsequent frame f_{m+1} and the current frame f_m in order, and if their similarity is greater than the threshold of user tolerance $\beta = 1-\alpha$, we skip the frame f_{m+1} and continue to compare the next frame f_{m+2} with f_m, otherwise end the comparison. The detailed steps are given in Algorithm 1 (line 1-5). The similarity function $SimilarityBetweenFrames(\cdot)$ details in line 6-20. We first compare the numbers of detected objects in frame f_l and f_m. If not equal, it directly returns zero (line 7-8). Otherwise, we calculate the minimum IoU of each match (line 13) and find the most appropriate match which value is maximum (line 14-16).

Algorithm 1 The Oracle Planner Algorithm

Require:
the tolerant deviation: α, the current frame: f_m.
Ensure: the count skipped and exempted from detection for the frames received after it when the m-th frame received.
1: $\beta \leftarrow 1-\alpha, i \leftarrow 1$
2: **while** $SimilarityBetweenFrames(f_{m+i}, f_m) \geq \beta$ **do**
3: $\quad i \leftarrow i+1$
4: **end while**
5: **return** $i-1$
6: **function** $SimilarityBetweenFrames(f_l, f_m)$
7: **if** $|O_l| \neq |O_m|$ **then**
8: $\quad$ **return** 0
9: **else**
10: $\quad$ the set of n elements for all permutation: $P = \{P_1, \ldots, P_j, \ldots\}$, $P_j = (p_j^1, p_j^2, \ldots, p_j^n)$;
11: $\quad similarity \leftarrow 0$
12: $\quad$ **for** P_j in P **do**
13: $\quad\quad IoU_{min} \leftarrow \min(\{IoU(b_{l,1}, b_{m,p_j^1}), \ldots\})$
14: $\quad\quad$ **if** $IoU_{min} > similarity$ **then**
15: $\quad\quad\quad similarity \leftarrow IoU_{min}$
16: $\quad\quad$ **end if**
17: $\quad$ **end for**
18: $\quad$ **return** $similarity$
19: **end if**
20: **end function**

B. The learning-based Planner

Using the historical videos captured from the cognitive assistance as the training dataset, we tailor the deep reinforcement learning DQN algorithm for the online planner training.

Goal. Trained with the oracle decisions, the online planner acquires the optional strategy for the current captured video frame and makes decisions that the oracle would make with a similar observation of video frames. For the current captured video frame f_i, the oracle planner uses the oracle strategy $\Pi^*(f_i)$ to decide the skipped frame count. For example, the oracle planner decides the skipped frame count is 2, it means the frame f_{i+1} and f_{i+2} is skipped and exempted from object detection, and the next detected frame is f_{i+3}. So our goal is minimizing the bias between the oracle planner's strategy $\Pi^*(f_i)$ and the online planner's strategy $\Pi(f_i)$ for each decision when captured the frame f_i. We quantize it as the below formula (N denotes the total decision count in training):

$$\min|\Pi(\cdot) - \Pi^*(\cdot)| = \min(\frac{1}{N} \cdot \sum_{i=1}^{N} |\Pi(f_i) - \Pi^*(f_i)|) \quad (1)$$

State space. The observation vector consists of the recent M frame in the video stream. We convert each frame to a gray-scale image and resize it to (416,416) as the same as the input size of the object detection model. Our experiment empirically chooses $M = 4$.

Action space. The action space contains L actions {0, 1, 2, ..., L-1}, and each action value represents the skipped frame count. $action = 0$ means that no frame can be skipped. Our experiment empirically chooses $L = 11$.

Reward. We make reward signals to guide agents to find a good solution for our goal: minimize the bias between the oracle planner's and the online planner's strategy. To avoid skipping the frames with large changes, we adopt a conservative method. We define the reward function as below:

$$reward(f_i) = \begin{cases} -(\Pi^*(f_i) - \Pi(f_i)) & \text{if } \Pi^*(f_i) \geq \Pi(f_i) \\ -(\Pi(f_i) - \Pi^*(f_i))^2 & \text{otherwise} \end{cases} \quad (2)$$

$\Pi^*(f_i)$ and $\Pi(f_i)$ indicate the skipped frame count when detecting the frame f_i deciding by the oracle planner and the learning-base planner, respectively.

Online prediction & cost. Once trained, we run the trained DRL model on the cognitive assistance timely. Our tailored DQN model contains six hidden layers (three convolutional layers, two dense layers, and one flatten layer). It brings a slight impact on the energy consumption and inference latency, compared to the Yolov4-tiny object detection model containing 38 hidden layers. We evaluate its overhead in Section VII-D.

VI. Algorithm for Selecting the DVFS Gear

In this section, we propose a lookup algorithm for selecting the DVFS gear scheme for energy-saving to build the runtime DVFS profile. According to the profile, we steer the object detection actuator to process the received frames at the appropriate DVFS gears in an orderly way.

We propose the lookup algorithm of selecting the DVFS gear scheme detailed in Algorithm 2. By the online planner's decision, we need to handle separately whether the frame shall be detected or not. For the frames exempted from detection, the object detection actuator does not need to do any operation and keeps in the standby state, and we switch to the lowest standby power consumption DVFS gear d' in the period (line 2-5). For detected frames, it needs to detect the current frame first and then switch to the standby state. So we first find the gear d with the lowest calculated energy consumption that meets the user's tolerant latency (line 7-8), we switch to the gear d at the former and the gear d' at the latter (line 11).

Algorithm 2 Selecting the DVFS Gear Algorithm

Require:
the set of the DVFS gears:$D = \{d_i\}$, the standby power at DVFS gear d_i:$W(d_i)$, the inference latency per frame at DVFS gear d_i:$T(d_i)$, the computation energy per frame at DVFS gear d_i:$E(d_i)$, the user's tolerant latency: τ, the video frame rate: r, the current captured frame: f_m.
Ensure: the energy-efficient runtime DVFS profile:$S(\cdot)$.
1: Supposing the capture moment of the m-th frame is $T_{m,0}$, and the capture moment of the next frame is $T_{m+1,0}$;
2: $T_{m+1,0} \leftarrow T_{m,0} + 1/r$
3: $d' \leftarrow argmin(W(d_i)) \quad (d_i \in M)$
4: **if** $IsSkiped(f_m)$ **then**
5: $\quad S(t) = d' \quad (t \in [T_{m,0}, T_{m+1,0}))$
6: **else**
7: $\quad D' \leftarrow \{d_i, T(d_i) \leq \tau\}$
8: $\quad d \leftarrow argmin(E(d_i)) \quad (d_i \in D')$
9: $\quad$ Supposing the complete time of the m-th frame for object detection is $T_{m,1}$;
10: $\quad T_{m,1} \leftarrow T_{m,0} + T(d)$
11: $\quad$ the most energy-saving DVFS scheme $S(\cdot)$ is below:

$$S(t) = \begin{cases} d, & \text{if } t \in [T_{m,0}, T_{m,1}] \\ d', & \text{if } t \in (T_{m,1}, T_{m+1,0}) \end{cases} \tag{3}$$

12: **end if**
13: **return** $S(\cdot)$

VII. Evaluation

In this section, we evaluate the energy consumption of the system prototype, the overhead of the online planner, and the impact on inference accuracy and latency.

A. Methodology

Datasets. We use D^2-City [9], large-scale comprehensive videos collected by front-facing dashcams of passenger vehicles on the DiDi platform. It has a similar perspective as the first-person video stream captured by the cognitive assistance. Especially when a vehicle is stuck in traffic or slowing down at an intersection, the rate of scene changes is more similar to it due to their almost same speed. In training, we use its training set involving 3436 videos. In prediction, we use its validation set involving another 100 videos.

Tools. We use a Jetson TX2 as a cognitive assistance, select the YOLOv4-tiny model for object detection, implement the energy consumption measure program with Python based on its internal power monitor TI INA3221. To train the online planner, we implement the DQN algorithm [1] with Tensorflow and train the model on a server with two NVIDIA GTX 2080 Ti GPUs. Moreover, we tailor the DQN algorithm for the Jetson TX2 with C++ based on tkDNN, which network structure is the same as the model on the server. After the server trains the DQN model completely, we import the trained model to the Jetson TX2, and it runs the tailored DQN inference algorithm to make decisions on-device.

Comparatives. For contrast, we compare the energy consumption of our scheme with the vanilla scheme that has not been optimized.

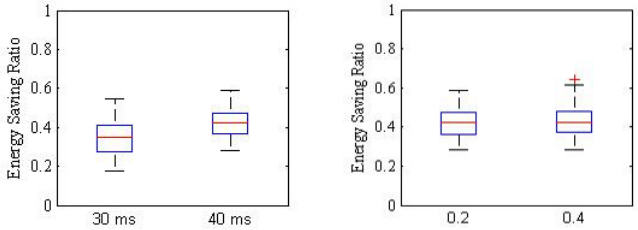
- **Our scheme.** Under the premise of meeting the needs of users, our scheme selects some frames instead of all frames of the captured video stream for inference, and autonomously adjusts the DVFS gears of the cognitive assistance device.
- **The vanilla scheme.** The vanilla scheme adopts the safest strategy: using all frames of the captured video stream, and always keeping the DVFS gear configuration that produces the shortest inference latency regardless of users' needs.

[1]Its hyper-parameter settings are shown below. The batch size, the number of the offline training iterations, the reward discount, the size of replay buffer, and the learning rate are 32, 110000, 0.99, 10000, and 0.0001, respectively.

B. Energy Consumption at the different tolerant latency

In this subsection, we emulate two users' needs. They set the same tolerant deviation α = 0.2. The difference is that the one sets the tolerant latency τ_1 = 30 ms while the other sets τ_2 = 40 ms. We measure the energy consumption of the cognitive assistance device at the above two schemes.

Figure 5(a) shows our scheme's energy-saving ratio at the different tolerant latency compared with the vanilla scheme. The average energy saving ratio at the tolerant latency 30 ms and 40 ms is 34.26% and 41.96%, respectively. The latter saves more energy due to selecting the more energy-saving DVFS gear Max-Q/MAX.



(a) tolerant latency (b) tolerant deviation

Fig. 5. Energy saving ratio of our scheme

C. Energy Consumption at the different tolerant deviation

In this subsection, we emulate another two users' needs. They set the same tolerant latency τ = 40 ms and the same video frame rate r = 25 FPS. The difference is that the one sets the tolerant deviation α_1 = 0.2 while the other sets α_2 = 0.4. Due to the different tolerant deviations, we separately train the DRL model using the training set. After trained, we validate the models on the validation set respectively and measure their energy consumption at the above two schemes.

Figure 5(b) shows our scheme's energy-saving ratio at the two different tolerant deviations compared with the vanilla scheme. The average energy saving ratio at the tolerant deviation 0.2 and 0.4 is 41.96% and 43.31%, respectively. Due to the conservative design for the reward function and the increase of the skipped frame count as the tolerance deviation increases, the latter has a slight increase in the average energy saving rate.

D. Overhead of the online planner

In this subsection, we evaluate the energy consumption and execution time of the online planner. We run it 100 times on the Jetson TX2 at the different DVFS gears, and compare these measurement results and the previous measure for object detection in Section III-A. Due to limited space, we ignore the relevant charts.

The energy consumption for making a decision is 8.89% to 12.24% of the energy consumption for detecting a frame, and

it achieves a minimum of $0.0279\,J$ at the Max-Q/Max gear. The average execution time for making a decision is 4.71 ms to 6.63 ms. So its overhead is acceptable both in terms of time and energy consumption compared with object detection. To ensure a crisp user experience, we switch the Max-N/Max gears to run the online planner, and its inference latency and energy consumption are 4.71 ms and 0.03712 J, respectively.

E. Impact on Inference Accuracy

In the experiment of Section VII-C, we also record the traces and detection results for each frame at the two tolerant deviations. And then, we use the Yolov4-tiny model to detect each frame as the ground truth. Finally, we develop a program to compare the metrics: Precision, Recall, F1 score, and IoU at the two tolerant deviations.

As Figure 6 shown, the average of Precision, Recall, F1 score and IoU are 96.44%, 96.49%, 95.49% and 97.32% respectively. Compared with the vanilla scheme, our scheme lost 3.56%, 3.51%, 4.51%, and 2.68% in the above metrics respectively. Compared with the inference accuracy at the tolerant inference deviation 0.2 and 0.4, The latter is on average 0.0048 lower than the former. So our scheme saves almost 40% of energy at the cost of 3.57% loss of inference accuracy on average.

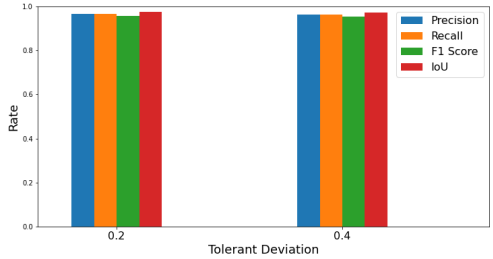


Fig. 6. Inference accuracy at the different tolerant deviations

F. Impact on Processing Latency

In the experiment of Section VII-B, we also record the traces and processing latency for each video at the two tolerant latency. Then we detect the videos by the vanilla scheme and meanwhile record the processing latency.

Figure 7 shows the processing latency per frame by the vanilla scheme and our scheme at two tolerant latency. There is the average smallest processing latency of 23.55 ms in the vanilla. In our scheme, its average processing latency exceeds the pre-defined value by 1.94% at the tolerant latency 30 ms, while it has a 2.96% headroom at 40 ms. So our scheme saves almost 40% of energy with negligible impact of processing latency.

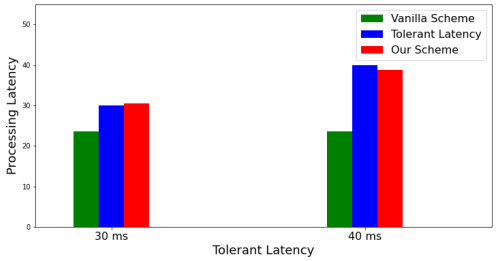


Fig. 7. Processing latency at the different tolerant latency

VIII. Conclusion

As assistance systems, wearable devices offer a glimmer of hope to users in cognitive decline. They should function in the wild with sufficient functionality, performance, and usability to be valuable. Their scene interpretation relies on the DNN inference of continuous video streams. In many cases, they have to execute the inference tasks autonomously and timely. The inference is very challenging to wearable devices because it can quickly drain batteries. Therefore, we propose an energy-efficient and content-aware DNN inference scheme for these assistive devices. First, we present the architecture of the assistance system. Secondly, leveraging the temporal correlation of video frames, we present a novel online planning method that carries out deep reinforcement learning with only a selected subset of frames to save energy. Moreover, we also employ a lookup algorithm to select the DVFS gear scheme for further energy-saving. At last, we implement a system prototype and evaluate its energy consumption. As the experimental results illustrate, on average our scheme saves almost 40% of the energy with negligible impact on inference accuracy and latency compared to the current solution.

References

[1] S. Gauthier, P. Rosa-Neto, J. Morais, and C. Webster, "World alzheimer report 2021: Journey through the diagnosis of dementia," *Alzheimers Disease International*, 2021.

[2] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," *MobiSys 2014*, pp. 68–81, 2014.

[3] G. Guo and J. Zhang, "Energy-efficient incremental offloading of neural network computations in mobile edge computing," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020, pp. 1–6.

[4] P. Macken, M. Degrauwe, M. Van Paemel, and H. Oguey, "A voltage reduction technique for digital systems," in *1990 37th IEEE International Conference on Solid-State Circuits*, 1990, pp. 238–239.

[5] M. Verucchi, G. Brilli, D. Sapienza, M. Verasani, and M. Solieri, "A systematic assessment of embedded neural networks for object detection," in *ETFA 2020*, 2020.

[6] S. Bateni and C. Liu, "Neuos: A latency-predictable multi-dimensional optimization framework for dnn-driven autonomous systems," in *USENIX ATC 2020*. USENIX Association, 2020, pp. 371–385.

[7] Y. Wu, K. Ni, C. Zhang, L. P. Qian, and D. H. Tsang, "Noma-assisted multi-access mobile edge computing: A joint optimization of computation offloading and time allocation," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 12, pp. 12 244–12 258, 2018.

[8] M. Xu, X. Zhang, Y. Liu, G. Huang, X. Liu, and F. X. Lin, "Approximate query service on autonomous IoT cameras," *MobiSys 2020*, pp. 191–205, 2020.

[9] Z. Che, G. Li, T. Li, B. Jiang, X. Shi, X. Zhang, Y. Lu, G. Wu, Y. Liu, and J. Ye, "D^2-city: A large-scale dashcam video dataset of diverse traffic scenarios," 2019.