HAPPi-Net

Hardware-Aware Performant Perception of Neural Networks

Alexander Frickenstein

Fakultät für Elektrotechnik und Informationstechnik
Technische Universität München

HAPPi-Net: Hardware-Aware Performant Perception of Neural Networks

Alexander Frickenstein

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:
Prof. Dr.-Ing. Georg Sigl

Prüfende der Dissertation:
1. Prof. Dr.-Ing. Walter Stechele
2. Prof. Maurizio Martina, PhD.

Die Dissertation wurde am 14.12.2020 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 19.03.2021 angenommen.

Berichte aus der Informatik

Alexander Frickenstein

HAPPi-Net

Hardware-Aware Performant Perception of Neural Networks

Shaker Verlag
Düren 2021

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Zugl.: München, Techn. Univ., Diss., 2021

Printed in Germany.

ISBN 978-3-8440-8069-8
ISSN 0945-0807

Shaker Verlag GmbH • Am Langen Graben 15a • 52353 Düren
Phone: 0049/2421/99011-0 • Telefax: 0049/2421/99011-9
Internet: www.shaker.de • e-mail: info@shaker.de

Abstract

Artificial neural networks are dominating a vast majority of application scenarios to date, and will surely extend their lead in the near future. Especially, the superior performance of convolutional neural networks (CNNs) for image processing tasks presents a promising use case in innovative and cutting-edge domains, such as robotics and autonomous driving. However, their dominance emerges from an ever-increasing memory intensity and computational complexity. In contrast to the increasing resource demand, real-world applications on embedded devices pose major challenges with regard to limited computing power, memory resources and available energy and/or latency budget for the deployment of CNNs in embedded settings. To counteract these challenges, this thesis presents a tripartite hardware-software co-design paradigm for the efficient application of CNNs on embedded accelerators. This allows the traversal through the design space by either a top-down, meet-in-the-middle or a bottom-up approach. Moreover, six novel optimization methods, on the three levels of abstraction, are presented in this dissertation, which further serve the illustration of the simplified design process. In detail, we present three different architectural optimization methods. One of which is the recently introduced autoencoder-based low-rank filter-sharing (ALF) technique. The compressed CNNs are applied to different central processing units (CPUs) and field programmable gate arrays (FPGAs) using algorithmic optimization techniques. By means of successive exploration and refinement steps, it is shown how more powerful CNN-based applications can be created and make use of orthogonal optimization methods like pruning, quantization and Winograd convolution. Furthermore, the increase in data-level parallelism is achieved by quantized neural networks. Here, binaryDAD-Net, a fully binarized neural network, is presented for semantic drivable area detection. In the same context, requirements of binary neural networks for the design of a runtime reconfigurable processing element, namely OrthrusPE, are made accessible. In summary, we show that the optimization of CNNs for embedded applications, such as in the field of autonomous driving, can only be achieved through the interaction of the three abstraction levels (using expert knowledge) and synergies of different compression techniques to arrive at a fruitful HW-CNN co-design.

Zusammenfassung

Künstliche neuronale Netze dominieren derzeit die Mehrheit der Anwendungsszenarien und werden ihren Vorsprung in Zukunft sicherlich noch ausbauen. Insbesondere die überlegene Leistungsfähigkeit von faltenden neuronalen Netzen (engl. Convolutional Neural Networks - CNNs) bei Bildverarbeitungsaufgaben stellt einen vielversprechenden Anwendungsfall in innovativen und zukunftsweisenden Bereichen wie der Robotik und dem autonomen Fahren dar. Die Vorherrschaft von CNNs rührt jedoch von einem fortwährend steigenden Speicherbedarf und wachsender Rechenkomplexität. Eine zentrale Herausforderung stellt dabei der steigende Ressourcenbedarf im Hinblick auf den Einsatz von CNNs in praktischen Anwendungen auf eingebetteten Systemen dar, im Sinne der begrenzten Rechenleistung und Speicherressourcen, des verfügbaren Energiebedarf oder der erlaubten Latenzzeiten. Um diesen Herausforderungen zu genügen, stellt diese Arbeit ein dreigliedriges Hardware-Software-Co-Design-Paradigma für die effiziente Anwendung von CNNs auf eingebetteten Beschleunigern vor. Dies erlaubt das Durchqueren des Designraumes mittels eines Top-Down-, Meet-in-the-Middle- oder Bottom-Up-Ansatzes. Darüber hinaus werden in dieser Dissertation sechs neuartige Optimierungsverfahren vorgestellt, welche auf den drei Abstraktionsebenen fußen, um den vereinfachten Entwurfsprozess weiter veranschaulichen. Im Detail werden drei unterschiedliche architektonische Optimierungsmethoden vorgestellt. Eine dieser Varianten basiert auf der erst kürzlich vorgestellten Autoencoder-basierten Low-Rank-Filter-Sharing (ALF) Technik. Die komprimierten CNNs werden hierbei unter zur Hilfenahme algorithmischer Optimierungstechniken auf verschiedenen Hardwarebeschleunigern (CPUs oder FPGAs) angewendet. Anhand von aufeinanderfolgenden Explorations- und Verfeinerungsschritten wird gezeigt, wie leistungsfähigere Anwendungen auf der Basis von CNNs erstellt werden können, welche orthogonale Optimierungsverfahren wie das Beschneiden, die Quantisierung und die Winograd-Faltung nutzen. Darüber hinaus wird die Erhöhung der Parallelität auf Datenebene durch quantisierte neuronale Netze erreicht. Hier wird binaryDAD-Net, ein vollständig binarisiertes neuronales Netz, zur semantisch Detektion eins befahrbaren Bereiches vorgestellt. Im gleichen Zusammenhang werden Anforderungen von binären neuronalen Netzen an den Entwurf eines zur Laufzeit rekonfigurierbaren Verarbeitungselements, durch OrthrusPE, zugänglich gemacht. Zusammenfassend wird gezeigt, dass die Optimierung von CNNs für eingebettete Anwendungen, beispielsweise im Bereich des autonomen Fahrens, nur durch das Zusammenwirken der drei Abstraktionsebenen (unter Verwendung des Expertenwissens) und den Synergien verschiedener Kompressionstechniken erreicht werden kann, dass in einem vorteilhaften Hardware-CNN Co-Design gipfelt.

Acknowledgments

First and foremost, I would like to thank my Ph.D. supervisor Prof. Walter Stechele. Walter is an excellent and experienced supervisor. His calm nature has always helped me over the past years, even in difficult moments, to have the necessary vision for further steps. In particular, his faith and certainty in the research project has helped to facilitate a successful cooperation between the university and BMW AG. Thanks to his enthusiasm, the initial idea has developed into a well working research team.

I would also like to thank my advisor Christian Unger for his excellent support. This is especially true for his engagement in challenging the ideas, questioning results and setting great goals.

Moreover, it was a great pleasure for me to work together with the Ph.D. colleagues Manoj-Rohit Vemparala and Nael Fasfous, to push and question our ideas and to inspire each other's work. I will miss the endless coffee, sandwiches and long working evenings on campus with them. Manoj and Nael have made a significant contribution to the success of this work, for which I am grateful to them.

For the endless opportunities to expand the research and the support with resources, I would like to thank my department head Simon Fürst from BMW AG. I would also like to thank all colleagues from the feature team Asimov and the BMW autonomous driving campus who have accompanied and supported the project in the past years.

Finally, I would like to thank my family, Mom, Dad, Lukas and Isabel, my girlfriend Julia Leopoldseder and my friends Matthias Kreuz, Florian Schoppe and Dieter Werner for their warm-hearted support over the years.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

AD autonomous driving.

ADMM alternating direction method of multipliers.

AI artificial intelligence.

ALF autoencoder-based low-rank filter-sharing.

ALU arithmetic logic unit.

AMC AutoML for Model Compression.

ANN artificial neural network.

ARCS Architecture of Computing Systems.

ASIC application-specific integrated circuit.

ASPP Atrous Spatial Pyramid Pooling.

AVX Advanced Vector Extensions.

BC Bayesian Compression.

BEV battery electric vehicle.

binaryDAD binary drivable area detection.

BLAS basic linear algebra subprograms.

BNN binarized neural network.

BPAC Binary Parallel Atrous Convolution.

BRAM block random-access memory.

CNN convolutional neural network.

CPU central processing unit.

CRS compressed row storage.

CRV Conference on Computer and Robot Vision.

CV computer vision.

CVPR Conference on Computer Vision and Pattern Recognition.

DAC Design Automation Conference.

DAD drivable area detection.

DATE Design, Automation & Test in Europe Conference and Exhibition.

DC Deep Compression.

DDPG deep deterministic policy gradient.

DL deep learning.

DLA Deep Learning Accelerator.

DNN deep neural network.

DNNDK Deep Neural Network Deployment Kit.

DNS Dynamic Network Surgery.

DPU Deep Learning Processing Unit.

DRAM dynamic random-access memory.

DSC dense-sparse convolution.

DSD dense-sparse-dense.

DSP digital signal processor.

ECU electronic control unit.

EIE Efficient Inference Engine.

FCN Fully Convolutional Network.

FFT fast Fourier transform.

FPGA field programmable gate array.

FPGM Filter Pruning via Geometric Median.

GAN Generative Adversarial Network.

GCC GNU Compiler Collection.

GEMM general matrix-matrix multiplication.

GOPS giga operations per second.

GPU graphics processing unit.

HDL hardware description language.

HIL hardware-in-the-loop.

HLS high-level synthesis.

HW hardware.

HW-CNN hardware-convolutional neural network.

IC integrated circuit.

ICRA International Conference on Robotics and Automation.

L-BFGS limited-memory Broyden-Fletcher–Goldfarb-Shanno.

L-OBS Layer-wise Optimal Brain Surgeon.

L2P Learning to Prune.

LCNN Lookup-CNN.

LRN Local Response Normalization.

LUT look-up table.

LWC Learning both Weights and Connections.

MAC multiply–accumulate.

MIT Massachusetts Institute of Technology.

MKL Math Kernel Library.

ML machine learning.

MLP multi-layer percetron.

MMX Multi Media Extension.

MSE mean squared error.

NAS neural architecture search.

NCC normalized compute complexity.

NN neural network.

NoC network-on-chip.

Op operation.

Param parameter.

PCI Peripheral Component Interconnect.

PE processing element.

QNN quantized neural network.

RAO resource-aware optimization.

ReLU rectified linear unit.

RL reinforcement learning.

RNN recurrent neural network.

RTL register-transfer level.

SDK software development kit.

SGD stochastic gradient descent.

SIMD single instruction multiple data.

SoC system-on-chip.

SotA state-of-the-art.

SQNR signal-to-quantization-noise ratio.

SRAM static random-access memory.

SSE3 Streaming SIMD Extensions 3.

STE straight-through estimator.

SVM support vector machine.

SW software.

TDG training data generator.

TPU Tensor Processing Unit.

VHDL Very High Speed Integrated Circuit Hardware Description Language.

VLSI very-large-scale integration.

VPU Vision Processing Unit.

XPE Xilinx Power Estimator.

List of Symbols

A Inverse Winograd transformation matrix.

A^{l-1} Input feature map.

A^{l-1} Binarized version of the input feature map.

A_B^{l-1} Input feature map of binaryDAD's bottleneck.

A_D^{l-1} Input feature map of binaryDAD's decoder.

A_E^{l-1} Input feature map of binaryDAD's encoder.

A^l Output feature map.

$A^l_{\textbf{inter}}$ Intermediate features of an ALF-block.

B Winograd transformation matrix for activations.

B^l Set of binarized weights.

C_i Number of channels of an input feature map.

C_o Number of channels of an output feature map.

$C_{\textbf{code}}$ Number of channels of the low-rank approximation.

D Parallelization factor of WinoCNN.

G Winograd transformation matrix for weights.

H_i Height of an input feature map.

H_o Height of an output feature map.

I Input image.

K Kernel dimensions.

L Layer of a neural network.

M Parallelization factor of WinoCNN.

$M_{\textbf{ABC}}$ Number of binary weight bases of ABC-Net.

N Classes for classification.

$N_{\textbf{ABC}}$ Number of binary activation bases of ABC-Net.

$N_{\textbf{hadamard}}$ Number of bits per Hadamard product.

P Padding (measured in pixels).

$P_{\textbf{Orthrus}}$ Set of pixels of OrthrusPE.

Q Bit-width of quantized weights and activations.

S Stride (measured in pixels).

W Set of weights/trainable parameters.

W^l Set of weights of a given layer.

W_B^l Set of weights of binaryDAD-Net's bottleneck.

W_D^l Set of weights of binaryDAD-Net's decoder.

W_E^l Set of weights of binaryDAD-Net's encoder.

W_i Width of the input feature map.

W_o Width of the output feature map.

$W_{\textbf{code}}$ Low-rank approximation of the weights.

$W_{\textbf{dec}}$ Decoder filters of ALF.

$W_{\textbf{enc}}$ Encoder filters of ALF.

$W_{\textbf{rec}}$ Reconstructed parameters of ALF.

X Set of input neurons.

Y Ground-truth label.

Θ_l Batch norm parameters.

α Scaling factor for binarized weights.

β Scaling factor for binarized input activations.

ℓ_1 Absolute value criterion.

ℓ_2 Quadratic mean square criterion.

ϵ Precision of quantized values.

η_S Theoretical speedup of Amdahl's law.

λ Strength of regularization.

$\lambda_{\textbf{Prune}}$ Pruning rate dependent scaling factor for ALF.

$\lambda_{\textbf{wd}}$ Weight decay scaling factor for the training of an ALF-block.

$\mathbf{A}$ 2D-input feature map matrix for the GEMM operation.

$\mathbf{B}$ 2D-weight matrix for the GEMM operation.

$\mathbf{C}$ 2D-output feature map matrix for the GEMM operation.

$\mathcal{D}$ Dataset gathering paired samples of images I and corresponding labels Y.

$\mathcal{K}_H$ Estimation of the memory requirement of Huffman coded data.

$\mathcal{L}_{\textbf{CE}}$ Cross entropy loss.

$\mathcal{L}_{\textbf{ae}}$ Autoencoder-specific loss function.

$\mathcal{L}_{\textbf{prune}}$ Regularize the pruning mask.

$\mathcal{L}_{\textbf{reg}}$ Regularization term.

$\mathcal{L}_{\textbf{task}}$ Task-specific loss function.

$\mathcal{M}$ Binary mask used for pruning.

$\mathcal{O}$ Overall search complexity.

`BS` Batch size.

`FB` Number of fractional bits.

`IB` Number of integer bits.

`Pr` Pruning rate which is the ratio of pruned (zero) to the overall number of weights.

`S` Sign bit of quantized values.

μ SGD-based momentum.

ν Learning rate of CNN optimizer.

$\nu_{\textbf{ae}}$ Autoencoder learning rate.

σ Activation function.

τ Sparsity threshold.

θ Fraction of zeroized to non-zeroized fiters in an ALF-block.

$\tilde{A}^l$ Linear combinations of intermediate features.

$\tilde{Y}$ Prediction of the CNN.

$\tilde{\mu}_i$ Fixed-point quantized centroid.

ζ ALF-specific hyperparameter to distinguish if a filter is active or dropped.

a Single output pixel.

a^{l-1} Tensor slice/tile of the input feature map.

a^{l} Tensor slice/tile of the output feature map.

b Individual bias trainable parameter.

b^{l} Binarize slice of the weights.

c Index of a class/centroid.

d Dilation rate.

e Learning epochs, *i.e.* one iteration throughout all training samples.

f Frequency.

f_p Parallel fraction code.

f_s Sequential fraction code.

$f_{\textbf{max}}$ Upper bound of the quantization limits.

$f_{\textbf{min}}$ Lower bound of the quantization limits.

g Noise scale.

g_A Gradients of the activations.

g_B Gradients of the binary weights.

g_W Gradients of the weights.

h^{l-1} Binarize tensor slice of the input feature map.

i Index parameter.

l Index parameter for a layer.

m_i Individual elements of the binary pruning mask.

$m_{\textbf{slope}}$ Slope of the sensitivity of the ALF-specific pruning factor.

n_p Number of parallel threads/cores.

p Number of quantization intervals.

p_i Pseudo probability.

$pr_{\textbf{high}}$ Upper bound of binary search based pruning.

$pr_{\textbf{low}}$ Lower bound of binary search based pruning.

$pr_{\textbf{max}}$ Maximum desired pruning rate of an ALF-block.

$pr_{\textbf{mid}}$ Actual pruning rate of binary search based pruning.

t Training step.

v_t SGD-based accumulator at training step t.

w Individual weight/trainable parameter.

w^l Tensor slice of the weights.

x Individual input neuron.

y Individual output neuron of a neural network.