Nona: Accurate Power Prediction Model Using Neural Networks

HoSun Choi, Chanho Park, Euijun Kim, and William J. Song
School of Electrical and Electronic Engineering, Yonsei University, Seoul, South Korea
{hosunhc,ch.park,cwchris,wjhsong}@yonsei.ac.kr

ABSTRACT

This paper proposes a neural-network-based power model, *Nona*, that accurately predicts the power consumption of heterogeneous CPUs on a commercial mobile device. With aggressive on-device power management in action, it becomes increasingly challenging to make accurate power predictions for diverse applications. To overcome the limitations of the existing power models based on linear regression, Nona uses a lightweight neural network with a small number of performance monitoring counters (PMCs) chosen from a system analysis and a loss function designed for power prediction. Experiments on Google Pixel 6 show that Nona has a 3.4% average prediction error, improving on prior work by 2.6x.

CCS CONCEPTS

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

KEYWORDS

Neural network, power, prediction, hardware measurement

1 INTRODUCTION

For diverse application behaviors and aggressive on-device power management, accurate power prediction becomes increasingly more challenging. Past work has presented several linear regression models based on hardware performance monitoring counters (PMCs). However, they showed inaccurate results for assuming linear correlations between processor power and PMC values [11, 13], which actually have nonlinear relations under on-device power management such as voltage and frequency scaling, throttling, etc. To solve the nonlinearity of power prediction, Lin et al. [9] performed the case studies of using neural networks (NNs) for power estimation. They used a large collection of software profiling data yet struggled with inaccurate results due to unoptimized NN models and using software metrics that are known to be poor indicators of power behaviors [5]. As such, the existing models did not provide effective solutions for the accurate power prediction of various applications on real hardware under active on-device power management.

This paper proposes an NN-based power model named *Nona* that accurately predicts the runtime power of heterogeneous CPUs on a commercial mobile device. The proposed NN model takes a set of PMCs, operating voltage and frequency, and temperature as input features and predicts the power consumption of heterogeneous CPUs. Nona resolves two practical challenges of on-device runtime power prediction. First, commercial devices offer many measurable PMCs but allow monitoring only a small number of counters at a time because of hardware resource limitations. Second, on-device power management frequently intervenes and pollutes power samples, which affects the distribution of NN training datasets.

Nona resolves the problems via a novel PMC selection strategy and systematic loss function. The selection strategy picks the PMCs that cover different logic activities in a processor and show the highest correlations with power behaviors. The systematic loss function guides Nona to learn the power behaviors without bias to the distribution of collected power data. With the nonlinear PMC selection strategy and the systematic loss function design, Nona implements a lightweight NN that achieves high prediction accuracy for a variety of real mobile applications. Experiments on Google Pixel 6 demonstrate that Nona predicts the runtime power of heterogeneous CPUs with only a 3.4% average error, which is 2.6x more accurate than the previous power model.

2 BACKGROUND

2.1 Performance Monitoring Unit

A performance monitoring unit (PMU) in a processor monitors runtime performance data, such as the number of instructions and cache accesses, through dedicated hardware registers, i.e., PMCs. Embedded in the processor, the PMU can collect low-level hardware statistics with minimal overhead [4]. The PMU has a long list of measurable PMCs but constrains the number of counters that can be collected at a time because of hardware resource limitations.

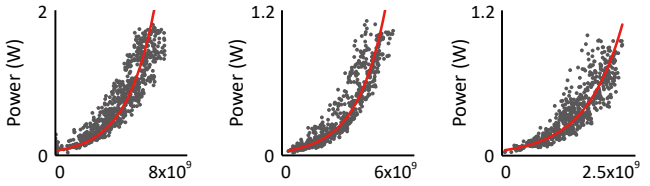
A Google Pixel 6 device used in our experiments is based on ARM DynamIQ big.LITTLE technology that consists of three CPU clusters: i) two high-performance Cortex-X1 cores, ii) two performance-efficient Cortex-A76 big cores, and iii) four energy-efficient Cortex-A55 LITTLE cores. We refer to these three CPU types as *big*, *middle*, and *little* clusters in the remainder of this paper. Each of these clusters has 118, 107, and 111 PMCs, respectively, but only six PMC values can be measured simultaneously per core [1–3].

2.2 Neural Network Design

The strength of a neural network lies in the ability to make predictions without expert knowledge. To improve prediction accuracy, the NN must be properly designed by tuning the network architecture, hyperparameters, and loss functions. The first two factors can be resolved, though time-consuming, by searching for an optimal NN structure, including the number of layers, learning rate, batch size, etc. On the other hand, devising a loss function $\mathcal{L}$ requires an in-depth understanding of the target model. The loss function quantifies how well the model learns and guides the training process. For example, classification problems generally use cross-entropy functions, and regression problems use mean squared errors (MSEs).

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06.
https://doi.org/10.1145/3649329.3655668



(a) Big-core cluster (b) Middle-core cluster (c) Little-core cluster

Figure 1: Power measurement data shows nonlinear relations to PMC values (i.e., # retired instructions) for all CPU clusters.

Other problems use customized loss functions [10]. In this paper, Nona presents a novel loss function for accurate power prediction.

3 RELATED WORK

Related literature of processor power prediction based on hardware PMCs can be categorized into two approaches: linear regression [11, 13] and nonlinear models [9, 12]. Walker et al. [13] proposed a linear-regression-based power model of heterogeneous CPUs. They introduced a PMC selection method, assuming that processor power and PMCs are linearly correlated. However, the linear prediction model produces inaccurate results, and the PMC selection method chooses non-optimal counters, aggravating the prediction accuracy. Oboril et al. [11] presented a linear regression model of McPAT [8] and calibrated it using hardware-measured power and PMC values. However, the correct use of McPAT requires detailed architecture and circuit information, which is subject to large errors [14].

To solve the nonlinearity problems, Lin et al. [9] performed the case studies of NNs for power estimation. They compared the accuracy of a few existing NN models without any new features. We demonstrate in our experiments that such a naive use of NNs leads to large prediction errors. Sagi et al. [12] applied nonlinear transformations to architectural stats in a simulation environment and compared power estimations to simulation outputs. However, we find that the methodology does not work well on real hardware and produces highly erroneous results.

4 MOTIVATION

Most of the previous power models were established based on the assumption that processor power and PMCs are linearly correlated or can be modeled via linear regression [11, 13]. To prove that the relations are actually nonlinear, Fig. 1 plots the distribution of measured power samples and PMC counter values (i.e., `INST_RETIRED`) for each CPU cluster on Google Pixel 6 while executing `stress-ng` benchmarks [7] under active on-device power management. The graphs clearly display nonlinear relations between the CPU cluster power and the PMC. Other PMCs also exhibit similar trends as `INST_RETIRED` (i.e., # retired instructions). Therefore, a linear estimation is potentially subject to a large error depending on how a linear regression line is drawn and where an operating point is. To overcome such limitations, this paper proposes an NN-based model that can accurately predict the runtime power of heterogeneous CPUs for a variety of mobile applications via a novel PMC selection strategy and systematic loss function.

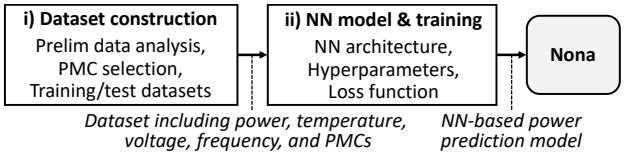


Figure 2: Overall procedure of Nona to build an NN-based power prediction model. i) Prelim datasets are analyzed to select PMCs, and training/test datasets are collected for the selected PMCs. ii) The NN model is designed and trained with a systematic loss function.

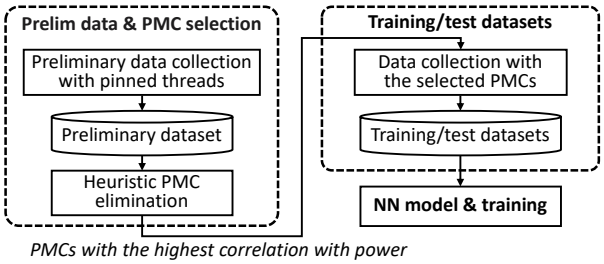


Figure 3: The procedure of dataset construction. A preliminary dataset is created by collecting all PMCs. Then, heuristic feature elimination identifies the most relevant PMCs for power prediction. Training/test datasets are collected with the selected PMCs.

5 METHODOLOGY

5.1 Overall Procedure

The overall procedure of developing Nona consists of two major steps, i) dataset collection and ii) NN model and training, as shown in Fig. 2. In the first step, *preliminary dataset* is collected using a few test workloads based on hardware measurement in a controlled environment (i.e., fixed frequency, pinned CPU cores). The measurement is repeated to collect all PMCs for all test workloads while sweeping the operating frequency. Then, the collected preliminary dataset is used to eliminate similar PMCs and the ones showing low correlations with power data. The *PMC selection* process finds the most relevant PMCs to power behaviors, which can be measured simultaneously by the PMU. Lastly, the selected PMCs are collected for many mobile applications to create *NN training and test datasets*.

The second step of Fig. 2 determines an *NN architecture* and its *hyperparameters* based on the training dataset collected in step 1. Importantly, Nona presents a novel *systematic loss function* for optimal training and improved prediction accuracy. The remainder of this section explains the detailed procedure of these steps.

5.2 Dataset Construction

The objective of the dataset construction step is to generate a high-quality training dataset for an NN model through *preliminary data collection* and *heuristic PMC elimination*. The overall process of dataset construction is outlined in Fig. 3.

5.2.1 Preliminary Data Collection. As each PMC covers a specific logic activity in a core, an ideal case would be to access all existing PMCs simultaneously, but it is not possible due to hardware resource limitations explained in Section 2.1. The logic activities of different PMUs are not completely independent of each other. Therefore, a preliminary dataset is collected for all PMCs through

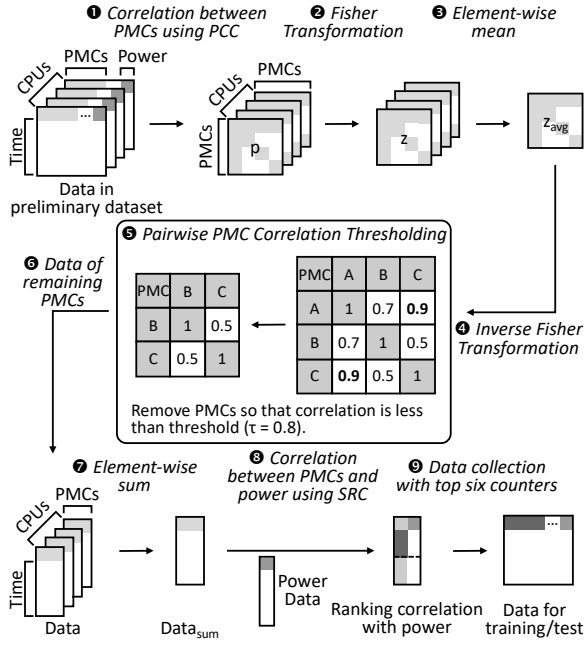


Figure 4: The process of heuristic PMC elimination that removes similar PMCs and the ones showing low correlations with power. This PMC selection process finds the most relevant PMCs to power behaviors, which can be measured simultaneously by the PMU.

repeated measurements, and then the most relevant PMCs showing the highest correlations with processor power are identified based on the preliminary dataset via a PMC selection strategy.

On a Google Pixel 6 device, PMCs and CPU cluster power can be measured using *Google Perfetto* and *Android Simpleperf*. Other hardware data, such as voltage, frequency, and temperature, can be probed by reading system files. We used the test workloads that can be pinned to cores (i.e., `Dhrystone`, `LMBench`, `stress-ng`, and `Sysbench`) to prevent thread migration. Since the PMU can monitor only six PMCs per core at a time, the preliminary dataset is collected by splitting the PMC list into groups of six PMCs to measure them all. To ensure consistent data measurements, each of these groups runs the same workload multiple times, and the measured data traces are averaged per time sample. Once the datasets of all groups measuring mutually exclusive PMCs are collected, they are concatenated to create a single dataset. This process is repeated for all test workloads to collect the whole preliminary dataset, which is then used for heuristic feature elimination for PMC selection.

5.2.2 Heuristic PMC Elimination. Fig. 4 shows a heuristic feature elimination process to find PMCs that are independent of each other and have the highest correlations with power. The elimination process is performed separately for each CPU cluster. The example in the figure shows the case of a four-core cluster similar to a little-core cluster. In the beginning, PMCs of zero variances are removed since they do not affect the output. ❶ For each core, a Pearson correlation coefficient (PCC) matrix is calculated to evaluate correlations between PMCs. ❷ Each element of PCC matrices is processed through Fisher transformation in Eq. (1), which approximates skewed PCCs to normal distributions.

Algorithm 1 Pairwise PMC Correlation Thresholding

```
Parameters: pcc_matrix: PMC PCC matrix [# of pmc, # of pmc]
                    p: PCC in pcc_matrix
                    τ: correlation max threshold
   /* Absolute values of PCC matrix */
1: pcc_matrix ← |pcc_matrix|
   /* Elimination of correlated PMCs */
2: while any p in pcc_matrix > τ; do
       /* Find a PMC pair with maximum PCC in the matrix. */
3:     (pmc_X, pmc_Y) ← argmax(pcc_matrix)
       /* Correlation of selected PMCs with unselected PMCs */
4:     vec_corr_pmc_X = pmc_X.corr(vec_pmcs_not_Y)
5:     vec_corr_pmc_Y = pmc_Y.corr(vec_pmcs_not_X)
       /* Arithmetic mean of correlations */
6:     avg_pmc_X = mean(vec_corr_pmc_X)
7:     avg_pmc_Y = mean(vec_corr_pmc_Y)
       /* Remove the PMC with greater average correlation. */
8:     if avg_pmc_X > avg_pmc_Y then
9:        remove(pmc_X)
10:    else
11:       remove(pmc_Y)
12:    end if
13: end while
```

After normalizing the distributions, ❸ PCCs are averaged element-wise to collectively evaluate the PMC correlations across all cores. ❹ Then, inverse Fisher transformation is applied based on Eq. (2), which returns a single average PCC matrix. ❺ The average matrix is processed through a pairwise correlation thresholding algorithm [6] that iteratively eliminates PMCs until all correlation pairs become smaller than a threshold.

$$z = \frac{1}{2}\ln\left(\frac{1+p}{1-p}\right) \tag{1}$$

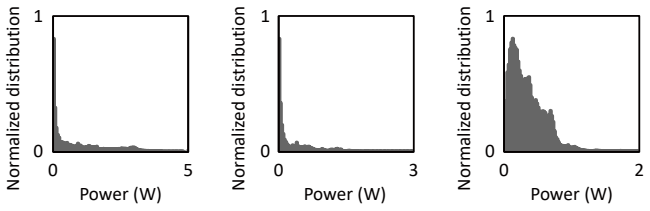
$$p = \frac{e^{2z}-1}{e^{2z}+1} \tag{2}$$

Algorithm 1 shows how PMCs are eliminated in the pairwise PMC correlation thresholding stage in Fig. 4. It first takes the absolute values of the averaged PCC matrix (line 1). The while loop checks if any PCC matrix value is greater than the threshold $\tau = 0.8$ (line 2). In the loop body, it finds a pair of PMCs with the largest PCC value (line 3). For each of the selected PMC pair (i.e., `pmc_X`, `pmc_Y`), the average of PCCs is calculated for all PMCs except for the two (lines 4-7). Then, the PMC with a greater PCC average is eliminated from the matrix since the removed PMC's logic activity is covered by other surviving PMCs (lines 8-12). The while loop is repeated until there are no elements greater than the threshold.

After eliminating all PMCs of large PCCs, the reduced set of PMCs has small correlations between them. ❻ The original preliminary dataset used in stage ❶ is reformed by removing the eliminated PMC's data. ❼ Then, the remaining PMC data are added element-wise across all cores to evaluate their impact on the cluster. ❽ We used a Spearman's rank correlation coefficient (SRC) to calculate a nonlinear correlation between each PMC and cluster power using Eq. (3). In the equation, $\mathrm{cov}(r_{\mathrm{PMC}}, r_{\mathrm{power}})$ is the covariance of r_{PMC} and r_{power} distributions, and r_{PMC} and r_{power} are the rank of a PMC and power sample in the distributions, respectively. $\sigma_{r_{\mathrm{PMC}}}$ and $\sigma_{r_{\mathrm{power}}}$ are the standard deviations of them.

$$\rho(r_{\mathrm{PMC}}, r_{\mathrm{power}}) = \frac{\mathrm{cov}(r_{\mathrm{PMC}}, r_{\mathrm{power}})}{\sigma_{r_{\mathrm{PMC}}}\sigma_{r_{\mathrm{power}}}} \tag{3}$$

❾ Lastly, PMCs are sorted by their SRC values, and the top six PMCs are chosen. For the selected PMCs, hardware measurement



(a) Big-core cluster (b) Middle-core cluster (c) Little-core cluster

Figure 5: CPU power data are biased to low-power states due to on-device power management. Such long-tail distributions cause NN training to be biased to the power numbers of large sample counts.

Table 1: Workloads for NN Training

Workloads	Components	Max Power (W)
AnTuTu Benchmark	CPU, Memory, I/O, Mixed	3.93
DeepBench	CPU	2.91
Dhrystone	CPU	2.25
LMBench	Memory	1.31
Geekbench	CPU, Memory, Mixed	3.78
PCMark	Mixed	2.61
stress-ng	CPU, Memory, File, Mixed	4.67
Sysbench	CPU, Memory, File	3.24

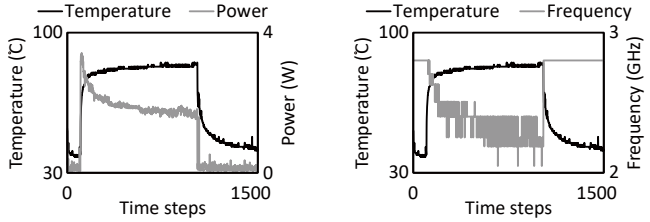
data are collected for a variety of mobile applications to create NN training and test datasets. Unlike the measurement for the preliminary dataset, workloads are not pinned to CPU cores, and their executions are governed by on-device power management.

5.3 NN Model and Training

5.3.1 NN Architecture & Hyperparameters. Nona uses a lightweight multi-layer perceptron (MLP) network made of one input layer, two hidden layers, and one output layer. All layers use ReLU activation. The input layer has 62 nodes, consisting of voltage and temperature per CPU cluster (i.e., 2 counters/cluster × 3 clusters) and frequency and six PMCs per core (i.e., 7 counters/core × 8 cores). The output layer has three nodes that represent the predicted power number of three CPU clusters. Input features are scaled via Z-score normalization to prevent bias, and output features are denormalized to generate an absolute power number.

5.3.2 NN Training Challenge. During the NN training phase, two major hurdles are raised: i) underestimation of high-power states and ii) negative-number power predictions. We observed that the NN model struggled with the prediction of high-power workloads due to a small number of high-power data in the training dataset, known as a long-tail distribution problem [15]. The training dataset comprises various workloads with diverse characteristics as listed in Table 1, but the collected power samples of CPU clusters are biased to low-power values as shown in Fig. 5.

Obtaining high-power samples is difficult, especially for big and middle-core clusters, since thermal throttling lowers their operating frequencies on entering high-power states. Fig. 6 displays a thermal throttling case in the big-core cluster. It shows that the cluster power starts to decrease as its temperature rises (Fig. 6a) because active on-device power management reduces the operating



(a) Power variation for thermal (b) Frequency variation for thermal

Figure 6: Thermal, power, and frequency of the big-core cluster while running the xOPS benchmark. Thermal throttling disturbs the collection of high-power data samples.

frequency (Fig. 6b). Since the MSE loss function calculates a loss by averaging the squared error of all sample points, it becomes biased toward large sample counts (i.e., low-power states) during NN training. In addition, MSE training can cause the NN model to predict negative power numbers. Since MSE is an arithmetic operation to minimize residuals, it does not prevent the negative-number predictions from occurring. Therefore, the NN model requires a systematic loss function that can guide optimal learning for imbalanced data and compensate for out-of-range power predictions.

5.3.3 Systematic Loss Function. Nona designs a novel loss function shown in Eq. (4). It has two terms, $\mathcal{L}_{\text{exp,i}}$ and $\mathcal{L}_{\text{neg,i}}$, and N is the number of samples. The first term in Eq. (5) has two hyperparameters, λ_{exp} and α_{exp}, and applies an exponential coefficient to MSE to increase the loss based on the magnitude of a power value. It multiplies the MSE of target power y_i and predicted power $\hat{y}_i$ with λ_{exp} and $e^{(\alpha_{\text{exp}} y_i)}$, which increases a penalty for underrepresented high-power samples. The second term in Eq. (6) has one hyperparameter, λ_{neg}, and penalizes negative power predictions. It takes the maximum of 0 and the negative of the predicted power $\hat{y}_i$ and multiplies it with λ_{neg}. For instance, if the NN model predicts power as -0.1, the max function penalizes it with a result of 0.1.

$$\mathcal{L}_{\text{sys}} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathcal{L}_{\text{exp,i}} + \mathcal{L}_{\text{neg,i}}\right) \tag{4}$$

$$\mathcal{L}_{\text{exp,i}} = \lambda_{\text{exp}} e^{(\alpha_{\text{exp}} y_i)}(y_i - \hat{y}_i)^2 \tag{5}$$

$$\mathcal{L}_{\text{neg,i}} = \lambda_{\text{neg}} \max(0, -\hat{y}_i) \tag{6}$$

The NN model is trained in two phases: i) initial training and ii) retraining. During the initial training phase, the NN is trained with conventional MSE. Then, in the retraining phase, the NN is fine-tuned and trained using the systematic loss function.

6 EVALUATION

6.1 Experiment Setup

We trained Nona using PyTorch with the initial training phase of 150 epochs and the learning rate of 10^{-3} that decreases by a factor of 10 every 30 epochs. The retraining phase took the network with the best validation loss in the initial training phase and retrained the NN over 100 epochs with the learning rate of 10^{-4} decreasing by a factor of 10 every 20 epochs. We used ReLU activation and the Adam optimizer with the weight decay of 10^{-3}. The hyperparameters of the systematic loss function are set to $\lambda_{\text{exp}} = 1$, $\alpha_{\text{exp}} = 5{\times}10^{-7}$, and

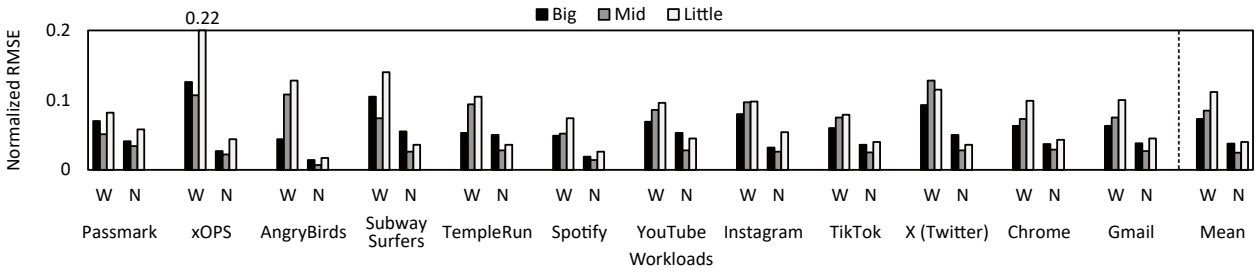
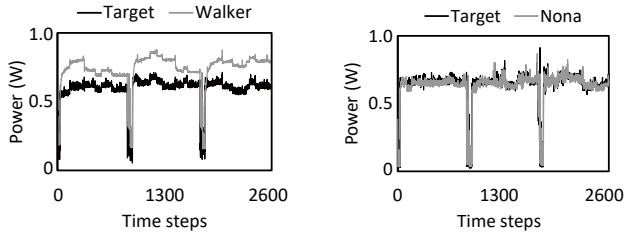


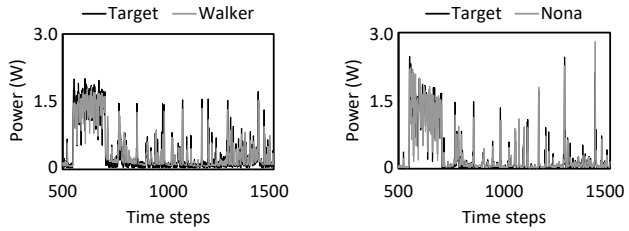
Figure 7: Normalized root-mean-square error (NRMSE) of power predictions by Walker (W) and Nona (N) for big, middle, and little clusters.

Table 2: Test Workloads for Power Prediction

Type	Applications
Stress Test	Passmark, xOPS
Game	Angry Birds, Subway Surfers, Temple Run
Media	Spotify, YouTube
Social Media	Instagram, TikTok, X (Twitter)
Miscellaneous	Chrome, Gmail



(a) Power prediction of little cluster by Walker and Nona on xOPS



(b) Power prediction of big cluster by Walker and Nona on Chrome

Figure 8: Time traces of power predictions by Walker and Nona for (a) the little cluster running xOPS and (b) big cluster running Chrome. Black lines are measured power, and gray lines are predictions.

$\lambda_{neg} = 5{\times}10^{-6}$. Notably, the lightweight NN model of Nona has an inference latency of only 0.11ms on mobile CPUs.

We compared Nona with the linear regression power model of Walker et al. [13] and partly with an NN model of Lin et al. [9]. The power models were trained with workloads in Table 1 and tested for applications in Table 2. The training workloads are selected to cover a wide variety of computation and power behaviors, and the test workloads are mostly user mobile applications and two high-power stress test benchmarks.

6.2 Accuracy of Power Prediction Model

The accuracy of power prediction models is evaluated using normalized root-mean-square error (NRMSE) that normalizes root-mean-square errors to the range of data samples. Fig. 7 plots the NRMSE of

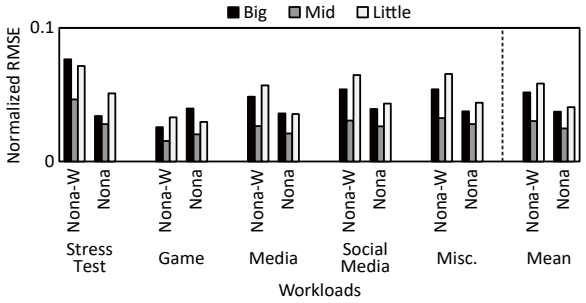


Figure 9: NRMSE of power predictions by Nona-W (Nona with Walker's PMC selections) vs. Nona.

power predictions by Walker and Nona for the test applications on Google Pixel 6. It shows that Nona has a 3.4% average error, which is 2.6x better than Walker. Since PMCs have nonlinear relations with power as shown in Fig. 1, the linear regression model struggles to capture the wide range of power variations, leading to high error rates. For instance, xOPS is a high-power stress benchmark, and it shows the largest error for Walker. Walker's power model has 12.6%, 10.7%, and 22.4% errors for big, middle, and little-core clusters, respectively. In contrast, Nona only has 2.7%, 2.2%, and 4.4% prediction errors for the respective CPU clusters.

Fig. 8 plots the time traces of measured power data and the power predictions of Walker (left) and Nona (right). Black lines in the graphs are measured power, and gray lines are predicted power numbers. Fig. 8a draws the power traces of the little-core cluster while running xOPS, and it shows large prediction errors in the case of Walker. Similarly, Fig. 8b plots the power traces of the big-core cluster running Chrome. Walker's power model shows visible prediction errors at the high-power and low-power states. On the other hand, Nona has consistently accurate power predictions regardless of application behaviors.

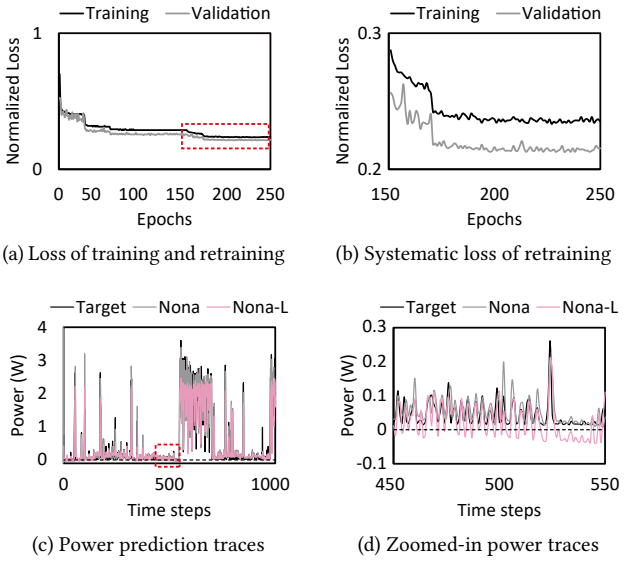
6.3 PMC Selection Strategy and Model Accuracy

To prove the significance of Nona's PMC selection strategy that reflects the nonlinear characteristics of power behaviors, it is compared to a Nona-W scheme. It has the same NN architecture, hyperparameters, and training datasets as Nona but uses different PMCs by Walker's selection method, which is based on the linear correlation assumption. The selected PMCs are compared in Table 3.

Fig. 9 charts the NRMSE of different application categories, showing that Nona outperforms Nona-W by 1.3x, 1.2x, and 1.4x for the big, middle, and little clusters, respectively. Nona slightly underperforms for gaming applications since two games (i.e., Subway Surfers

Table 3: Comparison of PMC Selections between Nona and Walker

Nona		Walker	
Address	Name	Address	Name
0x24	STALL_BACKEND	0x11	CPU_CYCLES
0x78	BR_IMMED_SPEC	0x1B	INST_SPEC
0x41	L1D_CACHE_WR	0x3C	STALL
0x7A	BR_INDIRECT_SPEC	0x13	MEM_ACCESS
0x73	DP_SPEC	0x16	L2D_CACHE
0x70	LD_SPEC	0x75	VFP_SPEC



(a) Loss of training and retraining

(b) Systematic loss of retraining

(c) Power prediction traces

(d) Zoomed-in power traces

Figure 10: (a) The change of loss values in the initial training phase with MSE. (b) The later retraining phase with the systematic loss function. (c) Power prediction traces of Nona and Nona-L for the big-core cluster running `Chrome`. (d) Zoomed-in power traces, where Nona-L makes unrealistic negative-number power predictions.

and `Temple Run 2`) activate `VFP_SPEC`, which is not in Nona's PMC list for the big cluster. For Nona, `VFP_SPEC` has the eighth highest SRC value. Nevertheless, the error difference in the gaming category is only 1.4% and 0.4% for the big and middle clusters, respectively.

6.4 Effect of Systematic Loss Function

To demonstrate the effect of Nona's systematic loss function, it is compared to a Nona-L scheme that uses conventional MSE similar to Lin et al., [9]. Both NN models are constructed with the same PMCs and trained with the same dataset. Fig. 10a shows that Nona converges to a low loss value in the later retraining phase via the systematic loss function. Fig. 10b specifically shows loss variations during the retraining phase, where the validation loss drops from 0.26 to 0.21. In the case of Nona-L, it converges to the loss value of 0.26 by training only with the MSE loss function. Although the difference between the two models seems small, the effect is considerable, as shown in Fig. 10c. The time trace reveals that Nona keeps track of rapid changes in power states, especially between the time steps of 550 and 700 with high power numbers. On the other hand, Nona-L has erroneous power predictions in this phase due to training bias toward low-power states explained in Section 5.3.2. Moreover, Nona-L even generates unrealistic predictions with negative power numbers, as shown in Fig. 10d during a low-power state. On average, Nona has 1.5x better prediction accuracy than Nona-L across all test applications.

7 CONCLUSION

This paper presents an accurate NN-based power prediction model, *Nona*, for heterogeneous CPUs. It leverages a novel PMC selection strategy to identify PMCs that have the highest correlations with power behaviors and can be measured simultaneously by the PMU. In addition, Nona designs an elegant systematic loss function that guides the NN model to have i) unbiased training for imbalanced data samples and ii) improved prediction accuracy. Experiments on a commercial mobile device demonstrate that Nona only has a 3.4% average prediction error, improving on the prior work by 2.6x.

ACKNOWLEDGMENT

This work was supported by Samsung Electronics Co., Ltd (task #IO231124-08041-01), National Research Foundation of Korea (grant #RS-2023-00283799), and Korea Evaluation Institute of Industrial Technology (grant #RS-2023-00236772). William Song is the corresponding author.

REFERENCES

[1] *Arm® Cortex®-A55 Core Technical Reference Manual*, Arm Limited, June 2023, Available: https://developer.arm.com/documentation/100442/latest/.
[2] *Arm® Cortex®-A76 Core Technical Reference Manual*, Arm Limited, June 2023, Available: https://developer.arm.com/documentation/100798/latest/.
[3] *Arm® Cortex®-X1 Core Technical Reference Manual*, Arm Limited, June 2023, Available: https://developer.arm.com/documentation/101433/latest/.
[4] S. Das, J. Werner, M. Antonakakis, M. Polychronakis, and F. Monrose, "SoK: The Challenges, Pitfalls, and Perils of Using Hardware Performance Counters for Security," *IEEE Symposium on Security and Privacy*, Sept. 2019, pp. 20–38.
[5] G. Dhiman, K. Mihic, and T. Rosing, "A System for Online Power Prediction in Virtualized Environments Using Gaussian Mixture Models," *ACM/IEEE Design Automation Conference*, June 2010, pp. 807–812.
[6] C. Dormann, J. Elith, S. Bacher, C. Buchmann, G. Carl, G. Carré, J. Marquéz, B. Gruber, B. Lafourcade, P. Leitão, T. Münkemüller, C. McClean, P. Osborne, B. Reineking, B. Schröder, A. Skidmore, D. Zurell, and S. Lautenbach, "Collinearity: A Review of Methods to Deal with It and A Simulation Study Evaluating Their Performance," *Ecography*, vol. 36, no. 1, pp. 27–46, May 2013.
[7] C. King, "stress-ng," Available: https://github.com/ColinIanKing/stress-ng.
[8] S. Li, J. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, "McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures," *International Symposium on Microarchitecture*, Dec. 2009, pp. 469–480.
[9] W. Lin, G. Wu, X. Wang, and K. Li, "An Artificial Neural Network Approach to Power Consumption Model Construction for Servers in Cloud Data Centers," *IEEE Transactions on Sustainable Computing*, vol. 5, no. 3, pp. 329–340, July 2020.
[10] F. Nie, H. Huang, X. Cai, and C. Ding, "Efficient and Robust Feature Selection via Joint ℓ2,1-Norms Minimization," *Advances in Neural Information Processing Systems*, Dec. 2010, pp. 1–9.
[11] F. Oboril, J. Ewert, and M. Tahoori, "High-Resolution Online Power Monitoring for Modern Microprocessors," *Design, Automation & Test in Europe Conference Exhibition*, Mar. 2015, pp. 265–268.
[12] M. Sagi, N. Doan, M. Rapp, T. Wild, J. Henkel, and A. Herkersdorf, "A Lightweight Nonlinear Methodology to Accurately Model Multicore Processor Power," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3152–3164, Oct. 2020.
[13] M. Walker, S. Diestelhorst, A. Hansson, A. Das, S. Yang, B. Al-Hashimi, and G. Merrett, "Accurate and Stable Run-Time Power Modeling for Mobile and Embedded CPUs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 1, pp. 106–119, Jan. 2017.
[14] S. Xi, H. Jacobson, P. Bose, G. Wei, and D. Brooks, "Quantifying Sources of Error in McPAT and Potential Impacts on Architectural Studies," *IEEE International Symposium on High-Performance Computer Architecture*, Feb. 2015, pp. 577–589.
[15] Y. Zhang, B. Kang, B. Hooi, S. Yan, and J. Feng, "Deep Long-Tailed Learning: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 9, pp. 10 795–10 816, Sept. 2023.