# Faculty of Engineering Sciences

Heidelberg University

Master Thesis
in Computer Engineering
submitted by
Constantin Nicolai
born in Bretten, Germany
Day/Month/Year Here

# YOUR TITLE HERE

This Master thesis has been carried out by Constantin Nicolai
at the
Institute of Computer Engineering
under the supervision of
Holger Fröning

# ABSTRACT

Briefly summarize the contents of your work in 150-250 words. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

# ZUSAMMENFASSUNG

Deutsche Version. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

*Nice quote here.*

— Some Author

# ACKNOWLEDGMENTS

Your acknowledgments here if desired

# CONTENTS

# 1 INTRODUCTION AND MOTIVATION

Introduction to your topic and motivation of your work. Example citation [1] (good book!). Table 1.1 shows an example table and Figure 1.1 an example plot.

**Table 1.1:** An Example table

| Memory [MiB] | Time [s] | Complexity |
|---|---|---|
| 40 | 10 | ∞ |

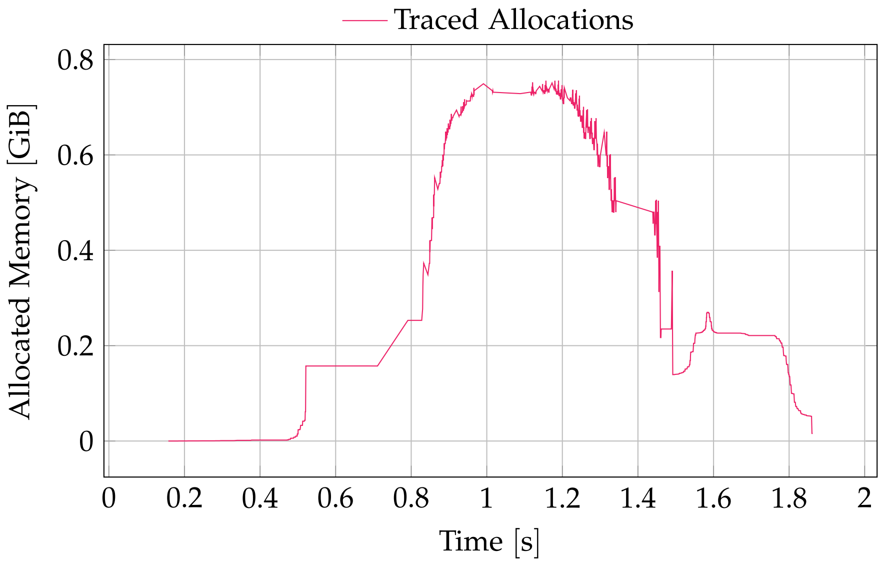

**Figure 1.1:** An example Figure

# 2 BACKGROUND

Sponner
metrics: power consumption, latency, memory footprint
prediction method: ERT extremely randomized trees, with XGBoost
framework: TVM, which I think allows the largest heterogenity of targets
Only inference, since we cannot match this work in any other measure, we at least need to beat it here by also including training

Ying Li: Path Beyond Simulators
metrics: latency
prediction method: Linear regression
framework: Pytorch
only inference
Based on architectural properties, not benchmarks

Daniel Justus: Predicting the Computational
metrics: latency
prediction method: regularized MLP
framework: Tensorflow
inference and training

Geoffrey X. Yu: A Runtime-Based ... Habitat
metrics: latency
predicion method: wave scaling and MLPs
framework: Pytorch
trainig
killer feature: does not require the GPU for predictions

## 2.1 TOPIC 1

The challenge of predicting neural network performance has invited a plurality of approaches. Apart from the methodological approaches they also differ in a number of aspects. While execution time is commonly the metric of choice, only few go further and also study metrics like power consumption and memory footprint. Another important distinction is the workload studied in the work, more specifically, whether both training and inference are studied. For practical reasons it is also relevant which machine learning framework is used and what hardware targets are required and can be predicted for. These many dimensions of possibility result in no work covering all possibilities, but allows for many different approaches which have use cases in a given situation.

There are also two fundamentally different philosophical approaches in this field. The first one has its origin in the hardware simulations. But the long simulation times make full simulations undesirable. Therefore one approach builds a performance model for DNNs using observations from a dataset of commonly used models. It uses linear regression in order to reflect the linear relationships observed between execution time and different properties of the neural network operations, such as input parameters, FLOPs and output parameters. While this work only allows predicting execution times for inference, due to its nature as a performance model based on hardware properties, it can be used for predictions of hardware targets outside its dataset and even for GPUs which might not be available to the user or may not even exist yet.

Another work called Habitat goes into a similar direction, in that it also utilizes hardware properties to predict performance on a different GPU based on the data collected on a local GPU at runtime. The beauty of this lies in the fact that this allows it to be used with any kind of model, since it just relies on runtime information. That makes it very appealing to researchers working on new or modified models. The prediction approach for Habitat is either a roofline model inspired scaling formula, called wave scaling, or an MLP based approach for operations which use different kernels based on their hardware target.

The second approach is comes from the idea of simply benchmarking the performance, but tries to generalize, simplify and accelerate this process. In order to do that the modular and repetitive nature of neural networks is employed. Since they are made up of many small operations which only vary in a few key parameters and are repeated numerous times in the training process and even during inference, measuring these building blocks and using these results to find full model performance is the obvious approach.

One work following this approach takes the step from the preceding works to replace the common linear regression with an MLP, trained on a subset of the many features of common DNN layers and their execution times. While this does include training time prediction, unfortunately it only focuses on the operation prediction and the combination into full model predictions, but fails to present the dataset collection and methodology, which is very crucial part in this data driven approach.

The last work to mention here is one that provides a great basis to start from for the measurement, rather then modeling approach side. It is built upon the TVM machine learning compiler, which provides great flexibility in the choice of hardware target, as it even employs target dependent automatic optimizations. Using TVM-built in tools to profile execution time, power consumption and even memory footprint for supporting hardware targets, it has both a broad and solid basis for its dataset of layer measurements. The actual prediction is performed using an ERT (extremely randomized tree) with XGBoosting. This leads to solid results on a wide variety of targets, the only major drawback is the lack of training support, since the work focuses solely on inference.

### 2.1.1 Subsection

Kaufmann et al. take an approach of performance modeling by means of the computation graph. They are however limited to the Google Tensor Processing Unit in this work.
Justus et al. take an approach exploiting the modular and repetitive nature of DNNs. Given the same operations are repeated over and over in training, often only varying in a few key paramters, these execution time for these base building blocks is measured. This is then done for one batch in the training process and generalized to the whole training process from there. There is however no presentation of the methodology for the execution time measurements.
Qi et al. present PALEO which employs an analytical approach towards predicting the execution for both training and inference of deep neural networks. The analytical approach brings both advantages and disadvantages with it. It does not require a dataset of measured execution times as a training set in the same way many other works do, but on the other hand it also is based on more fixed assumptions about the DNN execution than a more data driven approach.
Wang et al. approach with a mult-layer regression model to predict execution time for training and inference. Their work is however rather limited in terms of hardware targets and different DNNs studied.
Cai et al. focus their work, NeuralPower, on CNNs running on GPUs. For each target GPU, they collect a dataset and fit a sparse polynomial regression model to predict power, runtime, and energy consumption. While NeuralPower achieves good results, its usefulness has become limited due to its exclusive focus on CNNs, as other DNN architectures have grown in popularity.
Gianitti et al. also exploit the modular nature of DNNs in their approach. They define a complexity metric for each layer type, optionally including backpropagation terms, allowing them to predict execution times for both training and inference. However, their method faces significant limitations, as the complexity metric is only defined for a specific set of operations, making it incompatible with networks that include layers not covered in the original work. As a result, their approach is essentially limited to classic CNN architectures.
Velasco-Montero et el. also take the familiar per-layer approach. Their

predictions are based on linear regression models per type of layer, but again for a specific set of predefined operations. Given their focus on low-cost vision devices these restrictions are reasonable, but limit generalizability.

### 2.1.2 Other Subsection

Other details.

## 2.2 TOPIC 2

Second topic.

# 3 STATE OF THE ART AND RELATED WORKS

Talk about related works and state of the art, plus possibly problems with SOTA that you are fixing.

# 4 FIRST CONTRIBUTION

First contribution here.

## 4.1 SECTION

A section.

### 4.1.1 Subsection 1

Details.

### 4.1.2 Subsection 2

Yet another detail.

# 5 SECOND CONTRIBUTION

The second contribution goes here.

# 6 DISCUSSION AND OUTLOOK

Summary and discussion of the results and outlook/future work.

# a | APPENDIX

Appendix here

# BIBLIOGRAPHY

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. New York: Springer, 2006. 738 pp. ISBN: 978-0-387-31073-2.

# ERKLÄRUNG

Ich versichere, dass ich diese Arbeit selbständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

*Heidelberg, den Day/Month/Year Here*

Constantin Nicolai