# Poster: AI Waste Prevention: Time and Power Estimation for Edge Tensor Processing Units

Stefan Reif
Friedrich-Alexander-Unversität
Erlangen-Nürnberg

Benedict Herzog
Friedrich-Alexander-Unversität
Erlangen-Nürnberg

Judith Hemp
Friedrich-Alexander-Unversität
Erlangen-Nürnberg

Wolfgang Schröder-Preikschat
Friedrich-Alexander-Unversität
Erlangen-Nürnberg

Timo Hönig
Ruhr-Universität
Bochum

## ABSTRACT

Artificial Intelligence (AI) has changed our daily lives. The evolution from centralised cloud-hosted services towards embedded and mobile devices has shifted the focus from quality-related aspects towards the resource demand of machine learning. Its pervasiveness demands for "green" AI—both the development and the operation of AI models still include significant resource investments in terms of processing time and power demand. In order to prevent such *AI Waste*, this paper presents Precious, an approach, as well as practical implementation, that estimates execution time and power draw of neural networks (NNs) that execute on a commercially-available off-the-shelf accelerator hardware (i.e., Google Coral Edge TPU). The evaluation of our implementations shows that Precious accurately estimates time and power demand.

## CCS CONCEPTS

• **Hardware** → **Power and energy**; • **Computer systems organization** → *Embedded systems.*

## KEYWORDS

Green AI, Neural Network Accelerators, Resource Awareness

**ACM Reference Format:**
Stefan Reif, Benedict Herzog, Judith Hemp, Wolfgang Schröder-Preikschat, and Timo Hönig. 2021. Poster: AI Waste Prevention: Time and Power Estimation for Edge Tensor Processing Units. In *The Twelfth ACM International Conference on Future Energy Systems (e-Energy '21), June 28–July 2, 2021, Virtual Event, Italy.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3447555.3466579

## 1 INTRODUCTION

Machine learning has historically been associated with high resource demand. Due to their tremendous success, machine-learning applications have entered the domain of embedded systems, such as cars and smartphones [3, 9]. To enable machine learning in embedded devices, applications have to adhere to their limited resources [6, 8] in addition to providing a good quality of service (e.g., model prediction accuracy). The need for embedded machine learning has led to the development of special-purpose accelerator hardware for neural networks, such as the Coral Edge TPU [2, 5]. These hardware accelerators satisfy the growing demand and interest for machine-learning approaches [1, 4] and promise to execute the corresponding machine-learning workloads more efficiently than general-purpose hardware. For *green AI* [8], developers of deep learning applications need information on the resource demand, considering both hardware and software (i.e., the neural network). This poster, in particular, determines how *predictable* the resource demand of NN accelerator hardware is. To this end, we train various models that map statically analysable NN properties to the measured resource demand.

## 2 DESIGN AND IMPLEMENTATION

Our approach, Precious [7], comprises four phases. First, the *NN generation phase* creates random neural networks (NNs) that are either fully connected (i.e., dense), or contain convolutional layers, exclusively. Second, the *NN execution phase* executes all NNs on the TPU (i.e., it runs inferences) while measuring power draw and execution time. Third, the *training phase* creates models that map NN properties (as determined by a static analysis) to their resource demand (i.e., power draw and execution time). In the final *application phase*, application developers can apply the trained models to estimate the resource demand of their NNs.

In this poster, we extend Precious by a variety of models for power and execution time estimations. We evaluate the complexity of accurate resource estimation on embedded NN accelerators. Furthermore, we put the accuracy into context by outlining remaining limitations of resource predictability.

## 3 EVALUATION

*Model Prediction Accuracy.* We train three types of models: First, *dummy regressors* (DR) only compute the *mean* (MEA) or *median* (MED), ignoring all features. Second, linear models are trained using *linear* (LR), *huber* (HR), and *ransac* (RR) regressors with *linear* (LR) or *ridge* (R) as base regressors. Third, ensemble models are trained using *extra tree* (ETR), *random forest* (RFR), *decision tree* (DTR), and *adaboost* (ABR) regressors. Ensemble models are configurable by an error metric, which can be the *friedman mean squared error* (FMSE), the *mean absolute error* (MAE), or the *mean squared error* (MSE).

*e-Energy '21, June 28–July 2, 2021, Virtual Event, Italy*

ACM ISBN 978-1-4503-8333-2/21/06...$15.00
https://doi.org/10.1145/3447555.3466579

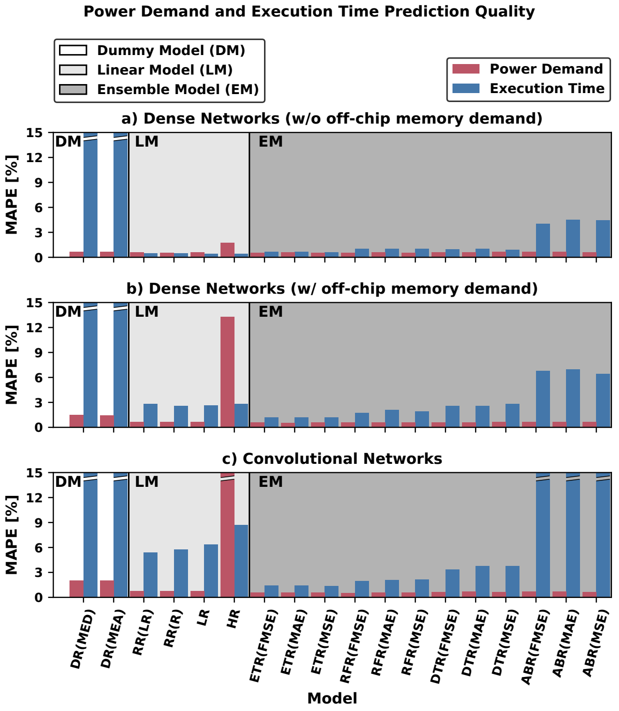

**Figure 1: Linear and ensemble model evaluation of the prediction quality for a) dense networks without and b) with off-chip memory demand, and c) convolutional networks.**

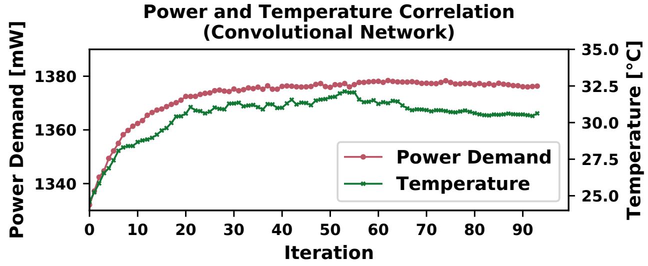

**Figure 2: Temperature and power demand increase for repeated execution of neural networks.**

The prediction accuracy of all models is summarised in Figure 1. We use the *mean absolute percentage error* (MAPE) as error metric, as it normalises the error to the measurement, and the individual resource demand can vary (NN-to-NN) significantly. The graphic is cut off at 15 % to maintain readability. For the *power draw*, the evaluation shows that most regressors achieve a similar performance—including the dummy regressors. Regarding the *execution time*, the ensemble models are generally most accurate, but for dense networks, linear models perform similar. However, some regressors do not converge in our experiment (e.g., ABR). Considering that an increase in regressor complexity results in a minor improvement in prediction accuracy, we conclude that the resource demand of TPUs is, in general, well-predictable.

*Limitations to Predictability.* The *power draw* of transistor-based logic circuits depends on the temperature, which is influenced by the ambient temperature and also self-induced heat. Both typically vary over time for most embedded systems, and are not precisely controllable. Consequently, the power draw of identical NNs varies between executions. A model trained with only statically available data cannot capture this variance. To obtain temperature traces, we use a Bosch Sensortec BME280 temperature sensor, attached to the casing of the TPU. Figure 2 shows the measured power and temperature trace of the running TPU. The surface temperature starts at ambient temperature (24.6 °C) and rises within 20 inferences by 5.8 °C. Then, the temperatures stabilises at 30.4 °C. The power draw shows the expected correlation with temperature, rises by 39.0 mW, and stabilises at 1332.1 mW, which is an increase of 2.9 %.

The *execution time* of the TPU depends on timings of the communication with the host system via USB. To verify this dependency between communication patterns and the execution time, we monitor the USB traffic with tshark. The traffic capture show that, between iterations, the host system communicates input and output data with the TPU. This means that the effective neural network execution time depends on the precise timings the USB bus, which is affected by interferences of other attached devices. We did not model the USB traffic in Precious because the communication timing is not known statically, as it depends on cross-traffic, and other embedded systems might use different buses and protocols.

## 4 CONCLUSION

This poster has presented an extended evaluation of Precious, a system to estimate the resource demand of NNs on embedded accelerator hardware, based on various regressors. Our evaluation shows that the NN accelerator generally behaves well-predictable. In many cases, estimations based on linear models are sufficiently accurate, considering that the estimation error is similar to the variation caused by temperature fluctuations and USB traffic.

## ACKNOWLEDGMENTS

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) – project number 146371743 – TRR 89 "Invasive Computing", under grants no. SCHR 603/13-1 ("PAX") and SCHR 603/15-2 ("eLARN"), and and from the Bundesministerium für Bildung und Forschung (BMBF) for the project AI-NET-ANTILLAS 16KIS1315.

## REFERENCES

[1] 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. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *Proc. OSDI'16*. USENIX, 265–283.

[2] S. Cass. 2019. Taking AI to the edge: Google's TPU now comes in a maker-friendly package. *IEEE Spectrum* 56, 5 (May 2019), 16–17.

[3] S. Greengard. 2020. AI on Edge. *CACM* 63, 9 (Aug. 2020), 18–20.

[4] Google LLC. 2020. Tensorflow Keras. https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras. Acc. 2020-02-20.

[5] Google LLC. 2020. USB Accelerator. https://www.coral.ai/products/accelerator. Acc. 2020-02-20.

[6] S. S. L. Oskouei, , H. Golestani, M. Hashemi, and S. Ghiasi. 2016. CNNdroid: GPU-Accelerated Execution of Trained Deep Convolutional Neural Networks on Android. In *Proc. MM'16*. ACM, 1201–1205.

[7] S. Reif, B. Herzog, J. Hemp, T. Hönig, and W. Schröder-Preikschat. 2020. Precious: Resource-Demand Estimation for Embedded Neural Network Accelerators. In *Proc. Challenge'20*.

[8] R. Schwartz, J. Dodge, N. Smith, and O. Etzioni. 2020. Green AI. *CACM* 63, 12 (Nov. 2020), 54–63.

[9] M. Xu, J. Liu, Y. Liu, F. X. Lin, Y. Liu, and X. Liu. 2019. A First Look at Deep Learning Apps on Smartphones. In *Proc. WWW'19*. ACM, 2125–2136.