2025 IEEE/ACM 47th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER) | 979-8-3315-3711-1/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICSE-NIER66352.2025.00031

Recommending Pre-Trained Models for IoT Devices

Parth V. Patil, Wenxin Jiang, Huiyun Peng,
Daniel Lugo, Kelechi G. Kalu
Purdue University, USA

Josh LeBlanc, Lawrence Smith,
Hyeonwoo Heo, Nathanael Aou
Purdue University, USA

James C. Davis
Purdue University, USA

***Abstract*—The availability of pre-trained models (PTMs) has enabled faster deployment of machine learning across applications by reducing the need for extensive training. Techniques like quantization and distillation have further expanded PTM applicability to resource-constrained IoT hardware. Given the many PTM options for any given task, engineers often find it too costly to evaluate each model's suitability. Approaches such as LogME, LEEP, and ModelSpider help streamline model selection by estimating task relevance without exhaustive tuning. However, these methods largely leave hardware constraints as future work—a significant limitation in IoT settings. In this paper, we identify the limitations of current model recommendation approaches regarding hardware constraints and introduce a novel, hardware-aware method for PTM selection. We also propose a research agenda to guide the development of effective, hardware-conscious model recommendation systems for IoT applications.**

***Index Terms*—Pre-Trained Models, Model Recommendation, IoT, Machine Learning**

I. Introduction

Many IoT applications today use deep neural networks (DNNs) for advanced functionality. Examples include object detection in autonomous vehicles [1], crop health monitoring in agriculture [2], and natural language processing for smart home assistants [3]. Given the cost of developing and training a DNN from scratch [4], engineers often leverage pre-trained models (PTMs) to expedite development and deployment processes. However, as highlighted by recent studies [5], selecting an appropriate PTM frequently requires manual evaluation of model performance on downstream tasks, a process that is time-intensive and prone to variability. Furthermore, a separate study [6] found that practitioners often encounter hardware constraints or lack the expertise needed to adapt DNN as the main hurdle. This underscores the value of a systematic approach to PTM recommendation, particularly one that addresses the resource limitations of IoT hardware.

PTM selection extends beyond IoT, with prior work making significant strides in recommending PTMs for specific tasks without model fine-tuning. Figure 1 summarizes the current PTM selection pipeline. These methods fall into two categories: heuristic-based [7]–[14] and learning-based [15]–[17], as detailed in II. However, their primary focus is maximizing model performance (*e.g.,* accuracy), with limited attention to hardware constraints. Deploying PTMs on IoT devices introduces many such constraints, including limited CPU, memory, energy, and low-bandwidth communication. Adapting current recommendation approaches to address these IoT-specific limitations is nontrivial, as many of these methods

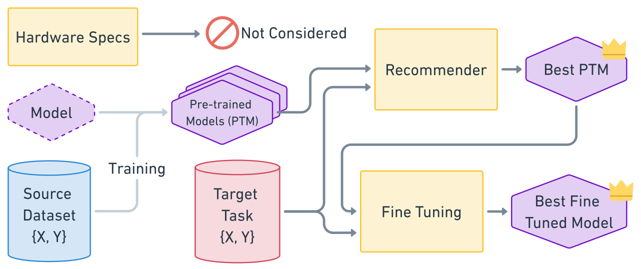

Fig. 1: A model recommendation process for selecting the best pre-trained model (PTM) for a target task. Notably, hardware specifications are not considered, limiting use on constrained devices. Models trained on a source dataset (X, Y) are stored in a model hub. For a target task with a non-overlapping dataset (X, Y), the system recommends the most suitable PTM for fine-tuning, resulting in the best model for the task.

rely on measures such as class similarity between source and target tasks or the model's capacity for distinguishing classes. Therefore, an effective IoT solution must incorporate both task suitability and hardware constraints, with potential further insights from modeling device energy consumption. We identify two critical gaps in the state-of-the-art PTM recommendation systems: (1) *Lack of IoT-specific inputs for model recommendation* (2) *Lack of Ground-Truth rankings for IoT devices*. Developing a hardware-aware recommender requires first establishing a ground truth model ranking on devices, which is essential for addressing the first gap.

In this work, we introduce approaches and research agendas to address these issues. First, we propose two modifications to the Model Spider framework [15] to enable hardware-aware recommendations, which we term *Model Spider Fusion* and *Model Spider Shadow*. Both approaches aim to address the first gap, each with distinct methods. Additionally, we outline methods for collecting raw data by defining essential metrics and generating custom rankings based on these hardware-specific performance indicators. We close by discussing future opportunities to support the ongoing development of hardware-conscious model recommendation systems.

Our contributions are:

1) We identify key gaps in PTM recommendation for IoT.
2) We introduce methods to address these gaps, focusing on hardware-specific metrics and tweaks to existing methods.
3) We outline a research agenda aimed at advancing PTM recommendations with a focus on hardware and sustainability.

II. Background and Related Work

This section covers background and related works on pre-trained model recommendation (§II-A) and model benchmarking in IoT systems (§II-B).

A. *Works for Model Recommendation*

Researchers have developed various methods to help software engineers identify the most suitable PTMs, aiming to minimize fine-tuning and forward passes to reduce computational costs, as summarized in Table I. This section reviews state-of-the-art approaches for PTM recommendations in downstream tasks, categorizing them into two main types: *heuristic-based* and *learning-based*.

Heuristic-Based Methods: These methods typically devise a novel heuristic or scoring system to rank a PTM's effectiveness for a target task. This category includes methods, such as NCE [7], and H-Score [8], which estimate the similarity between source and target labels directly. Additional methods, such as LEEP [9], N-LEEP [10], LogME [11], PACTran [12], GBC [13], and LFC [14], rely on a forward pass on the target dataset to capture representations. These representations are then analyzed for their usefulness based on target labels. However, heuristic approaches often struggle to capture the nuances of hardware specifications and their correlation with model performance, overlooking the practical challenges software engineers face when deploying models on hardware-constrained environments. For example, on IoT devices, larger models can leverage hardware-accelerated activation functions to boost performance, while smaller models with complex functions may still encounter bottlenecks.

Learning-Based Methods: Recent methods, including Model Spider [15], EMMS [16], and Fennec [17], employ machine learning to eliminate the need for forward passes. These methods encode both the PTM and target dataset into a latent space, using learning techniques to discern patterns and predict performance. Given their promising results, we focus on learning-based techniques for our hardware-aware recommendation solutions, as this approach is well-suited for software engineers to encode complex hardware specifications, such as CPU type, architecture, memory size, and I/O speed.

Model Spider tokenizes tasks and pre-trained models to generate recommendations that balance efficiency and accuracy [15]. The Model Spider approach encodes each pre-trained model into a token θ_m using an extractor Ψ, which encapsulates essential characteristics such as architectures and parameters. Simultaneously, tasks are embedded as other tokens $\mu(T)$, reflecting dataset statistics and task characteristics. The key contribution of Model Spider is its use of a multi-head attention mechanism to assess the similarity $sim(\theta_m, \mu(T))$ between model and task tokens. This similarity score serves as a reliable indicator of model performance, enabling a ranked selection of pre-trained models aligned with the task requirements. Optionally, it also allows for the application of a forward pass, leading to a refined token θ_m^* that further captures data-specific attributes. This is done for top K-ranked models from the first iteration. Unlike EMMS [16] and Fennec [17], Model Spider's tokenized, attention-based design is more suitable to capture the nuances of hardware specifications.

Table I: This table compares current model recommendation approaches and their requirements. ModelSpider is notably efficient, not requiring a forward pass on the target dataset; however, none address hardware constraints—a key limitation for resource-constrained IoT environments.

Recommendation Approach	**Requires**			**Hardware Aware**
	Forward Pass	*Source Labels*	*Target Labels*	
NCE [7]	No	-	-	×
H-Score [8]	No	-	-	×
OTCE [18]	-	-	-	×
LEEP [9]	-	No	-	×
N-LEEP [10]	-	No	-	×
LogME [11]	-	-	No	×
PACTran [12]	-	-	-	×
GBC [13]	-	No	-	×
LFC [14]	-	No	-	×
Model Spider [15]	No	No	No	×
EMMS [16]	No	No	-	×
Fennec [17]	No	No	-	×
MS Fusion	No	No	No	Yes
MS Shadow	No	No	No	Yes

B. *Works for Model Benchmarking on IoT*

Several studies have focused on benchmarking machine learning model performance on IoT devices, addressing specific applications and sustainability considerations relevant to software engineering workflows. For example, LwHBench [19] provides performance benchmarks to assess model performance, they use it for applications in agriculture, while Sayeedi et al. [20] emphasize the sustainability of models in terms of environmental impact. However, these approaches primarily evaluate models individually rather than comparing and ranking them. Sayeedi et al. introduce innovative metrics such as the Green Carbon Footprint, which combines power consumption, execution time, and other factors to provide a more holistic assessment of a model's environmental impact. Our work extends these studies by incorporating these metrics to provide a systematic way for software engineers to rank model performance across IoT devices, facilitating more informed and environmentally conscious model recommendations.

III. Motivation and Gap Analysis

While previous work has focused on model recommendations for downstream reuse, no studies or data currently support hardware-aware model selection. This section outlines key gaps in hardware awareness (§III-A) and the lack of ground truth (§III-B) in model recommender works.

A. *Lack of IoT specific inputs for model recommendation*

A key limitation of current model recommendation approaches is that they fail to consider the hardware performance implications while selecting models. For example, a recommendation system may rank models M1, M2, and M3, prioritizing M1 based on task performance. However,

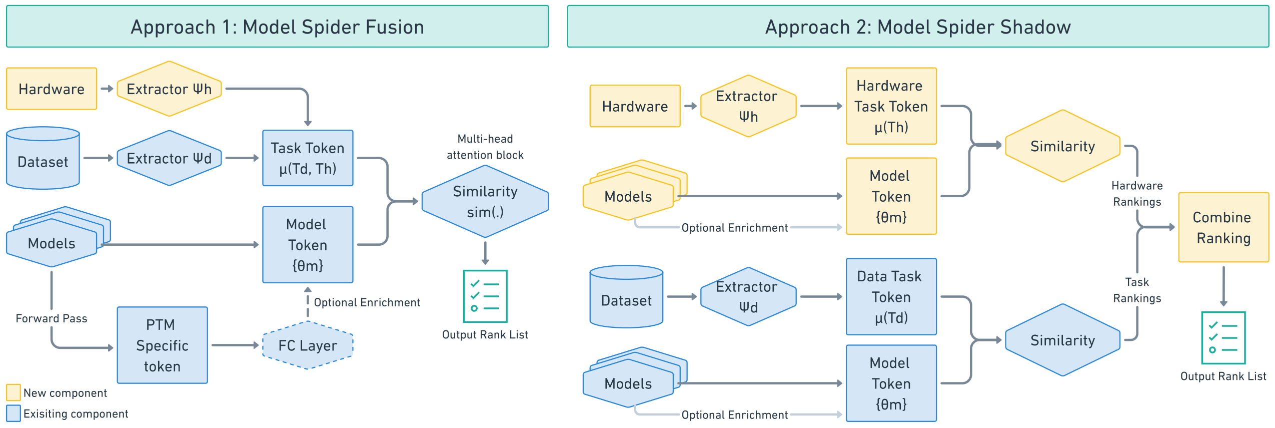

Fig. 2: Overview of our proposed IoT-specific model recommendation approaches. Yellow components indicate new modules introduced, and blue represents existing Model Spider components. *Model Spider Fusion* incorporates hardware specifications directly into task tokens via a hardware extractor. *Model Spider Shadow* creates dual ranking systems—task relevance and hardware compatibility—combined through Copeland's method for balanced recommendations.

if M1 is a large model requiring extensive time—such as 10 minutes for a forward pass on certain IoT hardware—it may be impractical for real-time applications, making M2 the more effective choice by balancing accuracy with practical performance constraints. Furthermore, deep learning compilers like ONNX could also influence model rankings by optimizing specific models for certain hardware types [21], [22]. Nevertheless, performance remains a major challenge for engineers when reusing PTMs, as highlighted by [23], and this issue is particularly critical in IoT contexts where resource limitations amplify its impact. As one participant in that study noted, "*Performance bugs are silent bugs. You will never know what happened until it goes to deployment.*" This emphasizes the need for a structured approach to assess a model's overall performance on IoT devices.

To further explore the identified research gaps, we propose the following research questions:

RQ1: How can different parameters be weighted to tailor recommendations? Specifically, can the solution be tuned to prioritize speed, cost-efficiency, or energy efficiency?
RQ2: How do model optimization techniques, such as quantization [24] and distillation [25], impact model rankings in the context of hardware-aware recommendations?

B. *Lack of Ground-Truth rankings for IoT devices*

A fundamental barrier to progress in the earlier gap is the **lack of empirical data comparing the performance of different models across various IoT devices**. This gap prevents researchers from finding patterns or correlations in model parameters that could provide a quick, reliable way to assess hardware compatibility. Previous benchmarking works [19], [20], only consider one model and do not compare or rank them. The lack of this data restricts our ability to develop predictive models to recognize patterns in rankings. Model Spider [15] includes a method to approximate ground truth by combining multiple approaches using Copeland's rank aggregation [26], [27]. However, since there are few heuristic ranking approaches in the hardware space, this strategy is currently infeasible. This lack of comprehensive performance data across the hardware landscape highlights the need for dedicated datasets and benchmarks to enable informed and hardware-aware model recommendations.

The following research questions will guide our analysis after data collection, aiming to uncover trends and correlations:

RQ3: Can specific trends be identified between hardware specifications and model performance?
RQ4: Is there a significant correlation between hardware resources and model characteristics, such as parameters, architecture, or problem type?

IV. Research Agenda

This section outlines a research agenda to enhance model recommendations for IoT applications. We introduce two approaches to make Model Spider hardware-aware (§IV-A) and propose methods for dataset creation (§IV-B) , improving adaptability across IoT environments. We then discuss extending our approach to broader deep learning systems and complex model reuse scenarios (§IV-C).

A. *Modifications to Model Spider*

To address the first gap identified, we propose two approaches to make the existing Model Spider framework hardware-aware. These build upon the existing framework, leveraging its robustness while enhancing its capability to account for hardware constraints with minimal modifications.

1) ***Model Spider Fusion – Augmenting Task Tokens****:* As shown in Figure 2, in this approach we propose to introduce a separate Extractor Ψ_h which would encode the hardware specification for any given hardware. Its inputs comprise hardware model, CPU specifications, RAM size, Memory Size, etc. We would append these to the existing Task Tokens $\mu(T_d, T_h)$. This modification enables the similarity block to

Table II: Categorizing Metrics into Groups

	Metric	Description	Sample
Hardware	Execution Time	Time for one forward pass	90 ms
	Memory Utilization	Memory used during execution	3 GB
	Power Consumption	Total energy consumed	5 W
	CPU Temperature	Temperature of the CPU	75°C
	Carbon Footprint [20]	Environmental impact	10.9 units
Model	Accuracy	% correct predictions	92%
	Precision	identified / predicted positives	88%
	Recall (Sensitivity)	identified / actual positives	85%
	F1 Score	Harmonic mean of last two	86%

learn correlations between model performance and the specific hardware in use, thereby enhancing the hardware-awareness.

*2) **Model Spider Shadow – New Hardware task**:* This approach builds on Model Spider's capability to recommend the best model for a downstream task, expanding it to include hardware requirements. By redefining a "downstream task" to encompass specific hardware, we replicate the framework with a hardware extractor Ψ_h that encodes hardware-specific features. This results in two ranking systems: a task-based selector for model suitability to tasks and a hardware-based selector for compatibility with hardware. We combine these rankings using Copeland's method [26], [27], yielding a balanced recommendation that accounts for both task and hardware. This framework can be further extended to include energy-based or cost-based selectors, enabling tailored recommendations.

B. Dataset Creation

Defining Metrics: We propose to use metrics laid out in Sayeedi *et al.* [20], categorized into two groups. These groups can either be combined to establish the ground truth in *Model Spider Fusion* (§IV-A1) or used independently to rank models in *Model Spider Shadow* (§IV-A2).

a) Methodology to collect data: To construct the ground truth data, we consider multiple PTMs and datasets. Each PTM is fine-tuned on all datasets. A methodology is then defined for collecting data from these (Model, Dataset) pairs, ensuring that experiments do not interfere with previous runs and that sensitive metrics are accurately reported.

b) Creating rankings from collected data: We propose a tunable ranking system that adjusts to specific requirements, ranking models based on selected metrics such as best execution time, best accuracy, or best energy consumption. To aggregate these metrics, we employ the weighted Copeland's rank-choice voting method. Each metric rank is assigned a weight w_i in $[0, 1]$, with $\sum w_i = 1$. The objective function (1) maximizes each model's combined score, incorporating both model performance and hardware efficiency [28], [29]. Here, $f(\alpha)$ denotes model performance for configuration α (*e.g.,* based on accuracy), and $\mathrm{HW}_i(\alpha)$ represents the i-th hardware metric (*e.g.,* execution time or power consumption). Each hardware metric is normalized by a threshold T_i and scaled by adjustable w_i to reflect the relevant importance.

$$\max_{\alpha \in A} f(\alpha) \cdot \sum_i \left(\frac{\mathrm{HW}_i(\alpha)}{T_i} \right)^{w_i} \tag{1}$$

C. Future Directions

We propose to extend the model recommender to a broader deep learning system (§IV-C1) and complex reuse (§IV-C2).

1) Extending to Broader Deep Learning Systems: Our proposed approach would extend Model Spider, which handles only basic tasks (*e.g.,* classification) on edge devices. However, hardware constraints are a crucial engineering consideration not only on edge devices but across a range of deep learning systems, including distributed learning frameworks (*e.g.,* federated learning [30]). Additionally, in real-world applications, the interaction between data collection and model prediction components must be addressed [31], [32]. We propose broadening our research to include DL systems, enhancing model reuse and adaptability across diverse environments.

2) Extending to More Complex Model Reuse Scenarios: Our work, along with most prior ML research, has primarily addressed basic tasks, such as classification and regression. Although Model Spider has explored model reuse in generative models (*e.g.,* Large Language Model) [15], broader applications of DL model reuse remain largely unexplored. In computer vision, for instance, classification serves as a foundation for more complex tasks, like object detection and segmentation, which build upon classification models as backbones [33]. These downstream tasks require fine-tuning the model and adding task-specific heads or decoders for complex objectives. Expanding our approach to support such reuse scenarios is essential.

Algorithm 1 Performance Evaluation Procedure for (Fine-Tuned Model, Dataset) Pairs

```
Input: Datasets and Models Fine-tuned on those
Output: Metrics contained in Table II
Function stabilize():
  Wait until:
    1) CPU and RAM utilization at nominal level
    2) Temperature within specified range
Function benchmark(Datasets, Models):
  Call stabilize()

  while dataset, model in pairs(Datasets, Models)
    for batch_size in range(1, 101)
      Sample batch with current batch_size
      Perform forward pass
      Measure performance metrics
    Set batch_size = 32
    for each batch in dataset
      Sample batch
      Perform forward pass
      Measure performance metrics
    Call stabilize()
```

V. Conclusion

In conclusion, this paper identifies and addresses critical gaps in PTM recommendation for IoT by proposing hardware-aware enhancements to the Model Spider framework. This work establishes a foundation for further research in optimizing PTM recommendations across diverse IoT environments.

REFERENCES

[1] K. Shah, C. Sheth, and N. Doshi, "A survey on IoT-based smart cars, their functionalities and challenges," vol. 210, pp. 295–300.

[2] K. S. Pratyush Reddy, Y. M. Roopa, K. Rajeev L.N., and N. S. Nandan, "IoT based smart agriculture using machine learning," in *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, pp. 130–134.

[3] P. J. Rani, J. Bakthakumar, B. P. Kumaar, U. P. Kumaar, and S. Kumar, "Voice controlled home automation system using natural language processing (NLP) and internet of things (IoT)," in *2017 Third International Conference on Science Technology Engineering & Management (ICONSTEM)*, pp. 368–373.

[4] M. Almeida, S. Laskaridis, A. Mehrotra, L. Dudziak, I. Leontiadis, and N. D. Lane, "Smart at what cost? characterising mobile deep neural networks in the wild," in *Proceedings of the 21st ACM Internet Measurement Conference*, ser. IMC '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 658–672. [Online]. Available: https://doi.org/10.1145/3487552.3487863

[5] W. Jiang, N. Synovic, M. Hyatt, T. R. Schorlemmer, R. Sethi, Y.-H. Lu, G. K. Thiruvathukal, and J. C. Davis, "An empirical study of pre-trained model reuse in the hugging face deep learning model registry," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pp. 2463–2475, ISSN: 1558-1225.

[6] N. K. Gopalakrishna, D. Anandayuvaraj, A. Detti, F. L. Bland, S. Rahaman, and J. C. Davis, ""if security is required": engineering and security practices for machine learning-based IoT devices," in *Proceedings of the 4th International Workshop on Software Engineering Research and Practice for the IoT*. ACM, pp. 1–8.

[7] A. T. Tran, C. V. Nguyen, and T. Hassner, "Transferability and hardness of supervised classification tasks," pp. 1395–1405.

[8] Y. Bao, Y. Li, S.-L. Huang, L. Zhang, L. Zheng, A. Zamir, and L. Guibas, "An information-theoretic approach to transferability in task transfer learning," in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 2309–2313, ISSN: 2381-8549.

[9] C. Nguyen, T. Hassner, M. Seeger, and C. Archambeau, "LEEP: A new measure to evaluate transferability of learned representations," in *Proceedings of the 37th International Conference on Machine Learning*. PMLR, pp. 7294–7305, ISSN: 2640-3498.

[10] Y. Li, X. Jia, R. Sang, Y. Zhu, B. Green, L. Wang, and B. Gong, "Ranking neural checkpoints," pp. 2663–2673.

[11] K. You, Y. Liu, J. Wang, and M. Long, "LogME: Practical assessment of pre-trained models for transfer learning," in *Proceedings of the 38th International Conference on Machine Learning*. PMLR, pp. 12 133–12 143, ISSN: 2640-3498.

[12] N. Ding, X. Chen, T. Levinboim, S. Changpinyo, and R. Soricut, "PACTran: PAC-bayesian metrics for estimating the transferability of pretrained models to classification tasks," in *Computer Vision – ECCV 2022*. Springer, Cham, pp. 252–268, ISSN: 1611-3349.

[13] M. Pándy, A. Agostinelli, J. Uijlings, V. Ferrari, and T. Mensink, "Transferability estimation using bhattacharyya class separability," pp. 9172–9182.

[14] A. Deshpande, A. Achille, A. Ravichandran, H. Li, L. Zancato, C. Fowlkes, R. Bhotika, S. Soatto, and P. Perona, "A linearized framework and a new benchmark for model selection for fine-tuning."

[15] Y.-K. Zhang, T.-J. Huang, Y.-X. Ding, D.-C. Zhan, and H.-J. Ye, "Model spider: Learning to rank pre-trained models efficiently," vol. 36, pp. 13 692–13 719.

[16] F. Meng, W. Shao, Z. Peng, C. Jiang, K. Zhang, Y. Qiao, and P. Luo, "Foundation model is efficient multimodal multitask model selector," vol. 36, pp. 33 065–33 094.

[17] J. Bai, S. Wu, J. Song, J. Zhao, and G. Chen, "Pre-trained model recommendation for downstream fine-tuning," version Number: 1.

[18] Y. Tan, Y. Li, and S.-L. Huang, "OTCE: A transferability metric for cross-domain cross-task representations," pp. 15 779–15 788.

[19] P. M. Sánchez Sánchez, J. M. Jorquera Valero, A. Huertas Celdrán, G. Bovet, M. Gil Pérez, and G. Martínez Pérez, "LwHBench: A low-level hardware component benchmark and dataset for single board computers," vol. 22, p. 100764.

[20] M. F. A. Sayeedi, J. F. Deepti, A. M. I. M. Osmani, T. Rahman, S. S. Islam, and M. M. Islam, "A comparative analysis for optimizing machine learning model deployment in IoT devices," vol. 14, no. 13, p. 5459, number: 13 Publisher: Multidisciplinary Digital Publishing Institute.

[21] J. C. Davis, P. Jajal, W. Jiang, T. R. Schorlemmer, N. Synovic, and G. K. Thiruvathukal, "Reusing deep learning models: Challenges and directions in software engineering," in *2023 IEEE John Vincent Atanasoff International Symposium on Modern Computing (JVA)*, pp. 17–30.

[22] P. Jajal, W. Jiang, A. Tewari, E. Kocinare, J. Woo, A. Sarraf, Y.-H. Lu, G. K. Thiruvathukal, and J. C. Davis, "Interoperability in deep learning: A user survey and failure analysis of onnx model converters," in *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2024, pp. 1466–1478.

[23] W. Jiang, V. Banna, N. Vivek, A. Goel, N. Synovic, G. K. Thiruvathukal, and J. C. Davis, "Challenges and practices of deep learning model reengineering: A case study on computer vision."

[24] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," pp. 2704–2713.

[25] X. Liu, X. Wang, and S. Matwin, "Improving the interpretability of deep neural networks with knowledge distillation," in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 905–912, ISSN: 2375-9259.

[26] A. Copeland, "A reasonable social welfare function," in *Seminar on Applications of Mathematics to Social Sciences*, 1951, mimeographed Notes.

[27] D. G. Saari and V. R. Merlin, "The copeland method: I.: Relationships and the dictionary," vol. 8, no. 1, pp. 51–76.

[28] H. Benmeziane, K. E. Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang, "A comprehensive survey on hardware-aware neural architecture search," *ArXiv*, vol. abs/2101.09336, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:231699126

[29] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, " MnasNet: Platform-Aware Neural Architecture Search for Mobile ," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2019. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2019.00293

[30] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 1273–1282, ISSN: 2640-3498.

[31] A. Makhshari and A. Mesbah, "Iot bugs and development challenges," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 460–472.

[32] J. Garcia, Y. Feng, J. Shen, S. Almanee, Y. Xia, Chen, and Q. Alfred, "A comprehensive study of autonomous vehicle bugs," in *Proceedings of the ACM/IEEE 42nd international conference on software engineering*, 2020, pp. 385–396.

[33] V. Banna, A. Chinnakotla, Z. Yan, A. Vegesana, N. Vivek, K. Krishnappa, W. Jiang, Y.-H. Lu, G. K. Thiruvathukal, and J. C. Davis, "An experience report on machine learning reproducibility: Guidance for practitioners and tensorflow model garden contributors," *arXiv preprint arXiv:2107.00821*, 2021.