G-Power: Architecture-level GPU Power Modeling with Aggregated Knowledge Foundations from Known GPUs

Qijun Zhang[1], Yao Lu[1], Shang Liu[1], Mengming Li[1], Chen Zhang[2], Dongbo Wang[3], Zhiyao Xie[1*]
[1]Hong Kong University of Science and Technology, [2]Shanghai Jiao Tong University, [3]UniVista
{qzhangcs, yludf, sliudx, mengming.li}@connect.ust.hk, chenzhang.sjtu@sjtu.edu.cn, wdb@univista-isg.com
eezhiyao@ust.hk

ABSTRACT

Graphics Processing Units (GPUs) have been serving as critical computation resources for large-scale parallel computations. With increasing chip complexity, power efficiency has become an important design objective for modern GPUs. GPU power optimization relies on fast power evaluation, requiring architecture-level GPU power model. However, because of the time-consuming power label collection, only simple microbenchmarks are adopted for training. The limitation of microbenchmarks as training data incurs low accuracy for existing architecture-level GPU power models like AccelWattch.

To address the limitation of microbenchmarks as training data, we propose G-Power, an architecture-level GPU power modeling framework that utilizes additional known GPU chips to provide additional knowledge. G-Power utilizes the aggregated knowledge foundation from additional known GPU chips and then performs fine-tuning on our target GPU. To provide foundations with additional known GPU chips and capture the similarity to utilize these foundations for fine-tuning, G-Power adopts a three-phase algorithm consisting of 1) pre-training with additional known chips, 2) attention-inspired aggregation, and 3) fine-tuning on our target GPU. We evaluate G-Power on four modern NVIDIA GPUs, demonstrating high accuracy. G-Power can achieve a low MAPE of 14% and a high correlation coefficient R of 0.88 on average, which are 22% lower MAPE and 0.36 higher R than AccelWattch.

CCS CONCEPTS

• **Hardware → Power estimation and optimization**.

KEYWORDS

Power model, machine learning

ACM Reference Format:
Qijun Zhang[1], Yao Lu[1], Shang Liu[1], Mengming Li[1], Chen Zhang[2], Dongbo Wang[3], Zhiyao Xie[1]. 2026. G-Power: Architecture-level GPU Power Modeling with Aggregated Knowledge Foundations from Known GPUs. In *63rd ACM/IEEE Design Automation Conference (DAC '26), July 26–29, 2026, Long Beach, CA, USA*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3770743.3804165

*Corresponding Author

DAC '26, July 26–29, 2026, Long Beach, CA, USA

ACM ISBN 979-8-4007-2254-7/2026/07
https://doi.org/10.1145/3770743.3804165

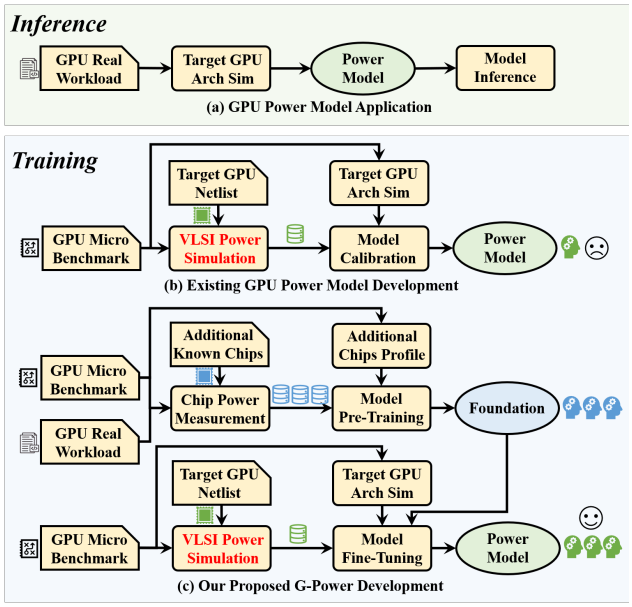


Figure 1: Workflow comparison between existing GPU power modeling [7] and our proposed G-Power. G-Power learns from additional known chips through the pre-training foundation to enable accurate modeling for our target GPU.

1 INTRODUCTION

Graphics Processing Units (GPUs) have become critical computation resources for accelerating large-scale parallel computations, such as machine learning, big data, and scientific computing [6, 9, 22, 25]. With the increasing complexity of GPU chips, power efficiency is becoming an increasingly important design objective for modern GPU designs, where even consumer-grade GPUs like RTX3090 can reach 350W. Power optimization of GPUs relies on power evaluation. Standard power simulation flow goes through RTL simulation, logic synthesis, physical design, and power simulation [1, 2, 24]. Such a power simulation flow is time-consuming, motivating architecture-level power models.

Power Model: Recent general RTL-stage power models [8, 27, 28, 36] eliminate the requirement of the VLSI flow, but RTL simulation is also prohibitively time-consuming. There have also been many architecture-level power models for CPU [3, 11, 30–35], while GPU models [7, 10] are few. GPU [7, 10]. GPU power modeling is more challenging than CPUs because the lack of high-quality open-source GPU designs forces researchers to use real GPU chips for power label collections, where no intermediate data is available.

Architecture-level GPU Power Models: AccelWattch [7], the updated version of GPUWattch [10], has been widely adopted for

GPU power modeling at the architecture level. As shown in Figure 1(a), for our target GPU design, AccelWattch performs architecture-level performance simulation for workloads to generate architectural events, and then takes these events to estimate GPU power. The development flow of AccelWattch is illustrated in Figure 1(b). For each GPU design, the AccelWattch is calibrated with linear regression. Because of the time-consuming power label collection, only short microbenchmarks with hundreds of instructions are adopted for calibration. After calibration, the model can predict the power consumption of long real workloads that have millions of instructions with fast architectural simulation. However, our measurement shows that when applying AccelWattch to Ampere and Ada GPUs, the mean absolute percentage error (MAPE) can be over 30%, and the correlation R can be under 0.6. Such a low accuracy is unacceptable to guide GPU power optimization.

Limitation of Microbenchmarks as Training Data: The low accuracy of the existing GPU power model is due to limitations of microbenchmarks as training data: the microbenchmarks' data distribution of the feature and label is different from real workloads. These microbenchmarks are short and simple, demonstrating a different data distribution from the long and complex real workloads that can not be adopted for training because of the time-consuming power simulation for our target GPU. This distribution difference results in low accuracy of the existing GPU power model workflow.

Our Solution–Use Data of Additional Known Chips: To address the limitation above, we propose G-Power, an architecture-level GPU power model that learns from additional known GPU chips with a pre-training foundation. Compared with the existing GPU power model that is trained with only our target GPU, G-Power utilizes additional known GPU chips to provide additional knowledge. Figure 1(c) illustrates the development flow of our proposed G-Power framework. G-Power learns from multiple additional known GPU chips, where data from both microbenchmarks and real workloads are available because of fast real chip power measurement.

How to Use Additional Known Chips: We point out that there are both similarities and differences among different GPUs. The similarity exists in both overall architecture and pipeline microarchitecture: 1) GPUs adopt a similar architectural design paradigm with SM, NoC, and memory partition. 2) GPUs have a similar pipeline design following the same hyper-threading microarchitecture that hides memory access latency. However, GPUs also have differences: 1) Different GPUs have different hardware configurations that are unavailable, such as the width of each pipeline stage, the number of processing units, and the cache configurations. 2) Different GPUs have different microarchitecture designs. For example, the warp scheduler, tensor core design, and memory access unit may differ across GPUs. Therefore, the key to G-Power is to *provide foundations with additional known GPU chips and capture the similarity to utilize these foundations for fine-tuning.*

G-Power adopts a three-phase algorithm: 1) *Pre-training with additional known chips.* It builds one foundation model for each additional known GPU chip. The foundation model is trained with both the microbenchmarks and the real workloads, incorporating comprehensive knowledge across programs. 2) *Attention-inspired aggregation.* This step captures the similarity. It estimates the similarity and then aggregates these foundation models based on this similarity to generate an aggregated foundation. 3) *Fine-tuning on our target GPU.* It fine-tunes the aggregated foundation for our target GPU with microbenchmark data from our target GPU.

Domain	Architectural Event	
SM Array	Total PTX Instruction	Total SASS Instruction
	Float and Integer Instruction	Fp64 Instruction
	DCache Read Hit	DCache Read Miss
	DCache Write Hit	DCache Write Miss
	Constant Cache Hit	Constant Cache Miss
	Shared Memory Access	Textural Cache Access
	ALU Access	Tensor Core Access
	FMA Access	DFMA Access
	DPU Access	EXU Access
	SM Activity	Warp Activity
NoC & L2Cache	L2Cache Read Hit	L2Cache Read Miss
	L2Cache Write Hit	L2Cache Write Miss
	NoC Request	NoC Response
Global Memory	Memory Read	Memory Write

Table 1: Our adopted architectural events in G-Power. It includes 28 events from the three GPU domains.

Contributions: Our contributions are summarized below.

- We propose G-Power, an architecture-level GPU power modeling framework that utilizes additional known chips to provide additional knowledge.
- G-Power introduces a three-phase algorithm to provide foundations with additional known GPU chips and capture the similarity to utilize these foundations for fine-tuning. It includes pre-training with additional known chips, attention-inspired aggregation, and fine-tuning on our target GPU.
- Evaluation of G-Power on four modern NVIDIA GPUs shows that G-Power can achieve a low MAPE of 14% and a high correlation R of 0.88 on average, which are 22% lower MAPE and 0.36 higher R than AccelWattch.

2 G-POWER OVERVIEW

In the GPU, each component has its architectural events that trigger the activity of its internal circuit. Because the same event always triggers the same circuit part, we can formulate the GPU power consumption as a linear function of architectural events. Denoting the number of events as N, the maximum power of component i as p_i, the idle power as p_{idle}, the i-th architectural event normalized by execution time as e_i, GPU power is formulated in Equation 1.

$$P(\{e_i\}) = \sum_{i=1}^{N} e_i * p_i + p_{\text{idle}} \tag{1}$$

where architectural events e_i are model inputs and the p_i and p_{idle} are model parameters. In G-Power, we adopt 28 architectural events e_i ($N = 28$), covering important components of the GPU. These architectural events are listed in Table 1. These events are classified by GPU domains, including Streaming Multiprocessors (SMs), on-chip network (NoC) and L2 Cache, and global memory, where components within a domain are tightly-coupled.

Figure 2 illustrates the overview of our proposed G-Power, a methodology designed to enhance power modeling for our target GPU by leveraging additional known chip data. G-Power consists of three phases: pre-training with additional known chips, attention-inspired aggregation, and fine-tuning on our target GPU.

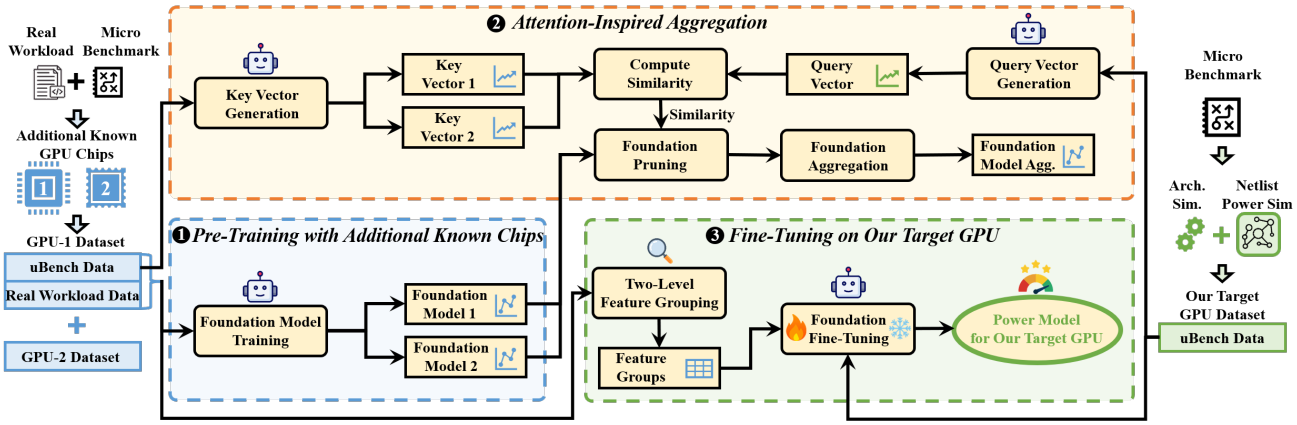


Figure 2: The three-phase framework of G-Power. 1) Pre-training with additional known chips. Multiple foundation models are trained based on datasets collected from multiple additional known GPU chips, where the dataset includes both the microbenchmark and real workload data. 2) Attention-inspired aggregation. Key and query vectors are generated based on microbenchmark data of additional known GPU chips and our target GPU, respectively. Foundations are aggregated based on similarities between the query vector and each key vector. 3) Fine-tuning on our target GPU. Features are grouped with two-level feature grouping, and the aggregated foundation is fine-tuned, where parameter ratios within each group are frozen.

1. Pre-training with additional known chips. Multiple foundation models are trained using datasets collected from a variety of additional known GPU chips. These datasets include both carefully designed microbenchmarks and real workload traces. A key advantage of this step is its ability to incorporate rich and diverse power characteristics from known GPU designs, thereby establishing a foundation that encapsulates the power behavior of additional known GPU chips. This provides additional knowledge that would otherwise be unavailable when modeling our target GPU.

2. Attention-inspired aggregation. Key and query vectors are generated from the microbenchmark data of additional known GPUs and our target GPU, respectively. The foundation models are then aggregated by evaluating the similarity between the target's query vector and each additional key vector. The strength of this approach lies in its deliberate and aligned comparison mechanism: since both the additional and target GPUs can use the same microbenchmarks to generate data, the similarity is computed in a consistent, aligned feature space. This ensures that the aggregation meaningfully captures architectural similarities, effectively transferring the most relevant knowledge to our target GPU.

3. Fine-tuning on our target GPU. In the final phase, we first group tightly coupled features via a two-level feature grouping strategy. The aggregated foundation model is then fine-tuned while keeping the parameter ratios within each feature group frozen. This design directly addresses the challenge of limited target-side data. By structurally constraining the tunable parameters, we effectively reduce the risk of overfitting, enabling stable adaptation to our target GPU. As a result, the knowledge encapsulated in pre-trained foundations is specialized to our target GPU in a robust way.

3 METHODOLOGY

We introduce our proposed three-phase algorithm in G-Power framework in this section. We describe pre-training with additional known chips in Section 3.1, attention-inspired aggregation in Section 3.2, and fine-tuning on our target GPU in Section 3.3.

3.1 Pre-Training with Additional Known Chips

This phase trains the foundation model for each additional known GPU chip. As discussed in Section 2, G-Power formulates the GPU power consumption as a linear function of architectural events. Therefore, we perform the linear regression for each chip dataset to build the foundation model for each additional known chip. Each chip dataset includes architectural events and power labels for both the microbenchmarks and real workloads, incorporating comprehensive knowledge about the GPU chip's power characteristics.

3.2 Attention-Inspired Aggregation

We propose attention-inspired aggregation to capture the similarity between our target GPU and each additional known GPU chip to utilize foundations wisely. Inspired by the attention [26], we represent each additional known chip as a key vector and our target GPU as the query vector, and then aggregate all foundation models based on similarity measured with query vector and key vector.

3.2.1 Key and Query Vector Generation. A key vector is generated for each additional known GPU chip, which is paired with its foundation model. The query vector is generated for our target GPU. Our insight is that the coefficients of the linear model reflect the characteristics of the power model. Therefore, we perform the linear regression on each additional known chip and our target GPU, and collect coefficients as key and query vectors, where the coefficients are normalized to guarantee these vectors are unit vectors.

Different from the foundation model that is trained on both the microbenchmark and real workload data, both the key and query vectors are trained on only the microbenchmark data. The reason is below: Only microbenchmark data is available to our target GPU, so the query vector can only be trained with microbenchmark data. For the key vector, it should be an equivalent counterpart to the query vector, so the key vector is also trained with only the microbenchmark data. Otherwise, the distribution difference between the microbenchmark and the real workload data will generate a biased vector that is not a counterpart to the query vector.

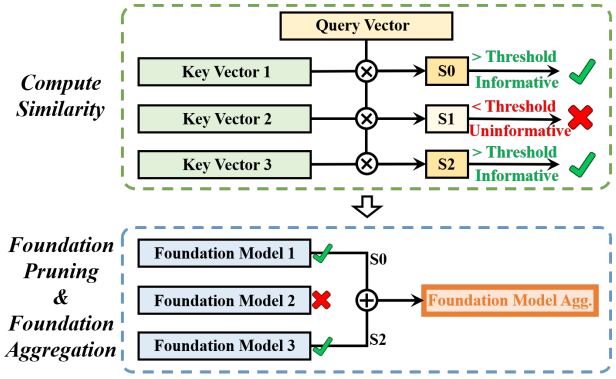


Figure 3: Illustration of attention-inspired aggregation. Each key vector is compared with the query vector to select informative foundations. All informative foundations are aggregated with similarities as weights.

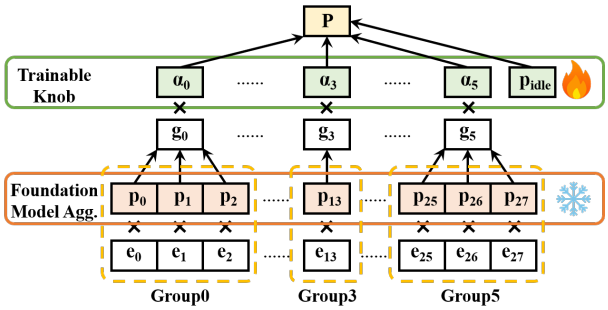


Figure 4: Illustration of fine-tuning on our target GPU. It introduces a trainable knob for each feature group. When fine-tuning, the given foundation model is frozen, and only the knobs are trained. After fine-tuning, knobs are multiplied by the foundation model to get the final GPU power model.

3.2.2 Foundation Pruning and Aggregation. With the key and query vectors generated, we aggregate all foundation models from additional known GPU chips based on similarities computed between the query vector and all key vectors. We first prune all foundation models with low similarities, and then aggregate the remaining foundation models by weighted-averaging them with similarities as weights. This enables G-Power to utilize the foundations wisely based on similarity.

The detail is illustrated in Figure 3. First, the query vector is compared with key vectors to compute similarities. The cosine similarities between the query vector of our target GPU and all key vectors of additional known GPU chips are calculated. With a predefined threshold, these foundations are classified as informative foundations [12] and uninformative foundations, based on whether the similarity is larger than the threshold or not. The informative foundation is a foundation with high similarity, indicating that this additional known GPU is similar to our target GPU, so the knowledge of its foundation is likely to be useful. Conversely, the uninformative foundation is a foundation with low similarity, where the additional known GPU is different from our target GPU, and the knowledge may not be useful for our target GPU.

Second, foundation pruning and foundation aggregation are performed with the identified informative foundations and calculated similarities to generate the final aggregated foundation model. The uninformative foundations are pruned because the knowledge of these may not be useful. Then, all informative foundations are weight-averaged to generate the final aggregated foundation model, where the weights are similarities normalized with softmax, similar to the operation in the attention mechanism.

3.3 Fine-Tuning on Our Target GPU

After the aggregated foundation model is generated, this foundation model is fine-tuned based on the dataset of our target GPU to build the final GPU power model for our target GPU design. The fine-tuning is based on the feature-grouping, where architectural events within the same group are tightly coupled from both the microarchitectural aspect and the data distribution aspect.

3.3.1 Foundation Fine-Tuning. We fine-tune the aggregated foundation model on our target GPU based on the feature grouping, which will be introduced in Section 3.3.2. The parameters within the same group should be updated in a coordinated approach because these are tightly coupled. Therefore, we freeze the ratio among parameters within each group during the foundation fine-tuning.

Figure 4 illustrates the details of the foundation fine-tuning. Compared with the original linear model that multiplies e_i and p_i and calculates the sum with an intercept, foundation fine-tuning introduces a trainable knob for each feature group, denoted as α_j in Figure 4. To be specific, $e_i * p_i$ within each feature group is summed up as g_j, and the g_j is then multiplied by α_j and finally summed up with the intercept p_{idle} to get the final power. During fine-tuning, the foundation model is frozen and only the trainable knob can be updated. In our implementation, we first calculate g_j of each feature group and form a vector of g_j. Then, this new vector is adopted as the feature of the linear regression, where the trained coefficients are α_j and the intercept is p_{idle}. After the fine-tuning, these trainable knobs are multiplied by the original foundation model to get the final GPU power model.

3.3.2 Two-Level Feature Grouping. The two-level feature grouping is performed to partition features into multiple groups that assist the parameter freezing in design fine-tuning. We propose a two-level feature grouping with both microarchitecture-aware grouping and data-driven grouping.

The microarchitecture-aware grouping partitions features at a coarse granularity, considering the microarchitecture coupling among features. We partition these features, which are architectural events, into three groups based on the domain. As listed in Table 1, the 28 architectural events belong to three domains, including SM array, NoC & L2Cache, and global memory. The insight is that the microarchitecture within a domain is tightly coupled, so the architectural events are also coupled and then considered to have similar variation trends across GPUs.

The data-driven grouping further partitions 20 features within SM array domain into a fine granularity. We regard a feature in all samples as the vector of this feature and normalize this vector to a unit vector. Then, the hierarchical clustering, which is feasible for high-dimensional data with few samples, is performed to partition features into multiple groups. The insight is that data-driven grouping identifies features with a similar variation trend across programs and GPUs, regardless of absolute value of each event.

GPU	Tech Node	SM Frequency	Memory Frequency
NVIDIA RTX A6000	8nm	1410MHz	2000MHz
NVIDIA RTX 3090	8nm	1395MHz	1219MHz
NVIDIA RTX 4090	5nm	2235MHz	1313MHz
NVIDIA RTX 5880 Ada	5nm	975MHz	2250MHz

Table 2: GPU designs adopted in our evaluation.

Kernel Name	Source Benchmark	Kernel Name	Source Benchmark
CUDA Samples 11.0			
cTensor_K1	cudaTensorCoreGemm	dct_K1	dct8x8
binOpt_K1	binomialOptions	dct_K2	dct8x8
walsh_K1	fastWalshTransform	histo_K1	histogram
walsh_K2	fastWalshTransform	msort_K1	mergesort
qrng_K1	quasirandomGenerator	msort_K2	mergesort
qrng_K2	quasirandomGenerator	sobol_K1	SobolQRNG
Rodinia 3.1			
bprop_K1	backprop	hsopt_K1	hotspot
bprop_K2	backprop	sradv1_K1	sradv1
btree_K1	b+tree	kmeans_K1	kmeans
btree_K2	b+tree		
CUTLASS 1.3 (cutlass-wmma)		Parboil	
cutlass_K1	input: 2560x16x2560	sgemm_K1	sgemm
cutlass_K2	input: 4096x128x4096	mriq_K1	mri-q
cutlass_K3	input: 2560x512x2560	sad_K1	sad

Table 3: Real GPU workloads adopted in our evaluation.

4 EXPERIMENTAL RESULTS

4.1 Experiment Setup

We adopt four modern NVIDIA GPUs with Ampere and Ada Lovelace architecture: RTX A6000 [18], RTX 3090 [17], RTX 4090 [20], and RTX 5880 Ada [21], as listed in Table 2. The 65 microbenchmarks are from AccelWattch [7]. The real workloads include CUDA Samples 11.0 [16], Rodinia 3.1 [4], CUTLASS 1.3 [14], and Parboil [23]. We perform kernel-level evaluation, with multi-kernel benchmarks broken into isolated kernels (Table 3).

To collect architectural events, we utilize the NVIDIA Nsight Compute [19]. It measures the real execution statistics when running kernels on the real GPU chips by using the hardware counters. Such an accurate architectural event measurement eliminates the interference of the inaccuracy of the architectural performance simulator, ensuring the evaluation focuses on the power aspect. The architectural events adopted in our experiment are listed in Table 1, including 28 events covering different components.

To collect power labels, we measure the power from the real GPU chips. In our measurement, we utilize NVIDIA Management Library (NVML) [15] and NVIDIA System Management Interface (nvidia-smi) [13]. To be specific, we use nvidia-smi to lock the frequency of the SM and memory before power measurement, and then execute an NVML-based monitor program to sample the power consumption during the kernel execution. Similar to the AccelWattch, we heat the GPU to 65°C with power-hungry programs before real power measurement. We perform 5 times for each kernel and use the average value as the final power label in our evaluation.

4.2 Summary of Baseline Methods

We adopt two representative existing works as our baselines for comparison. 1) AccelWattch [7] is the state-of-the-art architecture-level GPU power model, which is the updated version of GPUWattch [10]. This is built as an analytical power model with power calculation for each component, similar to the McPAT [11]. A calibration process that adopts a linear function is required before the prediction, training the model on the GPU microbenchmarks of the new target GPU. 2) McPAT-Calib [29] is a representative architecture-level CPU power model that we generalized to the GPU. It builds a machine learning model for power modeling. In our evaluation, the McPAT-Calib is trained on the GPU microbenchmarks of the new target GPU. The machine learning model adopted is XGBoost [5], the best model claimed in McPAT-Calib.

Other recent architecture-level CPU power models, such as PANDA [32], FirePower [30], and AutoPower [35], are not included in our evaluation because these works are customized to CPU and can not be generalized to GPU power modeling.

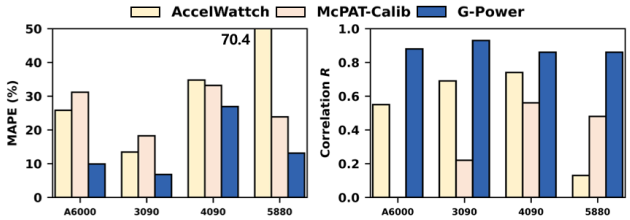


Figure 5: Accuracy comparison between G-Power and two baselines for four testing scenarios.

4.3 Training and Testing Data Setup

To reflect the real scenario, we adopt one GPU as our target and the remaining three as additional known chips, yielding four testing scenarios (A6000, 3090, 4090, and 5880). The three additional GPUs provide pre-training data with both microbenchmarks and real workloads. The target GPU uses microbenchmarks for fine-tuning and real workloads for testing. Baseline adopts the selected GPU using GPU microbenchmarks for training and real GPU workloads for testing for fair comparison.

4.4 Power Modeling Accuracy

Figure 5 demonstrates the accuracy comparison between our proposed G-Power and our two baselines in four testing scenarios. The accuracy is measured in MAPE and correlation R. It shows that our proposed G-Power can significantly outperform all baselines in all testing scenarios. Quantitatively, G-Power can achieve a low MAPE of 14% and a high correlation R of 0.88 on average. This MAPE is 22% and 13% lower than AccelWattch and McPAT-Calib, respectively. The correlation R is 0.36 and 0.57 higher than our two baselines. The superior performance of G-Power over all our baselines demonstrates the effectiveness of utilizing knowledge from additional known GPU chips.

For AccelWattch, the state-of-the-art architecture-level GPU power model, the prediction on our evaluated four modern GPU designs is inaccurate. The inaccuracy of AccelWattch is due to the limited training data and the data distribution difference between GPU microbenchmarks for training and real GPU workloads for testing. With the foundation that integrates knowledge from multiple additional known GPU chips with comprehensive programs measured, our proposed G-Power can address this data limitation, significantly improving the prediction accuracy.

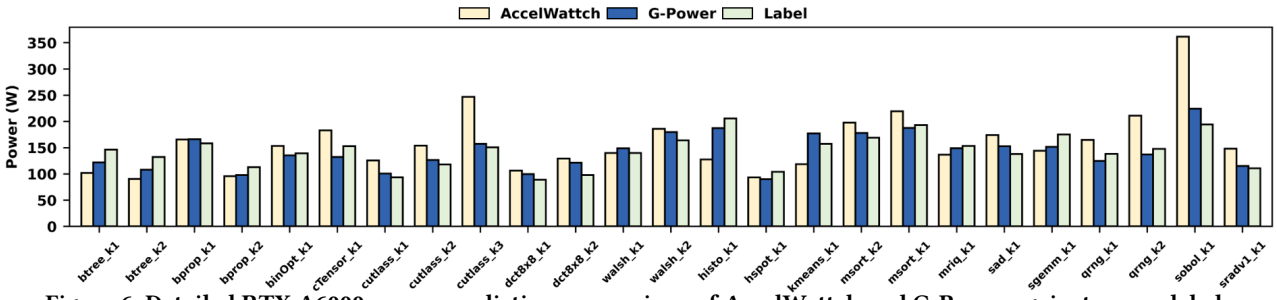
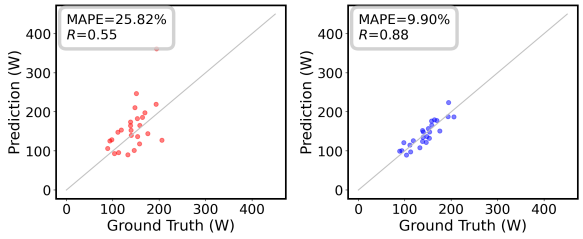
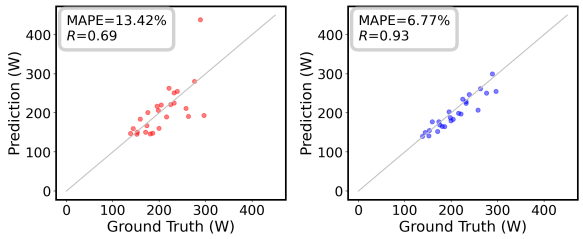


Figure 6: Detailed RTX A6000 power prediction comparison of AccelWatch and G-Power against power labels.



(a) A6000-AccelWatch

(b) A6000-G-Power



(c) 3090-AccelWatch

(d) 3090-G-Power

Figure 7: The power prediction visualization for A6000 and 3090 with the AccelWatch and G-Power.

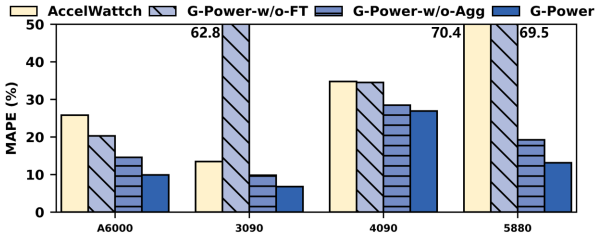


Figure 8: Ablation studies of G-Power.

Figure 6 shows the detailed power prediction comparison of AccelWatch and G-Power against the power labels for RTX A6000. It demonstrates that G-Power can achieve consistently higher accuracy over AccelWatch for all our evaluated real GPU workloads. Figure 7 visualizes the power prediction for RTX A6000 and 3090 with both the AccelWatch and G-Power. Each point in the figure represents the prediction for a real GPU workload, and the grey line indicates the accurate prediction. This visualization shows that the prediction of G-Power is close to the grey line, while the AccelWatch prediction is far from this line, indicating an advantage of our proposed G-Power. It also demonstrates that the correlation of AccelWatch prediction is dramatically low, which reflects that the power characteristics of different workloads can not be accurately differentiated. In comparison, G-Power achieves a high correlation, correctly identifying power behaviors of different workloads.

For McPAT-Calib, the ML-based power model, its prediction is also inaccurate, especially for the correlation R. We observe that the algorithm fits well on the training dataset with GPU microbenchmarks. This indicates that McPAT-Calib has an overfitting problem. The overfitting makes McPAT-Calib can make predictions within a reasonable range similar to the training data, but the relative value is significantly inaccurate, reflected with low correlation R.

4.5 Ablation Studies

Besides the two baselines discussed above, we also include two methods derived from G-Power for ablation studies: 1) G-Power-w/o-FT. The fine-tuning is not performed, directly adopting the aggregated foundation model as the final power model. 2) G-Power-w/o-Agg. Attention-inspired aggregation is not performed, directly selecting one foundation model for fine-tuning.

Figure 8 shows the results of our ablation studies, demonstrating the criticality of both fine-tuning and attention-inspired aggregation. The comparison between G-Power-w/o-FT and G-Power indicates that the fine-tuning is critical for G-Power. It is because although different GPU designs have similarities in the power behaviors, the resource amount is different, and there are also differences in microarchitecture design. The comparison between G-Power-w/o-Agg and G-Power shows that directly selecting one foundation model can only achieve sub-optimal accuracy. It is because some GPUs are divergent from our target GPUs, and attention-inspired aggregation can capture the similarity to utilize foundations wisely.

5 CONCLUSION

We propose G-Power, an architecture-level GPU power modeling framework that utilizes knowledge from additional known GPU chips. G-Power adopts a three-phase algorithm to wisely provide foundations with additional known GPU chips and capture the similarity to utilize these foundations for fine-tuning. This foundation-based framework boosts the GPU power modeling accuracy, which is a compelling addition to the GPU architects' toolbox.

ACKNOWLEDGEMENT

This work is supported by National Natural Science Foundation of China (NSFC) 62304192, Hong Kong Research Grants Council (RGC) CRF-YCRG C6003-24Y. It was partially conducted by ACCESS – AI Chip Center for Emerging Smart Systems, supported by the InnoHK initiative of the Innovation and Technology Commission of the Hong Kong Special Administrative Region Government.

REFERENCES

[1] 2021. Design Compiler® RTL Synthesis. https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/design-compiler-nxt.html.

[2] 2021. VCS® functional verification solution. https://www.synopsys.com/verification/simulation/vcs.html.

[3] David Brooks, Vivek Tiwari, and Margaret Martonosi. 2000. Wattch: A framework for architectural-level power analysis and optimizations. *ACM SIGARCH Computer Architecture News* 28, 2 (2000), 83–94.

[4] Shuai Che, Michael Boyer, Jiayuan Meng, David Tarjan, Jeremy W Sheaffer, Sang-Ha Lee, and Kevin Skadron. 2009. Rodinia: A benchmark suite for heterogeneous computing. In *2009 IEEE international symposium on workload characterization (IISWC)*. Ieee, 44–54.

[5] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[6] Forbes. 2019. NVIDIA Dominates The Market For Cloud AI Accelerators More Than You Think. https://www.forbes.com/sites/paulteich/2019/06/17/nvidia-dominates-the-market-for-cloud-ai-accelerators-more-than-you-think/#676dea375edb

[7] Vijay Kandiah, Scott Peverelle, Mahmoud Khairy, Junrui Pan, Amogh Manjunath, Timothy G Rogers, Tor M Aamodt, and Nikos Hardavellas. 2021. AccelWattch: A power modeling framework for modern GPUs. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*. 738–753.

[8] Donggyu Kim, Jerry Zhao, Jonathan Bachrach, and Krste Asanović. 2019. Simmani: Runtime power modeling for arbitrary RTL with automatic signal selection. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. 1050–1062.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012).

[10] Jingwen Leng, Tayler Hetherington, Ahmed ElTantawy, Syed Gilani, Nam Sung Kim, Tor M Aamodt, and Vijay Janapa Reddi. 2013. GPUWattch: Enabling energy optimizations in GPGPUs. *ACM SIGARCH computer architecture news* 41, 3 (2013), 487–498.

[11] Sheng Li, Jung Ho Ahn, Richard D Strong, Jay B Brockman, Dean M Tullsen, and Norman P Jouppi. 2009. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proceedings of the 42nd annual ieee/acm international symposium on microarchitecture (MICRO)*. 469–480.

[12] Sai Li, T Tony Cai, and Hongzhe Li. 2022. Transfer learning for high-dimensional linear regression: Prediction, estimation and minimax optimality. *Journal of the Royal Statistical Society Series B: Statistical Methodology* 84, 1 (2022), 149–173.

[13] NVIDIA. 2016. nvidia-smi - NVIDIA System Management Interface program. http://developer.download.nvidia.com/compute/DCGM/docs/nvidia-smi-367.38.pdf

[14] NVIDIA. 2019. CUTLASS: CUDA template library for dense linear algebra at all levels and scales. https://github.com/NVIDIA/cutlass

[15] NVIDIA. 2019. NVML API Reference. https://docs.nvidia.com/deploy/nvml-api/nvml-api-reference.html

[16] NVIDIA. 2020. CUDA Samples. https://docs.nvidia.com/cuda/archive/11.0/cuda-samples/index.html

[17] NVIDIA. 2020. GeForce RTX 3090 Family. https://www.nvidia.com/en-us/geforce/graphics-cards/30-series/rtx-3090-3090ti/

[18] NVIDIA. 2020. NVIDIA RTX A6000 Graphics Card. https://www.nvidia.com/en-us/products/workstations/rtx-a6000/

[19] NVIDIA. 2021. Nsight Compute. https://docs.nvidia.com/nsight-compute/NsightCompute/index.html

[20] NVIDIA. 2022. GeForce RTX 4090. https://www.nvidia.com/en-us/geforce/graphics-cards/40-series/rtx-4090/

[21] NVIDIA. 2024. NVIDIA RTX 5880 Ada Generation Graphics Card. https://www.nvidia.com/en-us/products/workstations/rtx-5880/

[22] Addison Snell and Laura Segervall. 2017. HPC application support for GPU computing. https://www.nvidia.com/content/intersect-360-HPC-application-support.pdf

[23] John A Stratton, Christopher Rodrigues, I-Jui Sung, Nady Obeid, Li-Wen Chang, Nasser Anssari, Geng Daniel Liu, and Wen-mei W Hwu. 2012. Parboil: A revised benchmark suite for scientific and commercial throughput computing. *Center for Reliable and High-Performance Computing* 127, 7.2 (2012).

[24] Synopsys. 2023. PrimePower: RTL to Signoff Power Analysis. https://www.synopsys.com/implementation-and-signoff/signoff/primepower.html

[25] TOP500.org. 2021. TOP500 List. https://www.top500.org/lists/top500/2021/06/

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[27] Zhiyao Xie, Shiyu Li, Mingyuan Ma, Chen-Chia Chang, Jingyu Pan, Yiran Chen, and Jiang Hu. 2022. DEEP: Developing extremely efficient runtime on-chip power meters. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 1–9.

[28] Zhiyao Xie, Xiaoqing Xu, Matt Walker, Joshua Knebel, Kumaraguru Palaniswamy, Nicolas Hebert, Jiang Hu, Huanrui Yang, Yiran Chen, and Shidhartha Das. 2021. APOLLO: An automated power modeling framework for runtime power introspection in high-volume commercial microprocessors. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*. 1–14.

[29] Jianwang Zhai, Chen Bai, Binwu Zhu, Yici Cai, Qiang Zhou, and Bei Yu. 2022. McPAT-Calib: A RISC-V BOOM microarchitecture power modeling framework. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* 42, 1 (2022), 243–256.

[30] Qijun Zhang, Mengming Li, Yao Lu, and Zhiyao Xie. 2025. Firepower: Towards a foundation with generalizable knowledge for architecture-level power modeling. In *Proceedings of the 30th Asia and South Pacific Design Automation Conference*. 1145–1152.

[31] Qijun Zhang, Mengming Li, Andrea Mondelli, and Zhiyao Xie. 2024. An Architecture-Level CPU Modeling Framework for Power and Other Design Qualities. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 44, 7 (2024), 2751–2764.

[32] Qijun Zhang, Shiyu Li, Guanglei Zhou, Jingyu Pan, Chen-Chia Chang, Yiran Chen, and Zhiyao Xie. 2023. PANDA: Architecture-level power evaluation by unifying analytical and machine learning solutions. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 01–09.

[33] Qijun Zhang, Shang Liu, Yao Lu, Mengming Li, and Zhiyao Xie. 2026. ReadyPower: A Reliable, Interpretable, and Handy Architectural Power Model Based on Analytical Framework. In *2026 31st Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 1223–1229.

[34] Qijun Zhang, Yao Lu, Mengming Li, Shang Liu, and Zhiyao Xie. [n. d.]. ArchPower: Dataset for Architecture-Level Power Modeling of Modern CPU Design. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[35] Qijun Zhang, Yao Lu, Mengming Li, and Zhiyao Xie. 2025. Autopower: Automated few-shot architecture-level power modeling by power group decoupling. In *2025 62nd ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–7.

[36] Yuan Zhou, Haoxing Ren, Yanqing Zhang, Ben Keller, Brucek Khailany, and Zhiru Zhang. 2019. PRIMAL: Power inference using machine learning. In *DAC*.