Federated Reinforcement Learning for Optimizing the Power Efficiency of Edge Devices

Benedikt Dietrich, Rasmus Müller-Both, Heba Khdr, Jörg Henkel
Chair for Embedded Systems, Karlsruhe Institute of Technology (KIT), Karlsuhe, Germany
benedikt.dietrich@kit.edu, rasmus.mueller@gmail.com, heba.khdr@kit.edu, henkel@kit.edu

***Abstract*—Reinforcement learning (RL) holds great promise for adaptively optimizing microprocessor performance under power constraints. It allows for online learning of application characteristics at runtime and enables adjustment to varying system dynamics such as changes in the workload, user preferences or ambient conditions. However, online policy optimization remains resource-intensive, with high computational demand and requiring many samples to converge, making it challenging to deploy to edge devices. In this work, we overcome both of these obstacles and present federated power control using dynamic voltage and frequency scaling (DVFS). Our technique leverages federated RL and enables multiple independent power controllers running on separate devices to collaboratively train a shared DVFS policy, consolidating experience from a multitude of different applications, while ensuring that no privacy-sensitive information leaves the devices. This leads to faster convergence and to increased robustness of the learned policies. We show that our federated power control achieves 57 % average performance improvements over a policy that is only trained on local data. Compared to a state-of-the-art collaborative power control, our technique leads to 22 % better performance on average for the running applications under the same power constraint.**

***Index Terms*—power control, DVFS, federated learning, reinforcement learning**

I. Introduction

Edge computing pushes processing power closer to the data sources, steadily increasing the computational load placed on edge devices that fuse information from connected endpoints through stream processing and machine learning (ML). At the same time, edge devices must operate under low-power conditions [1]. Dynamic voltage and frequency scaling (DVFS) is an indispensable lever in this trade-off, negotiating between application performance gains and increasing power consumption through the adjustment of the processor frequency at runtime[1]. The optimal DVFS policy is highly application-dependent [2], [3], as applications respond differently to frequency changes throughout their execution. The frequency controllers implemented in modern operating systems mostly ignore these application-specific characteristics and therefore, miss out on the potential for significant increases in power efficiency [4]. Instead, the state of the art relies on ML models developed at design time that predict the power/performance behavior of applications to guide runtime decision-making [3], [4].

[1]Modern processors expose a wide range of discrete frequency levels that can be changed in a matter of microseconds. For each frequency level there is a corresponding voltage that will be automatically applied when the frequency is set. Hence, the term voltage/frequency (V/f) level is commonly used.

Given the generalization capabilities of ML models, e.g., neural networks (NNs), these techniques even work well for unseen applications. However, their performance may degenerate for workloads that are substantially different from those for which they have been trained [2], [5]. At the same time, the variety of potential applications, device configurations and user preferences makes it challenging to derive a power control at design time that performs optimally in every potential runtime situation [5]. Therefore, adaptive power control using reinforcement learning (RL) has attracted the attention of researchers for its ability to learn the optimal frequency at runtime without requiring an elaborate design-time training process [6]. However, the intrinsic complexity of RL makes it challenging to deploy such techniques to edge devices: While the power controller should negotiate stringent performance requirements and power budgets in a near-optimal way, real-world edge devices cannot afford to spend a considerable fraction of their limited resources on deriving the control decisions themselves. Therefore, research has been conducted on how to reduce the overhead of training RL-based power controllers.

Transfer learning has been proposed to transfer knowledge obtained from a previous task (e.g., an application or device configuration) to a new one, thereby reducing the cost of adapting to the changing system dynamics. However, existing literature primarily focuses on transferring knowledge between consecutive versions of the same control policy on a single device [2], [5]. In contrast, few studies have proposed *collaborative learning* across multiple devices. We believe *multi-device learning bears greater potential for knowledge transfer, because policies can be trained on data from a broad range of applications simultaneously, rather than sequentially*. In [7] and [8], devices communicate performance and power traces with a server or with one another, allowing a power control policy to be trained using this shared information. In both cases, raw data samples leave the devices, granting malicious participants direct access to power and execution traces that can be used to recover sensitive information such as device activity or user data and cryptographic keys [9], [10]. The work in [11] presents privacy-preserving collaborative power control where only the policy, not raw traces, is shared with a central server. However, this work relies on tabular RL with limited representational capabilities compared to NN-based approaches[12], resulting in inferior performance of the DVFS policies [13]. Recent works have therefore shifted to NNs to

978-3-9826741-0-0/DATE25/© 2025 EDAA

represent the DVFS policy, e.g., [5], [8], [13]. *Yet, a method to collaboratively train an NN-based power control that does not leak sensitive information has not been presented so far.*

To address this gap in the existing literature, we explore the use of *federated learning (FL)* [14]–[16] to enhance processor power control. In FL, multiple independent devices collaborate to train a shared NN while keeping the training data local to each device. Examples such as voice assistants [17] and typing recommendation [18] on mobile devices demonstrate the practical applications of FL. Despite its prominence in the mobile field and the enhanced privacy it offers, FL has not yet been adopted in processor resource management, such as power control. State-of-the-art resource management techniques have increasingly integrated advanced ML models to enhance predictions and control decisions. FL can enable the deployment of even more sophisticated power controllers to edge devices; controllers that would be impractical to train with the limited computational power and workload diversity of a single device, leading to further improvements in power efficiency. In summary, we provide the following **novel contributions**.

- We introduce federated reinforcement learning for processor power control through neural-network-based DVFS.
- Our technique allows collaborative learning between several edge devices that run diverse workloads, without leaking sensitive information.
- We demonstrate on real edge devices that our federated power control significantly outperforms locally trained control policies.

II. Related Work

Performance optimization for processors considering power/energy/thermal trade-offs has been widely studied [19]. ML has become the leading method for this optimization problem due to its ability to model complex system dynamics and generalize to unseen applications. For example, an NN trained at design time can predict the impact of DVFS on application performance and power [3] or predict workload classes [4] to guide runtime frequency selection.

Although these design-time models generalize to unseen applications, it is practically infeasible to train a suitable model for all potential runtime situations [5]. Thus, online learning techniques have been developed that adapt policy parameters at runtime. For example, recursive least-squares estimators [20] can learn to predict the performance and power consumption of applications without design-time training. Alternatively, RL can be used to directly adjust processor frequency based on current performance counter and sensor measurements [5], [6], [21].

While effective, online RL remains notoriously resource-intensive, requiring significant computational power and many training examples. Therefore, researchers have explored transfer learning, which reuses knowledge from prior situations to speed-up adaptation to new ones. Three works adapt table-based RL to support transfer learning. In [22], workload changes initiate rescaling of the table values, while the work in [23] infers the values of unknown or under-explored states and actions by leveraging knowledge from neighboring states

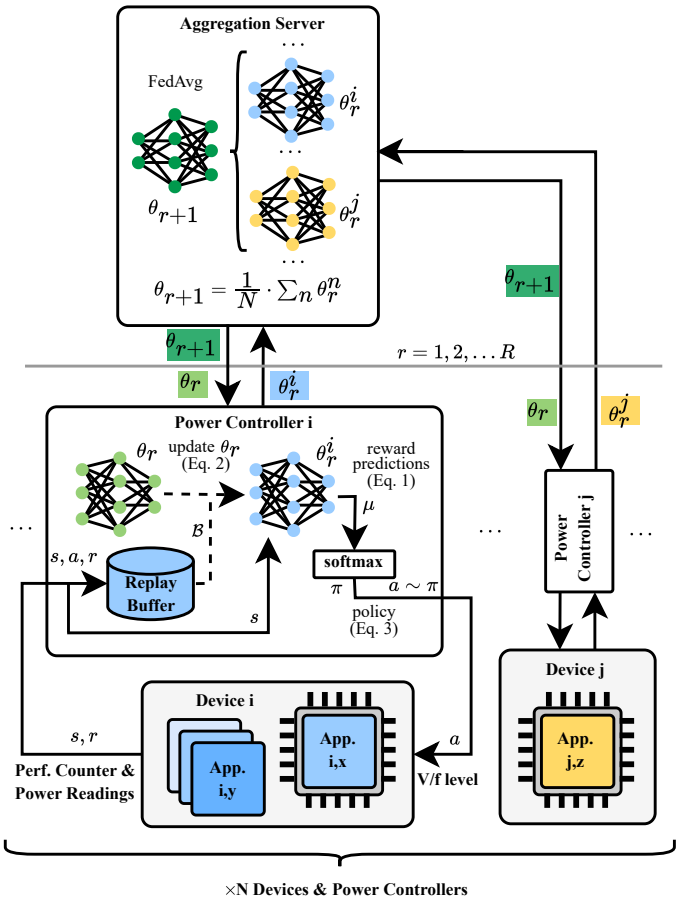

Fig. 1. Federated power control with N devices and a central aggregation server. All N power controllers are implemented as RL agents and alternate between observing the processor state and setting an approriate V/f level. Given these interactions with the processor, they update their local policy. At the end of each training round, the server joins the individual policies into a global version using federated averaging [14]. Afterward, it distributes the global policy back to the devices, initiating the next optimization round.

(actions). Selective knowledge transfer in [2] retains the previous application's value table as an expert policy and gradually shifts to one optimized for the current task. Similarly, the work in [24] triggers knowledge transfer when user preferences change, which leads to the agent re-exploring the new environment. The deep RL approach in [5] prioritizes samples with extreme rewards that mark changes in the environment settings during optimization.

Few works explore collaborative transfer across device boundaries. In [7], local power controllers operate on-device, while a shared policy is trained on a remote server using experience provided by all devices. The work in [8] studies direct collaboration between devices aimed at minimizing the energy-delay product. It uses a deep RL policy on each device trained with samples from the local processor as well as experience shared by the other devices. In both works, unprocessed performance counter and power traces leave the device, leaking sensitive information that allows malicious participants to infer device/user activity or even to exploit this side-channel leakage to extract further information [9], [10]. To counteract this, the authors of [11], [25] investigate collaborative resource man-

agement that preserves privacy. Their objective is to minimize energy under performance constraints. In [11], each device maintains a local and a global policy. The local policy is trained using the experience of the associated device. A central server integrates the local policies of all devices into a global one. Compared to [7], [8], only the policy is shared, not the raw samples. The work in [25] follows a similar pattern, but with direct communication between devices. Both works use tabular RL, which only supports small solution spaces as there is no generalization across states and features need to be discretized [12]. NNs provide a scalable alternative, supporting continuous feature spaces and thus are considered state-of-the-art for implementing DVFS policies [5], [8], [13] However, *no prior work has investigated privacy-preserving collaborative training of neural power controllers*.

III. Federated Power Control

We focus on a microprocessor with K discrete V/f levels. The workload consists of a sequence of single-threaded applications. Both the applications and their execution order are unknown at design time. The goal of this work is to use DVFS to maximize the performance, in terms of throughput or latency, of the application currently running while adhering to a configurable power constraint P_{crit} throughout its execution.

A power controller running on the processor performs DVFS to achieve the objective (see Fig. 1). We rely on an NN to implement the control policy. The power controller alternates between observing the processor's state, composed of performance counters and power measurements, and selecting a suitable frequency level. Using feedback acquired from the interaction with the processor, it can identify which frequencies led to violations under a similar workload in the past and select the one where the power consumption stays just below the threshold. The NN is updated as a regression model using gradient descent.

To achieve collaborative policy optimization, we use FL between N power controllers and one central aggregation server, following the federated averaging algorithm [14] (see Fig. 1). This algorithm is carried out over R rounds and each round starts with the server sending the current version of the shared model to the devices. On the devices, the power controllers optimize the model separately based on their interaction with the associated processor. Afterward, they send the locally optimized models back to the server, where the local models are combined to the next global version by averaging their parameters.

Compared to local training, power controllers benefit from collaborative exploration, resulting in faster and broader coverage of the solution space [26]. This allows consolidating knowledge from many different applications, not restricted to those running on each associated device. Furthermore, FL enables faster training due to the availability of more training examples and increased computational power [26]. The only additional requirement is a communication module in each device to transfer the local model to, and receive the global model from, the server. Unlike server-side training [7], devices do not share raw performance counters or power traces.

A. Local Power Controller

The power controller in each device is an RL agent [12]. Each frequency selection yields a reward signal reflecting our objective of maximizing power efficiency for the current workload. In general RL, the agent tries to learn the sequence of actions that maximizes the reward it accumulates over time. For power-constrained performance optimization, it is sufficient to identify the optimal frequency for the current state [21] since the effect of frequency selection is immediately observable in the power consumption in the next timestep, assuming that we neglect the impact of power consumption on temperature and temperature on leakage power[2]. Algorithm 1 describes the learning procedure implemented in each power controller. To identify the optimal frequency, the agent maintains an NN with parameters θ that estimates the expected reward for every action (V/f level) in a given state (see Eq. (1)).

Algorithm 1 Reinforcement Learning with Policy Network

```
Input: replay buffer, policy network θ, softmax temp. τ
for t = 1, 2...T do
    observe system state s_t
    infer expected reward μ(a, s, θ) for each action
    derive policy: π(a|s, θ, τ) (see Eq. (3))
    sample next action: a_t ~ π(a|s, θ, τ)
    execute the action a_t and observe the reward r_t
    store (s_t, a_t, r_t) in the replay buffer
    apply exponential decay to τ
    if t mod H = 0 then
        sample batch B from replay buffer
        update θ through backpropagation of loss L (see Eq. (2))
    end if
end for
```

$$\mu(s, a, \theta) \approx \mathbb{E}[r(s, a)] \tag{1}$$

The NN's parameters are continuously updated to minimize the prediction error over a batch of state-action-reward tuples $\mathcal{B} = \{(s, a, r)\}_{C_B}$ of size C_B sampled from a replay buffer [28] (see Eq. (2)).

$$\mathcal{L} = \frac{1}{|\mathcal{B}|} \cdot \sum_{(s,a,r)\in\mathcal{B}} \|\mu(s, a, \theta) - r(a, s)\|_2 \tag{2}$$

The buffer stores the C most recent state-action-reward samples from the interaction with the processor. The control policy is derived from the reward predictions using the softmax function Eq. (3) [12].

$$\pi(a|s, \theta, \tau) = \frac{exp(\frac{\mu(s,a,\theta)}{\tau})}{\sum_{a'} exp(\frac{\mu(s,a',\theta)}{\tau})} \tag{3}$$

At each timestep, the agent samples the next V/f level from the distribution $\pi(a|s, \theta, \tau)$. The temperature parameter τ controls the entropy of the distribution and decreases exponentially over time. Initially, the distribution is close to uniform and gradually becomes unimodal, with a peak at the optimal frequency for the workload. This leads to the agent exploring the solution

[2] A scenario in which the agent is concerned about maximizing the immediate reward for a given state is also referred to as a *contextual bandit* problem [27].

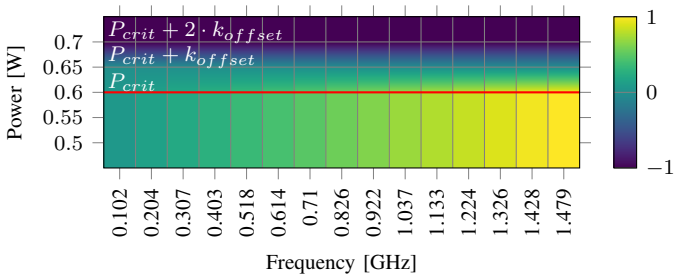

Fig. 2. Distribution of the reward signal for P_{crit} = 0.6 W and k_{offset} = 0.05 W. The frequency levels are those of the processor used in Section IV.

space in the early stages of training, learning about the reward distribution across frequency levels and, over time, exploiting this knowledge by selecting the one with the highest reward.

Reward: The reward signal trades off two opposing objectives: maximizing the performance of the application and adhering to the power constraint. We use the operating frequency normalized by the maximum frequency f_{max} as a surrogate metric for performance when the power consumption is below the threshold P_{crit} (see Eq. (4)). If it exceeds the threshold, the reward gradually decreases with increasing power consumption, becomes negative after $P_{crit} + k_{offset}$ and reaches the minimum of -1 at $P_{crit} + 2 \cdot k_{offset}$. Alternatively, the power constraint could be enforced with a hard cut, yielding a negative penalty in case of a violation. However, the behavior of the system is unlikely to deteriorate at the slightest overshoot. Therefore, we argue that our approach more closely implements power-efficient operation.

$$r_t = \begin{cases} \frac{f_{t+1}}{f_{max}} & P_{t+1} \leq P_{crit} \\ \frac{f_{t+1}}{f_{max}} \cdot \frac{P_{crit}+k_{offset}-P_{t+1}}{k_{offset}} & P_{t+1} \leq P_{crit} + k_{offset} \\ \frac{P_{crit}+k_{offset}-P_{t+1}}{k_{offset}} & P_{t+1} \leq P_{crit} + 2 \cdot k_{offset} \\ -1 & \text{otherwise} \end{cases} \tag{4}$$

State: The state of the local agent consists of the current frequency f, the instructions per cycle (IPC), the miss rate and the misses per kilo-instruction (MPKI) on the last-level cache, and the current power consumption.

$$s = (\, f,\, P,\, ipc,\, mr,\, mpki \,)$$

The IPC quantify the application's ability to utilize available hardware resources. A high value indicates significant instruction-level parallelism, with multiple functional units operating concurrently. This typically leads to higher performance, but also increased power consumption. The cache misses allow to further classify an application (or application execution phase) as memory-/compute-intensive. These application metrics provide insight on the power consumption in the next timestep, allowing the agent to proactively adjust the frequency according to the current workload.

Action: The agent's action space corresponds to the entire frequency range of the processor, enabling fine-grained DVFS.

$$\mathcal{A} = \{\, V/f_1 \ldots V/f_K \,\}$$

B. Federated Policy Optimization

To achieve collaborative optimization of the power control policy, we implement a horizontal FL system with N homogeneous client devices. This means that each device runs a local agent with identical state and action spaces [26]. The FL procedure is described in Algorithm 2. We consider a setting where each client participates in all R rounds. At the start of each round, the devices receive the parameters of the global model θ_r. Then, each device performs local updates of the model parameters based on samples from their own replay buffer. The buffer is maintained across all rounds and its content never leaves the device. At the end of the round, each of the N devices sends an updated local version θ_r^n of the NN to the server. The aggregation on the server is synchronous, i.e., the server waits for all devices to send their local models before computing the updated global model for the next round θ_{r+1}. It is unweighted, giving the same importance to each client.

Algorithm 2 Federated Averaging [14]

1: **Initialize:** global model θ_1
2: **for** round $r = 1, 2, \ldots R$ **do**
3: server broadcasts global model θ_r to all clients
4: **for** each client n in parallel **do**
5: perform multiple local updates to θ_r (see Algorithm 1).
6: send updated local model θ_r^n to the server
7: **end for**
8: server computes updated global model: $\theta_{r+1} = \frac{1}{N} \sum \theta_r^n$
9: **end for**

C. Configuration and Parameters

Table I lists the parameters used in our technique.

TABLE I
PARAMETERS OF OUR FEDERATED POWER CONTROL

Parameter	Value	Parameter	Value
Learning Rate (α)	0.005	#Hidden Layers	1
Max. Temp. (τ_{max})	0.9	#Neurons/Layer	32
Temp. Decay (τ_{decay})	0.0005	Pow. Constr. [W] (P_{crit})	0.6
Min. Temp. (τ_{min})	0.01	Pow. Offs. [W] (k_{offset})	0.05
Replay Capacity (C)	4,000	Ctrl. Intv. [ms] (Δ_{DVFS})	500
Batch Size (C_B)	128	#Rounds (R)	100
Optim. Intv. (H)	20	#Steps/Round (T)	100

The NN has a single layer and uses the *ReLU* activation function. The optimizer is *Adam* [29], and we use the *Huber* loss function, which penalizes small errors quadratically and larger errors linearly. Federated training is conducted over 100 rounds. In each round, the power controllers perform 100 steps in their local environment. They update their policy every 20th step, resulting in five local updates before sending the policy back to the server.

IV. EVALUATION

We conduct our experiments using two NVIDIA Jetson Nano boards [30] as representative edge devices. The system can be naturally extended to use more than two devices. The processors contain four ARM® Cortex®-A57 cores [31] with a shared clock signal. It supports 15 frequency levels, ranging

from 102 MHz to 1479 MHz. Our evaluation workload comprises twelve single-threaded applications from the *SPLASH-2* benchmark suite [32].

A. Comparison with a Local-Only Power Control

TABLE II
DISJUNCT TRAINING SET: APPLICATIONS PER DEVICE

Scenario	Device A		Device B	
1	fft	lu	raytrace	volrend
2	water-ns	water-sp	ocean	radix
3	fmm	radiosity	barnes	cholesky

Ideally, power controllers are trained using a broad set of applications with the execution order sampled uniformly at random. However, this approach does not adequately reflect many real-world scenarios, where applications are not uniformly distributed over time, nor do all devices execute the same applications. Devices often execute a few frequent workloads while occasionally encountering new ones at unpredictable times. When the assumption of uniformity is violated, controllers can struggle to maintain their training performance when faced with new applications [2], resulting in long adaptation times. This defeats the initial motivation for transitioning to online policy learning. Given that the non-uniform distribution of applications is intrinsic to many real-world settings, we argue that FL can improve the overall performance compared to locally trained policies by consolidating experience from many different devices. In the following, we present the results of our experiments, validating this hypothesis.

To simulate realistic conditions, we assign different applications to different devices. During training, each device is exposed to only two applications, limiting the workload diversity experienced by each device. For evaluation, all twelve applications are considered. We define three different scenarios based on the pairs of training applications used on each device (see Table II). For each scenario, we train one shared policy with our federated power control algorithm using data from both devices. In addition, we train two local-only policies, each of them constrained to one device, with no collaboration. After each training round, we evaluate the policies on each device using one of the twelve evaluation applications. During evaluation, the policies are not updated and the agents consistently exploit the action with the highest predicted reward. This approach provides an accurate estimate of performance on unseen applications.

The results are reported in Fig. 3. Each row corresponds to one evaluation scenario in Table II; the left column is for the local-only setting, and the right column is for our federated technique. We report the reward from the evaluation process for both devices in the local-only setting. In the federated setting, the reward is similar on both devices. We see that the reward curves of the federated policy are almost constant at just below 0.5 starting from early rounds. They are similar for different scenarios. In contrast, the local-only policies struggle to maintain training performance during evaluation. This is most pronounced for device B in the second scenario (L2),

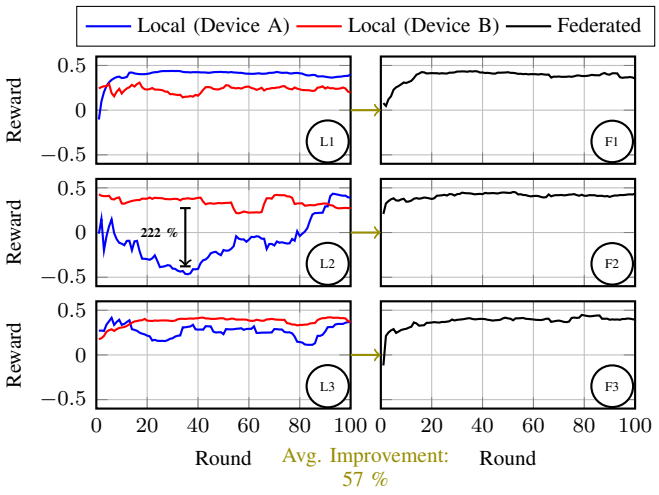

Fig. 3. Reward during the evaluation of the local-only and the federated policies for each scenario in Table II.

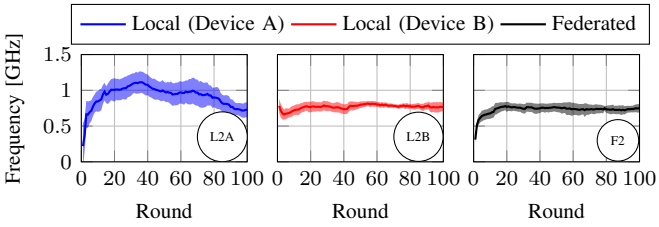

Fig. 4. Average frequency selection under the local-only and federated policies during the evaluation for the second scenario in Table II. The standard deviation for the selected frequencies is shown as a shaded area around the mean.

which uses the applications *ocean* and *radix* for training. Performance starts to drop right from the start with a minimum of -0.5 and the performance of device A is up to 222 % lower than of device B. The reward remains negative until evaluation round 80, indicating that the local policy frequently violates the power constraint (see Eq. (4)). The reason becomes apparent in Fig. 4, which shows the average frequency selected under each policy. The local-only policy on device A selects frequencies higher than the one on device B and the federated policy. In the two other scenarios, the reward curves are less dramatic, but the average reward of the local-only policies still falls short by 57 % compared to the federated counterpart. Interestingly, in each of the three scenarios, there is always one local-only policy that stands out negatively. This shows that *the federated system improves overall performance and robustness even with only two training applications available to each local device*. In this unbalanced setting, the benefits of FL stand out: Given the experience of peers, a stable policy can still be learned.

B. Comparsion with the State of the Art

To the best of our knowledge, no prior work has employed multi-device collaborative training of power controllers aimed at optimizing performance under a power constraint. In the absence of a natural comparison, we employ a state-of-the-art RL-based power control for the single-device setting, based on *Profit* [6], and extend it with a server component to allow collaboration between devices, inspired by the method described in [11], which we refer to as *CollabPolicy*.

Profit uses a table-based approach to maximize performance under a power constraint. The state of the agent is composed of the current frequency, power consumption, IPC and MPKI. The reward signal is a piecewise function that corresponds to the current instructions per second (IPS) if the power consumption remains below the constraint P_{crit}, and otherwise to a penalty of $-5 \cdot |P_{crit} - P|$. Exploration follows an ϵ-greedy strategy with exponential decay and we set the minimum value to 0.01, and the learning rate to 0.1, a typical value for table-based approaches. To achieve multi-device collaboration, *CollabPolicy* follows a similar principle as our a technique, alternating between local optimization and central aggregation. Each local controller trains a value table and in addition, maintains a copy of the global policy, represented by tuples $(\pi^*(s), \overline{r}(s), n(s))$, where π^* denotes the best action, $\overline{r}$ the average reward and n the visit count for each state s. Similarly, the local policy can be derived from the value table. When the average reward for the current state is higher under the local policy, it will consult the local policy, otherwise, the global policy. After one round of local optimization, the devices send their local policies to the server, which updates the global policy per state by considering average rewards and visit counts, then returns it to the devices.

For the comparison with the state of the art, *Profit+CollabPolicy*, we follow the same principle as in the previous section. To fairly evaluate the performance of the two approaches that come with different reward signals (with different magnitudes), we report the average execution times (for latency-sensitive settings), IPS (for throughput-sensitive settings) and power consumption during the evaluation. The results are shown in Table III. Our federated power control

TABLE III
RESULTS OF THE COMPARISON WITH THE STATE OF THE ART FOR THE SCENARIOS IN TABLE II (AVERAGE OVER ALL THREE SCENARIOS)

Category	Ours	Profit+CollabPolicy
Exec. Time [s]	24.24 (↓ 20 %)	30.38
IPS [$\times 10^6$]	0.92 (↑ 17 %)	0.79
Power[W]	0.52 (↑ 9%)	0.47

improves the average performance: It reduces the execution time by 20 % and increases the IPS by 17 %. Both techniques keep the average power consumption below the constraint.

Finally, we compare the two techniques using additional training applications. We split the application set into two halves, assigning six applications to each device, so that every application used in the evaluation has been seen during training by one of the two devices. Figure 5 shows the results. The values correspond to the average for each application in all evaluation rounds. Again, both techniques maintain the average power consumption below the constraint. Our technique closes the margin to the critical threshold for most applications. At the same time, applications finish faster by 22 % on average and 53 % at the maximum. This performance improvement is also reflected in the average throughput, as our technique leads to an average (maximum) increase of 29 % (95 %) in the IPS. The substantial increase in overall performance, while

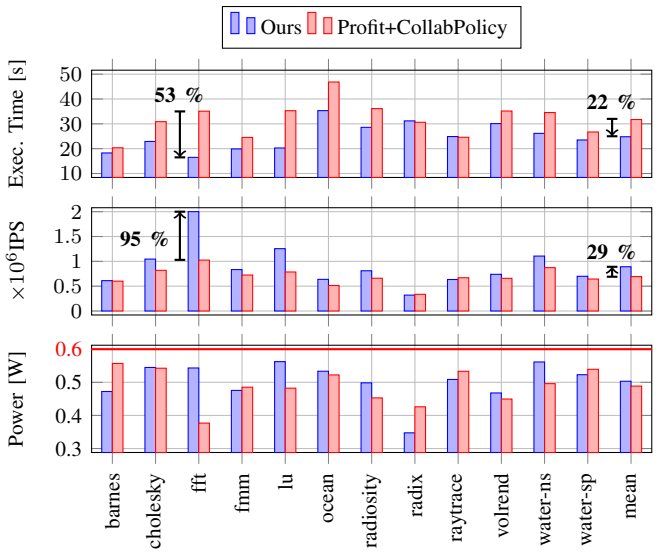

Fig. 5. Evaluation results for the comparison with the state of the art using six training applications per device. All twelve applications in the evaluation have been used for training on one of the two devices.

keeping power consumption below the threshold, demonstrates that our technique is more power-efficient than the state of the art. We conclude that the improvements are partly due to the increased expressiveness of a neural policy compared to table-based RL. Together with the results from the comparison with the local policy, this highlights the importance of a method to collaboratively train a neural power control using multiple devices.

C. Runtime Overhead

The average latency of our technique is 29 ms; the overhead relative to the control interval Δ_{DVFS} is 5.9 %. This overhead is already considered implicitly in the performance gains, as we use the execution time to compute them. Communication with the server amounts to 2.8 kB of data per transfer given the parameters in Table I, which can be considered negligible. The network must be stored on the devices and the replay buffer requires an additional 100 kB of storage.

V. CONCLUSION

In this paper, we explored the adoption of FL to enhance NN-based DVFS policies through collaboration between multiple devices in a privacy-preserving manner. We showed how FL enables the training of stable control policies, even with limited workload diversity on each device. Recent advances in resource management have continuously improved decision-making through the integration of more sophisticated ML methods. We anticipate that FL can further accelerate this trend by reducing the computational burden on each device while leveraging the collective experience of all devices. This paves the way for further increases in power efficiency and facilitates the deployment to real-world edge devices. Our work specifically focused on diversity in the applications across devices. Future research could expand our work by additionally considering varying objectives/user preferences and knowledge transfer between devices of different architecture.

REFERENCES

[1] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of Edge Computing and Deep Learning: A Comprehensive Survey," *IEEE Commun. Surv. Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.

[2] L. Chen, X. Li, F. Jiang, C. Li, and J. Xu, "Smart Knowledge Transfer-based Runtime Power Management," in *Design, Automation & Test in Europe Conference & Exhibition, DATE 2023*, IEEE, 2023, pp. 1–6.

[3] M. Rapp, M. B. Sikal, H. Khdr, and J. Henkel, "SmartBoost: Lightweight ML-Driven Boosting for Thermally-Constrained Many-Core Processors," in *58th ACM/IEEE Design Automation Conference, DAC, 2021*, IEEE, 2021, pp. 265–270.

[4] D. Huang, L. Costero, and D. Atienza, "Is the powersave governor really saving power?" In *24th IEEE International Symposium on Cluster, Cloud and Internet Computing, CCGrid 2024*, IEEE, 2024, pp. 273–283.

[5] S. Kim, K. Bin, S. Ha, K. Lee, and S. Chong, "zTT: learning-based DVFS with zero thermal throttling for mobile devices," in *MobiSys '21: The 19th Annual International Conference on Mobile Systems, Applications, and Services*, ACM, 2021, pp. 41–53.

[6] Z. Chen, D. Stamoulis, and D. Marculescu, "Profit: Priority and Power/Performance Optimization for Many-Core Systems," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 37, no. 10, pp. 2064–2075, 2018.

[7] G. Pan, B. C. Lai, S. Chen, and J. Jou, "A learning-on-cloud power management policy for smart devices," in *The IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2014*, IEEE, 2014, pp. 376–381.

[8] L. Chen, X. Li, and J. Xu, "Improve the Stability and Robustness of Power Management through Model-free Deep Reinforcement Learning," in *2022 Design, Automation & Test in Europe Conference & Exhibition, DATE*, IEEE, 2022, pp. 1371–1376.

[9] P. C. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference*, ser. Lecture Notes in Computer Science, vol. 1666, Springer, 1999, pp. 388–397.

[10] M. Lipp, A. Kogler, D. F. Oswald, *et al.*, "PLATYPUS: Software-based Power Side-Channel Attacks on x86," in *42nd IEEE Symposium on Security and Privacy, SP 2021*, IEEE, 2021, pp. 355–371.

[11] Z. Tian, Z. Wang, J. Xu, H. Li, P. Yang, and R. K. V. Maeda, "Collaborative Power Management Through Knowledge Sharing Among Multiple Devices," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 38, no. 7, pp. 1203–1215, 2019.

[12] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018, Second Edition.

[13] H. Khdr, M. E. Batur, K. Zhou, M. B. Sikal, and J. Henkel, "Multi-Agent Reinforcement Learning for Thermally-Restricted Performance Optimization on Manycores," in *Design, Automation & Test in Europe Conference & Exhibition, DATE 2024*, IEEE, 2024, pp. 1–6.

[14] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *20th International Conference on Artificial Intelligence and Statistics, AISTATS*, ser. Proceedings of Machine Learning Research, vol. 54, PMLR, 2017, pp. 1273–1282.

[15] P. Kairouz, H. B. McMahan, B. Avent, *et al.*, "Advances and Open Problems in Federated Learning," *Found. Trends Mach. Learn.*, vol. 14, no. 1-2, pp. 1–210, 2021.

[16] K. Pfeiffer, M. Rapp, R. Khalili, and J. Henkel, "Federated Learning for Computationally Constrained Heterogeneous Devices: A Survey," *ACM Comput. Surv.*, vol. 55, no. 14s, 334:1–334:27, 2023.

[17] K. Hao, *How Apple personalizes Siri without hoovering up your data*, en, Dec. 2019. [Online]. Available: https://www.technologyreview.com/2019/12/11/131629/apple-ai-personalizes-siri-federated-learning/ (visited on 09/10/2024).

[18] A. Hard, K. Rao, R. Mathews, *et al.*, "Federated Learning for Mobile Keyboard Prediction," *CoRR*, vol. abs/1811.03604, 2018.

[19] J. Henkel, H. Khdr, and M. Rapp, "Smart Thermal Management for Heterogeneous Multicores," in *Design, Automation & Test in Europe Conference & Exhibition, DATE 2019*, IEEE, 2019, pp. 132–137.

[20] U. Gupta, M. Babu, R. Ayoub, M. Kishinevsky, F. Paterna, and Ü. Y. Ogras, "STAFF: online learning with stabilized adaptive forgetting factor and feature selection algorithm," in *55th Annual Design Automation Conference, DAC 2018*, ACM, 2018, 177:1–177:6.

[21] B. Donyanavard, T. Mück, A. M. Rahmani, *et al.*, "SOSA: Self-Optimizing Learning with Self-Adaptive Control for Hierarchical System-on-Chip Management," in *52nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2019*, ACM, 2019, pp. 685–698.

[22] R. A. Shafik, S. Yang, A. Das, L. A. Maeda-Nunez, G. V. Merrett, and B. M. Al-Hashimi, "Learning Transfer-Based Adaptive Energy Minimization in Embedded Systems," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 35, no. 6, pp. 877–890, 2016.

[23] D. Jenkus, F. Xia, R. A. Shafik, and A. Yakovlev, "Runtime Energy Minimization of Distributed Many-Core Systems using Transfer Learning," in *2022 Design, Automation & Test in Europe Conference & Exhibition, DATE 2022*, IEEE, 2022, pp. 1209–1214.

[24] A. Surhonne, F. Maurer, T. Wild, and A. Herkersdorf, "LCT-TL : Learning Classifier Table (LCT) with Transfer Learning for runtime SoC performance-power optimization," in *16th IEEE International Symposium on Embedded Multicore/Many-core Systems-on-Chip, MCSoC*, IEEE, 2023, pp. 73–80.

[25] Z. Tian, Z. Wang, J. Xu, H. Li, P. Yang, and R. K. V. Maeda, "Collaborative Power Management Through Knowledge Sharing Among Multiple Devices," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 38, no. 7, pp. 1203–1215, 2019.

[26] J. Qi, Q. Zhou, L. Lei, and K. Zheng, "Federated Reinforcement Learning: Techniques, Applications, and Open Challenges," *CoRR*, vol. abs/2108.11887, 2021.

[27] G. Gerogiannis and J. Torrellas, "Micro-Armed Bandit: Lightweight & Reusable Reinforcement Learning for Microarchitecture Decision-Making," in *56th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2023*, ACM, 2023, pp. 698–713.

[28] L. J. Lin, "Self-Improving Reactive Agents Based On Reinforcement Learning, Planning and Teaching," *Mach. Learn.*, vol. 8, pp. 293–321, 1992.

[29] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[30] NVIDIA Corporation, *NVIDIA Jetson Nano*, https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/product-development/, (accessed: 2024-08-13), 2024.

[31] ARM Limited, *Cortex-A57*, https://developer.arm.com/Processors/Cortex-A57, (accessed: 2024-08-13), 2024.

[32] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 Programs: Characterization and Methodological Considerations," in *22nd Annual International Symposium on Computer Architecture, ISCA '95*, ACM, 1995, pp. 24–36.