E4: Energy-Efficient DNN Inference for Edge Video Analytics Via Early-Exit and DVFS

Ziyang Zhang[1], Yang Zhao[2], Ming-Ching Chang[3], Changyao Lin[1], Jie Liu[2]

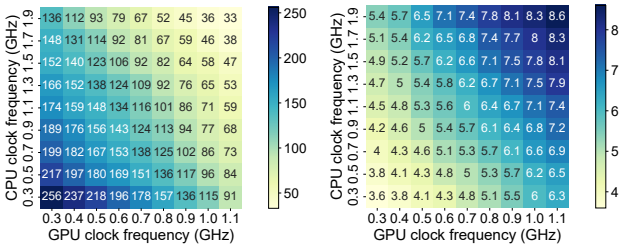
[1]Harbin Institute of Technology
[2]State University of New York at Albany
[3]Harbin Institute of Technology (Shenzhen)
zhangzy@stu.hit.edu.cn, yang.zhao@hit.edu.cn, mchang2@albany.edu, lincy@stu.hit.edu.cn, jieliu@hit.edu.cn

Abstract

Deep neural network (DNN) models are increasingly popular in edge video analytic applications. However, the compute-intensive nature of DNN models pose challenges for energy-efficient inference on resource-constrained edge devices. Most existing solutions focus on optimizing DNN inference latency and accuracy, often overlooking energy efficiency. They also fail to account for the varying complexity of video frames, leading to sub-optimal performance in edge video analytics. In this paper, we propose an Energy-Efficient Early-Exit (E4) framework that enhances DNN inference efficiency for edge video analytics by integrating a novel early-exit mechanism with dynamic voltage and frequency scaling (DVFS) governors. It employs an attention-based cascade module to analyze video frame diversity and automatically determine optimal DNN exit points. Additionally, E4 features a just-in-time (JIT) profiler that uses coordinate descent search to co-optimize CPU and GPU clock frequencies for each layer before the DNN exit points. Extensive evaluations demonstrate that E4 outperforms current state-of-the-art methods, achieving up to 2.8× speedup and 26% average energy saving while maintaining high accuracy.

Introduction

Advances in deep neural network (DNN) models and GPU hardware accelerators have significantly advanced video analytics in edge intelligence applications, including object detection (Zou et al. 2023; Zhao et al. 2019), action recognition (Ghodrati, Bejnordi, and Habibian 2021; Jhuang et al. 2013), and pose estimation (Andriluka et al. 2014; Toshev and Szegedy 2014; Andriluka et al. 2018), *etc.* To protect data privacy and ensure low-latency quality of service (QoS), many of these applications are deployed on edge devices close to the data sources (Liang et al. 2023). However, the increasing demand for higher video quality results in greater video frame complexity, making DNN models computationally intensive for tasks like multi-object detection and tracking. On the hardware side, edge devices face limitations in cost and size, resulting in fewer computational resources compared to cloud servers (Bhardwaj et al. 2022; Padmanabhan et al. 2023; Khani et al. 2023). The diverse network structures of DNN models (Cui et al. 2022) and the varying complexity of video frames (Menon et al. 2022) (*e.g.*, the spatial correlation between consecutive frames) introduce new challenges for edge video analytics.



(a) Inference latency (ms) (b) Energy consumption (W)

Figure 1: The impact of CPU and GPU clock frequencies on (a) inference latency (ms) and (b) energy consumption (W), based on running EfficientNet-B0 on an Nvidia Xavier NX edge GPU with 8GB DRAM.

Dynamic voltage and frequency scaling (DVFS) is a widely used power management technique that balances energy consumption and computing performance on edge devices by adjusting CPU and GPU voltage-frequency in real-time. While previous studies have developed various learning-based DVFS governors (Kim et al. 2021; Yeganeh-Khaksar et al. 2020; Lin et al. 2023), we find that their performance suffers when applied to edge video analytics due to a mismatch between CPU/GPU frequency settings and the specific demands of video processing at the edge.

As a motivating example, we tested *zTT* (Kim et al. 2021), a state-of-the-art learning-based DVFS governor, on an Nvidia Jetson Xavier NX edge device running the EfficientNet-B0 DNN model (Tan and Le 2019). The results, shown in Fig. 1, highlight the trade-off between inference latency and energy consumption. Specifically, we observe that *higher processor clock frequencies reduce inference latency but significantly increase energy consumption.* For instance, achieving real-time video analytics at 30 frames per second (fps), which requires a 30ms inference latency, necessitates setting the CPU and GPU frequencies to their highest levels of 1.9GHz and 1.1GHz, respectively. However, this leads to a sharp rise in energy consumption of 8.6W. Similar trends were observed on other edge devices as well.

Based on the experiments and observations, we identified that the challenges stem from the diversity in video frame complexity and DNN models. As illustrated in Fig. 2, differ-

ent video frames vary in the number of objects they contain, leading to varying levels of complexity for object detection. We use the term *video frame complexity* to describe these differing DNN inference demands in general video analytics applications. State-of-the-art DNN models are designed to detect multiple objects with high accuracy, but not all network layers are necessary for frames with fewer objects. For these low-complexity frames, DVFS should adaptively reduce CPU and GPU clock frequencies. However, current learning-based DVFS approaches do not account for video frame complexity. Additionally, edge video analytics often deploy different DNN models for various tasks like detection and tracking, which further complicates performance optimization. Conventional DVFS methods are application-agnostic and do not consider the specific workload characteristics of DNNs. In this paper, our proposed framework addresses both video frame complexity and DNN model diversity to optimize edge performance.

In this paper, we introduce the **Energy-Efficient Early-Exit (`E4`)** framework that improves DNN inference efficiency and latency for edge video analytics. Early-Exit is a mechanism that adaptively exits DNN inference early based on video frame complexity and DNN model diversity (Teerapittayanon, McDanel, and Kung 2016; Laskaridis et al. 2020; Zhang, Zhao, and Liu 2023b). We introduce two key design implementing `E4`: (1) an attention-based cascade module that determines DNN exit points by analyzing video frame complexity, and (2) a novel DVFS governor that automatically adjusts CPU and GPU frequencies for each layer before the DNN exit points using a Just-In-Time (JIT) profiler (You, Chung, and Chowdhury 2023) based on coordinate descent search. We conduct comprehensive experiments on two widely-used datasets and two representative video analytics DNN models across five heterogeneous edge devices. The evaluation results demonstrate that `E4` achieves lower latency and higher energy efficiency compared to state-of-the-art methods, while maintaining accuracy. The main contributions of this paper are as follows:

- We propose `E4`, an energy-efficient inference framework that enables adaptive DNN exit points and power management configurations tailored to dynamic video frames and DNN models.
- We design an attention-based cascade module that determines optimal exit points by analyzing the spatial correlation between consecutive video frames.
- We develop a novel power management approach using the coordinate descent search algorithm to automatically scale CPU and GPU frequencies for each network layer. Additionally, a lightweight DNN-based prediction model minimizes performance interference between multiple DNN models.
- Extensive experimental results demonstrate that `E4` outperforms state-of-the-art early-exit methods, achieving up to 2.8× speedup and 26% average energy savings.

Related Works

Power management on edge devices: Previous works have introduced various DVFS governors for power management on edge devices. For instance, zTT (Kim et al. 2021), GearDVFS (Lin et al. 2023) and Ring-DVFS (Yeganeh-Khaksar et al. 2020) use learning-based approaches to automatically optimize CPU and GPU frequencies to reduce energy consumption, often at the cost of inference performance. In contrast, methods like Road-RuNNer (Kakolyris et al. 2023), DVFO (Zhang et al. 2024; Zhang, Zhao, and Liu 2023a) and AppealNet (Li et al. 2021) use adaptive partitioning to enable cloud-edge collaborative inference, reducing energy consumption by running parts of DNN models on edge devices. Zeus (You, Chung, and Chowdhury 2023) focuses on optimizing energy during DNN training with a multi-arm bandit-based power optimizer that balances energy consumption and latency. Additionally, techniques like model compression (Han, Mao, and Dally 2015) and neural architecture search (NAS) (Ren et al. 2021) are complementary to `E4` and can further enhance energy efficiency by using lightweight DNN models.

Early-exit DNN inference: The early-exit mechanism (Teerapittayanon, McDanel, and Kung 2016) is a type of dynamic inference that allows DNNs to exit at different layers or sub-networks, once an accuracy threshold is met, thus reducing computation costs while maintaining accuracy. Since its introduction in BranchNet (Teerapittayanon, McDanel, and Kung 2016), various early-exit strategies have been developed. For instance, HarvNet (Jeon et al. 2023) uses NAS to automatically determine exit points, Delen (Liang et al. 2023) employs conditional execution for adaptive control of latency, accuracy, and power. PAME (Zhang et al. 2022) focuses on reducing batch inference latency with precision-aware early exits. However, these approaches do not integrate power management techniques like DVFS for energy-efficient dynamic inference. While methods such as EdgeBERT (Tambe et al. 2021), EENet (Li et al. 2023b) and Predictive Exit (Li et al. 2023a) combine early exits with DVFS, they overlook the impact of video frame complexity on DNN inference. In contrast, our work analyzes video frame complexity to optimize energy consumption and inference latency, while maintaining DNN inference accuracy in edge video analytics.

Edge video analytics: DNN-based video analysis on edge devices near the data source is a promising approach. Previous works have focused on optimizing inference latency, accuracy, and memory overhead. For example, Ekya (Bhardwaj et al. 2022), RECL (Khani et al. 2023) and AdaInf (Shubha and Shen 2023) use continuous learning (Wang et al. 2024) for incremental training on edge devices, which addresses accuracy degradation from data drift. Gemel (Padmanabhan et al. 2023) reduces memory overhead through model merging. Remix (Jiang et al. 2021) partitions video frames by the number of objects, using complex DNN models for regions with more objects and simpler models for regions with fewer objects to reduce inference latency. These approaches are complementary to `E4`, which further enhances inference performance.

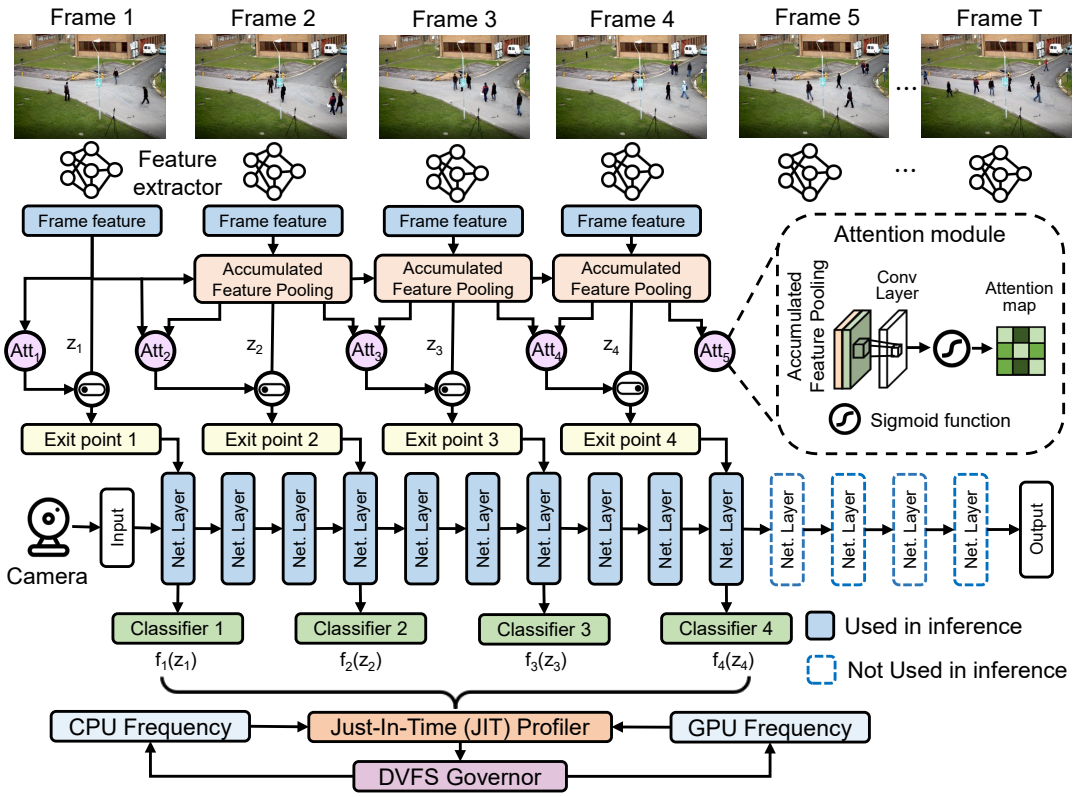


Figure 2: Overview of the proposed E4 efficient edge DNN video analytic inference framework. Given a video input, we sample T frames with varying complexities, such as different numbers of detectable objects. The feature extractor processes each frame and aggregates these features to assess video frame complexity. An attention module and its corresponding gate are trained to determine DNN early exit points. The Just-In-Time (JIT) Profiler and DVFS Governor are then employed to search and scale CPU and GPU clock frequencies for each layer before the DNN exit points.

Methodology

Design Overview

Our proposed E4 framework addresses energy-efficient DNN inference for edge video analytics. The core idea of E4 is to determine optimal exit points for different video frames based on their complexity. A learning-based DVFS governor then co-optimizes CPU and GPU clock frequencies for each network layer before these exit points during the DNN inference. As illustrated in Fig. 2, E4 consists of four key components: (1) *Accumulated Feature Pooling* analyzes video frame complexity. (2) *Attention-Based Early-Exit* determines the appropriate DNN exit points. (3) *Just-In-Time (JIT) Profiler* dynamically co-optimizes CPU and GPU frequencies for each layer before the exit points. (4) *DVFS governor* manages dynamic voltage and frequency scaling. The framework uses historical video frame data to train the early-exit DNN model, leveraging the temporal correlation between frames. Each classifier performs inference at the layer corresponding to each early exit point. This dynamic power management approach makes informed decisions based on the complexity of video frames and DNN models. Details of each component are discussed in the following sections.

Accumulated Feature Pooling

Given a set of video frames and their corresponding labels, *e.g.*, objects to be detected, the feature extractor generates features for each frame. In our experiments, we use the EfficientNet-B0 (Tan and Le 2019) and MobileNet-v2 (Sandler et al. 2018) as the feature extraction backbones. We aggregate these features using *Accumulated Feature Pooling*, which is based on max-pooling (Ghodrati, Bejnordi, and Habibian 2021). This method captures temporal relationships between T consecutive video frames. Let Φ denote the feature extraction network parameterized by hyper-parameters θ. We implement the temporal aggregation function Ψ using a two-layer long short-term memory (LSTM). For a video frame x_t and its extracted features $\Phi(x_t;\theta)$, the accumulated features z_t of x_t are obtained by:

$$z_t = \Psi\left(z_{t-1}, \Phi(x_t;\theta)\right). \quad (1)$$

Attention-Based Early-Exit

We design an attention-based cascade module to extract spatio-temporal correlations between successive T video frames for determining DNN exit points. Each attention module is implemented as a lightweight two-layer neural network with an 1×1 convolution kernel, featuring 64 neu-

rons in the first layer and 32 neurons in the second layer. Let β denote the attention module with parameter σ that takes the accumulated features z_t and z_{t-1} as input to generate attention weights. The aggregated features τ_t are obtained from the accumulated features z_t as:

$$\tau_t = W_1 z_{t-1} \cdot \tanh(W_2 z_t), \tag{2}$$

where W_1 and W_2 are the weights to be optimized. The normalized weight $\beta(\sigma_t)$ proportional to video frame complexity can then be obtained from the aggregated features τ_t:

$$\beta(\sigma)_t = \frac{\exp(\tau_t)}{\sum_t \exp(\tau_t)}. \tag{3}$$

The early-exit DNN model contains E exit points, each equipped with a gate unit that enables early exiting. Each gate is represented as a binary decision function $\rho()$ parameterized by μ, which determines whether a video frame has reached the required accuracy threshold for early exit during DNN inference. This way, each gate $\rho()$ is designed to evaluate the video frame complexity by processing the accumulated features z_t and z_{t-1} along with the attention weight $\beta(\sigma)_t$. Let t^* denote the earliest frame that meets the exit condition. The early-exit function e_{t^*} is formulated as:

$$e_{t^*} = \arg\min_t \rho\left((z_{t-1}, z_t, \beta(\sigma)_t); \mu\right). \tag{4}$$

If no gate meets the exit condition, the entire DNN model is used for inference.

In the `E4` framework, each DNN exit point corresponds to a classifier. For each exit point, the layer l of each classifier f_t within the given DNN layer indexed by frame t is dynamically determined. Each classifier f_t takes the accumulated features z_t as input and predicts the final video label. The training optimizes the parameters γ of the classifier f_t using the standard cross-entropy loss ℓ_{CE}:

$$\mathcal{L}_{cls} = \frac{1}{T}\sum_{t=1}^{T} \ell_{CE}\left(f_t(z_t;\gamma), y\right), \tag{5}$$

where y is the label of the video frame.

In `E4`, each gate is implemented as a multi-layer perceptron (MLP) with low computational overhead. Specifically, the two-layer MLP processes the aggregated features z_t and z_{t-1} along with the normalized weights $\beta(\sigma_t)$, with 64 and 32 neurons per layer, respectively. We optimize the gate parameter μ using the binary cross-entropy (BCE) loss:

$$\mathcal{L}_{gate} = \frac{1}{T}\sum_{t=1}^{T} \ell_{BCE}\left(\rho((z_{t-1}, z_t, \beta(\sigma)_t); \mu), y\right). \tag{6}$$

We also use the standard cross-entropy (CE) loss to optimize the parameters σ of the attention network β:

$$\mathcal{L}_{att} = \frac{1}{T}\sum_{t=1}^{T} \ell_{CE}(\beta(z_t;\sigma), y). \tag{7}$$

The final loss to train the early-exit DNN model is:

$$\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{gate} + \mathcal{L}_{att}. \tag{8}$$

Algorithm 1: CDS-Based Frequency Scaling Algorithm

Input: The DNN exit points e_t, the CPU-GPU clock frequency range $[C_{min}, C_{max}], [G_{min}, G_{max}]$

Output: The optimal CPU-GPU clock frequency pair (C_f^*, G_f^*) for each layer before e_t

```
Initialize a dictionary D ← {(C_{f_0}, G_{f_0}); power}.
for round r = 1 to R do
    while C_f ∈ [C_min, C_max] and G_f ∈ [G_min, G_max] do
        for i = 1 to e_t do
            Sample N candidates (C_f, G_f).
            for n-th candidate do
                Profile the energy consumption p.
                D ← {(C_f, G_f); power}
            end for
            Update the CPU-GPU clock frequency to the
            one with the lowest energy consumption.
        end for
    end while
end for
Sort D by the profiled energy consumption.
return (C_f^*, G_f^*).
```

Just-In-Time (JIT) Profiler

After analyzing frame complexity and determining the DNN exit point, we implemented a *Just-In-Time (JIT) Profiler* to efficiently identify the optimal power management configurations via coordinate descent search (CDS). CDS is a non-gradient optimization method focusing the search on one dimension at a time while keeping the values of other dimensions fixed. By alternating between dimensions, it converges to the optimal solution. This approach effectively transforms multi-variable optimization problems into single-variable ones, enhancing sampling efficiency. We also compare CDS performance with other methods like random search, evaluating both latency and energy consumption.

Algorithm 1 outlines our CDS-based dynamic frequency scaling approach using dynamic voltage and frequency scaling (DVFS) technology. The algorithm takes DNN exit points e_t and the CPU and GPU clock frequency ranges $[C_{min}, C_{max}], [G_{min}, G_{max}]$ as input. It initializes a dictionary $\mathcal{D}$ to track CPU and GPU clock frequencies along with the corresponding energy consumption p for each round. For the searching of optimal settings, the CDS algorithm treats the layer before each DNN exit point as a separate coordinate. It then alternates the search for the optimal CPU-GPU clock frequencies for each layer, keeping other coordinates fixed at their previous optimal values. With each iteration, CPU-GPU clock frequencies are updated layer by layer. The process continues until all layers have been optimized, returning the best CPU-GPU clock frequency configuration. Compared to random search, CDS can find near-optimal solutions more quickly, as demonstrated in our performance evaluation experiments.

In summary, the CDS-based JIT Profiler effectively finds the optimal low-power CPU and GPU clock frequencies for each layer before the DNN exit points, ensuring low-latency.

Table 1: Configurations of edge devices used in our experiments.

Edge GPU	Computing Power	DRAM	CPU	GPU	Max Power
Jetson Nano	0.47TFLOPS (FP16)	4GB	4×Cortex-A57@1.4GHz	128×Maxwell@0.9GHz	10W
Jetson TX2	1.33TFLOPS (FP16)	8GB	6×Cortex-A57@1.4GHz	256×Pascal@1.3GHz	15W
Jetson Xavier NX	21TOPS (INT8)	8GB	6×Carmel@1.4GHz	384×Volta@1.1GHz	20W
Jetson Orin Nano	40TOPS (INT8)	8GB	6×Cortex-A78AE@1.5GHz	512×Ampere@0.6GHz	15W
Jetson AGX Orin	275TOPS (INT8)	64GB	12×Cortex-A78AE@2.2GHz	2048×Ampere@1.3GHz	60W

DVFS Governor Settings

We next describe how we customize the DVFS governor specifically for edge video analytics, adapting it to both video frames and DNN models. Specifically, we use the Nvidia `nvpmodel` (Nvidia Corporation 2022), a performance governor for the Jetson series edge devices, which allows users to dynamically adjust the CPU and GPU clock frequencies. For example, the Jetson AGX Orin allows its CPU and GPU clock frequencies to scale between 0.1 and 2.2GHz, and 0.1 to 1.13GHz, respectively. Based on the identified DNN exit points, the DVFS governor dynamically adjusts the CPU and GPU clock frequencies according to the optimal power setting provided by the JIT Profiler. Since DVFS governor operates with millisecond-level latency, the primary overhead is attributed to the JIT Profiler, which we will evaluate in detail later.

Implementation and Evaluation

Implementation of E4

`E4` prototype: We implemented `E4` using Python 3.6 across five heterogeneous Nvidia edge devices. The specific configurations of edge devices are summarized in Table 1. The minimum clock frequency for all edge devices is set to 0.1GHz to ensure basic system operation. We utilize textttjetson-stats (R Bonghi 2024) to measure the energy consumption of edge devices during DNN inference.

Baselines: We compare `E4` with the following alternatives.
• **EENet** (Li et al. 2023b) is a state-of-the-art energy efficient inference method incorporating early-exit and DVFS. It provides frequency and voltage calibration advice over short timescales by predicting where the DNN model will exit based on inference workloads and timing constraints.
• **zTT** (Kim et al. 2021) is a recent workload-aware DVFS governor that utilizes deep reinforcement learning (DRL) to adopt to ambient temperature, aiming to optimize CPU and GPU clock frequencies for energy savings.
• **Ring-DVFS** (Yeganeh-Khaksar et al. 2020) is a DRL enhanced DVFS governor designed for multi-core embedded systems, aimed at reducing energy consumption. This approach is applied to edge devices with heterogeneous CPU-GPU processors using Nvidia `nvpmodel` (Nvidia Corporation 2022) for a fair comparison.
• **E4-R** is a reduced baseline of our proposed `E4` with CDS replaced with random search. Random search samples CPU-GPU clock frequencies and power management configurations randomly, and profiles their energy consumption as the cost. A cache is used to record all schedules and their associated costs. After a specified number of search rounds, random search returns the schedule with the lowest found energy consumption.

Experimental Setup

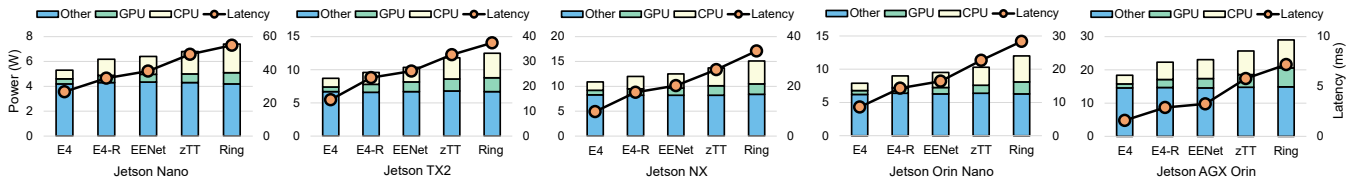
DNN models: We utilize EfficientNet-B0 (Tan and Le 2019) and MobileNet-v2 (Sandler et al. 2018) for edge video analytics. Both DNN models are pretrained on the ImageNet dataset, with five DNN exit points configured for each model. We remove the last classification layer from the backbone and replace it with a fully-connected layer with 1024 neurons. The early-exit DNN models are trained offline using four Nvidia 3080 GPUs with a mini-batch size of 512. Training is conducted with the Adam optimizer and a learning rate of 10^{-4}.

Datasets: We conduct experiments on two large-scale datasets: ActivityNet-v1.3 (Caba Heilbron et al. 2015) and Mini-Kinetics (Kay et al. 2017). ActivityNet-v1.3 is a long-range action recognition dataset comprising 20,000 videos across 200 classes (10,024 videos for training, 4,926 for validation, and 5,044 for testing). Each video has an average duration of 167 seconds and 1.5 labels. Mini-Kinetics, provided by (Meng et al. 2020), is a short-range action recognition dataset featuring 200 classes from the Kinetics dataset (Caba Heilbron et al. 2015), with 121,215 training and 9,867 test videos, each averaging 10 seconds in duration. We use *top-1 accuracy* to evaluate multi-label classification performance in ActivityNet, and *mean average precision (mAP)* for multi-class classification on Mini-Kinetics.

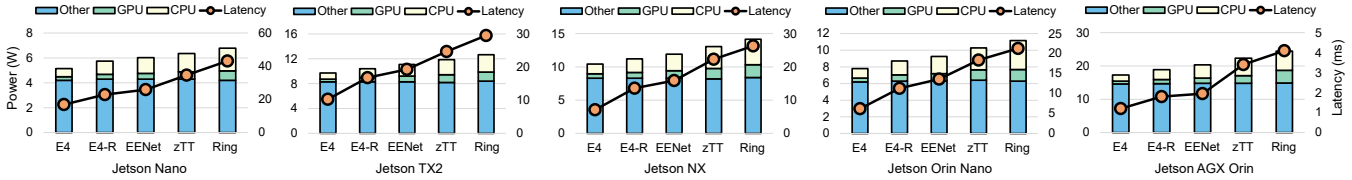
Performance Evaluation

Comparison of energy and latency: We compare the inference performance of `E4` with other methods across two DNN models and two datasets, using the five heterogeneous edge devices listed in Table 1.

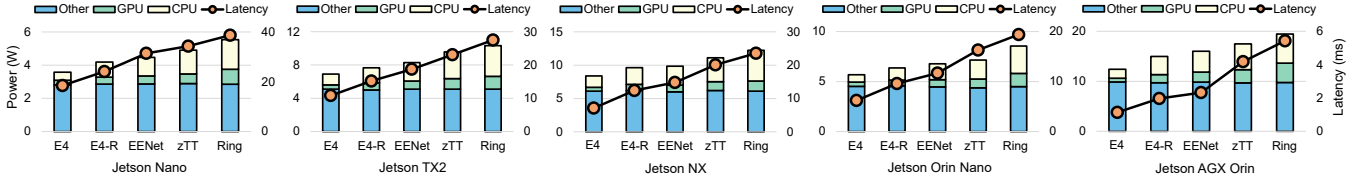
Fig. 3(a,c) shows that for video analysis with EfficientNet-B0 (Tan and Le 2019), energy consumption is primarily driven by non-video analytics system operation (denoted as "Other"), followed by CPU and GPU usage for data preprocessing, transmission, and parallel computing. `E4` improves energy efficiency, achieving 20% to 37% energy saving and a 1.3× to 2.2× speedup. The performance improvement of `E4` is attributed to DNN's early-exit mechanism, which significantly reduces computation cost and energy consumption by partial DNN inference. In comparison, zTT (Kim et al. 2021) and Ring-DVFS (Yeganeh-Khaksar et al. 2020) focus solely on optimizing the clock frequency of individual heterogeneous processors. Even compared to EENet (Li et al. 2023b), a state-of-the-art energy-efficient inference framework that



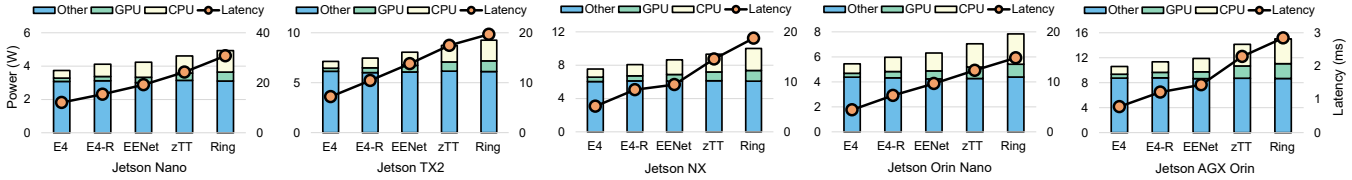
(a) EfficientNet-B0 on ActivityNet-v1.3 dataset



(b) MobileNet-v2 on the ActivityNet-v1.3 dataset



(c) EfficientNet-B0 on the Mini-Kinetics dataset



(d) MobileNet-v2 on the Mini-Kinetics dataset

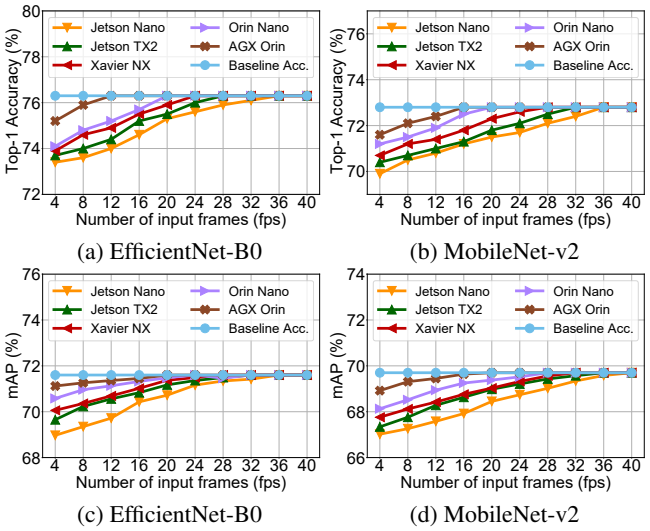
Figure 3: Comparison of energy consumption and inference latency for E4 *vs.* baseline approaches (EENet, zTT, and Ring).

coordinates early exit and DVFS, E4 has lower inference latency and higher energy saving, thanks to an attention-based cascade that cautiously guides early exit and DVFS governor by analyzing the complexity of video frames.

In Fig. 3(b) and Fig. 3(d), we further examine the video analytics task using MobileNet-v2 (Sandler et al. 2018). It can be observed that E4 reduces 18%$\sim$30% energy consumption and achieves $1.4\times\sim1.9\times$ inference speedup, respectively. Since the computation complexity of MobileNet-v2 (Sandler et al. 2018) is lower than EfficientNet-B0 (Tan and Le 2019), the search space of our CDS-based *JIT Profile* is limited, which causes the energy-efficient improvement to be slightly lower than EfficientNet-B0 (Tan and Le 2019).

Moreover, E4 has better performance than E4-R for five heterogeneous edge devices. The results reveal that although the random search algorithm is simple, it could significantly reduce the runtime energy consumption and latency, highlighting the advantages of our framework design. Interestingly, we find that compared with edge devices with low computing power, E4 brings more significant performance improvement to edge devices with high computing power. For instance, the performance improvement of Jetson AGX Orin, which has the highest computing power, is 37% higher on average than that of Jetson Nano with the lowest computing power. The results are attributed to the fact that high computing power means a larger frequency range, so that E4 has a larger optimization space.

Impact of number of input frames on accuracy: The accuracy of early-exit in E4 is closely related to the number of input frames. Intuitively, the higher the number of input frames, the richer the information of the frames extracted by the aggregated feature pool, which means higher accuracy, but higher computational overhead. We evaluate the accuracy of E4 by scaling the number of input frames. As shown in Fig. 4, we report the accuracy of E4 on ActivityNet and Mini-Kinetics dataset with various frame rates, respectively. In particular, edge devices with high computing power can process more video frames within the same time window, which means better adaptability to frame rates ($e.g.$, the accuracy loss of Jetson AGX Orin is within 1%). For instance, E4 leverages more video frames for complex objects, and conversely uses fewer frames to inference simple objects. It can also be observed that the accuracy of E4 improves with the number of input frames , but up tp a certain limit. On the one hand, complex objects may require more frames to be recognized. On the other hand, the reason why an excessively high number of frames cannot further improve the accuracy is attributed to the inability of convolutional neural



(a) EfficientNet-B0 (b) MobileNet-v2

(c) EfficientNet-B0 (d) MobileNet-v2

Figure 4: Effect of input frame rate *vs.* inference accuracy on (a,b) ActivityNet-v1.3 and (c,d) Mini-Kinetics datasets.

networks in leveraging temporal information. Therefore, we set $T = 20$ in our experiments to keep the balance between accuracy and computational overhead.

Ablation study: We inspect the performance improvement of different components in E4, including the CDS-based DVFS governor and the attention-based early-exit framework. For all ablations, we follow the same training procedure explained in Section Experimental Setup. Jetson AGX Orin is uesd for DNN inference on Activitynet-v1.3 and Mini-Kinetics dataset, respectively. Table 2 reports the energy consumption and inference latency for all combinations. We first evaluate the behavior of DVFS governor. Despite increasing 12% inference latency on average, DVFS governor reduces 26% energy consumption, which is desirable. We then inspect the impact of attention-based DNN's early-exit mechanism. It can be observed that inference latency and energy consumption are significantly reduced by 31% and 28% respectively, which is attributed to partial DNN inference, achieving energy saving while reducing computation cost. Compared with the original DNN inference without DVFS and early-exit, the performance improvement of E4 is the most significant, reducing the energy consumption and inference latency by up to 46% and 73%, respectively. It reveals the effective complementarity of DVFS governor and DNN's early-exit mechanism in co-optimizing energy consumption and inference latency.

Memory usage: As shown in Fig. 5, we report the memory usage of E4 compared to other approaches. Clearly, the memory usage of E4 consistently outperforms other approaches. Compared to the baseline (without early exit and DVFS), the average memory usage is reduced by up to 55%, which is promising for resource-constrained (especially memory) edge devices, further facilitating the deployment of video analytics to edge devices or even microprocessor units (MCU) with lower resource and power consumption, thereby reducing costs. Furthermore, E4 reduces memory usage by 45%∼49% compared to *zTT* (Kim et al. 2021) and *Ring-DVFS* (Yeganeh-Khaksar et al. 2020) without early exit, which is mainly attributed to the memory saving from early exit. In comparison, the performance improvement of DVFS technology for energy saving is limited, saving only 6%∼10% in memory usage compared to baseline. Although *EENet* (Li et al. 2023b) coordinates early exit and DVFS, it does not focus on the complexity of video frames, thus increasing memory usage by 23% compared to E4. Overall, early exit dominates memory usage, while the DVFS technology can further reduce memory usage by fine-grained tuning of the CPU and GPU clock frequencies.

Table 2: Ablation studies of DVFS and early-exit on Nvidia Jetson AGX Orin.

DNN	DVFS	Early-Exit	Latency (ms)	Power (W)
Eff-B0 on Act	✗	✗	6.3	34.6
	✓	✗	7.1	23.1
	✗	✓	4.6	24.5
	✓	✓	**1.6**	**18.4**
Mob-v2 on Act	✗	✗	4.1	27.3
	✓	✗	4.9	21.6
	✗	✓	3.4	23.1
	✓	✓	**1.2**	**17.3**
Eff-B0 on Mini	✗	✗	3.9	23.7
	✓	✗	4.3	18.2
	✗	✓	2.4	16.7
	✓	✓	**1.1**	**12.4**
Mob-v2 on Mini	✗	✗	3.2	22.7
	✓	✗	3.6	16.9
	✗	✓	1.7	14.2
	✓	✓	**0.8**	**10.6**

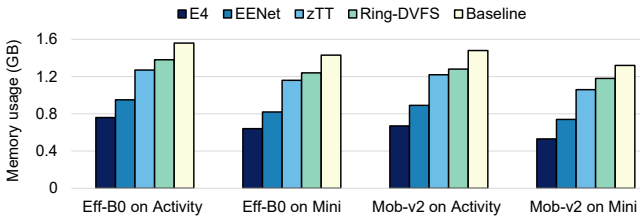


Figure 5: Comparison of memory usage between E4 and other approaches using two DNN models for edge video analysis on the ActivityNet-v1.3 and Mini-Kinetics datasets.

Conclusions

In this paper, we propose E4, an energy-efficient DNN inference framework for edge video analytics. E4 introduces two design knobs to enable energy-efficient DNN inference: the early-exit mechanism of DNN models and a Just-in-Time profiler to obtain optimal CPU and GPU clock frequency configurations. Comprehensive experiments with prototype implementations on heterogeneous edge devices show that E4 outperforms state-of-the-art early-exit approaches in terms of energy consumption and inference latency without sacrificing accuracy significantly. We earnestly hope that E4 will inspire the community to consider energy as a first-class resource in DNN optimization in edge video analytics.

References

Andriluka, M.; Iqbal, U.; Insafutdinov, E.; Pishchulin, L.; Milan, A.; Gall, J.; and Schiele, B. 2018. Posetrack: A benchmark for human pose estimation and tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5167–5176.

Andriluka, M.; Pishchulin, L.; Gehler, P.; and Schiele, B. 2014. 2d human pose estimation: New benchmark and state of the art analysis. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*, 3686–3693.

Bhardwaj, R.; Xia, Z.; Ananthanarayanan, G.; Jiang, J.; Shu, Y.; Karianakis, N.; Hsieh, K.; Bahl, P.; and Stoica, I. 2022. Ekya: Continuous learning of video analytics models on edge compute servers. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 119–135.

Caba Heilbron, F.; Escorcia, V.; Ghanem, B.; and Carlos Niebles, J. 2015. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the ieee conference on computer vision and pattern recognition*, 961–970.

Cui, W.; Zhao, H.; Chen, Q.; Wei, H.; Li, Z.; Zeng, D.; Li, C.; and Guo, M. 2022. {DVABatch}: Diversity-aware {Multi-Entry}{Multi-Exit} batching for efficient processing of {DNN} services on {GPUs}. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*, 183–198.

Ghodrati, A.; Bejnordi, B. E.; and Habibian, A. 2021. Frameexit: Conditional early exiting for efficient video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15608–15618.

Han, S.; Mao, H.; and Dally, W. J. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

Jeon, S.; Choi, Y.; Cho, Y.; and Cha, H. 2023. HarvNet: Resource-Optimized Operation of Multi-Exit Deep Neural Networks on Energy Harvesting Devices. In *Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services*, 42–55.

Jhuang, H.; Gall, J.; Zuffi, S.; Schmid, C.; and Black, M. J. 2013. Towards understanding action recognition. In *Proceedings of the IEEE international conference on computer vision*, 3192–3199.

Jiang, S.; Lin, Z.; Li, Y.; Shu, Y.; and Liu, Y. 2021. Flexible high-resolution object detection on edge devices with tunable latency. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, MobiCom '21, 559–572.

Kakolyris, A. K.; Katsaragakis, M.; Masouros, D.; and Soudris, D. 2023. RoaD-RuNNer: Collaborative DNN partitioning and offloading on heterogeneous edge systems. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 1–6. IEEE.

Kay, W.; Carreira, J.; Simonyan, K.; Zhang, B.; Hillier, C.; Vijayanarasimhan, S.; Viola, F.; Green, T.; Back, T.; Natsev, P.; et al. 2017. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*.

Khani, M.; Ananthanarayanan, G.; Hsieh, K.; Jiang, J.; Netravali, R.; Shu, Y.; Alizadeh, M.; and Bahl, V. 2023. {RECL}: Responsive {Resource-Efficient} continuous learning for video analytics. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, 917–932.

Kim, S.; Bin, K.; Ha, S.; Lee, K.; and Chong, S. 2021. zTT: Learning-based DVFs with zero thermal throttling for mobile devices. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, 41–53.

Laskaridis, S.; Venieris, S. I.; Almeida, M.; Leontiadis, I.; and Lane, N. D. 2020. SPINN: synergistic progressive inference of neural networks over device and cloud. In *Proceedings of the 26th annual international conference on mobile computing and networking*, 1–15.

Li, M.; Li, Y.; Tian, Y.; Jiang, L.; and Xu, Q. 2021. AppealNet: An efficient and highly-accurate edge/cloud collaborative architecture for DNN inference. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 409–414. IEEE.

Li, X.; Lou, C.; Chen, Y.; Zhu, Z.; Shen, Y.; Ma, Y.; and Zou, A. 2023a. Predictive exit: Prediction of fine-grained early exits for computation-and energy-efficient inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 8657–8665.

Li, X.; Shen, Y.; Zou, A.; and Ma, Y. 2023b. EENet: Energy Efficient Neural Networks with Run-time Power Management. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 1–6. IEEE.

Liang, Q.; Hanafy, W. A.; Bashir, N.; Ali-Eldin, A.; Irwin, D.; and Shenoy, P. 2023. Dělen: Enabling Flexible and Adaptive Model-serving for Multi-tenant Edge AI. In *Proceedings of the 8th ACM/IEEE Conference on Internet of Things Design and Implementation*, 209–221.

Lin, C.; Wang, K.; Li, Z.; and Pu, Y. 2023. A Workload-Aware DVFS Robust to Concurrent Tasks for Mobile Devices. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, 1–16.

Meng, Y.; Lin, C.-C.; Panda, R.; Sattigeri, P.; Karlinsky, L.; Oliva, A.; Saenko, K.; and Feris, R. 2020. Ar-net: Adaptive frame resolution for efficient action recognition. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, 86–104. Springer.

Menon, V. V.; Feldmann, C.; Amirpour, H.; Ghanbari, M.; and Timmerer, C. 2022. VCA: video complexity analyzer. In *Proceedings of the 13th ACM multimedia systems conference*, 259–264.

Nvidia Corporation. 2022. nvpmodel. Accessed on 2024-03-18, Version 35.1.

Padmanabhan, A.; Agarwal, N.; Iyer, A.; Ananthanarayanan, G.; Shu, Y.; Karianakis, N.; Xu, G. H.; and Netravali, R. 2023. Gemel: Model Merging for {Memory-Efficient},{Real-Time} Video Analytics at the Edge. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, 973–994.

R Bonghi. 2024. jetson-stats. Accessed on 2024-04-06, Version 4.2.7.

Ren, P.; Xiao, Y.; Chang, X.; Huang, P.-Y.; Li, Z.; Chen, X.; and Wang, X. 2021. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)*, 54(4): 1–34.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4510–4520.

Shubha, S. S.; and Shen, H. 2023. AdaInf: Data Drift Adaptive Scheduling for Accurate and SLO-guaranteed Multiple-Model Inference Serving at Edge Servers. In *Proceedings of the ACM SIGCOMM 2023 Conference*, 473–485.

Tambe, T.; Hooper, C.; Pentecost, L.; Jia, T.; Yang, E.-Y.; Donato, M.; Sanh, V.; Whatmough, P.; Rush, A. M.; Brooks, D.; et al. 2021. Edgebert: Sentence-level energy optimizations for latency-aware multi-task nlp inference. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, 830–844.

Tan, M.; and Le, Q. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, 6105–6114. PMLR.

Teerapittayanon, S.; McDanel, B.; and Kung, H.-T. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, 2464–2469. IEEE.

Toshev, A.; and Szegedy, C. 2014. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1653–1660.

Wang, L.; Zhang, X.; Su, H.; and Zhu, J. 2024. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Yeganeh-Khaksar, A.; Ansari, M.; Safari, S.; Yari-Karin, S.; and Ejlali, A. 2020. Ring-DVFS: Reliability-aware reinforcement learning-based DVFS for real-time embedded systems. *IEEE Embedded Systems Letters*, 13(3): 146–149.

You, J.; Chung, J.-W.; and Chowdhury, M. 2023. Zeus: Understanding and optimizing {GPU} energy consumption of {DNN} training. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, 119–139.

Zhang, S.; Cui, W.; Chen, Q.; Zhang, Z.; Guan, Y.; Leng, J.; Li, C.; and Guo, M. 2022. PAME: Precision-aware multi-exit dnn serving for reducing latencies of batched inferences. In *Proceedings of the 36th ACM International Conference on Supercomputing*, 1–12.

Zhang, Z.; Zhao, Y.; Li, H.; Lin, C.; and Liu, J. 2024. DVFO: Learning-Based DVFS for Energy-Efficient Edge-Cloud Collaborative Inference. *IEEE Transactions on Mobile Computing*.

Zhang, Z.; Zhao, Y.; and Liu, J. 2023a. DVFO: Dynamic Voltage, Frequency and Offloading for Efficient AI on Edge Devices. In *Proceedings of the 22nd International Conference on Information Processing in Sensor Networks*, 304–305.

Zhang, Z.; Zhao, Y.; and Liu, J. 2023b. Octopus: SLO-Aware Progressive Inference Serving via Deep Reinforcement Learning in Multi-tenant Edge Cluster. In *International Conference on Service-Oriented Computing*, 242–258. Springer.

Zhao, Z.-Q.; Zheng, P.; Xu, S.-t.; and Wu, X. 2019. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11): 3212–3232.

Zou, Z.; Chen, K.; Shi, Z.; Guo, Y.; and Ye, J. 2023. Object detection in 20 years: A survey. *Proceedings of the IEEE*, 111(3): 257–276.

AAAI Paper Checklist

1. **This paper:**

   Includes a conceptual outline and/or pseudocode description of AI methods introduced. (yes)

   Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results. (yes)

   Provides well marked pedagogical references for less-familiar readers to gain background necessary to replicate the paper. (yes)

2. **Does this paper make theoretical contributions? (No)**

3. **Does this paper rely on one or more datasets? (yes)**

   A motivation is given for why the experiments are conducted on the selected datasets. (yes)

   All novel datasets introduced in this paper are included in a data appendix. (NA)

   All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (NA)

   All datasets drawn from the existing literature (potentially including authors' own previously published work) are accompanied by appropriate citations. (yes)

   All datasets drawn from the existing literature (potentially including authors' own previously published work) are publicly available. (yes)

   All datasets that are not publicly available are described in detail, with explanation why publicly available alternatives are not scientifically satisficing. (NA)

4. **Does this paper include computational experiments? (yes)**

   Any code required for pre-processing data is included in the appendix. (no).

   All source code required for conducting and analyzing the experiments is included in a code appendix. (no)

   All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (yes)

   All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from. (yes)

   If an algorithm depends on randomness. then the method used for setting seeds is described in a way sufficient to allow replication of results. (yes)

   This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system: names and versions of relevant software libraries and frameworks.(yes)

   This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics. (yes)

   This paper states the number of algorithm runs used to compute each reported result. (yes)

   Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average: median) to include measures of variation, confidence, or other distributional information. (yes)

   The significance of any improvement or decrease in performance is judged using appropriate statistical tests(e.g., Wilcoxon signed-rank). (yes)

   This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments.(NA)

   This paper states the number and range of values tried per (hyper-)parameter during development of the paper, along with the criterion used for selecting the final parameter setting. (yes)