PowerLens: An Adaptive DVFS Framework for Optimizing Energy Efficiency in Deep Neural Networks

Jiawei Geng[ab], Zongwei Zhu*[b], Weihong Liu[ab], Xuehai Zhou[ab], Boyu Li[b]
[a]School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China
[b]Suzhou Institute for Advanced Research, University of Science and Technology of China, Suzhou 215123, China
{gjw1998,lwh2017,llbbyy}@mail.ustc.edu.cn,{zzw1988*,xhzhou}@ustc.edu.cn

ABSTRACT

To address the power management challenges in deep neural networks (DNNs), dynamic voltage and frequency scaling (DVFS) technology is garnering attention for its ability to enhance energy efficiency without modifying the structure of DNNs. However, current DVFS methods, which depend on historical information such as processor utilization and task computational load, face issues like frequency ping-pong, response lag, and poor generalizability. Therefore, this paper introduces PowerLens, an adaptive DVFS framework. Initially, we develop a power-sensitive feature extraction method for DNNs and identify critical power blocks through clustering based on power behavior similarity, thereby achieving adaptive DVFS instrumentation point settings. Then, the framework adaptively presets the target frequency for each power block through a decision model. Finally, through a refined training and deployment process, we ensure the framework's effective adaptability across different platforms. Experimental results confirm the effectiveness of the framework in energy efficiency optimization.

KEYWORDS

DNNs, energy efficiency, adaptive DVFS, power characterization

1 INTRODUCTION

The capabilities of deep learning and deep neural networks (DNNs) in solving complex challenging problems have been widely recognized. With the increasing complexity of these models, effectively managing their power consumption has emerged as a critical issue in both research and practical applications [11].

To address this issue, dynamic voltage and frequency scaling (DVFS) has attracted widespread attention in research for its ability to optimize energy efficiency without modifying the structure of DNNs (e.g., pruning that alters DNN structure [17]). While effective in traditional domains, DVFS faces challenges in deep learning, highlighting the need for tailored solutions.

Current research on DVFS in deep learning often extends traditional strategies, relying on hardware states and historical data for runtime frequency adjustments. Although recent methods have incorporated features of DNNs, such as considering the computations in convolutional layers [21], they essentially just expand the use of historical data without breaking out of this paradigm. This essentially heuristic approach encounters significant limitations:

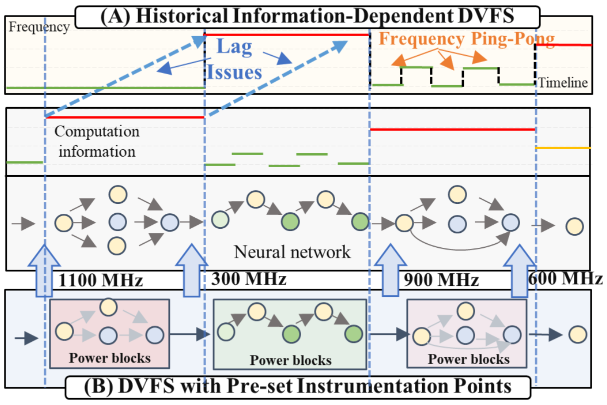


Figure 1: An illustration of the two DVFS methods.

① **Frequency Ping-Pong and Lag Issue.** As shown in Figure 1(A), the reliance on historical information [5, 12] (e.g., processor utilization) and heuristic rules [10], often leads to lag in response, creating a misalignment between computation needs and frequency adjustments. Moreover, rapid workload changes can cause DVFS systems to oscillate between settings, diminishing performance.
② **Frequency Tuning Accuracy Issue.** DVFS methods that rely on historical information may struggle to accurately predict the actual workload and performance demands of DNNs [12].
③ **Platform Generalizability Issue.** While existing methods may be effective in certain scenarios, they encounter challenges when transitioning to new hardware platforms. For example, transferring the method in [5] to different devices necessitates recalibrating the governor's utilization representations and thresholds.

Deep learning tasks possess structures suitable for pre-analysis [13], distinguishing them from general tasks. Therefore, we introduce the adaptive DVFS framework, PowerLens, emphasizing adaptability in three key aspects to address previous limitations:

Adaptive DVFS Instrumentation Points Granularity. To address frequency ping-pong and lag issues, we propose a power-sensitive feature extraction method and power behavior similarity clustering. As shown in Figure 1(B), this approach accurately identifies key power blocks, serving as the fundamental units for DVFS, thereby preventing frequency ping-pong. Moreover, this method alleviates lag issues by presetting DVFS instrumentation points before each block, enabling more precise frequency tuning.

Adaptive Target Frequency Setting. To address the issue of accuracy in frequency tuning, the framework employs a decision

DAC '24, June 23–27, 2024, San Francisco, CA, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0601-1/24/06…$15.00
https://doi.org/10.1145/3649329.3655956
∗ Corresponding author: Zongwei Zhu

model to predict and adaptively preset the target frequency for each block. This ensures that the frequency settings align with the varying performance and power requirements of each power block.

Adaptability to Hardware Platforms. To achieve platform generalizability, PowerLens provides a complete model training and framework deployment workflow without human intervention, ensuring stable performance across different hardware platforms.

The contributions of this work are summarized below.

- We propose a power-sensitive feature extraction method, paired with clustering based on power behavior similarity to identify critical power blocks in DNNs.
- An offline adaptive DVFS framework is developed. It optimizes the energy efficiency by accurately determining DVFS instrumentation points and their target frequencies through two deep learning-based prediction models.
- The framework has been successfully deployed on two hardware platforms, confirming its adaptability and effectiveness.

2 POWERLENS FRAMEWORK

This section delves into the adaptive DVFS framework, PowerLens, covering two core aspects as illustrated in Figure 2. Firstly, we explore the workflow and essential modules of PowerLens, which include power-sensitive feature extraction methods, clustering based on power behavior similarity, and adaptive DVFS decisions. These methods, connected through the intermediate representation. Subsequently, the model training stage encompasses a unified process from dataset generation to model training. This process yields two key prediction models: ① A model for predicting clustering hyperparameters, enabling the selection of appropriate clustering schemes for different DNNs. ② A decision model used to determine target frequency of each power block. Finally, we systematically analyze the offline and runtime overhead introduced by the framework.

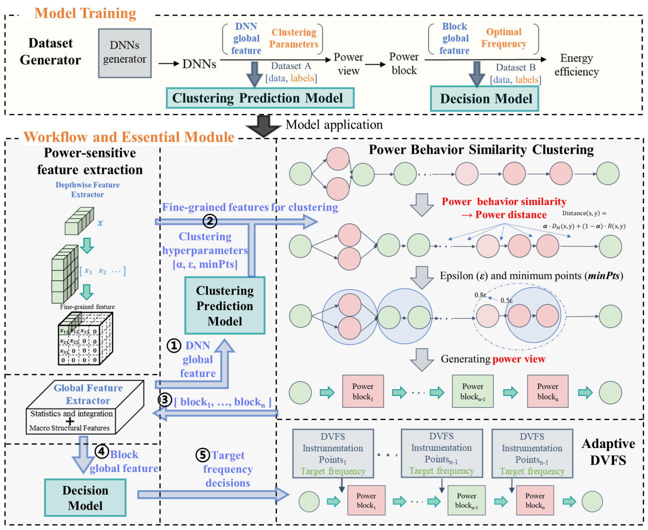

Model Training
Dataset Generator
DNNs generator
DNNs
DNN global feature
Clustering Parameters
Dataset A [data, labels]
Power view
Power block
Block global feature
Optimal Frequency
Dataset B [data, labels]
Energy efficiency
Clustering Prediction Model
Decision Model
Model application
Workflow and Essential Module
Power-sensitive feature extraction
Depthwise Feature Extractor
Fine-grained feature
Fine-grained features for clustering
Clustering hyperparameters [α, ε, minPts]
Clustering Prediction Model
① DNN global feature
②
③ [block1, ..., blockn]
Global Feature Extractor
Statistics and integration
Macro Structural Features
④ Block global feature
Decision Model
⑤ Target frequency decisions
Power Behavior Similarity Clustering
Power behavior similarity → Power distance
Epsilon (ε) and minimum points (minPts)
Generating power view
Power block
DVFS Instrumentation Points
Target frequency
Adaptive DVFS

Figure 2: The adaptive DVFS framework PowerLens.

2.1 Workflow and Essential Module

The PowerLens integrates modules including power-sensitive feature extraction, power behavior similarity clustering, and adaptive target frequency decision-making. The interconnection of various modules is facilitated through intermediate representations and prediction modules.

2.1.1 Workflow. As illustrated in figure 2, the framework operates as follows. ① Global features of the DNNs are extracted by a global feature extractor and are then used as inputs for the clustering hyperparameter prediction model. This model predicts clustering hyperparameters for the current DNNs. ② Subsequently, these clustering hyperparameters are combined with the network's fine-grained deepwise features for power behavior similarity clustering. ③ The clustering process maps the "power distance" between operators by calculating the *Mahalanobis distance* [3] and regularization, thus forming a power view. ④ The block information extracted from the power view is further processed by the global feature extractor to extract each block's global feature, serving as input for the decision model. This model outputs target frequency decisions for each block. ⑤ Based on these optimization decisions, target frequencies are preset in advance at each power block within the DNN. The entire framework achieves a closed-loop control from input to optimization decision, ensuring that the neural network achieves energy efficiency optimization while maintaining performance.

2.1.2 Power-Sensitive Feature Extraction. We delve into power-sensitive feature extraction methods of mixed granularity. Our goal is to construct a comprehensive representation that accurately reflects the power characteristics of DNNs through feature extraction and integration. Therefore, we have adopted two complementary methods: one is the *Depthwise Feature Extractor*, which delves into each layer of the network to extract **fine-grained layer-level features**; the other is the *Global Feature Extractor*, which operates from a global perspective to extract coarse-grained macroscopic features of the network. Subsequently, these features will be effectively expressed and fused to form intermediate representation, laying a solid foundation for subsequent clustering and prediction.

Depthwise Feature Extractor. The complexity and diversity of component types in DNNs, along with the heterogeneity of their functions, necessitate the extraction of fine-grained, layer-by-layer deepwise features. Specifically, it involves extracting crucial attributes such as computational load, number of parameters, volume of memory access, operator type, the number of input and output channels, and the dimensions of the feature map. For operator types that have a significant impact on power consumption, further extraction of their deep features are carried out to enhance the representational capability of these features. For instance, in the case of convolutional layers, we pay close attention to key parameters such as the dimensions of the convolutional kernels, the quantity of filters, and the stride length. As for Transformer modules, their inherent complexity necessitates a comprehensive examination. This includes delving into aspects like the number of heads in the attention mechanism, the dimensions of the matrices involved, as well as the parameters governing the fully connected and normalization layers. These attributes collectively reflect the computational and storage requirements of the layers, establishing a direct correlation with their power consumption.

Global Feature Extractor. We also need to engage in feature extraction at the level of entire blocks or DNNs, aiming to encapsulate the global power feature. Incorporating global features helps to circumvent potential misrepresentations that may arise from the locality of features. Specifically, this task includes two facets.

① Extraction of Macro Structural Features: This encompasses analyzing macro parameters of DNNs, such as the number of layers, depth, types, residual, and branching structures. These parameters serve to illustrate the overall scale and the complexity of its topological structure, providing insights into the global power patterns.

② Statistics and Aggregation of Features: This entails aggregating all the fine-grained features to generate a comprehensive summary, including calculations of the network's overall FLOPs, total number of parameters, and so forth. Additionally, an analysis of the proportions of various components can be conducted to elucidate the computational patterns.

2.1.3 Power Behavior Similarity Clustering. We divide the operators in the network based on their power feature by employing a power behavior similarity clustering method. This process not only maps and quantifies the **power distance** between operators but also divides different power blocks based on this mapping. The **power view**, composed of these **power blocks**, serves as a logical intermediate representation that intuitively presents the main paths and areas where power usage is concentrated within the network. Such clustering is crucial for subsequent energy efficiency optimization, as it allows us to target groups of operators with similar power characteristics for more effective DVFS instrumentation set.

Algorithm 1 embodies the core logic and implementation process of the clustering algorithm. Firstly, considering that in a multi-dimensional feature space, different features may have different scales and dimensions. The *Mahalanobis distance* [3] naturally adjusts the scale of these features through the covariance matrix. Therefore, we quantify the **power behavior similarity** in the feature space using the *Mahalanobis distance*.

Next, by introducing an inter-operator distance regularization term in the distance calculation, we ensure that only physically adjacent operators are considered, avoiding the erroneous clustering of operators that are not adjacent but have similar features.

Then, based on the neighborhood radius (ϵ) and minimum number of points in a cluster (*minPts*) hyperparameter of the DBSCAN algorithm [2], the network is divided into multiple power blocks, forming a power view. This process not only simplifies the complex network structure but also provides an intuitive and effective basis for making DVFS target frequency decisions, greatly enhancing the operability and precision of adaptive DVFS.

Finally, The post-processing of clustering results is primarily aimed at adjusting and optimizing the outcomes of clustering to ensure that the generated power blocks are continuous and practically feasible within the network's hierarchical structure. Post-processing not only deals with isolated points to ensure that each block is appropriately addressed but also involves adjusting size, shape, or membership of clusters to achieve better power view.

In this part, building on basic power behaviors and relationships, we have introduced clustering to enhance the understanding of DNN power patterns. This clustering has facilitated the implementation of adaptive DVFS instrumentation points granularity.

2.1.4 Adaptive Target Frequency Decision-Making. Utilizing the previously constructed power view, we identify power blocks, guiding the direction and methodology for precise energy efficiency control. In this phase, DVFS instrumentation points are set before each block of the DNNs, based on the established power view. In this process, the PowerLens adaptive presets target frequencies at instrumentation points based on the characteristics and requirements of each power block. Specifically, as illustrated in Figure 2, we employ a decision model where the input is the global features of each power block, and the output is the target frequency for that block. For example, in the case of computation-intensive blocks, the decision model will opt to increase the target frequency to alleviate computational load. Conversely, for memory-intensive blocks, considering energy efficiency requirements, it will decide to reduce the frequency. Then, during the inference of DNNs, adaptive DVFS are performed according to the preset instrumentation points and target frequencies, thereby enhancing energy efficiency.

Algorithm 1 Power Behavior Similarity Clustering

```
Require: Scaled power-sensitive deepwise features $X$;
    Clustering hyperparameter: neighborhood radius $\epsilon$, least number of operators $minPts$; $\alpha$, $\lambda$ for distance calculation.
Ensure: Power view
 1: $n \leftarrow$ number of layers in $X$
 2: Compute covariance matrix $C$ of $X$
 3: $P \leftarrow$ pseudo-inverse of $C$
 4: Initialize distance matrix $D \in \mathbb{R}^{n\times n}$ with zeros
 5: for each pair of layers $i, j$ in $X$ do
 6:     $D[i,j] \leftarrow \sqrt{(X[i]-X[j])^\top P(X[i]-X[j])}$
        {Mahalanobis distance}
 7: end for
 8: Initialize spacing regularization matrix $R \in \mathbb{R}^{n\times n}$ with zeros
 9: for each pair of layers $i, j$ in $X$ do
10:     $R[i,j] \leftarrow \exp(-\lambda \cdot |i-j|)$ {operator spacing}
11: end for
12: $Distance_{\text{final}} \leftarrow \alpha \cdot D[i,j] + (1-\alpha) \cdot R[i,j]$
13: $clusters \leftarrow \text{DBSCAN}(D_{\text{final}}, \epsilon, minPts)$
14: $newClusters \leftarrow \text{processClusters}(clusters)$
    {Ensure clusters are contiguous and non-overlapping}
15: return $newClusters$
```

2.2 Model Training Phase

The model training phase is a foundational phase of the PowerLens, encompassing multiple steps such as dataset generation, model training. In this phase, through a carefully designed data generator and training process, key datasets for prediction are constructed, and two core prediction models are trained: the *clustering parameter prediction model* and the *target frequency decision model*.

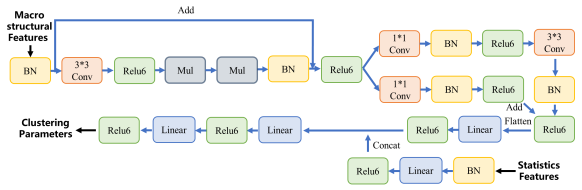


Figure 3: The clustering parameter prediction model.

As illustrated in Figure 2, the role of the dataset generator is to generate two groups of datasets. It first uses a DNN generator to produce a large variety of neural networks by randomly combining features mentioned in section 2.1.2. These networks are then subjected to clustering algorithms with varying hyperparameters, resulting in corresponding power views. Each block in the power view is deployed at all frequencies to select test data that achieves the optimal energy efficiency. Ultimately, two groups of datasets are outputted: Dataset A contains the global features of the neural networks and their corresponding clustering hyperparameters. Meanwhile, Dataset B includes the global features of each block and its optimal frequency.

The *clustering hyperparameter prediction model* is trained with the Dataset A, with its network structure depicted in Figure 3. As described in the global feature extractor of section 2.1.2, the global features of the DNN are divided into macro structural features and statistics features. Structural features are input at the beginning stage of the model to establish a basic understanding of the DNN structure. Statistics features are input during the mid-stage of the network to further enhance the prediction accuracy based on the existing structural understanding. This model achieves an accuracy of 92.6% [1] on the test set.

The *decision model*, trained with Dataset B, is shown in Figure 4. It predicts the target frequency settings based on the global features of each block. This is fundamentally a classification task, involving choosing between several frequency levels supported by the hardware, and it achieves an accuracy of 94.2% [1] on the test set. Moreover, even in cases of prediction deviation, the predicted target frequency is only one or two level away from the actual optimal frequency, thereby not significantly compromis.

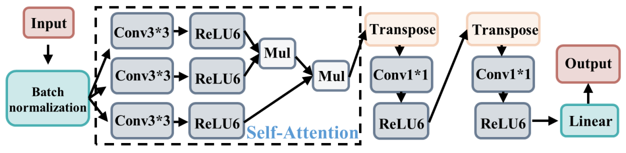


Figure 4: The target frequency decision model.

After extensive training and validation, prediction models provides solid predictive data support for the entire framework, ensuring the accuracy and effectiveness of optimization.

2.3 Framework Overhead Analysis.

2.3.1 Offline Overhead Analysis. PowerLens works offline, and its offline overhead primarily arises from model training and workflow. Training the clustering hyperparameter prediction model and the decision model requires data collection and training until convergence. Moreover, this method eliminates the need for manual intervention, as transferring it to a new hardware platform simply involves the automated generation of datasets and training. The overhead from workflow mainly involves feature extraction, hyperparameter prediction, clustering, predicting target frequencies, and generating power views. These are all completed before executing inference, hence they do not affect the performance of runtime.

2.3.2 Runtime Performance Overhead Analysis. The runtime overhead brought by the framework stems from two sources: ① The DVFS commands consume processor resources, thereby creating overhead. However, PowerLens achieves adaptive DVFS instrumentation granularity through clustering, thereby targeting only the key power blocks and reducing performance degradation caused by frequent frequency adjustments. Consequently, this precise control can effectively manage and balance such overhead, ensuring that it there is no significantly impact the overall runtime performance. ② PowerLens adaptively sets frequencies for each block, including turning down frequency for some blocks, which may lead to increased time overhead for these blocks. However, this performance sacrifice is made to achieve a higher energy efficiency ratio. This trade-off is often worthwhile, as it maintains or enhances overall energy efficiency while sacrificing only a small portion of performance. Importantly, this approach does not alter the structure or computational intensity of the neural network, thus not affecting the accuracy of inference.

[1]A total of 8000 random networks were generated, resulting in 31,242 block data. They are divided into training, validation, and test sets in an 80%-10%-10% ratio.

3 EXPERIMENTAL EVALUATION

3.1 Experimental Setup

Hardware Platforms. We deploy the PowerLens on two platforms, NVIDIA Jetson AGX and NVIDIA Jetson TX2. Some necessary runtime environments are installed, such as Jetpack 4.6.2, Ubuntu 16.04, CUDA 10.2, torchvision 0.12, PyTorch 1.12.0, working mode MAXN on AGX and TX2. Two platforms are configured with different GPU frequencies and batch sizes: On the AGX, frequencies ranges from 114MHz to 1370MHz across 14 levels. On the TX2, frequencies ranges from 114MHz to 1300MHz across 13 levels. To calculate the energy efficiency, we monitor real-time power by the performance management tool (tegrastats) in Jetson platform.

Models and Datasets. As shown in Table 1, 12 DNNs from torchvision with different sizes and computational complexity are used in our prediction accuracy and energy efficiency comparison experiments. The image data used for our test inference process is from ImageNet. Each energy efficiency test is required to be run 50 times on randomized different inputs to obtain an average result.

Baselines for comparison. To demonstrate the effectiveness and superiority of the proposed PowerLens, we use three baselines for comparison: ① The built-in method (**BiM**) [7]. We choose the ondemand method built into both hardware platforms, which are widely used frequency scaling methods relying on historical hardware. ② The **FPG** [5]. A new heuristic DVFS method is called **FPG-C+G** in this section. This method dynamically adjusts the CPU and GPU frequencies during runtime based on performance, power, energy delay product, and CPU/GPU utilization. ③ The **FPG-G** is a variant of the **FPG-C+G** that retains the ondemand for CPU and only adjusts the frequency scaling strategy for the GPU.

Experimental Metric. To quantitatively assess energy efficiency, we employ the equation 1 to calculate energy efficiency [20]. Energy efficiency is considered a positive indicator, with higher values indicating better performance in terms of energy utilization.

$$EE_{model} = \frac{Images}{E} = \frac{FPS \times t}{\bar{P} \times t} = \frac{FPS}{\bar{P}} (Images/J), \quad (1)$$

where *images* is the number of images processed by the model. E is the energy consumption, $\bar{P}$ is the average power, and t is the inference time. *FPS* represents frames per second.

3.2 Results

3.2.1 Energy efficiency improvement. In Table 1, we present the experimental results for energy efficiency improvement. Compared to BiM, PowerLens achieves an average improvement of 57.85% on

TX2 and an average of 119.42% on AGX. In addition, PowerLens achieves an average energy efficiency improvement of 18.39% on TX2 and 27.31% on AGX compared to the FPG-G method. Compared to the FPG-C+G method, which configures both CPU and GPU frequencies simultaneously, PowerLens achieves an average energy efficiency improvement of 13.53% on TX2 and 15.97% on AGX, despite only configuring GPU frequencies for PowerLens.

Table 1: Results of energy efficiency improvement.

(a) Energy efficiency improvement on TX2

model name	Block[1]	BiM[2]	FPG-G[2]	FPG-CG[2]
alexnet	1	38.60%	2.94%	1.31%
googlenet	1	30.10%	6.89%	4.32%
vgg19	2	43.40%	23.00%	20.76%
mobilenet_v3	1	29.76%	6.55%	3.96%
densenet201	3	35.76%	7.32%	5.53%
resnext101	4	79.79%	25.97%	21.07%
resnet34	1	41.86%	4.82%	1.45%
resnet152	3	59.85%	32.88%	24.10%
regnet_x_32gf	3	123.80%	15.47%	11.23%
regnet_y_128gf	4	131.71%	29.12%	20.59%
vit_base_16	1	36.95%	40.46%	24.70%
vit_base_32	1	42.67%	25.32%	23.39%
Average		57.85%	18.39%	13.53%

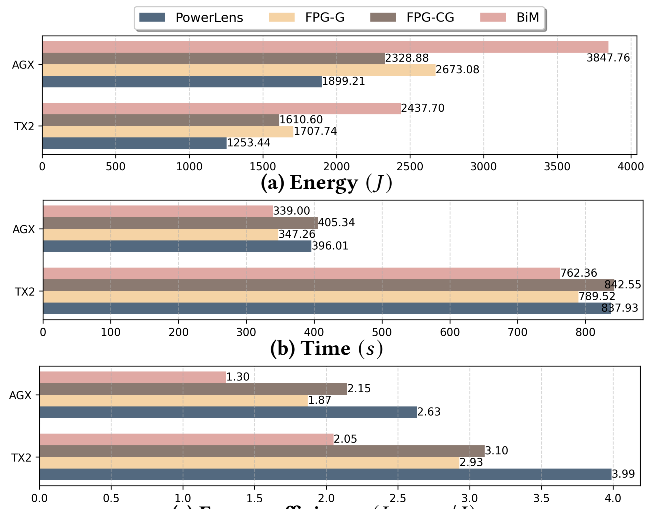
(b) Energy efficiency improvement on AGX

model name	Block[1]	BiM[2]	FPG-G[2]	FPG-CG[2]
alexnet	1	26.17%	10.55%	3.80%
googlenet	2	113.78%	7.55%	5.81%
vgg19	2	134.30%	37.78%	20.66%
mobilenet_v3	1	144.37%	6.40%	3.56%
densenet201	2	132.36%	11.49%	9.35%
resnext101	3	131.40%	38.78%	20.11%
resnet34	2	133.72%	3.97%	2.34%
resnet152	4	129.27%	49.87%	36.98%
regnet_x_32gf	2	129.40%	12.39%	8.89%
regnet_y_128gf	6	144.34%	45.37%	24.30%
vit_base_16	1	104.87%	67.90%	36.21%
vit_base_32	1	104.87%	67.90%	36.21%
Average		119.42%	27.31%	15.97%

From Table 1, we can also make the following observations: ① Smaller networks, such as *AlexNet* and *MobileNetV*3, lack a sufficient number of operators for clustering. This limitation hinders the potential to achieve significant gains with adaptive DVFS approaches. As a result, there is only few effect on improving energy efficiency. ② Complex networks can be clustered into more power blocks, and the number of blocks is positively correlated with energy efficiency improvement. For example, the comparison between *ResNet*34 and *ResNet*152, and between *RegNet_X_*32*GF* and *RegNet_Y_*128*GF*. ③ For networks composed of repeated components, PowerLens treats continuous repeated components as a whole block and makes decisions for the optimal frequency. This is facilitated by the considering of power behavior similarity in clustering. For instance, PowerLens treats the connections of repeated *transformer* modules in the *ViT* model as a large power block.

3.2.2 The performance in the inference task flow. We randomly assemble 100 inference tasks by combining the DNNs listed in Table 1. Each task includes 50 three-channel 224 × 224 images. As shown in Figure 5, in the experimental results for task flow processing, PowerLens exhibits the lowest energy consumption and the highest energy efficiency among the four methods. However, there is a certain degree of increase in inference time, due to the runtime overhead discussed in section 2.3.2. Compared to FPG-G, FPG-CG, and BiM, PowerLens achieves energy reductions of 26.60%, 22.18%, and 48.58% on TX2, and 28.95%, 18.45%, and 50.64% on AGX, respectively. In terms of time, PowerLens increases task flow processing time by 6.13%, -0.54%, and 9.91% on TX2, and 14.03%, -2.30%, and 16.82% on AGX, respectively. Moreover, PowerLens achieves energy efficiency gains of 36.24%, 28.49%, and 94.48% on TX2, and 40.75%, 22.62%, and 102.60% on AGX, respectively.

[1]Blocks represent the number of power blocks obtained in this dnn by clustering.
[1]The BiM, FPG-G and FPG-C+G columns are the energy efficiency (EE) gains of PowerLens relative to the comparative benchmark methods ($EE_{powerlens}$-EE_{BiM})/EE_{BiM}.



(a) Energy (J)

(b) Time (s)

(c) Energy efficiency $(Images/J)$

Figure 5: The experimental results in task flow processing.

3.2.3 Ablation studies. To demonstrate the effectiveness of power behavior similarity clustering in Algorithm 1, we compare PowerLens with two variations that employ different strategies. Specifically, P-R represents the approach where the clustering algorithm is replaced with random block partitioning, and P-N denotes the approach that does not use any clustering algorithm and directly makes frequency decisions for entire DNN. The energy efficiency losses relative to PowerLens for these two methods are shown in Table 2. PowerLens with Algorithm 1 outperforms the others.

Table 2: The energy efficiency loss for different clustering

DNN Models	TX2		AGX	
	P-R	**P-N**	**P-R**	**P-N**
alexnet	-26.49%	-20.55%	-31.49%	-3.45%
googlenet	-34.06%	-8.15%	-99.43%	-8.06%
vgg19	-30.57%	-25.75%	-74.25%	-17.36%
mobilenet_v3	-49.31%	-19.18%	-43.02%	-10.18%
densenet201	-25.23%	-9.13%	-27.71%	-14.73%
resnext101	-69.52%	-31.88%	-23.85%	-28.95%
resnet34	-66.84%	-6.25%	-85.46%	-8.62%
resnet152	-62.35%	-21.59%	-49.05%	-27.49%
regnet_x_32gf	-35.78%	-16.61%	-69.37%	-18.17%
regnet_y_128gf	-21.40%	-16.37%	-50.17%	-68.55%
vit_base_16	-42.62%	-5.06%	-96.81%	-11.29%
vit_base_32	-47.06%	-1.58%	-21.33%	-2.46%
Average	-42.60%	-15.17%	-55.99%	-18.28%

3.3 Overhead Analysis

The time overhead of PowerLens can be divided into offline stage overhead (discussed in section 3.3.1) and runtime stage overhead (discussed in section 3.3.2). The **offline overhead** of PowerLens is shown in Table 3. Moreover, to understand the **runtime overhead**, we have changed the DVFS level for 100 times and measured its average time overhead, which is 50ms for the device used in the experiments. The time overhead caused by turning down frequency depends on the specific scenario and is analyzed in Figure 5.

Table 3: Offline overhead of PowerLens.

Phase		TX2	AGX
Model Training	clustering hyperparameter prediction model	20h	15h
	decision model	6h	4.5h
Workflow	feature extraction	10s	10s
	hyperparameter prediction	320ms	150ms
	clustering	60s	60s
	decision of each block	220ms	130ms

4 RELATED WORK

Model modification. Numerous works has been conducted on modifying models to achieve energy efficiency improvements [4, 17], such as pruning [17], quantization [4], knowledge distillation [8]. In addition, some studies have shortened the inference time by selectively skipping certain operators or blocks [16]. However, these methods inevitably disrupt the original design and structure of DNNs, and may even result in reduced accuracy [18].

DVFS methods are favored due to their ability to optimize energy efficiency without modifying DNNs, and they are typically orthogonal to most model modification research [14]. Traditional DVFS methods typically follow predefined rules and struggle to effectively manage complex workloads [6]. Hence, researchers have explored various rule-based [7] and learning-based [9] DVFS. Recently, NeuOS [1] has been customized for DNN tasks, and zTT [6] introduces factors like quality of service. However, these methods primarily rely on hardware states and historical information gathered from previous executions [14], essentially following a heuristic thinking. They still cannot fully address the limitations mentioned in section 1, such as frequency ping-pong and lag responses.

Recent approaches have explored synergizing DVFS technology with factors like batchsize [15], adaptively balancing throughput and power, which however are orthogonal to the adaptive DVFS in this paper. Some other methods considers expanding the control scope of DVFS, such as integrating CPU and GPU control strategies [5, 19]. However, the substantial differences between CPU and GPU tasks can lead to a significant declines in control efficiency for one side due to resource changes on the other side. Nevertheless, the idea of coordinating multiple components for DVFS inspires our future directions for extending PowerLens.

5 CONCLUSION

The PowerLens extracts power-sensitive features from DNNs and performs clustering based on power behavior similarity, accurately identifying critical power blocks. We preset DVFS instrumentation points and target frequencies in front of each power block based on the clustered power view. The experimental results show that PowerLens achieves an average energy efficiency improvements of 88.64%, 22.85%, and 14.75%, respectively, compared to three baseline methods. In the future, we will incorporate more configurable optimization options into PowerLens, such as CPU DVFS and batchsize. Additionally, we plan to apply PowerLens in cloud servers, where more complex and diverse tasks can yield greater benefits.

ACKNOWLEDGMENTS

This work was partially supported by the National Key R&D Program of China (2022YFB4501600), the National Natural Science Foundation of China (No.62102390), the special fund for Jiangsu Natural Resources Development (Innovation Project of Marine Science and Technology, No.JSZRHYKJ202218), and the National Key Laboratory of Science and Technology on Space Micrwave (No.HTKJ2022KL504021).

REFERENCES

[1] Soroush Bateni and Cong Liu. 2020. NeuOS: A Latency-Predictable Multi-Dimensional Optimization Framework for DNN-driven Autonomous Systems. In *USENIX Annual Technical Conference*. 371–385.

[2] Rupanka Bhuyan and Samarjeet Borah. 2023. A Survey of Some Density Based Clustering Techniques. *CoRR* abs/2306.09256 (2023).

[3] Qiang Chen, Weizhong Yu, and et al. 2023. Rooted Mahalanobis distance based Gustafson-Kessel fuzzy C-means. *Inf. Sci.* 644 (2023), 118878.

[4] Insu Choi, Jae-Youn Hong, JaeHwa Jeon, and Joon-Sung Yang. 2023. RQ-DNN: Reliable Quantization for Fault-tolerant Deep Neural Networks. In *60th ACM/IEEE Design Automation Conference, DAC 2023*. IEEE, 1–2.

[5] Meruyert Karzhaubayeva, Aidar Amangeldi, and Jurn-Gyu Park. 2023. CNN Workloads Characterization and Integrated CPU-GPU DVFS Governors on Embedded Systems. In *IEEE Embedded Systems Letters*. 1–1.

[6] Seyeon Kim, Kyungmin Bin, Sangtae Ha, Kyunghan Lee, and Song Chong. 2021. zTT: Learning-Based DVFS with Zero Thermal Throttling for Mobile Devices. *GetMobile Mob. Comput. Commun.* 25, 4 (2021), 30–34.

[7] Young Geun Kim, Joonho Kong, and Sung Woo Chung. 2018. A Survey on Recent OS-Level Energy Management Techniques for Mobile Processing Units. *IEEE Trans. Parallel Distributed Syst.* 29, 10 (2018), 2388–2401.

[8] Shaojie Li and et al. 2023. Distilling a Powerful Student Model via Online Knowledge Distillation. *IEEE Trans. Neural Networks Learn. Syst.* 34, 11 (2023).

[9] Xiao Li, Lin Chen, Shixi Chen, and et al. 2022. Power Management for Chiplet-Based Multicore Systems Using Deep Reinforcement Learning. In *IEEE Computer Society Annual Symposium on VLSI, ISVLSI 2022*. IEEE, 164–169.

[10] Xinmei Li, Lei Mo, and et al. 2023. Approximation-Aware Task Deployment on Heterogeneous Multicore Platforms With DVFS. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 42, 7 (2023), 2108–2121.

[11] Xiangjie Li, Yingtao Shen, An Zou, and Yehan Ma. 2023. EENet: Energy Efficient Neural Networks with Run-time Power Management. In *60th ACM/IEEE Design Automation Conference, DAC 2023*. IEEE, 1–6.

[12] Chengdong Lin, Kun Wang, Zhenjiang Li, and Yu Pu. 2023. A Workload-Aware DVFS Robust to Concurrent Tasks for Mobile Devices. In *ACM MobiCom 2023, Madrid, Spain, October 2-6, 2023*. ACM, 19:1–19:16.

[13] Weihong Liu, Jiawei Geng, Zongwei Zhu, and et al. 2022. Sniper: cloud-edge collaborative inference scheduling with neural network similarity modeling. In *DAC '22: 59th ACM/IEEE Design Automation Conference*. ACM, 505–510.

[14] Francisco Mendes and et al. 2022. Decoupling GPGPU voltage-frequency scaling for deep-learning applications. *J. Parallel Distributed Comput.* 165 (2022), 32–51.

[15] Seyed Morteza Nabavinejad, Sherief Reda, and Masoumeh Ebrahimi. 2022. Coordinated Batching and DVFS for DNN Inference on GPU Accelerators. *IEEE Trans. Parallel Distributed Syst.* 33, 10 (2022), 2496–2508.

[16] Lois Orosa, Skanda Koppula, and et al. 2022. EcoFlow: Efficient Convolutional Dataflows for Low-Power Neural Network Accelerators. *CoRR* (2022).

[17] Richard Petri, Grace Li Zhang, Yiran Chen, and et al. 2023. PowerPruning: Selecting Weights and Activations for Power-Efficient Neural Network Acceleration. In *60th ACM/IEEE Design Automation Conference, DAC 2023*. IEEE, 1–6.

[18] Adrian Schwaiger, Kristian Schwienbacher, and Karsten Roscher. 2022. Beyond Test Accuracy: The Effects of Model Compression on CNNs. In *Proceedings of the Workshop on Artificial Intelligence Safety 2022 (SafeAI 2022)*, Vol. 3087.

[19] Qizhen Weng, Wencong Xiao, and et al. 2022. MLaaS in the Wild: Workload Analysis and Scheduling in Large-Scale Heterogeneous GPU Clusters. In *19th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2022*.

[20] Chunrong Yao and et al. 2021. Evaluating and analyzing the energy efficiency of CNN inference on high-performance GPU. *Concurr. Comput. Pract. Exp.* (2021).

[21] Chunrong Yao and et al. 2022. EAIS: Energy-aware adaptive scheduling for CNN inference on high-performance GPUs. *Future Gener. Comput. Syst.* 130 (2022).