Nadjib Mammeri, Markus Neu, Sohan Lal, Ben Juurlink

Performance Counters based Power Modeling of Mobile GPUs using Deep Learning

Mammeri, Nadjib; Neu, Markus; Lal, Sohan; Juurlink, Ben (2019): Performance Counters based Power Modeling of Mobile GPUs using Deep Learning. Accepted for International Conference on High Performance Computing & Simulation (HPCS 2019) - The 17th Annual Meeting; July 15 – 19, 2019; Dublin, Ireland.

Performance Counters based Power Modeling of Mobile GPUs using Deep Learning

Nadjib Mammeri, Markus Neu, Sohan Lal, Ben Juurlink
Embedded Systems Architecture Group
Technische Universität Berlin
Email: {mammeri}, {m.neu}, {sohan.lal}, {b.juurlink}@tu-berlin.de

***Abstract*—GPUs have recently become important computational units on mobile devices, resulting in heterogeneous devices that can run a variety of parallel processing applications. While developing and optimizing such applications, estimating power consumption is of immense importance as energy efficiency has become the key design constraint to optimize for on these platforms. In this work, we apply deep learning techniques in building a predictive model for estimating power consumption of parallel applications on a heterogeneous mobile SoC. Our model is an artificial neural network (NN) trained using CPU and GPU hardware performance counters along with measured power data. The model is trained and evaluated with data collected using a set of graphics OpenGL workloads as well as OpenCL compute benchmarks. Our evaluations show that our model can achieve accurate power estimates with a mean relative error of 4.47% with respect to real power measurements. When compared to other models, our NN model is about 3.3x better than a statistical linear regression model and 2x better than a state-of-the-art NN model.**

***Index Terms*—Power, Modeling, Mobile, GPU, Neural Network, Deep Learning, Performance Counter**

I. Introduction

With the recent shift towards multicore designs and parallel processing, Graphics Processing Units (GPUs) emerged as a dominant platform for parallel computing thanks to their massively parallel architecture and general purpose programmability [1]. This trend is observed in high performance and cloud computing systems as well as in modern mobile computing devices. Recent embedded and portable devices such as smartphones and tablets contain heterogeneous system-on-chips (SoCs) that include several high-end CPU and GPU cores. Understanding and predicting power consumption of these devices is of paramount importance because energy efficiency is often considered the key parameter to optimise for on these devices.

Power modeling and estimation is traditionally performed at design-time using RTL power simulations or using architectural simulators such as GPUSimPow [2] and GPUWattch [3]. The slow performance of these simulations, however, makes such models inappropriate for real-time and online analysis. Researchers often resort to building analytical power models that are based on a variety of techniques ranging from statistical [4] and linear regression techniques [5] to static code analysis [6] and machine learning [7]. These analytical models are orders of magnitude faster, making them appropriate for real-time and online analysis. Examples of use cases of such models include smart DVFS controllers [8] and power-aware scheduling algorithms [9]. However, the accuracy of these analytical models comes into question and is often the determining factor in whether they are adopted or not. Hence the challenge is to not only design a model with a reasonably good performance, making it fit for online analysis, but to achieve as higher accuracy as possible.

In this paper, we propose a highly-accurate power model for estimating power consumption of parallel applications on a heterogeneous mobile SoC. Our power model is an artificial neural network that uses hardware performance counters as inputs and provides a power estimate as output. We apply deep learning techniques, which have shown success in improving the accuracy of many applications such as image classification and voice recognition, in developing our accurate power model. In fact, we trained our model using CPU and GPU performance counter data along with measured power. The model was trained and evaluated with data collected using a set of OpenGL graphics workloads as well as OpenCL compute benchmarks. The measured power data was collected using a real power measuring device, as described in Section III-B.

Although our model was developed for one particular platform, the Intel Z3560 SoC employing an Imagination PowerVR G6430 GPU, one of the main objectives of this work is to come up with a methodology encompassing a model topology and a set of benchmarks and techniques that can later be used in developing similar models for other devices. One of the techniques that we devised and contributed significantly to improving the accuracy of our model, is the use of one of the hardware performance counters as a data coverage measure for the quality of our training data. This is described in detail in Section III-C.

Our experimental evaluations show that our model can achieve power estimates with mean relative error of 4.47% with respect to real power measurements. To further evaluate the effectiveness of our approach, we developed a linear regression model and trained it using the same dataset. The NN model resulted in about 3.3x better accuracy than that of the linear regression model, indicating that the deep learning approach is more effective. Similarly, if compared to prior work utilising similar deep learning approach, our model is 2x better than a state-of-the-art NN based model that was developed for AMD GPUs [10]. This validates our methodology towards improving the quality of the training data by using

a data coverage measure. We also performed further analysis exploring how the model reacts to changes to its topology, by varying the number of layers and number of neurons per layer, and conducted analysis of the overhead and cost involved with such modeling approach.

In particular we make following contributions:

- We propose an accurate performance counter based power model for estimating power consumption of parallel applications on a heterogeneous mobile SoC. The model is an artificial neural network trained with graphics and compute applications.
- We detail our methodology encompassing the model architecture, a set of training benchmarks and data quality techniques with the aim of making it applicable for creating similar models for other platforms.
- We evaluate the effectiveness of our approach by comparing our model to other analytical models and explore further options on how it can be optimised.

II. Related Work

Power modeling has been extensively studied in prior literature. Different approaches were proposed ranging from pure empirical and pure analytical to hybrid which combines the bests of both approaches. The empirical approach is entirely based on measurement data obtained from a particular device. Several works empirically modeled GPU power consumption [11, 4, 12, 13]. Hong and Kim [11] propose an integrated performance and power prediction model for a GPU predicting the optimal number of active processors for a given application. Unlike most previous empirical power models which require measured execution time, hardware performance counters, their model use predicted execution time to estimate dynamic power events. The geometric mean of the error between predicted and measured power is 2.5% for microbenchmarks and 9.2% for GPGPU kernels. Ma et al. [4] present a statistical scheme for analyzing and modeling the power consumption of GPUs. Based on the measured power consumption and runtime workload signals, they build a statistical regression model to dynamically estimate the power consumption of a GPU. The geometric mean of the prediction error is 8.9%. Nagasaka et al. [14] also develop a statistical power prediction model for GPUs by using performance counters and report an average error of 4.7%. Wang and Chen [12] also develop a statistical power model for GPU and show that the power consumption is directly proportional to the computational intensity and the number of active SMs. The reported average relative error is less than 6%. A similar work is done by Zhang et al. [13]. The empirical power models have higher accuracy for the architecture they are built for, but they lack the flexibility to make accurate predictions for GPUs with different architectural parameters and designs. Wang [15] build a high level, purely analytical power model for the main functional units of GPUs by integrating the gpgpu-sim [16], Wattch [17], and Orion [18]. Analytical power modeling is parameterized and it has more flexibility than an empirical power modeling. However, the analytical approach typically cannot provide reasonable absolute accuracy due to lack of measured data.

TABLE I: Platform Specification

Google Nexus Player employing Intel Z3560 SoC	
Operating System	Andorid 7.1
CPU	1.8 GHz Intel Atom(TM) x4
GPU	Imagination PowerVR Rogue G6430
APIs	OpenGL 3.x, 4.x, OpenGL ES 3.0 & OpenCL 1.2

Recently several researchers have combined empirical and analytical approaches to deliver reasonable accuracy as well as flexibility to model different architectures [2, 3]. Both GPUSimPow [2] and GPUWattch [3] are designed using a hybrid power modeling approach. Both simulators integrate a cycle accurate architectural simulator called gpgpu-sim [16] with a heavily modified variant of McPAT [19]. On average GPUSimPow has a relative error of 10.8% for GTX580, while GPUWattch has an average relative error of 9.9 % for GTX 480. Wattch [17] is another widely used CPU power simulator. Compared to McPAT, Wattch only models dynamic power and does not consider timing and area.

In contrast to previous works which are focused on power modeling and estimation of desktop GPUs, our work targets power modeling for mobile GPUs. Moreover, we use deep learning techniques in developing our power model instead of traditional approaches. The closest related work to our approach is [10], which provide a deep learning model for AMD desktop GPUs. Their power data is collected using digital counters inside the hardware that estimate dynamic power and excludes static power. The accuracy of these power measuring counters can be put into question. One can argue that they trained a deep learning model using another model. Our work on the other hand uses a real power measuring device, meaning that our model is able to predict both dynamic and static power. Another major difference is that they trained their model using GPGPU compute workloads only, making their model not able to predict power for graphics workloads because when training only with compute workloads, the graphics sub-components of the GPU such as rasterisation units are not exercised. In contrast, our model can predict both compute as well graphics workloads because it was trained using an extensive set of graphics and compute benchmarks.

III. Methodology

In this section, we describe the architecture of our NN model and the process and methodology employed in its design and training.

A. Platform

Although our methodology is not tied to a particular platform, we mainly carried out our work on the Google Nexus player representing a typical modern portable device powered with a heterogeneous mobile SoC: the Intel Z3560, which is also used in many other consumer devices such as the Asus Zenpad S tablet. Table I provides a detailed specification of the platform used in this work.

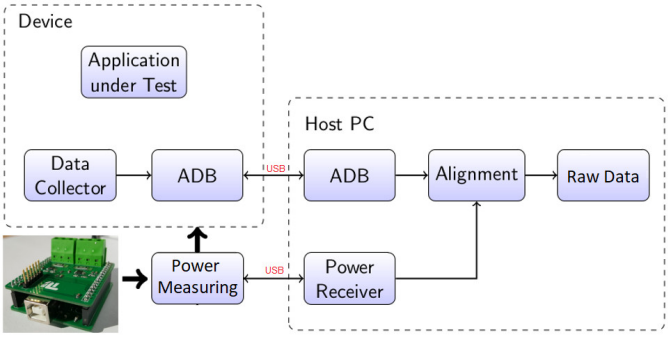
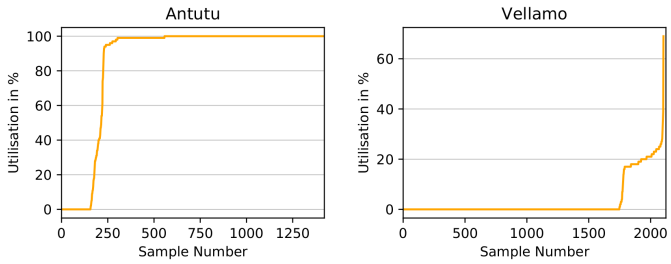
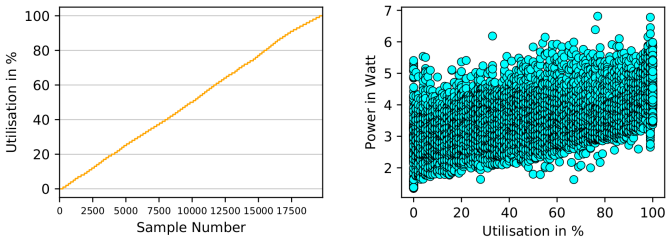


Fig. 1: Data Collection Workflow



(a) Initial GPU Utilisation Data of Antutu & Vellamo Benchmarks



(b) Training Data Coverage of the Antutu Benchmark

Fig. 2: Training Data Quality

This particular SoC was chosen because it employs an Imagination PowerVR GPU and provides access to different GPU performance counters through Imagination's PVRScope API. In total, we used 26 hardware counters including 14 GPU counters obtained using Imagination's PVRScope API and 12 CPU counters obtained using the Perf API. Full list of the hardware counters used in this work is provided in Figure 4.

The power consumption of the SoC is measured using a special device, depicted in Figure 1. This power measuring device [20] was designed specifically for mobile and embedded systems using high precision components and can provide accurate real-time power measurements with sampling rates of up to 125 KHz.

B. Data Collection

Training data collection requires running a set of benchmarks and collecting performance counters data along with power measurements. Figure 1 shows a diagram describing this process. Both the Nexus player and the power measuring device communicate to a host computer via USB. After every data collection step, we perform a further processing step to ensure proper alignment between the hardware counters and the measured power. This is achieved by running a predesigned micro-benchmark that we know it would cause a spike in power consumption before starting data collection for the running benchmark. We later detect this spike and align the data accordingly. Several Shell and Python scripts were created to automate this process.

Unlike other previous works, we used both OpenGL graphics workloads and OpenCL compute workloads as our training benchmarks set. The OpenGL applications used are: 3DMark 1.7.35, AnTuTu 6.1.1, Vellamo 3.2.6, gl2jni, PowerVR SDK appliactions (ParticleSystem, Bumpmap, Glass, Skinning). For the OpenCL workloads we mainly used the Rodinia Suite 3.1 along with self-developed micro benchmarks. We gathered about 16GB of raw data representing days of running benchmarks on the device under test. The data was then post-processed to exclude any zero or redundant samples.

C. Training Data Quality

By examining the initially collected data, we observed that some benchmarks tend to under-utilise the GPU and some benchmarks, in particular heavy OpenGL applications, tend to fully-utilise the GPU. Figure 2a shows examples of initially collected data of the GPU utilisation performance counter obtained by running the Antutu and Vellamo benchmarks respectively. It is evident that the model would not achieve good coverage of the state space if we naively use this data as our training dataset. To mitigate this, we decided to consider the GPU utilisation counter as a measure of data coverage and repeat running the benchmarks several times with different input sizes under different load conditions with the aim of obtaining good coverage of GPU utilisation values.

Figure 2b shows the GPU utilisation of our final dataset for the Antutu benchmark as well as its power distribution with respect to GPU utilisation. It can be seen that the data represents a good spectrum of GPU utilisations ranging from 0 to 100% and a wide spread distribution of power values for every utilisation percentage ranging from 0.5 to 7 Watts. It is worth noting that the maximum power consumption of our device is 7 Watts. We repeatedly run the benchmarks and examined the GPU utilisation until good utilisation spectrum was achieved. Most of our benchmarks were repeatedly run 400 times, whereas the PowerVR SDK applications required to be repeatedly run 1000 times. On the other hand, for some benchmarks in particular the OpenCL ones, full utilisation coverage was not achieved. With this strategy, we managed to gather a large dataset totaling 134k samples including 87k samples from OpenGL applications and 47k from OpenCL applications.

D. Training Data Validation

To validate our final dataset, we performed correlation analysis on all the collected counter samples including power. Figure 4 shows the dataset's correlation matrix as a heatmap

TABLE II: Different model configurations

Learning Rate	Number of Hidden Layers	Neurons per Layer	Test Set Size (%)	Error (MSE)
0.001	2	24	20	0.045856
0.001	2	24	50	0.05068
0.001	2	5	5	0.043573
0.001	2	5	20	0.057422
0.0002	2	50	5	0.045971
0.0004	1	50	5	0.045657
0.0004	**2**	**50**	**5**	**0.042244**
0.0004	3	50	5	0.042798
0.0004	2	5	5	0.049604
0.0008	2	24	5	0.043661
0.0008	2	5	20	0.046521
0.0008	2	5	5	0.058982
0.0008	2	50	5	0.046737
0.0008	2	500	5	0.057934
0.0008	2	1000	5	0.04523
0.0008	2	1500	5	0.04964

on the lower left of the figure and its mirrored correlation coefficients on the upper right of the figure.

By analysing Figure 4, several observations can be made. In general there is very low correlation between different groups of counters. None of the CPU counters depend on any of the GPU counters and vice versa. This is expected because these counters are collected independently capturing the behaviour of two completely different components, CPU and GPU in this case.

When looking at the CPU counters, there is a very strong positive correlation between the number of hardware instructions and their corresponding number of CPU cycles for every core. This is again expected because the number of CPU cycles taken by a certain core depends on the number instructions executed by that core.

When examining the GPU counters on the other hand, we observe two strong negative correlations between the Compute Active counter and the Renderer Active and frames per second counters. The Compute Active is set whenever the GPU is performing a compute workload and similarly the Renderer Active counter is set whenever the GPU is rendering graphics. This reveals information about our process of collecting data, meaning that the GPU was either performing compute or graphics. In fact, this is true because when collecting data, we did not mix running compute and graphics workloads.

All of these observations and interpretations give us confidence in our dataset, however the most important observation that we can make is by examining the power data. It can be seen that all correlation coefficients between power and other counters are lower than 0.5 with a mean of 0.21. This means that there is no strong correlation or in other words no linear relationship between power and any of the other counters. Hence, our deep learning approach using a non-linear activation function is necessary in trying to learn the correlations between the different counters and power. On the top of this, our problem is clearly non-linear and building a linear model with this dataset would not yield good results.

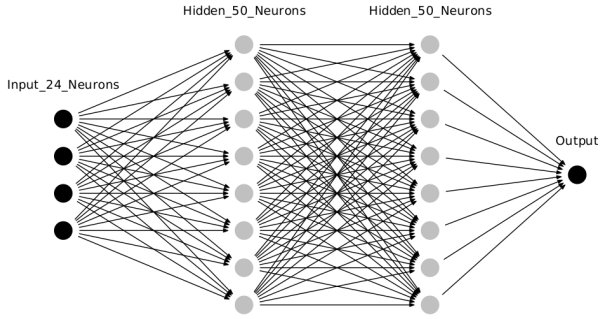


Fig. 3: NN Model Topology

E. NN Power Model Architecture

Determining the right architecture of a neural network upfront is a hard task. Based on initial investigations, we made our model a fully connected one with 4 layers (1 input, 2 hidden layers and 1 output). Since we have a limited number of features 26, making a deeper network would probably result in overfitting. For the hidden layers, *sigmoid* was chosen as the activation function to be able to represent the inherent non-linear relationships between the measured power and the counter values as concluded in Section III-D.

Our model was implemented using the Keras framework running Google Tensorflow [21] as its backend. Training the model with our dataset takes time in the order of days when running on a CPU machine but the training time can be reduced to the order of hours by executing our training runs on a GPU accelerated cluster.

To determine other configuration parameters, we repeated the model training several times varying one parameter at a time and comparing the mean squared error (MSE) of the prediction results. Before training the model, the collected dataset was divided into training set and test set and the samples in each set are then shuffled to ensure that the model does not learn a very specific pattern. Table II shows the obtained MSE error by varying the learning rate, number of layers, number of neurons per layer and the size of of the test dataset.

It can be seen that the best configuration is the one with 2 hidden layers, a learning rate of 0.0004, 50 neurons per layer and a test dataset size of 5%. Having only one hidden layer results in less accuracy and similarly increasing the number of layers above 2 does not yield best accuracy. Increasing the number of neurons per layer, improves accuracy but a bigger change in the number of neurons per layer does not seem to affect the MSE that much. Large numbers of neurons per layer above 500 result in worst performance. Changing the learning rate on the other hand variably affects the MSE. A learning rate of 0.0004 being the best and both smaller and higher learning rates results in less accurate model. Increasing the test dataset size negatively affects the model accuracy, which is expected because the model is trained with less data. The best configuration was achieved with 5% test dataset size, corresponding to 127.3k samples for training data and 6.7k

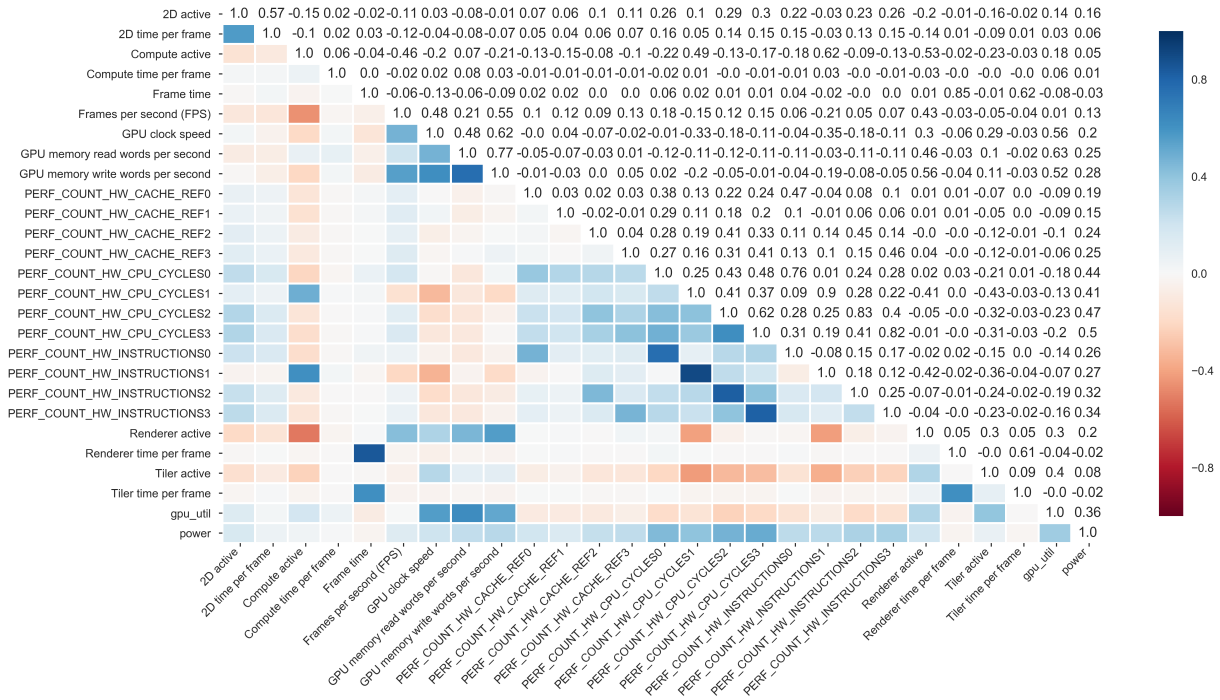


Fig. 4: Correlation Matix of the Entire Dataset

samples for test data.

Beyond these parameters, we also studied the effects of using different activation functions (linear, relu, selu, tanh, softsign softmax, softplus). For the sake of brevity the results are not included but sigmoid resulted in better accuracy. Based on the above results, our final model consists of one input layer, two hidden layers and one output layer. We used sigmoid as the activation function for our hidden layers. The topology of the network is depicted in Figure 3.

IV. Evaluation Results

To determine the effectiveness of our approach we evaluated our model in two different ways. The first using the measured power as a baseline and the second comparing our model to a linear regression model trained with the same dataset.

A. Model Evaluation

To be able to determine how well our model performs, we implemented a new metric in Keras calculating the relative mean error of the predicted power with respect to measured power. The builtin MSE metric is good for comparisons but it does not give an indication of the model's accuracy with respect to real power measurements.

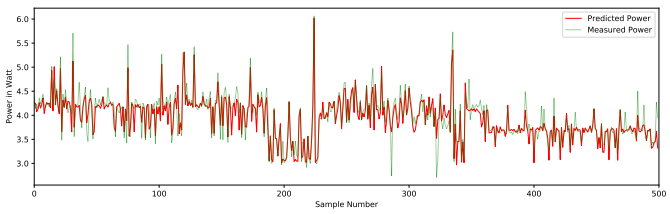
Figure 5a shows the power predicted by our NN model versus the actual measured power of the first 500 samples of the test dataset and Figure 5c shows the corresponding squared errors of these samples. For brevity, we limited both figures to 500 samples but the average error values are calculated for the entire test dataset. Figure 5a clearly shows that the predicted power represented by the red line is pretty well aligned to the actual measured power represented with the green line. This signifies that our model has a good prediction accuracy with a relative mean error of 4.47% over the entire test data set of 6.7k samples. This is confirmed by the corresponding squared errors of individual samples depicted in Figure 5c. Most squared errors are very small with a couple of peaks all less than 1 and a mean squared error of 0.04222.

To give these number a perspective, our model is 2x better than a state-of-the-art deep learning based model reported in [10], which has a relative mean error of 10%. Although the latter model was developed for desktop AMD GPUs, it gives an indication of the effectiveness of our approach. We attribute the good results obtained in this work to our strict methodology and the quality of our training dataset.

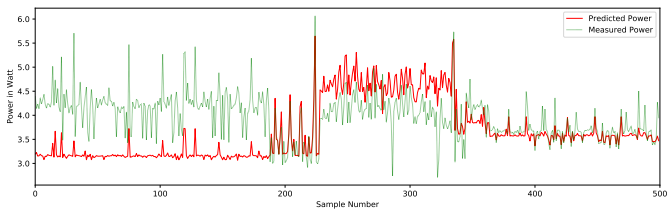
Despite these good results, our model suffers from a couple of limitations. Although we tried to develop a general methodology, our model was only implemented and tested for one particular platform. It would be easier to port the model to other platforms employing an Imagination GPU because we would have access to the same GPU counters. However, developing a model for a platform with a different GPU requires significant work. We believe that the type of techniques and analysis described in this work can be applied in developing other models as long as there is a way to access some GPU and CPU counters.

B. Linear Regression Model Comparison

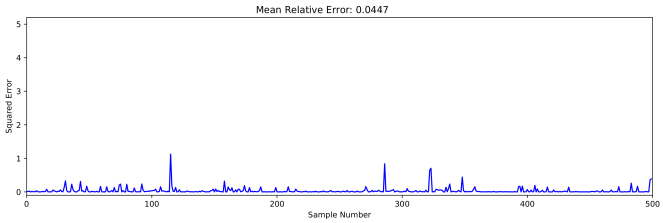
To determine the effectiveness of using deep learning techniques in building such predictive power models, we developed a linear regression model and compared its results with our NN



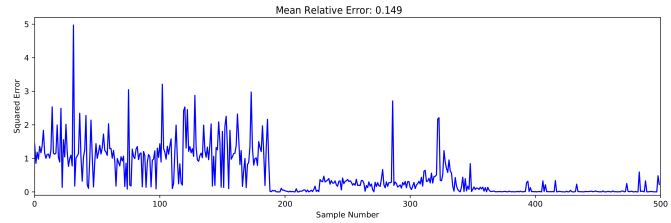
(a) NN Model Predicted Power



(b) Linear Regression Model Predicted Power



(c) NN Model Squared Errors



(d) Linear Regression Model Squared Errors

Fig. 5: NN Power Model Evaluation Results. Only 500 samples are shown but the average error values are calculated for the entire test dataset of 6.7K samples.

model. The linear regression model can be represented using the following equation:

$$Power = w_0 + \sum_{n=1}^{k} w_n * c_n \tag{1}$$

where k is the total number of hardware counters and w_n represents the weight of the nth counter c_n. Assuming we make m data samples including power measurements, we can write the model as an mxk system of linear equations. Then the $k+1$ weights can be calculated by finding the approximate solution that gives the best fit for the mxk system of liner equations minimising the sum of squared errors, as stated by the following equation:

$$\min_{x} \sum_{i=1}^{m} r_i^2 \tag{2}$$

wehere r is the error of the predicted power of the ith sample.

The linear regression was trained with the same dataset used in training the NN model. Figure 5b shows the power predicted by the linear model versus the measured power and Figure 5d shows the corresponding squared errors of individual samples. Again both figures were limited to 500 samples only. In contrary to the NN model results, Figure 5b depicts that the linear model predicted power represented by the red line is not well aligned to the measured power represented by the green line. There is a significant vertical shift of about 1.2 watts between the predicted and measured power values for the first 200 samples then some good alignment for about 50 samples and after that another big discrepancy followed by better alignment for the last 150 samples. This results in a mean relative error of 14.9%. Looking at the squared errors depicted in Figure 5d, they are much larger than those of the NN model resulting in peaks reaching 5 and a mean squared error of 0.5751.

Clearly the NN model is 3.3x better than the linear regression model, implying that the deep learning approach is more effective than linear regression for building such models. These results confirm our findings made in Section III-D stating that the problem in not linear and requires somehow a non-linear solution, which in the case of the NN model is provided by the sigmoid activation function.

C. Overhead Analysis

When deploying a predictive model, it is often necessary to consider the overhead involved with its execution. One advantage of the linear regression model is its simplicity and its low execution overhead. It requires only about 24 MAC operations whereas our proposed NN model with 2 hidden layers and 50 neurons per layer requires 3.85k MAC operations. This number of MAC operations is dependent on the network's number of weights and biases which in turn are dependent on the number of neurons present within the network. On the top of this execution overhead, the number of weights and biases also adds extra memory and storage requirements.

In this section, we try to find the best configuration for our model considering both accuracy and overhead represented by the network's number of weights and biases. Figure 6 shows how the model's mean relative error (MRE) changes by varying the number of neurons per layer and the corresponding changes in the number of weights and biases. It can be seen that there is an exponential decrease in the mean relative error as we increase the number of neurons per layer stabilizing at around 60 neurons with an MRE of 4.4%. Strangely, for 75 neurons the MRE increases to 4.45%. This can be attributed to a bad training run because our training and test data sets are randomly chosen at the beginning of every training run. In the contrary, the number of weights and biases exponentially increases as we increase the number of neurons

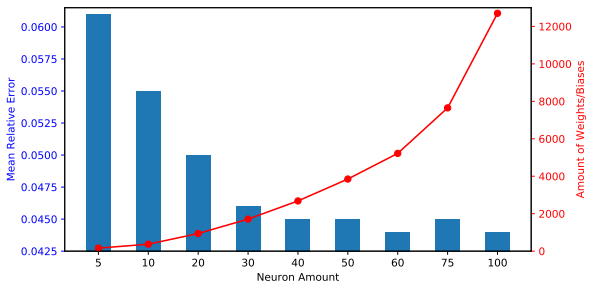


Fig. 6: NN Model Cost Analysis

from 161 parameters for 5 neurons and rapidly reaching 12.7k parameters for 100 neurons.

The intersection point of 30 neurons per hidden layer represents the optimum configuration resulting in mean relative error of 4.62% and 1.7k weights and biases. This represents a 3.35% drop in accuracy and 55% less overhead, implying that the less overhead achieved comes at cost manifested in a drop in accuracy.

V. Conclusion

This paper presented a power model for estimating power consumption of parallel applications on a heterogeneous mobile SoC that has several CPU and GPU cores. Our power model is an artificial neural network that is based on hardware performance counters. We demonstrated that the model is highly accurate with mean relative error of 4.47% and about 2x better than a state-of-the-art model built for desktop GPUs. We also demonstrated that power modeling using deep learning can improve upon statistical techniques such as linear regression by a factor of 3.3x.

As with most machine learning techniques, the model accuracy highly depends on the data used in training. Indeed, we demonstrated that our model accuracy was a result of the quality of our training data, and the methodology and techniques employed in acquiring such data. The type of techniques and analysis applied in this work can be applied in developing power models for other platforms as long as there is a way to access some GPU and CPU counters.

As future work, this power model can be extended for cross-platform prediction or can be utilised in applications such as power-aware task scheduling algorithms on heterogeneous SoCs.

References

[1] J. D. Owens, D. Luebke, N. Govindraju, M. Harris, J. Kruger, A. E. Lefohn, and T. J. Purcell, "A Survey of General Purpose Computation on Graphics Hardware," *Computer Graphics Forum*, vol. 26, pp. 80–113, 2006.

[2] J. Lucas, S. Lal, M. Andersch, M. Alvarez-Mesa, and B. Juurlink, "How a single chip causes massive power bills GPUSimPow: A GPGPU power simulator," in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 4 2013, pp. 97–106.

[3] J. Leng, T. Hetherington, A. ElTantawy, S. Gilani, N. S. Kim, T. M. Aamodt, and V. J. Reddi, "GPUWattch: Enabling Energy Optimizations in GPGPUs," in *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ser. ISCA '13. New York, NY, USA: ACM, 2013, pp. 487–498. [Online]. Available: http://doi.acm.org/10.1145/2485922.2485964

[4] X. Ma, M. Dong, L. Zhong, and Z. Deng, "Statistical Power Consumption Analysis and Modeling for GPU-based Computing," in *Proc. of ACM SOSP Workshop on Power Aware Computing and Systems (HotPower)*. Citeseer, 2009.

[5] T. Jin, S. He, and Y. Liu, "Towards Accurate GPU Power Modeling for Smartphones," in *Proceedings of the 2nd Workshop on Mobile Gaming - MobiGames '15*. New York, New York, USA: ACM Press, 2015, pp. 7–11.

[6] S. S. Baghsorkhi, M. Delahaye, S. J. Patel, W. D. Gropp, W.-m. W. Hwu, S. S. Baghsorkhi, M. Delahaye, S. J. Patel, W. D. Gropp, and W.-m. W. Hwu, "An adaptive performance modeling tool for GPU architectures," in *Proceedings of the 15th ACM SIGPLAN symposium on Principles and practice of parallel programming - PPoPP '10*, vol. 45, no. 5. New York, New York, USA: ACM Press, 2010, p. 105.

[7] J. Lucas and B. Juurlink, "ALUPower: Data Dependent Power Consumption in GPUs," in *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 9 2016, pp. 95–104. [Online]. Available: http://ieeexplore.ieee.org/document/7774570/

[8] M. J. Walker, S. Diestelhorst, A. Hansson, A. K. Das, S. Yang, B. M. Al-Hashimi, and G. V. Merrett, "Accurate and Stable Run-Time Power Modeling for Mobile and Embedded CPUs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 1, pp. 106–119, 1 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7464834/

[9] R. Barik, N. Farooqui, B. T. Lewis, C. Hu, and T. Shpeisman, "A black-box approach to energy-aware scheduling on integrated CPU-GPU systems," in *Proceedings of the 2016 International Symposium on Code Generation and Optimization - CGO 2016*. New York, New York, USA: ACM Press, 2016, pp. 70–81.

[10] G. Wu, J. L. Greathouse, A. Lyashevsky, N. Jayasena, and D. Chiou, "GPGPU performance and power estimation using machine learning," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2 2015, pp. 564–576.

[11] S. Hong and H. Kim, "An Integrated GPU Power and Performance Model," in *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ser. ISCA '10. New York, NY, USA: ACM, 2010, pp. 280–289. [Online]. Available: http://doi.acm.org/10.1145/1815961.1815998

[12] H. Wang and C. Qingkui, "Power Estimating Model and Analysis of General Programming on GPU," *Journal of*

Software, vol. 7, 2012.

[13] Y. Zhang, Y. Hu, B. Li, and L. Peng, "Performance and Power Analysis of ATI GPU: A Statistical Approach," in *2011 IEEE Sixth International Conference on Networking, Architecture, and Storage*, 7 2011, pp. 149–158.

[14] H. Nagasaka, N. Maruyama, A. Nukada, T. Endo, and S. Matsuoka, "Statistical power modeling of GPU kernels using performance counters," in *International Conference on Green Computing*. IEEE, 8 2010, pp. 115–122.

[15] G. Wang, "Power analysis and optimizations for GPU architecture using a power simulator," in *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, vol. 1, 8 2010, pp. 1–619.

[16] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt, "Analyzing CUDA workloads using a detailed GPU simulator," in *2009 IEEE International Symposium on Performance Analysis of Systems and Software*, 4 2009, pp. 163–174.

[17] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," in *Proceedings of 27th International Symposium on Computer Architecture (IEEE Cat. No.RS00201)*, 6 2000, pp. 83–94.

[18] A. B. Kahng, B. Li, L. Peh, and K. Samadi, "ORION 2.0: A fast and accurate NoC power and area model for early-stage design space exploration," in *2009 Design, Automation Test in Europe Conference Exhibition*, 4 2009, pp. 423–428.

[19] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures," in *Proceedings of the 42Nd Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 42. New York, NY, USA: ACM, 2009, pp. 469–480. [Online]. Available: http://doi.acm.org/10.1145/1669112.1669172

[20] B. Juurlink, J. Lucas, N. Mammeri, M. Bliss, G. Keramidas, C. Kokkala, and A. Richards, "The LPGPU2 Project," in *Proceedings of the 20th International Workshop on Software and Compilers for Embedded Systems - SCOPES '17*. New York, New York, USA: ACM Press, 2017, pp. 76–80. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3078659.3078672

[21] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A system for large-scale machine learning," 5 2016.