Long Short-Term Memory Neural Network-based Power Forecasting of Multi-Core Processors

Mark Sagi*, Martin Rapp†, Heba Khdr†, Yizhe Zhang*, Nael Fasfous*,
Nguyen Anh Vu Doan*, Thomas Wild*, Jörg Henkel†, Andreas Herkersdorf*
*Chair of Integrated Systems (LIS), Technical University of Munich (TUM), Germany
{mark.sagi, yizhe.zhang, nael.fasfous, anhvu.doan, thomas.wild, herkersdorf}@tum.de
†Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany
{martin.rapp, heba.khdr, henkel}@kit.edu

***Abstract*—We propose a novel technique to forecast the power consumption of processor cores at run-time. Power consumption varies strongly with different running applications and within their execution phases. Accurately forecasting future power changes is highly relevant for proactive power/thermal management. While forecasting power is straightforward for known or periodic workloads, the challenge for general unknown workloads at different voltage/frequency (v/f)-levels is still unsolved. Our technique is based on a long short-term memory (LSTM) recurrent neural network (RNN) to forecast the average power consumption for both the next 1 ms and 10 ms periods. The run-time inputs for the LSTM RNN are current and past power information as well as performance counter readings. An LSTM RNN enables this forecasting due to its ability to preserve the history of power and performance counters. Our LSTM RNN needs to be trained only once at design-time while adapting during run-time to different system behavior through its internal memory. We demonstrate that our approach accurately forecasts power for unseen applications at different v/f-levels. The experimental results shows that the forecasts of our LSTM RNN provide 43% lower worst case error for the 1ms forecasts and 38% for the 10ms forecasts, compared to the state of the art.**

***Index Terms*—Power Forecasting, Recurrent Neural Network, Multi-/Many-Core**

I. Introduction

Power and temperature are first-class constraints in modern multi-/many-core processors [1]. This has led to a plethora of techniques for power and thermal management [2], [3]. The dynamic behavior of workloads is the main challenge because it requires constantly optimizing the system to changing conditions. As an example, Fig. 1 (top) shows the rapidly-changing power consumption of the *SPLASH-2* [4] *fft* benchmark. Only *proactive* management techniques, which anticipate future system behavior, can be optimal in such a scenario, whereas *reactive* techniques, which only react after the system behavior has changed, always lag one step behind [2]. This has two effects: 1) After the system state has changed, the actions taken do not match the changed system state until the management algorithm reacts. Therefore, the employed actions are not optimal, resulting in performance/power/energy losses. 2) The system constraints (e.g., temperature) may be violated during the reaction time. To avoid such violations, reactive approaches can enforce guard bands. This is, however, conservative and causes significant performance losses.

A key component to achieve proactive management is the *forecast* of the future system state [2]. Fig. 1 illustrates the

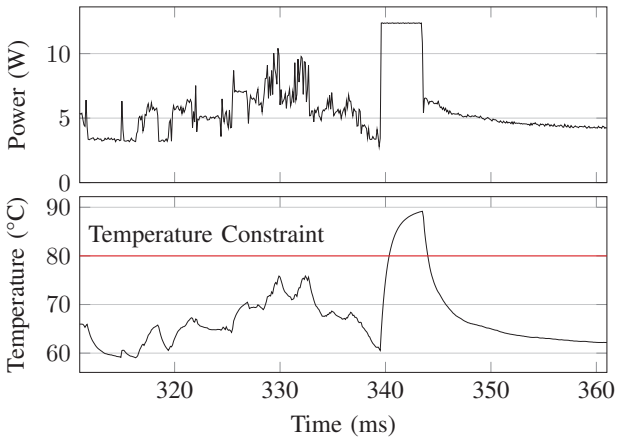

Fig. 1. Motivational example: Splash-2 FFT at 40ms, consistently high power consumption leads to a thermal violation. Anticipating such thermal violations by the means of accurate forecasts is key for avoiding them.

importance of proactive management for thermal management. Around t=330 ms, instantaneous power consumption reaches up to 10.4 W. However, because these peaks are short and are interleaved with phases of lower power, temperature stays below the constraint. Without a forecast, resource management would need to throttle the processor to avoid a potential emergency, unnecessarily decreasing the performance. However, if the forecast is available, throttling can be avoided. At t=339.6 ms, power suddenly increases sharply. This leads to a thermal violation at t=340.4 ms, i.e., only 0.8 ms later, which is too fast for a reactive technique. Only with a forecast available can the violation be avoided. Forecasts of the power consumption can be used to avoid thermal violations. Power consumption is also an essential metric for power and energy management. Therefore, forecasting power is key for many different management techniques.

The term "prediction" is often used in literature with different meanings: 1) Many works have tackled the problem of *estimating the current* power consumption due to the lack of a power sensor [5]. Estimation does not involve aspects of future time, but is still referred to as prediction in some cases. 2) A second type of problem is the *prediction of the impact* of management actions, e.g., the impact of task migration [6]. These techniques inherently assume that the workload does not change soon after

978-3-9819263-5-4/DATE21/©2021 EDAA

applying the management action. They, therefore, predict the sensitivity of the current workload on various v/f-levels, micro-architecture, etc. 3) The third problem is *forecasting future* properties and events. We tackle the problem of forecasting future power consumption, e.g., due to workload phases. Most of the related work has focused on the first two categories, both of which do not involve an aspect of time, and, therefore, are generally easier to solve. However, as demonstrated above, forecasting the dynamics of the workload is mandatory for proactive resource management.

Modern on-chip systems are subject to many contradictory objectives, such as maximizing the performance or minimizing power consumption. Thus, operating systems switch between several different resource management policies. While it is possible to directly learn the management actions from system observations, e.g., with reinforcement learning, such techniques require separate training for each policy [7]. Learning properties of the system that are independent of the management policy allow for reusing the same model for different policies. We aim at forecasting power, and therefore, our model can be employed as an input for many (changing) resource management policies, irrespective of their objectives and constraints.

Power forecasting belongs to the problem class of time-series forecasting. However, compared to similar problems, not only is the past power trace available as a feature for forecasts, but also the trace of the performance counter readings, which allow for additional insights into the application behavior. There are many challenges involved in power forecasting. Autoregressive (AR) models, autoregressive–moving-average (ARMA) models, or variants thereof, are often-used simple linear models. However, the recognition of patterns in the performance and power data, which enables the forecasting of power changes, can require both non-linear modeling and the ability to dynamically discard/forget previous forecast values, e.g., one-time shocks. Another key drawback of ARMA-like models is the limited history length that prevents learning long-term dependencies. The resource management policy may change the v/f-levels at any time, which imposes another challenge for power forecasting because power and performance counter traces may be observed at different v/f-levels.

LSTM RNNs [8] have been successfully employed for time-series forecasting. They overcome the limitations of ARMA-like models because 1) they can learn non-linear dependencies and 2) they can learn long-term dependencies with the help of an internal memory. We propose in this work to employ LSTM RNNs to the problem of power forecasting. We will overcome the challenge of changing v/f-levels by normalizing both inputs and outputs of the model to the v/f-levels. This allows us to use a single model for all v/f-levels. However, there are several challenges when employing an LSTM RNN for power forecasting. To achieve a high prediction accuracy on unseen applications, i.e., generalization, it is crucial that we create a single model that is used for all applications. In addition, the timeframe of forecasts (e.g., forecasting the next 1 ms or the next 10 ms) depends on the management algorithm that makes use of the forecasts with 1 ms and 10 ms being common decision periodicities for power and thermal management. For example, v/f-scaling operates much faster than task migration, which results in very different requirements for the forecast timeframe. A forecasting technique must be easily adaptable to different timeframes.

We are the first to perform power forecasting with LSTM RNNs. This allows us to make the following novel contributions:

- We forecast future workload-dependent core power consumption at run-time considering arbitrary-length history of power and performance counters with high accuracy.
- Our technique is fully-compatible with dynamic voltage and frequency scaling (DVFS), comprising feature traces and forecasting outputs at different v/f-levels with a single model.
- We demonstrate that our model generalizes to unseen applications with high accuracy.

II. Related Work

Machine learning (ML) provides powerful and well-established means to build prediction models. Many works in literature have employed ML in multi-core systems to predict workload power/performance, while only a few of them propose to forecast the workload in the future. In the following we review the most recent papers in both categories; i.e, *prediction* and *forecasting*.

The state-of-the-art techniques in the prediction category employ ML to predict the change in the workload performance/power for potential management decisions, such as DVFS and task migration. For instance, the technique proposed in [9] employs recursive least squares (RLS) algorithm to predict the change in the application performance if DVFS is executed, in order to proactively select the suitable v/f-level for each application towards optimizing for performance. In [10], the change in the power consumption at different v/f-levels has been predicted by incorporating three ML techniques; sequential minimal optimization regression, simple linear regression, and decision trees. In [11], a neural network (NN) model uses performance counters to predict the performance of a thread if it would be migrated to another core in S-NUCA architecture. Another NN model is presented in [6] to predict both power and performance of a thread once migrated to another core on a heterogeneous platform. Support Vector Machine (SVM) has been employed in [12] to predict thread performance indicators, like instructions per cycle (IPC) and last level cache misses (LLCM), for potential core type changes. None of the techniques in this category forecast the workload in the future.

As previously mentioned, the forecasting problem is more difficult than estimation or prediction problems. Due to its complexity, only a few works have tackled the challenge of forecasting, despite its relevance. Some of those works aim at forecasting performance characteristics of the workload, e.g. [13]–[15], while others forecast the power consumption of the workload, e.g. [16]–[18], which are the most closely related works to ours. An early study proposed in [16] introduced a power-forecasting technique by using a history table to identify

application phase patterns based on their active phase lengths. Average power consumption is encoded for each observed phase. If a recurring phase pattern is recognized, the following phase in this pattern and its encoded phase power-level are used as power forecast. The main limitation of this work is its coarse granularity in regard to the length of identified power phases (10-100ms) which does not account for the substantial intra-phase variation of power consumption. Another work [17] forecasts the power consumption by employing a model predictive control. First, the model forecasts performance counters of future kernel executions based on tables that contain profiling information of the kernels. Then, it forecasts power and performance based on the performance counters with design-time models. These forecasts have been used to optimize v/f-levels. More recently, the work proposed in [18], called *SmartDPM*, employs a linear predictor to capture sporadic variations in the workload and accordingly forecasts the power. This forecast allows selecting the appropriate v/f-levels that minimize the power consumption.

The main drawback of these works is that they are designed for known applications. In contrast, our proposed model is able to forecast the power of unknown applications at high accuracy.

III. Problem Formulation

In this work, we target the problem of forecasting the near-future workload-dependent core-level power consumption with high accuracy while maintaining a low overhead. We define the near-future power consumption as the average power consumption within a short future time interval. The accuracy is measured with both the mean absolute percentage error (MAPE) of the forecasts, which is defined as:

$$\text{MAPE} = 100\% \cdot E\left(\left|\frac{P_{forecast} - P_{actual}}{P_{actual}}\right|\right), \qquad (1)$$

as well as the instantaneous worst-case (WC) error over all time steps. We use the complexity of the LSTM NN (e.g., number of layers) as a proxy variable for the overhead. The inputs to the forecasting technique are observations of the current system state. This comprises power values and performance counter readings. The observations may be obtained at a shorter period than the forecasting duration. The following summarizes the problem formulation:

Target: Forecast average core-level power consumption of the future time period τ.

Optimization Goal: Find a trade-off between prediction error (MAPE, WC error) and LSTM RNN complexity (number of layers).

Available Inputs:

- Core power values at observation period $\tau_M \leq \tau$
- Performance counter readings at observation period τ_M
- Employed v/f-levels at observation period τ_M
- Target v/f-level for prediction

IV. Methodology for Generating LSTM NNs for Power Forecasting

An overview of our methodology is shown in Fig. 2. There are two distinct phases. First, at design time, power and performance counter traces for a multitude of benchmarks are aggregated, the LSTM RNN architectures are optimized and the final LSTM RNN for power forecasting is trained. At run-time, the LSTM RNN forecasts power for periods of τ.

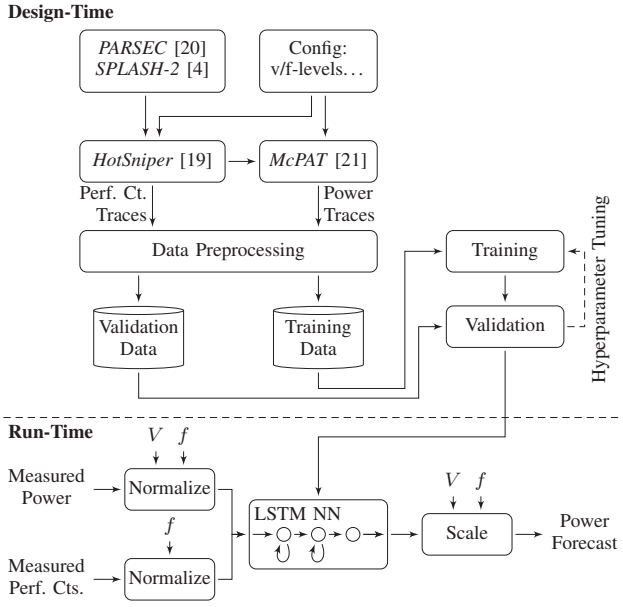

Fig. 2. Overview of our methodology to create the LSTM RNN at design-time and employ it at run-time.

A. Data Preprocessing

The LSTM RNN itself needs to be invariant in regard to voltage and frequency, i.e., the same network needs to accurately forecast power at different v/f-levels. Therefore, we normalize all performance counters PC – where necessary – to the operating frequency of the current data trace. To avoid arithmetic underflows we normalize it to the frequency in GHz for each discrete point k in time as follows:

$$PC_{inv}[k] = \frac{PC[k]}{f_{GHz}[k]}. \qquad (2)$$

The power traces are normalized to the current operational v/f-level as

$$P_{inv}[k] = \frac{P_{dyn}[k]}{f_{GHz}[k] \cdot V^2[k]}. \qquad (3)$$

In addition, the training input is generated in form of average power consumption over τ points in time as:

$$\bar{P}_{inv}[k+\tau] = \frac{1}{\tau}\sum_{i=k}^{k+\tau} P_{inv}[i]. \qquad (4)$$

Afterwards, PC_{inv}, $\bar{P}_{inv}$ and P_{inv} are scaled to the range of $(0, 1)$ for more stable RNN weight updates.

Finally, the data traces of the benchmarks are pseudorandomly distributed into a training data set, a validation data set and a holdout data set. Traces from a single benchmark are only used for one set, and never mixed. Thereby, we maintain independence between the sets. The training data set and the validation data set are used for hyperparameter tuning and for

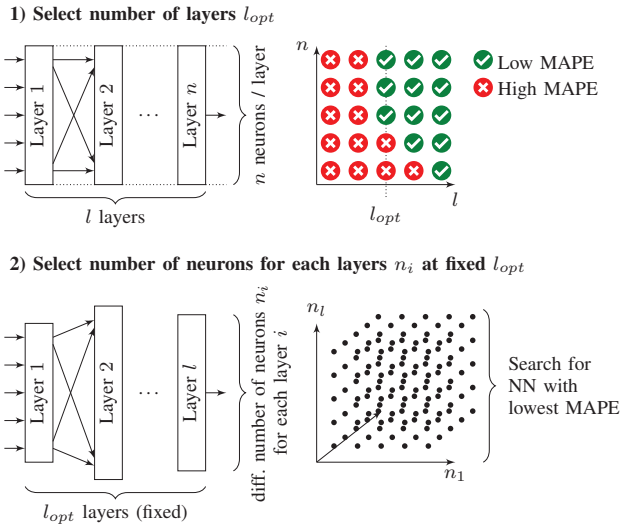

Fig. 3. We select the LSTM NN topology in a two-stage DSE. We first select the depth of the NN (number of layers) l_{opt} under the simplifications of having the same number of neurons per layer, and then optimize the number of neurons for each layer n_i for the selected l_{opt}.

training the final power forecasting LSTM RNN. The holdout data set, is solely used to assess the forecasting performance of the parameter-optimized and trained network in Section V.

B. Hyperparameter Optimization

The input dimensionality of the LSTM RNN is the number of performance counters while the output dimensionality is the number of neurons in the last LSTM layer. These outputs are fed to a dense layer with an output dimensionality of 1. During training, the LSTM RNN is fed with batches of $PC_{inv}[k, k-1, ..., k-m]$ and $P_{inv}[k, k-1, ..., k-m]$ going back m discrete time points and with $\bar{P}_{inv}[k+\tau]$ as a target function. To achieve high forecasting accuracy while reducing run-time overhead, the hyperparameters have to be optimized through design space exploration (DSE). For training the LSTM RNN, we first empirically determine the algorithmic hyperparameters by generating a set of LSTM RNNs with identical architecture and only varying a single hyperparameter for training. The Adam optimizer showed the highest forecasting accuracy compared to the stochastic gradient descent (SGD) and Nesterov accelerated gradient (NAG) optimizers. The maximum number of epochs were set to 10 for τ=1 and 25 for τ=10 where no accuracy improvement on the training data were observable for the largest network architectures. mean squared error (MSE) was chosen as loss function due to its stability. In addition, we use early stopping (patience = 5) with model checkpointing to avoid overfitting the training data. We chose `ReLU` because it achieves the highest training accuracy and a negligible overhead over a set of different activation functions (`ReLU`, `tanh`, `sigmoid`).

The remaining model hyperparameters, i.e., number of layers and number of neurons per each layer, are determined through two successive heuristic grid-searches. An exhaustive parameter search (over all possible combinations of layers and neurons per individual layer) would be computationally infeasible. The model hyperparameters have a large impact on run-time overhead, therefore, we try to find parameter values which are minimal and which generate networks with acceptable MAPE$\leq$10% and MAPE$\leq$30% values for 1ms and 10ms forecasts, respectively. The two grid-searches and their respective optimization goals are shown in Fig. 3.

LSTM RNNs with $l \in \{1, 2, \ldots, 12\}$ layers and $n \in \{1, 2, \ldots, 20, 25, \ldots, 100\}$ neurons per layer are generated in the first-grid search. The goal of the first grid-search is to determine a minimal viable layer count and minimal viable number of neurons per layer. For this, we generate LSTM RNN along a two-dimensional grid with number of layers in one dimension and number of neurons in the other dimension. The resulting network architectures are rectangular, i.e., have always the same number of neurons per layer. We use MAPE in the following to compare the forecasting accuracy of different networks as this metric is independent of the underlying average power levels, for example for different v/f-levels. In contrast, we chose MSE as loss function because the dependence on the power level is not relevant within training particular networks and MSE cannot suffer from small denominators. The optimal layer count l_{opt} is chosen based on the MAPE distributions for the different number n neurons per layer. The smallest layer count for which an n_{min} exists where the MAPE values fall within the acceptable MAPE range for $n \geq n_{min}$ is then the fixed layer count l_{opt} for the second grid-search. In addition, n_{min} is used to constrain the minimum number of neurons per layer in the second grid-search. A parameter n_{max} constraining the maximum number of neurons is also determined based on the n values for which further increases yield no statistically significant improvements in the MAPE. As we expect different forecasting time-frames τ – in our case 1 ms and 10 ms – to require different architectures, we execute a separate first-grid search for each τ value.

In the second grid-search, we aim to find good neuron distributions over a fixed number of layers. The LSTM RNN architectures are thus *not* rectangular anymore. For each parameter point, we generate 10 networks to have more statistically reliable results. The second grid-search is again separately executed for each τ value. The median MAPE value is then computed over the generated networks and the architecture with lowest median MAPE is chosen as architecture for the run-time LSTM RNN. The final LSTM RNN for run-time deployment, generates the dynamic power forecasts $\hat{P}_{dyn}$ as follows:

$$\hat{P}_{dyn}[k+\tau] = \texttt{LSTM}(P_{inv}[k, ..., k-l], \boldsymbol{PC}_{inv}[k, ..., k-l]) \cdot f_{GHz}[k] \cdot V^2[k]. \quad (5)$$

Total power forecast is obtained by adding the static power, which only depends on the v/f-level, and not on the workload:

$$\hat{P}_{total}[k+\tau] = \hat{P}_{dyn}[k+\tau] + P_{static}(V[k], f[k]). \quad (6)$$

V. Experimental Evaluation

A. Experimental Setup

For obtaining the trace data, the HotSniper multicore simulator [19] is used, which is an extended version of the Sniper

TABLE I
PERFORMANCE COUNTER INPUTS

Performance Information		
Processor Unit	Performance Counter	
Core	Instructions per Cycle	IPC
	# Branch Instructions	BPU
	# Floating Point Instructions	FP
	% C0 State Residency of a Core	C0
L2 & L3 Cache	# Load Instructions	LxLI
	# Store Instructions	LxSI
	# Load Misses	LxLM
	# Store Misses	LxSM
	% Cycles Lost due to Cache Misses	LxCLK

TABLE II
BENCHMARKS DISTRIBUTION ON THE TRAINING AND ON THE HOLDOUT DATA SET

Data Set	PARSEC	SPLASH-2
Training/ Validation	bodytrack, dedup, x264, streamcluster, fluidanimate, swaptions, freqmine, raytrace	fft, fmm, lu, ocean, radiosity, radiy, raytrace
Holdout	blackscholes, canneal, facesim	cholesky, barnes

simulator [22]. A 16-core system with 2×2 tiles with four cores each, Intel Gainestown microarchitecture and a NoC interconnect is simulated at a 22 nm technology node. Performance counter values and core-level power—generated by McPAT [21]—are traced with a sampling rate of 10 kHz. The memory architecture is as follows: 32 KB private L1 cache, 265 KB private L2 cache and 8 MB per-tile shared L3 cache. The performance counters used as input for the LSTM RNNs are given in Table I. Table II shows the distribution of the PARSEC [20] benchmarks as well as the SPLASH-2 benchmarks [4] into the training/validation data set and the holdout data set.

B. Hyperparameter Optimization Results

For the first grid-search, we found for $\tau = 1$ ms a minimal layer count of $l_{opt} = 5$. The forecasting accuracy showed significant improvements starting at a number of neurons $n = 10$ and diminishing improvements starting at around $n = 30$. Based on this, in the second grid-search we decided to investigate architectures with number of neurons per layer $n_i \in \{10, 30, 60\}$. Our reasoning for including $n_3 = 60$ is that a single or multiple larger layers might lead to overall smaller neuron numbers on the remaining layers. With this, $3^5 = 243$ different architectures are investigated in the second grid-search and overall 2430 LSTM RNNs are generated. Of these architectures, a `30-60-30-30-10` neuron distribution showed the smallest median MAPE.

For $\tau = 10$ ms were found an $l_{opt} = 3$ and minimal neuron number of $n = 10$, however, the accuracy improvement for larger neuron numbers was far less distinct compared to the 1ms results. Therefore, we chose $n_i \in \{10, 20, 40, 60, 80, 100\}$ which leads to $6^3 = 216$ different architectures and overall 2160 generated LSTM RNNs. The optimal architecture was `20-80-40`. The training time for one LSTM RNN is around one minute and overall training time for both grid-searches and the final evaluation is around 90 hours when using a Nvidia Titan RTX for training.

TABLE III
FORECASTING MAPE VALUES AND INSTANTANEOUS WORST CASE-ERROR VALUES FOR OUR LSTM RNN APPROACH AND REFERENCE APPROACHES

Error	History Table [16]	Autoregressive Model [18]	LSTM RNN
1ms Forecasting Period			
MAPE	30 %	8 %	10 %
Inst. WC	5.3 W	3.5 W	2.0 W
10ms Forecasting Period			
MAPE	31 %	24 %	25 %
Inst. WC	5.4 W	5.8 W	3.6 W

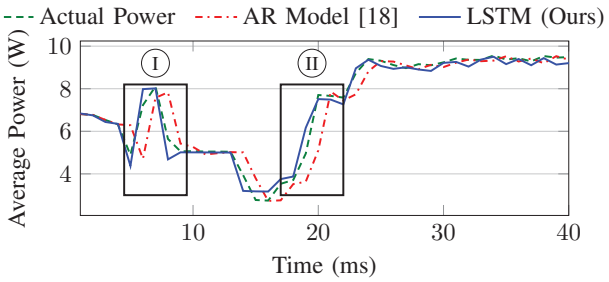

Fig. 4. Power forecasts over time-frames of $\tau = 1$ ms by reference AR model [18] and the proposed LSTM RNN for the unseen PARSEC *facesim* benchmark.

C. Forecasting Accuracy

To assess the forecasting accuracy of the optimized LSTM RNN architectures, we train 100 networks each for the 1 ms and 10 ms architectures using training and validation data at 3 GHz operating frequency. The forecasting accuracy is determined by running inference on the holdout data and then compared to the following baseline techniques: 1) history table forecasting based on [16], 2) autoregressive (AR) forecasting based on [18]. For the AR forecasting methodology, the last 4 average core power values were used as input as this showed the highest forecasting accuracy when parameterizing the AR model for 1 ms and 10 ms forecast durations. The forecasting accuracy of our LSTM RNN approach and the baselines are given in Table III. The error values are computed for all techniques on the same holdout benchmarks. However, the history table-based forecasting [16] has to have observed the power phases at least once to be able to make a power phase forecast. Therefore, the history table-based model observes the holdout data once to generate phase patterns such that it is able to forecast power. For $\tau = 1$ ms, the history table-based approach has significant intra-phase errors due to long phase durations.

The AR based approach shows less MAPE error than our LSTM RNN methodology but higher instantaneous worst-case error. For $\tau = 10$ ms, all approaches have significant MAPE errors with the LSTM RNN having smaller worst-case errors.

To highlight the differing forecasting capabilities, an example of power forecasts for $\tau = 1$ ms is given in Fig. 4 for the PARSEC *facesim* benchmark which is part of the holdout data. We can observe that the AR approach [18] distinctively lags behind the actual core power. This is visible when the power consumption shows sudden changes, e.g., Phases I and II. However, it also adapts to constant power levels which is the main reason for the AR approach showing lower MAPE values compared to our LSTM RNN.

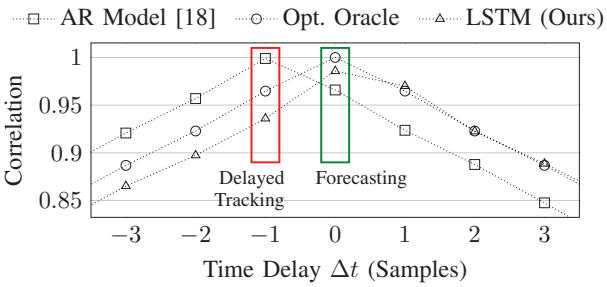

Fig. 5. Cross-correlation of forecasted and actual power trace demonstrating that our model actually forecasts future power.

TABLE IV
FORECASTING ERROR FOR OUR LSTM RNNS TRAINED ON 3 GHZ AND FORECASTING AT 2 GHZ AND 1 GHZ

Error	**2 GHz**	**1 GHz**	**2 GHz**	**1 GHz**
	1ms Forecasting Period		**10ms Forecasting Period**	
MAPE	12 %	8.9 %	28 %	20 %
Inst. WC	1.9 W	0.4 W	2.3 W	0.7 W

We further investigate the forecasting capabilities of the techniques by calculating the cross-correlation between the power trace and the forecasted traces for the full traces of all 5 holdout benchmarks. Fig. 5 shows the resulting cross-correlation values as well as a (hypothetical) oracle policy that perfectly forecasts the power trace. It shows a peak at $\Delta t = 0$ samples, as expected. The correlation decreases slowly towards $\Delta t = \pm\infty$ samples because most of the time, power values only change slightly. Our proposed technique based on the LSTM RNN closely follows this trend, and—most importantly—also has a peak at $\Delta t = 0$ samples. This demonstrates that our technique forecasts the next (future) power value and does *not* simply follow the measured (past) power trace. In contrast, the AR model shows a peak at $\Delta t = -1$ samples, i.e., the forecasted value correlates most strongly with the last observed power value, and *not* with the actual value (delayed tracking).

Finally, we investigate the forecasting ability of the previously generated LSTM RNNs when used at 2 GHz and 1 GHz. The resulting error values are given in Table IV and show that our LSTM RNNs can be used over a wide range of frequencies.

VI. CONCLUSION

This paper presents LSTM RNNs as a promising solution for the power forecasting problem, which is indispensable to enable proactive power and thermal management for on-chip systems. Our proposed technique is able to accurately forecast the power consumption for unknown applications while minimizing LSTM RNN complexity, i.e., number of layers. The experimental evaluation has proved the superiority of LSTM RNN to solve the power forecasting problem.

ACKNOWLEDGMENT

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Projektnummer 146371743 – TRR 89 "Invasive Computing".

REFERENCES

[1] J. Henkel, H. Khdr, S. Pagani, and M. Shafique, "New trends in dark silicon," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2015, pp. 1–6.

[2] A. K. Singh, S. Dey, K. R. Basireddy, K. McDonald-Maier, G. V. Merrett, and B. M. Al-Hashimi, "Dynamic Energy and Thermal Management of Multi-Core Mobile Platforms: A Survey," *IEEE Design & Test*, 2020.

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

[4] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 Programs: Characterization and Methodological Considerations," *Int. Symp. on Computer Architecture (ISCA)*, vol. 23, no. 2, pp. 24–36, 1995.

[5] D. Lee and A. Gerstlauer, "Learning-Based, Fine-Grain Power Modeling of System-Level Hardware IPs," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 23, no. 3, p. 30, 2018.

[6] Y. Kim, P. Mercati, A. More, E. Shriver, and T. Rosing, "P^4: Phase-Based Power/Performance Prediction of Heterogeneous Systems via Neural Networks," in *International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2017, pp. 683–690.

[7] M. Rapp, H. Amrouch, M. C. Wolf, and J. Henkel, "Machine Learning Techniques to Support Many-Core Resource Management: Challenges and Opportunities," in *Workshop on Machine Learning for CAD (MLCAD)*. ACM/IEEE, 2019.

[8] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to Forget: Continual Prediction with LSTM," 1999.

[9] U. Gupta, M. Babu, R. Ayoub, M. Kishinevsky, F. Paterna, S. Gumussoy, and U. Y. Ogras, "An Online Learning Methodology for Performance Modeling of Graphics Processors," *IEEE Transactions on Computers (TC)*, vol. 67, no. 12, pp. 1677–1691, 2018.

[10] B. Dutta, V. Adhinarayanan, and W.-c. Feng, "GPU Power Prediction via Ensemble Machine Learning for DVFS Space Exploration," in *International Conference on Computing Frontiers (CF)*, 2018, pp. 240–243.

[11] M. Rapp, A. Pathania, T. Mitra, and J. Henkel, "Neural Network-based Performance Prediction for Task Migration on S-NUCA Many-Cores," *IEEE Transactions on Computers*, 2020.

[12] C. V. Li, V. Petrucci, and D. Mossé, "Predicting Thread Profiles Across Core Types via Machine Learning on Heterogeneous Multiprocessors," in *Brazilian Symposium on Computing Systems Engineering (SBESC)*. IEEE, 2016, pp. 56–62.

[13] S. J. Tarsa, A. P. Kumar, and H. Kung, "Workload Prediction for Adaptive Power Scaling using Deep Learning," in *International Conference on IC Design & Technology (ICICDT)*. IEEE, 2014, pp. 1–5.

[14] A. Iranfar, W. S. De Souza, M. Zapater, K. Olcoz, S. X. de Souza, and D. Atienza, "A Machine Learning-Based Framework for Throughput Estimation of Time-Varying Applications in Multi-Core Servers," in *International Conference on Very Large Scale Integration (VLSI-SoC)*. IEEE, 2019, pp. 211–216.

[15] M. G. Moghaddam, W. Guan, and C. Ababei, "Investigation of LSTM based prediction for dynamic energy management in chip multiprocessors," in *2017 Eighth International Green and Sustainable Computing Conference (IGSC)*, 2017, pp. 1–8.

[16] W. L. Bircher and L. John, "Predictive Power Management for Multi-Core Processors," in *International Conference on Computer Architecture (ISCA)*. ACM, 2010, p. 243–255.

[17] A. Majumdar, L. Piga, I. Paul, J. L. Greathouse, W. Huang, and D. H. Albonesi, "Dynamic GPGPU Power Management using Adaptive Model Predictive Control," in *High Performance Computer Architecture (HPCA)*. IEEE, 2017, pp. 613–624.

[18] P. D. Saj Manoj, A. Jantsch, and M. Shafique, "SmartDPM: Machine Learning-Based Dynamic Power Management for Multi-Core Microprocessors," *Jrnl. of Low Power Electronics (JOLPE)*, vol. 14, no. 4, 2019.

[19] A. Pathania and J. Henkel, "HotSniper: Sniper-Based Toolchain for Many-Core Thermal Simulations in Open Systems," *IEEE Embedded Systems Letters (ESL)*, 2018.

[20] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications," in *Int. Conf. on Parallel Architectures and Compilation Techniques (PACT)*. ACM, 2008.

[21] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "The McPAT Framework for Multicore and Manycore Architectures: Simultaneously Modeling Power, Area, and Timing," *ACM Transactions on Architecture and Code Optimization (TACO)*, 2013.

[22] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: Exploring the Level of Abstraction for Scalable and Accurate Parallel Multi-Core Simulation," in *Int. Conf. for High Performance Computing, Networking, Storage and Analysis (SC)*. ACM, 2011, p. 52.