ASP-DAC 2023
28th Asia and South Pacific Design Automation Conference
Date: January 16 – 19, 2023
ASIA SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE

Microarchitecture Power Modeling via Artificial Neural Network and Transfer Learning

Jianwang Zhai[1], Yici Cai[1], **Bei Yu**[2]

[1]Tsinghua University
[2]The Chinese University of Hong Kong

Jan. 18, 2023

Introduction

Power Modeling for Modern CPUs

Power Modeling

- With the slowdown of Moore's law and the breakdown of Dennard scaling, power consumption has become the main challenge in high power-efficiency CPU design.
- Accurate and robust power models are highly demanded to explore better designs.
- Time-to-market for newer generations of CPU is stringent, and designers want to perform modeling for the new target CPU at a lower cost.

Challenges

- High requirements: modeling speed, accuracy, and generality.
- Complex architecture & large-scale design space & advanced technology.
- The lengthy EDA design flow makes data collection very difficult.

Previous Solutions & Limitations

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.

Related Work

- Regression for power modeling using design parameters[1].
- PMC-based power models using equivalent microarchitecture events [2].
- Hierarchical analytical power modeling, McPAT[3].
- Re-weighting McPAT results by linear regression[4].
- Calibrate McPAT using a broader range of features and machine learning (ML)[5].

[1] B. C. Lee, et al. "Illustrative Design Space Studies with Microarchitectural Regression Models". In: *Proc. HPCA*, 2007.

[2] M. J. Walker, et al. "Accurate and Stable Run-Time Power Modeling for Mobile and Embedded CPUs". In: *IEEE TCAD*, 2017.

[3] S. Li, et al. "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures". In: *Proc. MICRO*, 2009.

[4] W. Lee, et al. "PowerTrain: A learning-based calibration of McPAT power models". In: *Proc. ISLPED*, 2015.

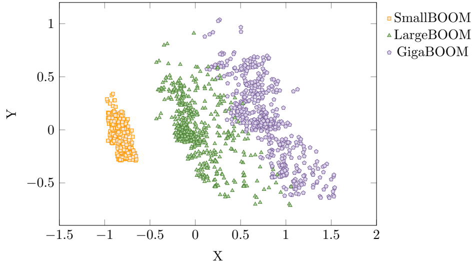
[5] J. Zhai, et al. "McPAT-Calib: A Microarchitecture Power Modeling Framework for Modern CPUs". In: *Proc. ICCAD*, 2021.

Limitations

- ML-based models usually assume that the training and test data are independent and identically distributed , *i.e.*, *i.i.d.*
- However, there are significant discrepancies in data distribution among different CPU designs, even when they are in the same design space.
- Thus, previous learning-based models ignore the issue of transferability and often require retraining the model by collecting sufficient data for a new target CPU design.

Gap between Training & Testing Data: Example

- Taking RISC-V BOOM[6] as an example, principal component analysis (PCA) is used to visualize the power data for three different configurations.
- Power models trained using one existing configuration are usually not accurate enough for new target configurations.



Data distribution.

Table: Modeling MAPE.

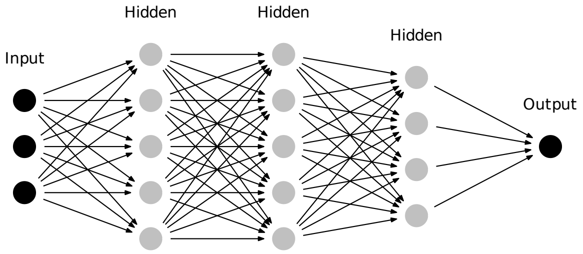
Ridge →	SmallBOOM	LargeBOOM	GigaBOOM
SmallBOOM	**6.93%**	11.98%	10.55%
LargeBOOM	61.29%	**8.22%**	15.24%
GigaBOOM	64.17%	15.48%	**6.74%**

[6] J. Zhao, et al. "SonicBOOM: The 3rd generation berkeley out-of-order machine." *Fourth Workshop on Computer Architecture Research with RISC-V*, 2020.

Preliminaries

Artificial Neural Network

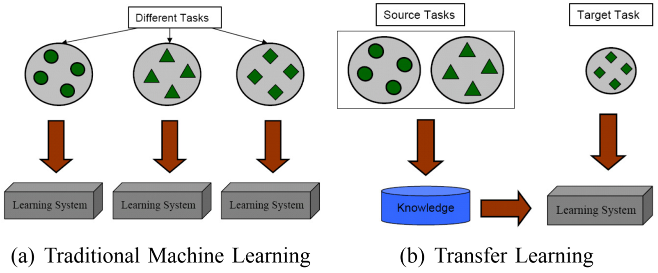
- Artificial neural network (ANN) is an ML model with powerful feature extraction and function fitting capabilities.
- Importantly, ANN models exhibit good transferability[7] and are widely used for prediction tasks in various disciplines.



[7] J. Yosinski, et al. "How transferable are features in deep neural networks?" In: *Proc. NeurIPS*, 2014.

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.
- Transfer Learning (TL)[8] aims to improve the performance of the target model by transferring knowledge from the source domain, thus reducing the dependence on target domain data.



[8] S. J. Pan, et al. "A survey on transfer learning". In: *IEEE TKDE*, 2010.

Problem Formulation

Definition (Power)

The total power can be expressed as:

$$P = P_{dynamic} + P_{static} = \underbrace{\alpha C V_{DD}^{2} f + V_{DD} I_{leakage}}_{\text{Transistor level}} = \underbrace{\sum \beta_n f_n(e_n, \boldsymbol{d}) + g(\boldsymbol{d})}_{\text{Microarchitecture level}} \tag{1}$$

Definition (Microarchitecture Configuration)

A CPU design characterized by a set of microarchitecture design parameters, such as *FetchWidth*, *DecodeWidth*, *FetchBufferEntry*, *etc.*.

Definition (Benchmark)

The workload program executed on the target CPU design.

Problem Formulation

- The objective of microarchitecture power modeling is to estimate the power of different benchmarks running on the target microarchitecture configuration.
- To handle the distribution discrepancies, we focus on how to transfer the knowledge gained in the source domain (*i.e.*, the existing configuration) to the target domain (*i.e.*, the new target configuration) to improve the target modeling performance.

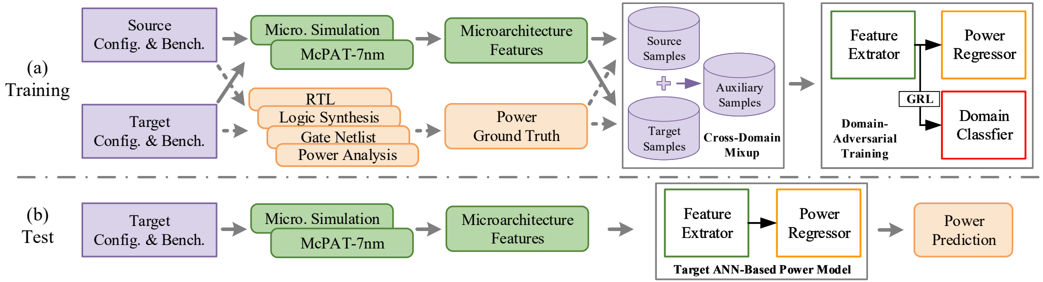
Problem (Microarchitecture Power Modeling)

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 power model that gives accurate power predictions for unlabeled samples, $\{(\boldsymbol{x}_i^t)\}_{i=n+1}^N$, on the target configuration $\mathcal{C}_{\mathcal{T}}$.

Method

Overview

- ANN-Based Microarchitecture Power Model
- Cross-Domain Mixup
- Domain-Adversarial Training

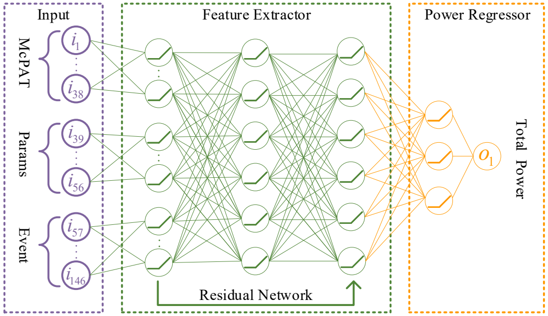


Our microarchitecture power modeling flow.

ANN-Based Power Model

- The total power is modeled directly using the modeling features.
- Feature sources: McPAT-7nm modeling results & microarchitecture design parameters & event statistics obtained by gem5 simulation.
- Features are extracted separately from different sources according to their importance; a residual network is added to enhance the gradient propagation.

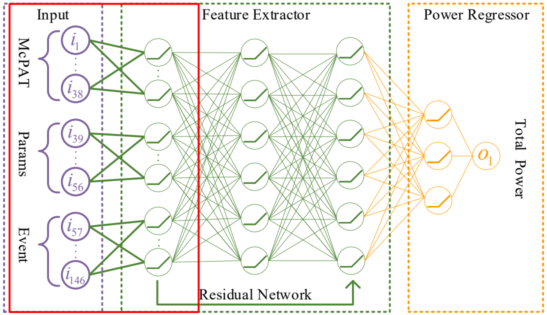
Network structure of ANN-based power model:



ANN-Based Power Model

- The total power is modeled directly using the modeling features.
- Feature sources: McPAT-7nm modeling results & microarchitecture design parameters & event statistics obtained by gem5 simulation.
- Features are extracted separately from different sources according to their importance; a residual network is added to enhance the gradient propagation.

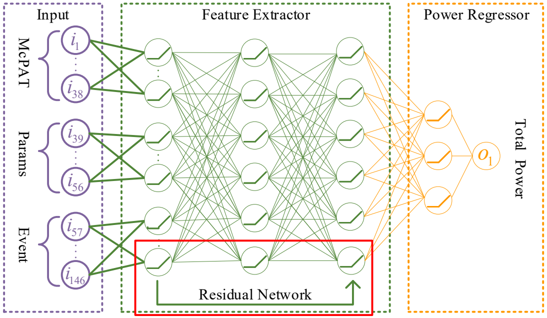
Network structure of ANN-based power model:



ANN-Based Power Model

- The total power is modeled directly using the modeling features.
- Feature sources: McPAT-7nm modeling results & microarchitecture design parameters & event statistics obtained by gem5 simulation.
- Features are extracted separately from different sources according to their importance; a residual network is added to enhance the gradient propagation.

Network structure of ANN-based power model:



Cross-Domain Mixup

Motivation

- The biggest challenge comes from collecting enough labeled samples.
- The distribution discrepancies are present in both feature and label space.

Cross-Domain Mixup

- The similarity can be considered as the potential for knowledge transfer:

$$\text{similarity} = \cos < \boldsymbol{x}^s, \boldsymbol{x}^t >= \frac{\boldsymbol{x}^s \cdot \boldsymbol{x}^t}{||\boldsymbol{x}^s||||\boldsymbol{x}^t||}. \tag{2}$$

- For each labeled target sample $(\boldsymbol{x}^t, y^t)$, we select the k most similar labeled source samples and perform mixup:

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

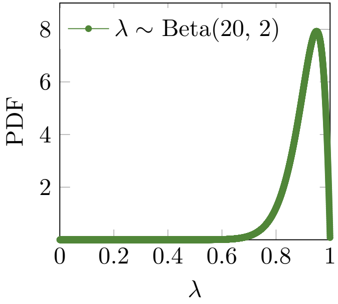
$$y_i^m = \lambda_i y^t + (1-\lambda_i) y_i^s, \qquad i = 1, ..., k \tag{4}$$

where $\lambda_i \sim \text{Beta}(20, 2)$, k is set to $\lceil \frac{m+N}{2n} \rceil$ to keep the balance among domains.

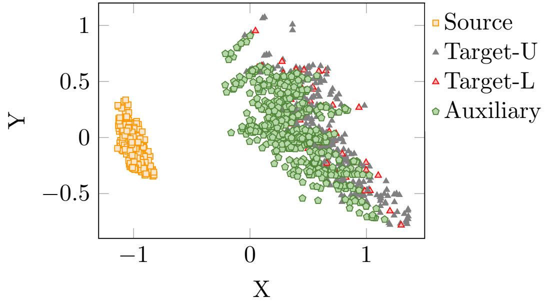
Auxiliary Domain

Data Distribution

- We can get the labeled auxiliary domain samples, $D_m = \{(\boldsymbol{x}_i^m, y_i^m)\}_{i=1}^{kn}$, that are closer to the target distribution.



(a) Beta distribution.



(b) Data distribution.

Domain-Adversarial Training

Motivation

- After cross-domain mixup, there are still distribution discrepancies between the different domains.
- How to take advantage of all labeled and unlabeled samples to train a better model for the target domain.

Domain-Adversarial Training

- Domain-adversarial training of neural networks[9] for image classification tasks can learn discriminative but domain-invariant features between the source and target domains.
- We extend it for a regression task (*i.e.*, power modeling) and perform domain-adversarial training on three domains: source, target, and auxiliary domains.

[9] Y. Ganin et al. "Domain-adversarial training of neural networks". In: *JMLR*, 2016.

Domain-Adversarial Training: Pipeline

- Input: all labeled and unlabeled samples from the three domains.
- Three major parts: the feature extractor $G_f(\cdot;\theta_f)$, the power regressor $G_y(\cdot;\theta_y)$, and the 3-class domain classifier $G_d(\cdot;\theta_d)$.
- GRL is a gradient reversal layer, aiming to make G_d unable to distinguish which domain the sample comes from, thereby extracting domain-invariant features.

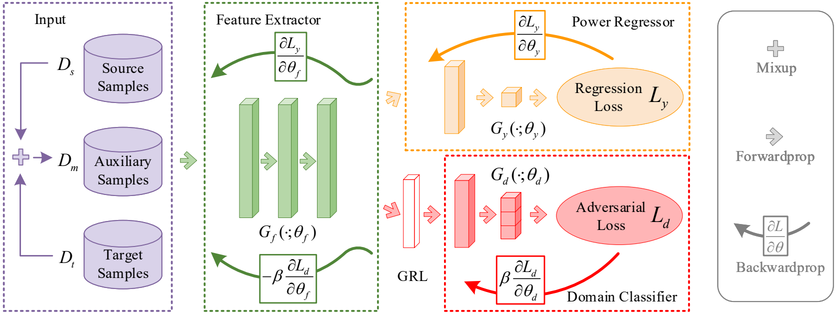


Illustration of the pipeline in improved domain-adversarial training.

Domain-Adversarial Training

Power Regression

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

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

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

$$\mathcal{L}_y^m(\theta_f, \theta_y) = \frac{1}{kn} \sum_{i=1}^{kn} \mathcal{L}_y(G_y(G_f(\boldsymbol{x}_i^m; \theta_f); \theta_y), y_i^m). \tag{7}$$

Domain-Adversarial Training

Adversarial Training

- 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_f and G_d, with losses:

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

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

$$\mathcal{L}_d^m(\theta_f, \theta_d) = \frac{1}{kn} \sum_{i=1}^{kn} \mathcal{L}_d(G_d(G_f(\boldsymbol{x}_i^m; \theta_f); \theta_d), d_i^m). \tag{10}$$

Domain-Adversarial Training

Optimization Objective and Solution

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

$$\begin{aligned} E(\theta_f, \theta_y, \theta_d) = \mathcal{L}_y^s(\theta_f, \theta_y) + \mathcal{L}_y^t(\theta_f, \theta_y) + \mathcal{L}_y^m(\theta_f, \theta_y) \\ -\beta(\mathcal{L}_d^s(\theta_f, \theta_d) + \mathcal{L}_d^t(\theta_f, \theta_d) + \mathcal{L}_d^m(\theta_f, \theta_d)), \end{aligned} \tag{11}$$

  where $\beta > 0$ is a hyper-parameter for trade-off.

- The saddle point $\hat{\theta}_f$, $\hat{\theta}_y$, $\hat{\theta}_d$ are given by:

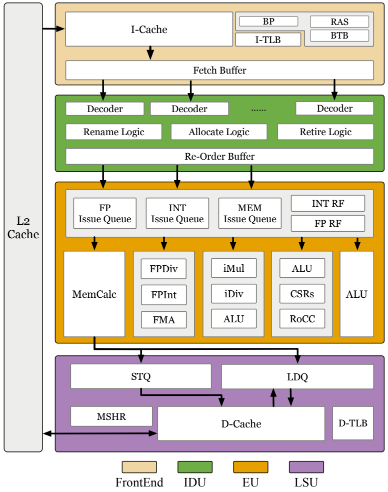
$$(\hat{\theta}_f, \hat{\theta}_y) = \underset{\theta_f, \theta_y}{\operatorname{argmin}} E(\theta_f, \theta_y, \hat{\theta}_d), \tag{12}$$

$$\hat{\theta}_d = \underset{\theta_d}{\operatorname{argmax}} E(\hat{\theta}_f, \hat{\theta}_y, \theta_d). \tag{13}$$

- The final target model: $y^t_{prediction} = G_y(G_f(\boldsymbol{x}^t; \hat{\theta}_f); \hat{\theta}_y)$.

Evaluation

RISC-V BOOM



Detailed BOOM pipeline.

RISC-V

- Free & Open source; Easy to start.
- Has received great attention and support from academia and industry.

BOOM

- A family of out-of-order RISC-V designs.
- High performance & Parametric microarchitecture design & Automatic design flow.
- There are various typical configurations for different application scenarios.

Experiment: Data & Task

- Three distinct RISC-V BOOM configurations; 100 commonly used benchmarks.
- Six transfer tasks: **L** → **S**, **G** → **S**, **S** → **L**, **G** → **L**, **S** → **G**, and **L** → **G**.

Table: Design parameters and power statistics of 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

Result: Power Modeling Baselines

- **HPCA07**[10]: design parameter-based modeling method.
- **TCAD17**[11]: microarchitecture event statistics-based method.
- **PowerTrain**[12]: re-weights the McPAT modeling results with L1 regularization.
- **McPAT-Calib**[13]: use a wider range of feature sources and advanced ML methods.
- **McPAT-CalibAL**[14]: use an active learning method to select labeled target samples.

Training Ways

- *Tgt O.*: train the target model using only the available labeled target samples.
- *Both*: train the target model using both the labeled source and target samples.

[10] B. C. Lee, et al. "Illustrative Design Space Studies with Microarchitectural Regression Models". In: *Proc. HPCA*, 2007.

[11] M. J. Walker, et al. "Accurate and Stable Run-Time Power Modeling for Mobile and Embedded CPUs". In: *IEEE TCAD*, 2017.

[12] W. Lee, et al. "PowerTrain: A learning-based calibration of McPAT power models". In: *Proc. ISLPED*, 2015.

[13] J. Zhai, et al. "McPAT-Calib: A Microarchitecture Power Modeling Framework for Modern CPUs". In: *Proc. ICCAD*, 2021.

[14] J. Zhai, et al. "McPAT-Calib: A RISC-V BOOM Microarchitecture Power Modeling Framework". In: *IEEE TCAD*, 2023.

Experiment: Transfer Learning Baselines

- **Fine-tune**[15] first pre-trains an ANN model using labeled source samples, and then fine-tunes the pre-trained model with labeled target samples.
- **DANN**[16] 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**[17] learns a new feature representation by minimizing the disparity discrepancy between the encoded source and target domains.
- **TrAdaBoostR2**[18] is based on the reverse boosting principle, which reduces the weights of poorly predicted source samples at each boosting iteration.

[15] J. Yosinski, et al. "How transferable are features in deep neural networks?" In: *Proc. NeurIPS*, 2014.

[16] Y. Ganin, et al. "Domain-adversarial training of neural networks". In: *JMLR*, 2016.

[17] Y. Zhang, et al. "Bridging theory and algorithm for domain adaptation". In: *Proc. ICML*, 2019.

[18] D. Pardoe, et al. "Boosting for regression transfer". In: *Proc. ICML*, 2010.

Experiment: Transfer Learning Baselines

- **KLIEP**[19] is a kernel-based sample bias correction method minimizing the KL-divergence between a reweighted source and target distributions.
- **KMM**[20] reweights source samples to minimize the maximum mean discrepancy (MMD) between source and target domains.
- **WANN**[21] relies on an adversarial weighting approach to minimize the $\mathcal{Y}$-discrepancy between domains.

[19] M. Sugiyama, et al. “Direct importance estimation with model selection and its application to covariate shift adaptation”. In: *Proc. NeurIPS*, 2007.
[20] J. Huang, et al. “Correcting sample selection bias by unlabeled data”. In: *Proc. NeurIPS*, 2006.
[21] A. de Mathelin, et al. “Adversarial weighting for domain adaptation in regression”. In: *Proc. ICTAI*, 2021.

Accuracy of ANN-Based Model

- Cross-validation is performed in each of the three configurations.
- Compared with SOTA ML models, ANN-based power models embedded with a prior knowledge can achieve comparable or even better modeling accuracy.

Table: 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%**

Comparison and Analysis of Power Models

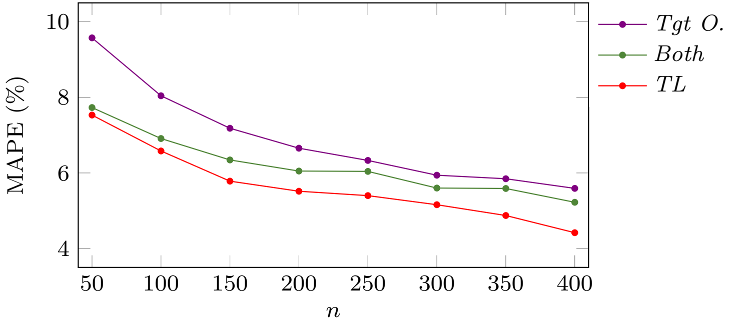
- Perform the six transfer tasks in the three distinct configurations.
- The use of source samples is not always beneficial (e.g. TCAD17 & PowerTrain).
- Compared with the SOTA results, our TL-based model achieves better results.

Table: Comparison with previous microarchitecture power models.

Task	HPCA07		TCAD17		PowerTrain		McPAT-Calib		McPAT-CalibAL		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**

Further Analysis on Target Domain Num n

- For ANN-based model, the introduction of source samples can always enhance the target modeling ability, but the performance gain decreases with the increase of n.
- Importantly, our TL method can improve the transferability of the model by utilizing the source knowledge more efficiently, always showing better transfer results.



Test MAPE with different values of n (labelled num in target domain).

Transfer Results

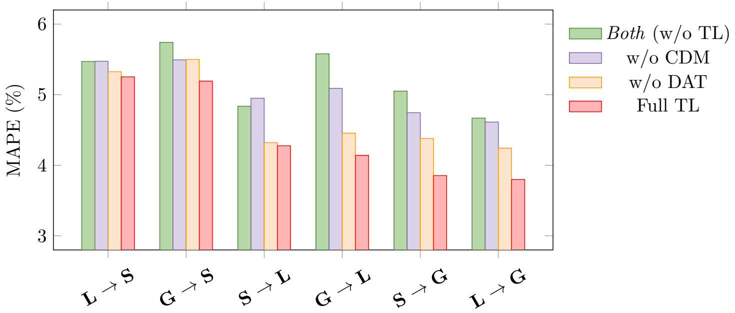
- All TL methods are implemented on our ANN-based model.
- Cross-domain mixup can fill the distribution discrepancies in both feature and label space, meanwhile the domain-adversarial training makes full use of both labeled and unlabeled target samples, thus achieving the best transfer performance.

Table: Comparison with previous TL methods.

TL Method	**L → S**	**G → S**	**S → L**	**G → L**	**S → G**	**L → G**	Avg.→
Fine-tune	5.33%	5.47%	4.81%	5.43%	5.09%	4.75%	5.15%
DANN	5.47%	5.49%	4.95%	5.09%	4.75%	4.61%	5.06%
MDD	5.49%	5.80%	4.78%	5.04%	5.19%	4.80%	5.18%
TrAdaB.	5.41%	5.35%	4.71%	4.81%	5.10%	5.00%	5.06%
KLIEP	6.20%	5.73%	4.77%	5.18%	5.19%	4.77%	5.31%
KMM	5.58%	5.52%	5.04%	5.13%	5.20%	5.12%	5.27%
WANN	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%**

Ablation Study

- "w/o CDM" means adversarial training only on the source and target domains without cross-domain mixup.
- "w/o DAT" represents training the model directly with labeled samples from the three domains, without domain-adversarial training.
- "Full TL" uses both techniques to achieve the best transfer results.



Ablation study of our TL method.

Conclusion

Why is our modeling method effective?

- **ANN-Based Power Model**: to ensure powerful feature extraction and transferability.
- **Cross-Domain Mixup**: can address the problem of insufficient labeled target samples and fill in the distribution discrepancie in both feature and label space.
- **Domain-Adversarial Training**: can extract domain-invariant features for further knowledge transfer and complete the target model construction.

Open problem

- How to improve the transferability between power models of highly heterogeneous processors.

THANK YOU!