Microarchitecture Power Modeling via Artificial Neural Network and Transfer Learning

Jianwang Zhai
Tsinghua University
Beijing, China
zhaijw18@mails.tsinghua.edu.cn

Yici Cai
Tsinghua University
Beijing, China
caiyc@tsinghua.edu.cn

Bei Yu
Chinese University of Hong Kong
Hong Kong, China
byu@cse.cuhk.edu.hk

Abstract

Accurate and robust power models are highly demanded to explore better CPU designs. However, previous learning-based power models ignore the discrepancies in data distribution among different CPU designs, making it difficult to use data from the historical configuration to aid modeling for new target configuration. In this paper, we investigate the transferability of power models and propose a microarchitecture power modeling method based on transfer learning (TL). A novel TL method for artificial neural network (ANN)-based power models is proposed, where cross-domain mixup generates more auxiliary samples close to the target configuration to fill in the distribution discrepancy and domain-adversarial training extracts domain-invariant features to complete the target model construction. Experiments show that our method greatly improves the model transferability and can effectively utilize the knowledge of the existing CPU configuration to facilitate target power model construction.

ACM Reference Format:
Jianwang Zhai, Yici Cai, and Bei Yu. 2023. Microarchitecture Power Modeling via Artificial Neural Network and Transfer Learning . In *28th Asia and South Pacific Design Automation Conference (ASPDAC '23), January 16–19, 2023, Tokyo, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3566097.3567844

1 Introduction

The replacement of modern CPUs is gradually accelerating, with stringent time-to-market and lengthy and time-consuming design flow becoming notorious challenges. Meanwhile, power consumption has become one of the core metrics of concern in CPU design. Complex CPU architectures need to be modeled to obtain accurate power estimates to ensure a good performance-power trade-off. Therefore, how to complete power modeling for the new CPU design quickly and accurately has become a key concern of the community.

To overcome the above challenges, people try to conduct power modeling to guide the CPU design at the early design stage, *i.e.*, the microarchitecture design stage. Lee *et al.* [1] use microarchitecture design parameters to perform regression-based power modeling to support DSE, but it could not accurately model different workload programs. McPAT [2] models power analytically in a hierarchical manner, *i.e.*, from the transistor level to the architecture level. However, McPAT has struggled to meet the latest design requirements due to its low modeling accuracy and lack of support for advanced process technologies. PowerTrain [3] re-weighs the power of each component from McPAT by linear regression with L1 regularization, but it lacks transferability for different designs. Runtime power model [4] based on performance monitoring counters (PMC) allows design-time power modeling using equivalent microarchitecture events, but typically comes with some loss of accuracy.

Zhai *et al.* propose the McPAT-Calib framework [5], which utilizes machine learning (ML) methods to calibrate McPAT-7nm. Although their framework achieves high performance in experiments, the limitation is still observed. McPAT-Calib uses several typical architectures evenly distributed throughout the design space as training sets, thus ensuring high generality of the power calibration model. However, in practical applications, the CPU design space is too large [6], and people often only focus on the design and optimization of a certain configuration. Therefore, it is difficult to obtain so many typical configurations, while it is more practical to use the existing historical configuration to assist in modeling the target configuration. ML-based models usually assume that the training and test data are independent and identically distributed (*i.i.d.*), but for CPU design, even in the same design space, there are still large distribution discrepancies for different configurations. Existing power models ignore this problem, and thus have poor transferability and difficulty in using historical data and knowledge of existing configurations.

Taking RISC-V BOOM [7] as an example, it has five typical configurations for different application scenarios, *e.g.*, SmallBOOM (**S**) focuses on power efficiency, GigaBOOM (**G**) pursues higher performance, and LargeBOOM (**L**) aims at PPA balance. Based on the features selected by McPAT-Calib, Figure 1 visualizes the power data of these three configurations by principal component analysis (PCA). We can intuitively observe a large discrepancy in data distribution, which will pose two problems. (1) A power model trained with an

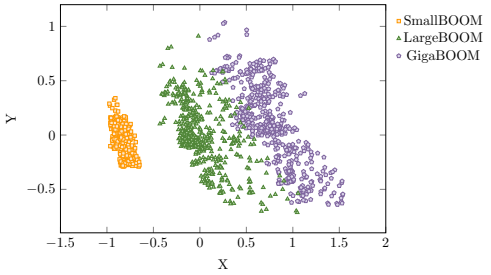


Figure 1: Data distribution for three BOOM configurations.

Table 1: Modeling MAPE for different configurations.

Ridge →	SmallBOOM	LargeBOOM	GigaBOOM
SmallBOOM	**6.93%**	11.98%	10.55%
LargeBOOM	61.29%	**8.22%**	15.24%
GigaBOOM	64.17%	15.48%	**6.74%**

ASPDAC '23, January 16–19, 2023, Tokyo, Japan

ACM ISBN 978-1-4503-9783-4/23/01.
https://doi.org/10.1145/3566097.3567844

Table 2: Microarchitecture design parameters and power statistics for three BOOM configurations.

Parameters	SmallBOOM (**S**)					LargeBOOM (**L**)					GigaBOOM (**G**)				
	−−	−	Default	+	++	−−	−	Default	+	++	−−	−	Default	+	++
FetchWidth	4	4	4	4	4	8	8	8	8	8	8	8	8	8	8
DecodeWidth	1	1	1	1	1	3	3	3	3	3	5	5	5	5	5
FetchBufferEntry	5	6	8	12	16	18	21	24	27	30	30	30	35	35	40
RobEntry	16	24	32	40	48	81	90	96	105	114	125	130	130	130	140
IntPhysRegister	36	44	52	60	68	88	94	100	105	112	108	118	128	130	140
FpPhysRegister	36	42	48	52	56	88	92	96	105	112	108	118	128	130	140
LDQ/STQEntriy	4	6	8	12	16	16	20	24	28	32	24	28	32	34	36
BranchCount	6	7	8	9	10	14	15	16	16	16	18	19	20	21	22
MemIssue/FpIssueWidth	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2
IntIssueWidth	1	1	1	1	2	2	3	3	3	3	5	5	5	5	5
DCache/ICacheWay	2	4	4	4	8	8	8	8	8	8	8	8	8	8	8
DCache/ICacheTLBEntry	8	8	8	8	16	16	16	16	16	32	32	32	32	32	32
DCacheMSHR	2	2	2	2	4	4	4	4	4	4	8	8	8	8	8
ICacheFetchBytes	2	2	2	2	2	4	4	4	4	4	4	4	4	4	4
Min. Power (mW)	9.54	9.73	10.22	10.64	12.11	21.12	22.14	22.31	22.98	28.03	36.84	38.33	34.12	35.41	36.70
Max. Power (mW)	14.62	16.01	18.10	17.06	19.94	38.07	39.15	42.56	43.85	50.52	61.80	59.94	59.75	60.43	65.23
Avg. Power (mW)	11.84	12.60	13.51	13.67	15.55	27.27	28.45	29.53	30.32	35.26	44.86	45.88	42.96	43.56	45.91
Std. Power (mW)	1.32	1.53	1.71	1.69	1.78	4.57	4.49	4.82	4.97	5.30	5.78	5.50	6.58	5.97	7.16

existing configuration is usually not accurate enough for the new target configuration, especially for some more complex target samples. As shown in Table 1, the average mean absolute percentage error (MAPE) for known configurations is only 7.30%, while for unknown configurations is up to 29.79%. (2) Directly using data from an existing configuration to aid in target model construction may have negative effects and instead reduce modeling accuracy (see Section 5.3 for details). Therefore, it is urgent to handle different distributions and improve the transferability of power models.

To handle distribution discrepancies, we can treat different configurations as different domains and use TL to complete microarchitecture power modeling. Specifically, we use an ANN-based power model and use cross-domain mixup to generate auxiliary samples, and finally, use improved domain-adversarial training to complete target model construction. Our work is the first to apply TL to power modeling, which can effectively utilize the existing configuration to enhance the power modeling of the target configuration.

The main contributions can be summarized as follows:

- We introduce prior knowledge and design an ANN-based microarchitecture power model, to automatically extract better features and complete nonlinear power modeling.
- We propose a cross-domain mixup approach to generate auxiliary samples that are closer to the target configuration, thus addressing the problem of insufficient labeled target samples and filling in distribution discrepancies.
- We perform domain-adversarial training on the source, target, and auxiliary domains to extract domain-invariant features for knowledge transfer and model construction.

2 Preliminaries

RISC-V BOOM. RISC-V is an open-source instruction set architecture (ISA) suited for various applications. Berkeley out-of-order machine (BOOM) is a ten-stage pipeline out-of-order design that has gained widespread attention from academia and industry. Thanks to the parametric microarchitecture design, designers can generate BOOM cores with different configurations, *e.g.*, SmallBOOM (**S**), LargeBOOM (**L**), GigaBOOM (**G**). We consider them as three different domains and give five sub-architectures respectively, and the design parameters are listed in Table 2, along with the measured power statistics under 100 commonly used benchmarks (*i.e.* workload programs).

Artificial Neural Network. Artificial neural network (ANN) [8] is an ML model with powerful feature extraction and function fitting capabilities. In recent years, ANN has become more and more popular, and has been widely used in prediction tasks (*e.g.* classification and regression) in various disciplines achieving more competitive results than traditional statistical models. Moreover, the ANN model exhibits good transferability [9]. In this paper, we use an ANN-based microarchitecture power model by introducing prior knowledge.

Transfer Learning. For traditional ML models, once the data distribution is changed, new labeled training data needs to be re-collected to rebuild the model [10]. For microarchitecture power modeling, if the configuration changes significantly, it is expensive to re-collect sufficient labeled data due to the lengthy and time-consuming design flow. Therefore, how to transfer knowledge from one configuration to another related configuration becomes increasingly important. TL [11] can be used to solve such problems, aiming to improve the performance on target domain by transferring the knowledge from source domain, thereby reducing the dependence on a large amount of target domain data. To the best of our knowledge, the use of TL in the field of power modeling has not been discussed before.

3 Problem Formulation

Problem 1 (Microarchitecture Power Modeling)**.** Given labeled samples of size m from a source configuration $\mathcal{C}_\mathcal{S}$, $D_s = \{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^m$, and labeled samples of size n from the target configuration $\mathcal{C}_\mathcal{T}$, $D_t = \{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^n$. The objective is to construct a target microarchitecture power model that gives accurate power predictions for unlabeled samples, $\{(\boldsymbol{x}_i^t)\}_{i=n+1}^N$, on the target configuration $\mathcal{C}_\mathcal{T}$.

Considering the correlation and distribution discrepancies between different configurations, our focus is on how to transfer the knowledge gained in the source domain (*i.e.* existing configuration) to the target domain (*i.e.* new target configuration) to improve target modeling performance.

4 Methodologies

Figure 2 illustrates our ANN- and TL-based microarchitecture power modeling flow. For given configuration and benchmark, we

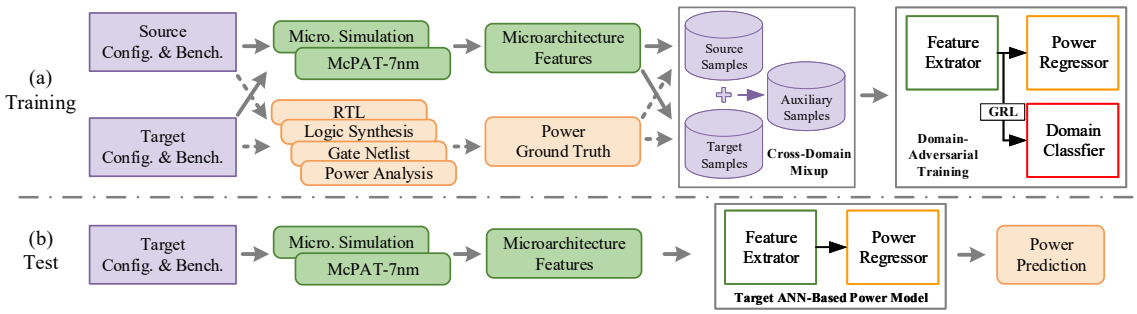


Figure 2: Our microarchitecture power modeling flow.

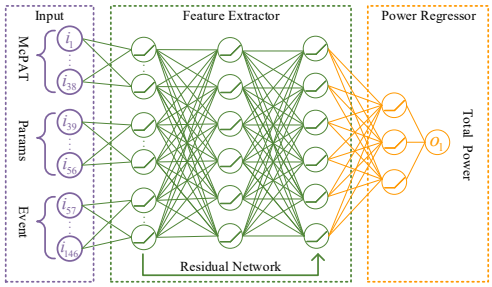
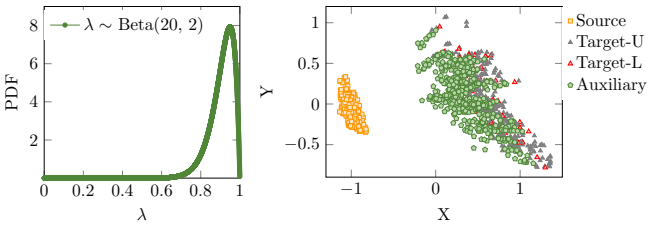


Figure 3: Network structure of our ANN-based power model.



(a) (b)

Figure 4: (a) Probability density function (PDF) of Beta distribution. (b) Data distribution after cross-domain mixup, where "Target-U" and "Target-L" indicate unlabeled and labeled target samples, respectively.

perform microarchitecture simulation (using gem5) and McPAT-7nm modeling to obtain microarchitecture modeling features $\boldsymbol{x}$, and use golden gate-level power analysis flow to obtain power ground truth y. The source samples are historical data from the existing configuration, while the training target samples need to be re-collected.

Our modeling method aims to optimize target model construction using historical source samples. First, based on prior knowledge of microarchitecture power modeling, we use an ANN-based power model to automatically extract better features from different feature sources and accomplish nonlinear modeling. Then, we use a cross-domain mixup approach to generate auxiliary samples close to the target configuration, fully utilizing the labeled samples of source configuration to overcome the insufficient labeled target samples. By this way, we can narrow the inter-domain discrepancy. Finally, we perform improved domain-adversarial training between the source, target, and auxiliary domains to extract domain-invariant features, to complete knowledge transfer and ANN-based model construction.

4.1 ANN-Based Microarchitecture Power Model

Zhai *et al.* [5] identified a wide range of feature sources for microarchitecture power modeling, including: (1) McPAT-7nm modeling results (38-dimension), contain preliminary estimates of dynamic power for all levels of components, as well as *Core.Area* and *Core.Leakage*; (2) microarchitecture design parameters (18-dimension) that provide configuration information, shown in Table 2; and (3) event statistics (90-dimension), obtained by gem5 simulation, reflect the critical activities of important CPU components. We use these three feature sources to model the total power directly.

Neural networks have shown dominance in ML, which relies on their ability to extract features. Even simple multilayer perceptron (MLP) models can adequately approximate arbitrary complex nonlinear functions. However, the MLP model has too many parameters, so it is not just difficult to train, but also easy overfitting the training data. To exploit the powerful feature extraction and transferability of ANN, we give an example ANN-based power model by introducing the prior knowledge of microarchitecture power modeling.

Figure 3 shows our model, including an input layer, a feature extractor, and a regressor for total power prediction. The input is the normalized 146-dimensional total features from three feature sources. The feature extractor contains three hidden layers. To reduce the complexity, the first hidden layer extracts features from different sources separately. According to the prior knowledge of the importance of features, the number of neurons is set to 20, 5, and 25 respectively. The second and third hidden layers are fully connected networks. To enhance gradient propagation and simplify the learning process, we add a residual network between the first and third layers. The power regressor contains a fully connected hidden layer and an output layer that predicts the total power value. The rectified linear unit (ReLU) activation function is used for each layer except the output layer to enhance the ability to model nonlinear relations.

4.2 Cross-Domain Mixup

When constructing a power model for the new target configuration, the biggest challenge comes from collecting enough labeled samples. Meanwhile, unlike the transfer of classification tasks, the distribution discrepancies in power modeling are present in both feature and label space. We propose a cross-domain mixup approach to address these problems, filling in the distribution discrepancies while enriching the labeled samples. Mixup [12] is an data augmentation method that supports simple linear behavior in-between training examples, regularizing neural networks by training them on convex combinations of random image pairs and their associated labels.

For samples belonging to the source and target configuration respectively, the similarity can be considered as the potential for

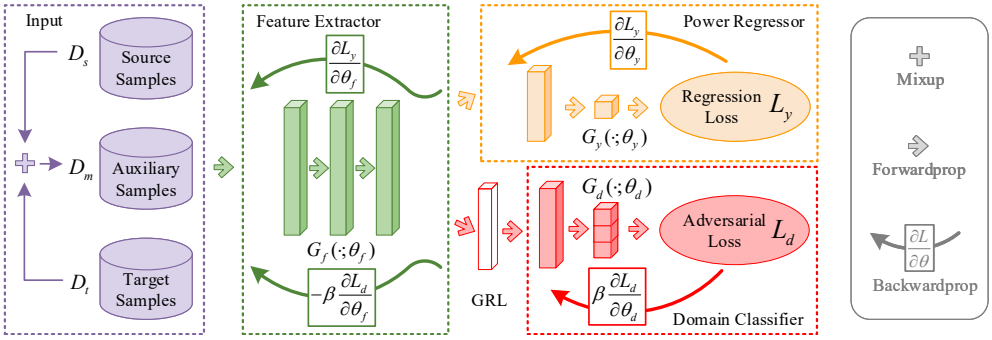


Figure 5: Illustration of the pipeline in improved domain-adversarial training.

knowledge transfer. We use the cosine measure between the source sample $\boldsymbol{x}^s$ and the target sample $\boldsymbol{x}^t$ to measure this similarity:

$$\text{similarity} = \cos < \boldsymbol{x}^s, \boldsymbol{x}^t >= \frac{\boldsymbol{x}^s \cdot \boldsymbol{x}^t}{||\boldsymbol{x}^s||||\boldsymbol{x}^t||}. \tag{1}$$

Then, for each labeled target sample $(\boldsymbol{x}^t, y^t)$, we select the k labeled source samples with the highest similarity to it and perform cross-domain mixup:

$$\boldsymbol{x}_i^m = \lambda_i \boldsymbol{x}^t + (1-\lambda_i)\boldsymbol{x}_i^s, \qquad i = 1, ..., k \tag{2}$$

$$y_i^m = \lambda_i y^t + (1-\lambda_i) y_i^s, \qquad i = 1, ..., k \tag{3}$$

where $\lambda_i \sim \text{Beta}(20, 2)$. This Beta distribution is shown in Figure 4(a), as we aim to enhance the ability to model the target configuration. Since the modeling feature $\boldsymbol{x}$ is a vector and the power label y is a scalar, the linear interpolation can be directly performed.

We treat the auxiliary samples generated by cross-domain mixup as a new domain of size kn, called auxiliary domain, *i.e.*, $D_m = \{(\boldsymbol{x}_i^m, y_i^m)\}_{i=1}^{kn}$. To keep the balance among three domains, k is set to $\lceil \frac{m+N}{2n} \rceil$. Figure 4(b) shows the data distribution of the three domains after cross-domain mixup. It can be seen that the distribution of auxiliary domain is close to the target distribution, and also helps to fill the gap between the target and source domains.

4.3 Domain-Adversarial Training

After cross-domain mixup, there are still distribution discrepancies between domains, so we aim to exploit the transferability of ANN to extract domain-invariant features to facilitate knowledge transfer. Recent works have shown that domain-adversarial training of neural networks [13] for image classification tasks can learn discriminative but domain-invariant features between the source and target domains. We improve it for a regression task (*i.e.* power modeling) and perform domain-adversarial training on three domains.

To implement adversarial training, a 3-class domain classifier is connected after the feature extractor to discriminate the source (corresponding domain labels are set to 0, *i.e.*, $d^s = 0$), target ($d^t = 1$), and auxiliary ($d^m = 2$) domains. In our task, domain-adversarial training is accomplished by jointly optimizing two networks: (1) the feature extractor and power regressor used both during training and test, and (2) the domain classifier that discriminates between the three domains during training.

Figure 5 illustrates the pipeline of the improved domain-adversarial training. Taking the samples of the three domains generated by cross-domain mixup as input, this neural network consists of three major parts, *i.e.*, the feature extractor $G_f(\cdot;\theta_f)$ with parameters θ_f, the power regressor $G_y(\cdot;\theta_y)$ with parameters θ_y, and the 3-class domain classifier $G_d(\cdot;\theta_d)$ with parameters θ_d. The two modules, G_f and G_y constitute the power model desired in Section 4.1.

We use mean squared error (MSE) as the regression loss $\mathcal{L}_y$ for power prediction, and train G_f and G_y with all labeled samples from the three domains, with losses:

$$\mathcal{L}_y^s(\theta_f, \theta_y) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}_y(G_y(G_f(\boldsymbol{x}_i^s;\theta_f);\theta_y), y_i^s), \tag{4}$$

$$\mathcal{L}_y^t(\theta_f, \theta_y) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_y(G_y(G_f(\boldsymbol{x}_i^t;\theta_f);\theta_y), y_i^t), \tag{5}$$

$$\mathcal{L}_y^m(\theta_f, \theta_y) = \frac{1}{kn}\sum_{i=1}^{kn} \mathcal{L}_y(G_y(G_f(\boldsymbol{x}_i^m;\theta_f);\theta_y), y_i^m). \tag{6}$$

The domain classifier G_d uses a softmax activation function with categorical cross-entropy (CCE) as the adversarial loss $\mathcal{L}_d$. We use all labeled and unlabeled samples to train G_d, with losses:

$$\mathcal{L}_d^s(\theta_f, \theta_d) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}_d(G_d(G_f(\boldsymbol{x}_i^s;\theta_f);\theta_d), d_i^s), \tag{7}$$

$$\mathcal{L}_d^t(\theta_f, \theta_d) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_d(G_d(G_f(\boldsymbol{x}_i^t;\theta_f);\theta_d), d_i^t), \tag{8}$$

$$\mathcal{L}_d^m(\theta_f, \theta_d) = \frac{1}{kn}\sum_{i=1}^{kn} \mathcal{L}_d(G_d(G_f(\boldsymbol{x}_i^m;\theta_f);\theta_d), d_i^m). \tag{9}$$

In order for G_f to extract domain-invariant features, *i.e.*, G_d cannot correctly perform domain classification, the complete optimization objective is as follows:

$$\begin{aligned} E(\theta_f, \theta_y, \theta_d) = \mathcal{L}_y^s(\theta_f, \theta_y) + \mathcal{L}_y^t(\theta_f, \theta_y) + \mathcal{L}_y^m(\theta_f, \theta_y) \\ -\beta(\mathcal{L}_d^s(\theta_f, \theta_d) + \mathcal{L}_d^t(\theta_f, \theta_d) + \mathcal{L}_d^m(\theta_f, \theta_d), \end{aligned} \tag{10}$$

where $\beta > 0$ is a hyper-parameter for trade-off, and the saddle point $\hat{\theta}_f, \hat{\theta}_y, \hat{\theta}_d$ given by

$$(\hat{\theta}_f, \hat{\theta}_y) = \underset{\theta_f, \theta_y}{\text{argmin}}\, E(\theta_f, \theta_y, \hat{\theta}_d), \tag{11}$$

$$\hat{\theta}_d = \underset{\theta_d}{\text{argmax}}\, E(\hat{\theta}_f, \hat{\theta}_y, \theta_d). \tag{12}$$

At the saddle point, the regression loss $\mathcal{L}_y$ on the three domains is minimized and the adversarial loss $\mathcal{L}_d$ is maximized. A *gradient reversal layer* (GRL) is inserted between the feature extractor G_f and the domain classifier G_d. During the forward propagation, GRL

passes the data normally. However, during the backward propagation, GRL takes the gradient from the domain classifier G_d and multiplies it by -1, and then passes it to the feature extractor G_f.

After completing domain-adversarial training, G_f and G_y form the desired target power model, and $G_y(G_f(\boldsymbol{x}^t; \hat{\theta}_f); \hat{\theta}_y)$ can be used to predicted the total power of unlabeled target sample $\boldsymbol{x}^t$.

5 Evaluation

As shown in Table 2, we chose three distinct configurations (*i.e.*, SmallBOOM (**S**), LargeBOOM (**L**), GigaBOOM (**G**)) of RISC-V BOOM, along with 100 commonly used benchmarks to evaluate the power models. We use 7 nm PDK ASAP7 [14] and commercial power analysis flow (Genus for logic synthesis, VCS for simulation @ 500MHz, and PrimeTime PX for power analysis) to obtain the power ground truth. Therefore, we can obtain data for three configurations, each containing 500 samples. Based on the given three configurations, we evaluate six transfer tasks: **L** → **S**, **G** → **S**, **S** → **L**, **G** → **L**, **S** → **G**, and **L** → **G**. MAPE is used as the metric to evaluate the accuracy of power modeling results. All experiments are conducted on Intel(R) Core(TM) i9-9900k CPU@3.60GHz with 64GB main memory.

5.1 Baselines

Microarchitecture Power Models. We compare our modeling method with previous representative microarchitecture power modeling approaches, including parameter-based method (HPCA07) [1], event statistics-based method (TCAD17) [4], and the model (PowerTrain) [3] that only re-weights the McPAT modeling results. McPAT-Calib [5] uses a wider range of feature sources and advanced ML methods to calibrate McPAT-7nm results. McPAT-CalibAL [15] uses an active learning method (PowerGS) to select labeled target samples. We use two naive ways to construct previous power models, *i.e.*, using only the labeled target samples (*Tgt O.*) and directly using both the labeled source and target samples (*Both*), to evaluate the impact of the source samples on target models.

Transfer Learning Baselines. Fine-tune [9] first pre-trains an ANN model using labeled source samples, and then fine-tunes the pre-trained model with labeled target samples. DANN [13] trains the target model using labeled source and target samples, and uses all samples to train a binary classifier used to discriminate source and target domains. MDD [16] learns a new feature representation by minimizing the disparity discrepancy between the encoded source and target domains. TrAdaBoostR2 [17] is based on the reverse boosting principle, which reduces the weights of poorly predicted source samples at each boosting iteration. KLIEP [18] is a kernel-based sample bias correction method minimizing the KL-divergence between a reweighted source and target distributions. KMM [19] reweights source samples to minimize the MMD between source and target domains. WANN [20] relies on an adversarial weighting approach to minimize the $\mathcal{Y}$-discrepancy between domains.

5.2 Accuracy of ANN-Based Model

We first verify the accuracy of the ANN-based power model described in Section 4.1. We perform 10-fold cross-validation in each of the three configurations, respectively, and compare our model using total features with various regression models combining feature selection in McPAT-Calib. As shown in Table 3, the ANN-based model obtains the best modeling results because better features can be automatically extracted from different feature sources and has strong nonlinear modeling capability. Compared to the MLP regressor with the same number of hidden layers and neurons, the introduction of prior knowledge leads to better results for our model.

Table 3: Modeling accuracy of different models.

Features	Model	S	L	G	Avg.
Selected	Linear Regressor	6.95%	8.12%	6.52%	7.20%
	Ridge Regressor	6.94%	8.13%	6.60%	7.22%
	Gaussian Process Regressor	7.01%	8.36%	6.92%	7.43%
	KNeighbors Regressor	6.60%	8.37%	5.82%	6.93%
	Support Vector Regressor	8.14%	9.69%	7.90%	8.58%
	Random Forest Regressor	5.48%	7.03%	6.08%	6.20%
	XGBoost Regressor	**4.56%**	5.45%	5.62%	5.21%
Total	MLP Regressor	5.27%	5.28%	5.62%	5.39%
	Our ANN-based model	5.42%	**4.89%**	**4.75%**	**5.02%**

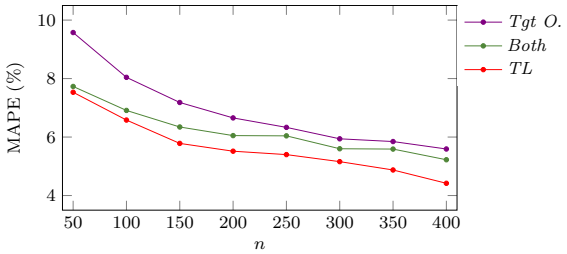


Figure 6: Test MAPE with different values of n.

5.3 Comparison and Analysis of Power Models

We compare our microarchitecture modeling method with the previous power models and investigate the transferability of these models. All previous models are constructed in two ways, *i.e.*, *Tgt O.* and *Both*, and compared with our TL-based modeling method. In each transfer task, all source samples are labeled, and for the target configuration, 80% of the samples are used as the training set, and the remaining 20% as the test set, *i.e.*, $n = 400$. To eliminate randomness, we repeat each experiment five times and report the average values of MAPE for each transfer task in Table 4. It can be observed that for TCAD17 and PowerTrain, directly using the source samples to assist the target modeling will bring negative effects and leads to a decrease in modeling accuracy, while no significant gain is observed for HPCA07. For McPAT-Calib and McPAT-CalibAL, the use of source samples is beneficial to improve the modeling accuracy of the target configuration. Compared with the state-of-the-art results (*i.e.* McPAT-CalibAL), our TL-based model achieves better results, being able to reduce the test MAPE by 18.6%.

To further explore the effectiveness of our TL method, we compare the results when n takes different values, and Figure 6 plots the average test MAPE for the six transfer tasks. As with previous models, we construct ANN-based models directly in two naive ways, and compare them with the proposed TL method (*TL*). By comparing *Tgt O.* and *Both*, it can be seen that for ANN-based model, the introduction of source samples can always enhance the modeling ability of target configurations, but the performance gain decreases with the increase of n. Importantly, our TL method can improve the transferability of the model by utilizing the knowledge of the source sample more efficiently, always showing better transfer results whether using fewer or more labeled target samples. Moreover, when achieving a similar target modeling accuracy as McPAT-CalibAL, our method requires fewer labeled target samples, *i.e.*, can reduce the dependence on labeled target samples. Since the training time of the power model

Table 4: Comparison with previous microarchitecture power models.

Task	HPCA07 [1]		TCAD17 [4]		PowerTrain [3]		McPAT-Calib [5]		McPAT-CalibAL [15]		Ours
	Tgt O.	*Both*	*Tgt O.*	*Both*	*Tgt O.*	*Both*	*Tgt O.*	*Both*	*Tgt O.*	*Both*	
L → S	10.30%	10.21%	7.81%	9.28%	7.30%	7.98%	5.15%	4.97%	4.94%	**4.67%**	5.25%
G → S	10.30%	10.23%	7.81%	10.82%	7.30%	8.93%	5.15%	5.09%	4.94%	**4.99%**	5.19%
S → L	13.63%	13.63%	10.21%	10.60%	8.23%	8.87%	5.97%	5.43%	5.54%	4.93%	**4.28%**
G → L	13.63%	13.51%	10.21%	12.61%	8.23%	8.75%	5.97%	6.10%	5.54%	5.89%	**4.14%**
S → G	11.62%	11.72%	7.55%	7.83%	6.67%	6.77%	6.56%	5.97%	6.35%	5.55%	**3.85%**
L → G	11.62%	11.58%	7.55%	8.33%	6.67%	6.75%	6.56%	5.63%	6.35%	5.42%	**3.80%**
Average	11.85%	11.81%	8.53%	9.91%	7.40%	8.01%	5.89%	5.53%	5.61%	5.24%	**4.42%**
Ratio	2.681	2.672	1.930	2.242	1.674	1.812	1.333	1.251	1.269	1.186	**1.000**

Table 5: Comparison with previous TL methods.

TL Method	**L → S**	**G → S**	**S → L**	**G → L**	**S → G**	**L → G**	Avg.→
Fine-tune [9]	5.33%	5.47%	4.81%	5.43%	5.09%	4.75%	5.15%
DANN [13]	5.47%	5.49%	4.95%	5.09%	4.75%	4.61%	5.06%
MDD [16]	5.49%	5.80%	4.78%	5.04%	5.19%	4.80%	5.18%
TrAdaB. [17]	5.41%	5.35%	4.71%	4.81%	5.10%	5.00%	5.06%
KLIEP [18]	6.20%	5.73%	4.77%	5.18%	5.19%	4.77%	5.31%
KMM [19]	5.58%	5.52%	5.04%	5.13%	5.20%	5.12%	5.27%
WANN [20]	5.91%	6.05%	4.98%	5.30%	4.75%	4.69%	5.28%
Ours	**5.25%**	**5.19%**	**4.28%**	**4.14%**	**3.85%**	**3.80%**	**4.42%**

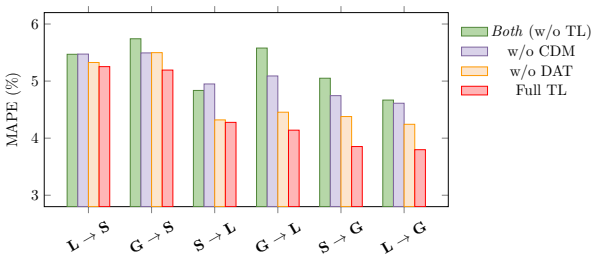


Figure 7: Ablation study of our TL method.

is much less than the acquisition time of the labeled samples, our method can effectively reduce the construction cost of target models.

5.4 Comparison and Analysis of TL Methods

After specifying the transfer task, we can compare our TL method with TL baselines, all of which are implemented on our ANN-based model. Similarly, 80% of the target samples are used as the training set. We can observe from Table 5 that our method achieves the best transfer performance. This is because most previous TL methods ignore the distribution discrepancies in the label space, and some are proposed for unsupervised domain adaptation. Our cross-domain mixup approach can fill the distribution discrepancies in both feature and label space, meanwhile the domain-adversarial training makes full use of both labeled and unlabeled target samples.

As the ablation study on our TL method, we explore the impact of the two proposed TL techniques, as shown in Figure 7. "w/o CDM" means adversarial training only on the source and target domains without cross-domain mixup, showing a slight improvement (*i.e.* Avg. MAPE=5.06%). "w/o DAT" represents training the ANN-based model directly with labeled samples from the three domains, without domain-adversarial training, and achieves better transfer improvements (*i.e.* Avg. MAPE=4.70%). "Full TL" uses both techniques to achieve the best transfer results (*i.e.* Avg. MAPE=4.42%).

6 Conclusion

We propose a novel microarchitecture power modeling method based on ANN and TL. It solves the problem of distribution discrepancies between different CPU configurations, effectively utilizing the available source configuration data to enhance the modeling ability of the target configuration. Our method improves the transferability of learning-based power models, meanwhile has higher modeling accuracy. However, how to enhance the transferability between power models from heterogeneous processors is still an open problem.

Acknowledgment

This work is supported by the National Natural Science Foundation of China 61834002. The authors thank Qiang Zhou and Xinyang Chen from Tsinghua University, Chen Bai and Xufeng Yao from CUHK, Wenlong Lyu and Zhitang Chen from Huawei Noah's Ark Lab, for their comments and feedback.

References

[1] B. C. Lee and D. M. Brooks, "Illustrative design space studies with microarchitectural regression models," in *Proc. HPCA*, 2007.
[2] S. Li, J. H. Ahn *et al.*, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proc. MICRO*, 2009.
[3] W. Lee, Y. Kim, J. H. Ryoo *et al.*, "PowerTrain: A learning-based calibration of McPAT power models," in *Proc. ISLPED*, 2015.
[4] M. J. Walker, S. Diestelhorst *et al.*, "Accurate and stable run-time power modeling for mobile and embedded CPUs," *IEEE TCAD*, 2017.
[5] J. Zhai, C. Bai, B. Zhu *et al.*, "McPAT-Calib: A microarchitecture power modeling framework for modern CPUs," in *Proc. ICCAD*, 2021.
[6] C. Bai, Q. Sun *et al.*, "BOOM-Explorer: RISC-V BOOM microarchitecture design space exploration framework," in *Proc. ICCAD*, 2021.
[7] J. Zhao *et al.*, "SonicBOOM: The 3rd generation berkeley out-of-order machine," in *Fourth Workshop on Computer Architecture Research with RISC-V*, 2020.
[8] O. I. Abiodun, A. Jantan *et al.*, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, 2018.
[9] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proc. NeurIPS*, 2014.
[10] W. Ji, T.-Y. Ho, and H. Yao, "Transfer learning-based microfluidic design system for concentration generation," in *Proc. DAC*, 2020.
[11] F. Zhuang, Z. Qi, K. Duan *et al.*, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, 2020.
[12] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *Proc. ICLR*, 2018.
[13] Y. Ganin, E. Ustinova, H. Ajakan *et al.*, "Domain-adversarial training of neural networks," *Journal of Machine Learning Research*, 2016.
[14] L. T. Clark, V. Vashishtha *et al.*, "ASAP7: A 7-nm finFET predictive process design kit," *Microelectronics Journal*, 2016.
[15] J. Zhai, C. Bai, B. Zhu *et al.*, "McPAT-Calib: A RISC-V BOOM microarchitecture power modeling framework," *IEEE TCAD*, 2022.
[16] Y. Zhang, T. Liu, M. Long, and M. Jordan, "Bridging theory and algorithm for domain adaptation," in *Proc. ICML*, 2019.
[17] D. Pardoe and P. Stone, "Boosting for regression transfer," in *Proc. ICML*, 2010.
[18] M. Sugiyama, S. Nakajima *et al.*, "Direct importance estimation with model selection and its application to covariate shift adaptation," in *Proc. NeurIPS*, 2007.
[19] J. Huang, A. Gretton, K. Borgwardt *et al.*, "Correcting sample selection bias by unlabeled data," in *Proc. NeurIPS*, 2006.
[20] A. de Mathelin, G. Richard, F. Deheeger *et al.*, "Adversarial weighting for domain adaptation in regression," in *Proc. ICTAI*, 2021.