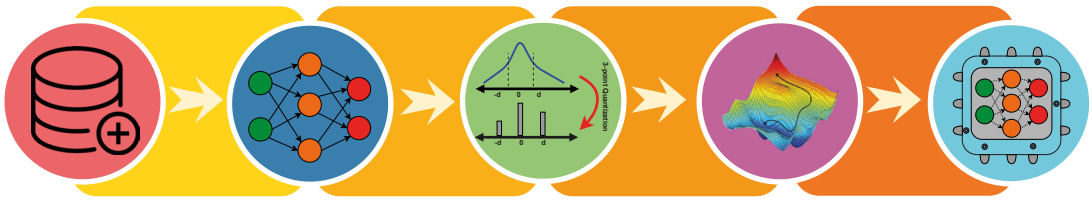
Mälardalen University Licentiate Thesis 299

DeepMaker: Customizing the Architecture of Convolutional Neural Networks for Resource-Constrained Platforms

Mohammad Loni



MÄLARDALEN UNIVERSITY
SWEDEN

Mälardalen University Press Licentiate Theses
No. 299

DeepMaker

CUSTOMIZING THE ARCHITECTURE OF CONVOLUTIONAL NEURAL NETWORKS FOR RESOURCE-CONSTRAINED PLATFORMS

Mohammad Loni

2020

School of Innovation, Design and Engineering

ISBN 978-91-7485-490-9
ISSN 1651-9256
Printed by E-Print AB, Stockholm, Sweden

Sammanfattning

Convolutional Neural Networks (CNNs) lider av energihungriga implementationer på grund av att de kräver enorm beräkningskapacitet och har en betydande minneskonsumtion. Detta problem kommer att framhävas mer när allt fler CNN implementeras på resursbegränsade plattformar i inbyggda datorsystem. I denna uppsats fokuserar vi på att minska resursåtgången för CNN, i termer av behövda beräkningar och behövt minne, för att vara lämplig för resursbegränsade plattformar. Vi föreslår två metoder för att hantera utmaningarna; optimera CNN-arkitektur där man balanserar nätverksnoggrannhet och nätverkskomplexitet, och föreslår ett optimerat ternärt neuralt nätverk för att kompensera noggrannhetsförluster som kan uppstå vid nätverkskvantiseringsmetoder. Vi utvärderade effekterna av våra lösningar på kommersiellt använda plattformar (COTS) där resultaten visar betydande förbättringar i nätverksnoggrannhet och energieffektivitet.

Abstract

Convolutional Neural Networks (CNNs) suffer from energy-hungry implementation due to requiring huge amounts of computations and significant memory consumption. This problem will be more highlighted by the proliferation of CNNs on resource-constrained platforms in, e.g., embedded systems. In this thesis, we focus on decreasing the computational cost of CNNs in order to be appropriate for resource-constrained platforms. The thesis work proposes two distinct methods to tackle the challenges: optimizing CNN architecture while considering network accuracy and network complexity, and proposing an optimized ternary neural network to compensate the accuracy loss of network quantization methods. We evaluated the impact of our solutions on Commercial-Off-The-Shelf (COTS) platforms where the results show considerable improvement in network accuracy and energy efficiency.

It is not the strongest of the species that survives,
nor the most intelligent that survives.
It is the one that is the most adaptable to change.
_ Charles Darwin _

Acknowledgments

This ongoing journey, as a pleasant part of my life, has been full of unique and memorable moments, and many people had supporting roles to play in different stages of the journey. First, my sincere thanks go to the great team of my supervisors. I would like to thank very much my main supervisor Prof. Mikael Sjödin for his big encouragement. I am deeply grateful to Prof. Masoud Daneshtalab for his very thoughtful technical guidance, caring, the excellence of patience, continuous energizing support and his acts of kindness as my co-supervisor. Thanks to both of you for believing in me and giving me the opportunity to progress.

I am very grateful to my colleagues, Dr. Arash Ghareh Baghi, Dr. Amin Majd, and Dr. Carl Ahlberg for their supports, discussions and feedbacks as co-authors in my published papers. Special thanks to my dear friends Dr. HamidReza Faragardi, and Mr. Masoud Ebrahimi for their advice and big motivation during my Ph.D. life.

Above all, I would like to express my deep gratitude to my parents, my brothers, my close friends, and Fateme Poursalim for all their supportive presence, understanding, and being patient with me. Without their support, I would not have reached here. My study at Mälardalen university has provided me with the opportunity of meeting new friends and working with great people. I would also like to thank all of them.

This research has been supported by Swedish Knowledge Foundation (KKS) via the DeepMaker and DPAC projects and the Swedish Research Council (VR) via the FAST-ARTS project.

Mohammad Loni, Västerås, December 2020

List of publications

Papers included in the thesis[1]

Paper A *Designing Compact Convolutional Neural Network for Embedded Stereo Vision Systems*, Mohammad Loni, Amin Majd, Abdolah Loni, Masoud Daneshtalab, Mikael Sjödin, Elena Troubitsyna. In the Proceedings of the 12th IEEE International International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC). Hanoi, Vietnam, September 2018. **Winner of the Best Paper Award**.

Paper B *NeuroPower: Designing Energy Efficient Convolutional Neural Network Architecture for Embedded Systems*, Mohammad Loni, Ali Zoljodi, Sima Sinaei, Masoud Daneshtalab, and Mikael Sjödin. In the Proceedings of the 28st International Conference on Artificial Neural Networks (ICANN). Munich, Germany, September 2019.

Paper C *DeepMaker: A multi-objective optimization framework for deep neural networks in embedded systems*, Mohammad Loni, Sima Sinaei, Ali Zoljodi, Masoud Daneshtalab, Mikael Sjödin. In the Microprocessors and Microsystems Journal (MICPRO), 2020, Elsevier.

Paper D *TOT-Net: An Endeavour Toward Optimizing Ternary Neural Networks*, Najmeh Nazari, Mohammad Loni, Mostafa E. Salehi, Masoud Daneshtalab, Mikael Sjödin. In the Proceedings of IEEE International Conference on Digital System Design (DSD 2019). Chalkidiki, Greece, August 2019.

[1] The included articles have been reformatted to comply with the thesis layout.

Paper E *DenseDisp: Resource-Aware Disparity Map Estimation by Compressing Siamese Neural Architecture*, Mohammad Loni, Ali Zoljodi, Daniel Maier, Amin Majd, Masoud Daneshtalab, Mikael Sjödin, Ben Juurlink and Reza Akbari. In the Proceedings of the IEEE World Congress on Computational Intelligence (WCCI). Glasgow (UK), July 2020.

Additional papers, not included in the thesis

1. *ADONN: Adaptive Design of Optimized Deep Neural Networks for Embedded Systems*, Mohammad Loni, Masoud Daneshtalab, Mikael Sjödin. In the proceeding of IEEE Conference on Digital System Design (DSD). Prague, Czech, August 2018.

2. *SoFA: A Spark-oriented Fog Architecture*, Neda Maleki , Mohammad Loni, Masoud Daneshtalab, Mauro Conti , Hossein Fotouhi. In the IEEE 45th Annual Conference of the Industrial Electronics Society (IECON). Lisbon, Portugal, October 2019.

3. *Embedded Acceleration of Image Classification Applications for Stereo Vision Systems*, Mohammad Loni, Carl Ahlberg, Masoud Daneshtalab, Mikael Ekström, Mikael Sjödin. In the proceeding of IEEE Design, Automation Test in Europe Conference Exhibition (DATE). Dresden, Germany, March 2018.

Contents

I

Thesis

Chapter 1

Introduction

Deep Neural Networks (DNNs) are increasingly becoming favored over machine learning methods in a wide range of applications such as intelligent transportation [1], natural language processing [2], medical diagnosis [3, 4], and e-commerce [5]. The main reasons of DNNs superiority comes from their high flexibility, more generalization proficiency for large-scale tasks and requiring less human intervention. Convolutional Neural Networks (CNNs) are a subset of DNN algorithms that their advantage in visual recognition and image classification tasks is obvious to everyone.

The benefits of CNNs are predicated upon performance efficiency and energy consumption delivered from hardware platforms. Recently, CNNs are becoming more complex models containing hundreds of deep layers and millions of floating-point operations to provide more accurate results. However, the failure of traditional performance and energy scaling paradigm in affording of computing requirements for modern applications leads CNN hardware implementation towards inefficiency [6]. The problem is more pronounced in deploying CNNs on resource-constrained platforms due to the limited processing and/or power budget. Many prior works attempted to reduce the computational complexity and frequent memory accesses of CNNs (see Section 3). Generally, to enhance the efficiency of the CNN implementation, academia and industry put forward four solutions:

1. Many CNN hardware accelerators are proposed to overcome the compu-

tational cost and huge memory-footprint of CNNs by parallel computing and efficient data reuse [7, 8, 9].

2. Network pruning is a practical method to minimize the size of the network and refine the network accuracy by removing the redundant network connections and fine-tuning the weights [10, 11, 12].

3. Designing optimized CNN architecture for resource-constrained platforms [13, 14, 15].

4. Neural network quantization techniques would also remarkably reduce memory-footprint and hence improve the energy efficiency and the model inference time [16, 17, 18, 19].

The main focus of this thesis is to propose a framework, named DeepMaker, to design the CNN architecture for resource-constrained platforms. DeepMaker benefits from the third (Paper A [20], Paper B [21], Paper C [22], and Paper E [23]) and the forth (Paper D [24]) approaches to make the hardware implementation of a CNN on resource-constrained platforms possible. Figure 1.1 shows the processing pipeline of the DeepMaker framework.

There exist a variety of customized CNN architectures for different tasks. However, finding a cost-efficient architecture is still challenging due to the lack of a general designing solution. In addition, many parameters have to be chosen in advance, while manual decisions of each of these parameters is extremely time-consuming and needs expertise. Neural Architecture Search (NAS) methods are proposed to build neural models without human intervention [25, 26]. NAS methods try to design the neural architecture with competitive or even better accuracy against the best results designed by experts.

DeepMaker leverages multi-objective evolutionary-based NAS techniques in the first processing stage to balance a trade-off between implementation efficiency and accuracy of CNNs. Paper A [20], Paper B [21], Paper C [22], and Paper E [23] focus on the first stage.

Network quantization is an impressive network compression method trying to reduce the memory-footprint of CNNs. The goal of network quantization is to represent the floating-point weights and/or activation functions with fewer bits. However, most of the network quantization techniques do not provide acceptable accuracy level. Paper D [24] proposes an optimized ternarizing

method that amortize the quantization accuracy loss of CNNs, as a disadvantages of quantization techniques (Stage 2).

The CNN training dynamic depends on properly selecting training parameters such as learning rate, momentum and weight initialization. DeepMaker automatically tweaks to the learning rate parameter in Stage 3. Paper B [21], Paper C [22], and Paper D [24] cover this contribution.

Finally, DeepMaker is able to deploy the optimized CNN architecture on a wide range of hardware platforms including CPU (x86, AArch64), GPU, embedded GPU, and FPGA (Stage 4).

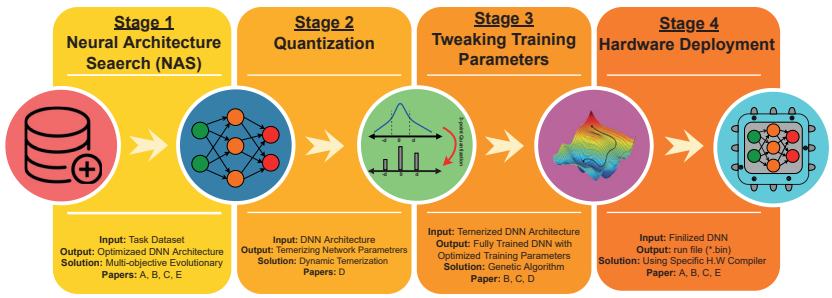


Figure 1.1: The overview of the DeepMaker framework.

In Section 1.1 and Section 1.2, we discuss the research challenges and motivations of using evolutionary-based techniques with regard to the existing issues of common techniques in optimizing CNN architecture.

1.1 Research Challenges

Figure 1.2 represents the overview of NAS structure. NAS starts with a set of predefined operations in order to form the search space. NAS uses a search strategy to explore among a large number of candidate architectures. All selected candidate architectures are trained and ranked. To evaluate the network architecture, we perform performance evaluation on the test set. Then, the search strategy is updated according to the ranking information of the previous candidates to obtain a set of new candidate architectures. The most promising network architecture is delivered to user as the final optimal architecture after

terminating the search process.

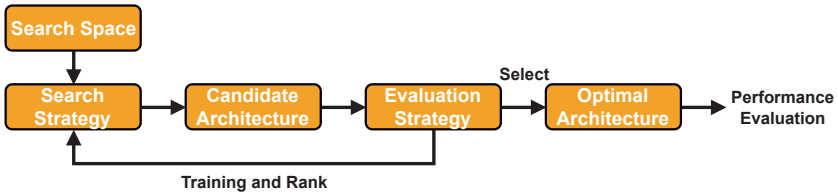


Figure 1.2: The overview of the NAS framework.

According to the NAS structure, designing the CNN architecture involves four essential challenges. In the rest of this section, we address the main barriers of designing CNN architecture and our proposed solutions that tackles this issues. The main NAS challenges are:

1. **Properly selecting search space and involved hyper-parameters.** The search space is defined by the predefined architectural hyper-parameters and corresponding operation set. For example, architectural template, kernel size and the number of channels of the convolutional layer, and connectivity method of operations are among the most important search space parameters. The influence of the search space on the final NAS performance is critical since these parameters determine which architectures can be searched by the NAS [25]. Therefore, the proper selecting the search space is necessary. We classify the search spaces in two essential categories including discrete and continuous search spaces. Discrete NAS search strategies are mainly categorized as *macro NAS* and *micro NAS* [27].

    - *Macro NAS* strategies directly search the entire neural network architecture. In other words, NAS finds an optimal network architecture within a huge search space with the granularity of operations. Although the *macro NAS* strategies yield a flexible search space, the larger the search space enforce the higher search cost. Figure 1.3 illustrates examples of two common *macro NAS* search spaces with a chain-based connection structure. Figure 1.3.a shows a simple example of a chain-based architecture. Figure 1.3.b shows

a chain-based architecture with supporting skip connections to provide more diversity.

- *Micro NAS* strategies, so-called cell-based NAS, use pre-learned neural cells, where each cell is usually well-optimized for comparatively proxy tasks. Figure 1.4 shows an example of NASNet [28] as one of the first studies using this *micro NAS* idea. *Micro NAS* strategies try to find the optimal interconnection among neural cells by stacking many copies of the cells. Although *micro NAS* strategies highly decrease the search time, they might not be optimal for unseen tasks [23, 29].

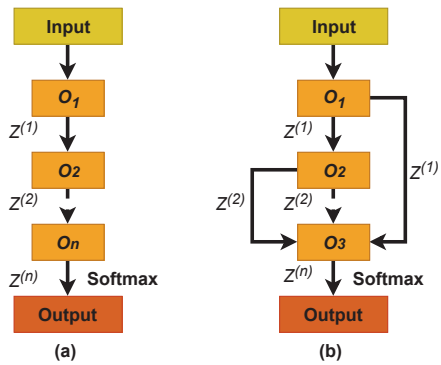


Figure 1.3: (a) A simple example of a chain-based architecture. O_i is an operation and the i^{th} operation in the architecture and $z^{(i)}$ is the o_i output feature map. (b) Extending the example by adding skip connections to provide more diversity. The input passes a series of operations to obtain the final output.

Therefore, there is a trade-off between selecting *macro NAS* or *micro NAS* search spaces since it has a high impact on search cost and quality of results. On the other hand, we have continuous search spaces which are almost optimized by gradient decent algorithm [25, 30]. DARTS [31] as one of the earliest implementation of the continuous search space, tries to continuously relax an originally discrete search space. DARTS uses gradients to efficiently optimize the search space. DARTS utilize the NASNet cell-based search space [28]. DARTS learns a neural cell as the key building block of the final architecture. The learned cells are

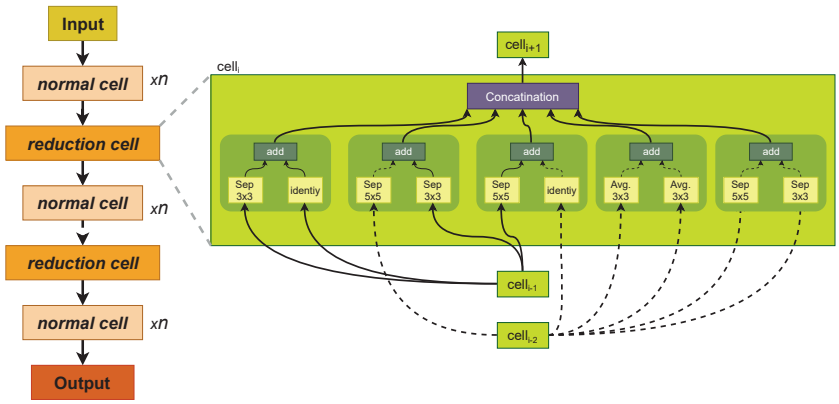


Figure 1.4: The structure of the search space leveraged in NASNet [28]. The search space is based on two cells including normal cell and reduction cell. Normal cell extracts advanced features without changing the spatial resolution. Reduction cell reduces the spatial resolution. In order to design the final architecture, multiple normal cells followed by a reduction cell are repeated, and this structure is repeated multiple times.

stacked to form either a convolutional network or a recurrent network if recursively be connected.

This cell is represented by a directed acyclic graph (DAG) constructed by N sequentially connected nodes. DARTS assumes each cell has two input nodes and one output node. To construct a convolutional cell, the input nodes are the output of the cells in previous two layers. To construct a recurrent cell, one input is from the current time step, and the other input is feed-backed from the previous time step. The output of cell is calculated by applying a concatenation operation to all intermediate nodes. For a discrete search space, each intermediate node can be expressed as $x^{(j)} = \sum_{i<j} o^{(i,j)}(x^{(i)})$ where $x_{(j)}$ is a potential feature representation in the cell, and $x_{(i)}$ is previous intermediate node $x_{(i)}$ through a directed edge operation $o_{(i,j)}$.

Therefore, to learn the cell architecture, operations on the DAG edges should be learned. DARTS makes the discrete search space continuous by relaxing the selection of candidate operations to a softmax of all possible operations. Figure 1.5 presents continuous relaxation and dis-

cretization of search space in DARTS [31].

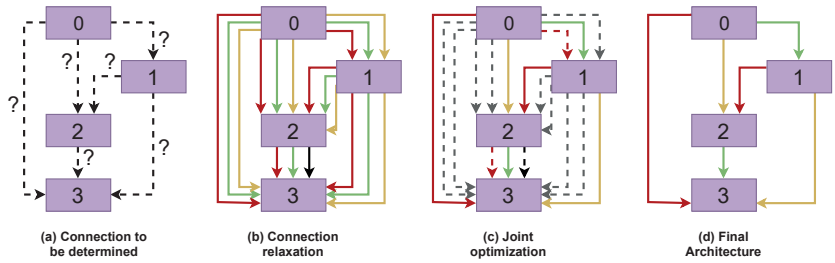


Figure 1.5: (a) The structure of a cell aiming to be learned. The operations on edges are unknown. (b) Illustrating the continuous relaxation of the cell-based search space. Each edge is a mixture of all candidate operations. (c) Joint optimizing the probability of mixed operations and network weights with gradient descent method. (d) Final network architecture.

2. **Properly selecting the search strategy.** The search strategy determines how to explore the search space which is often large or even unbounded. It is desirable to quickly find well-performing neural architectures, while trying to avoid being converged to a region of sub-optimal solutions. In other words, a suitable search strategy balances the exploration-exploitation trade-off. Recently, different search strategies are proposed to explore the space of neural architectures. Random search, Bayesian optimization, neuro-evolutionary methods, reinforcement learning (RL), and gradient-based methods are the most popular search strategies in community. In the following, these search strategies are briefly presented.

    - ***Random Search.*** Random search selects specific number of candidate architectures (a sample size) randomly from the architectural space. Random search evaluates the selected candidate architectures (e.g., by calculating accuracy). Then, it identifies the best architecture in the sample, stores it in the memory, and repeats this process. If the new architecture is better than the previous one, the previous architecture will be replaced by the new architecture. The search will be stopped after a pre-defined number of

iterations. Random search is proven to be a strong baseline for hyper-parameter optimization [32].

- ***Bayesian Optimization (BO).*** Bayesian Optimization is one of the most popular methods for hyper-parameter optimization. However, it has not been used by NAS experts since typical BO methods are based on Gaussian processes focusing on low-dimensional continuous problems.
- ***Reinforcement Learning (RL).*** RL methods are useful for modeling sequential Markov decision process where an agent interacts with an environment with the goal of maximizing its future benefit. To use RL for NAS problems, the design of a CNN architecture can be considered as the agent's action, with the action space identical to the search space. The agent's reward is the estimate of the performance of the trained architecture on test data.
- ***Neuro-Evolutionary Methods.*** Neuro-evolutionary methods are an alternative to RL approaches by using evolutionary algorithms for optimizing the neural architecture. Neuro-evolutionary algorithms consist of the following key operators including initialization, random parent selection, cross-over, mutation, and survivor selection. In general, neuro-evolutionary methods are highly sensitive to the choices for cross-over and mutation operators, and the fitness function that control the behavior of search process. Cross-over and mutation operators guide the diversity trade-off in the population. Similarly, the choice of fitness functions reflects the optimization objective.
- ***Gradient-Based Methods.*** While the methods above employ a discrete search space, Liu et al. [31] propose DARTS, a continuous relaxation to enable direct gradient-based optimization. DARTS optimizes both the network architecture and the network weights by alternating gradient descent steps on training data for weights and on validation data for architectural parameters.

3. **Properly selecting the evaluation strategy.** All the search strategies try to find a neural architecture that maximizes some performance measurements, such as accuracy. These strategies need to evaluate the perfor-

mance of a candidate architecture. The simplest way is to train the candidate architecture on training data and evaluate its performance on validation data. However, training each architecture require extensive amount of computing capacity, which is the main bottleneck of NAS methods. For example, NASNet [28] used RL to spend 2000 GPU days to design the best architecture in CIFAR-10 [33] and ImageNet [34]. Similarly, AmoebaNet [35] needs 3150 GPU days using neuro-evolutionary. This naturally raises the need of some methods for accelerating performance evaluation. For NAS, however, it is enough to know whether an candidate architecture is better or worse than the previous candidate. In general, there exist four techniques to reduce the evaluation cost during search process including:

(a) **Lower Fidelity Estimation:** Reducing the training time is performed by ❶ training with fewer epochs, ❷ training on a subset of dataset, ❸ down-scale models, and ❹ down-scale data. Although low-fidelity approximations remarkably reduce the computational cost, they also introduce bias in the estimation by performance underestimation. This may not be a problem if the search strategy only relies on ranking different architectures and the relative ranking remains stable and the difference between the approximations and the full evaluation is not too big [36].

(b) **Learning Curve Extrapolation:** Reducing the training time by performance extrapolation after just a few training epochs. Figure 1.6 shows an example of an early training termination in order to predict the final accuracy from the premature learning curve (solid line). This significantly reduces the number of required training iterations.

(c) **Weight Inheritance/Network Morphisms:** Initializing the weights of new candidate architectures based on weights of other architectures that have been trained before, e.g., a parent model, is another approach to speed up performance estimation. This avoids training from scratch.

(d) **One-Shot Models/Weight Sharing:** Treating all architectures as different sub-graphs of a super-graph (the one-shot model) and share the weights between architectures that have common edges

in the super-graph. This significantly improves performance estimation of architectures, since no training is required and only the evaluating performance on validation data is performed.

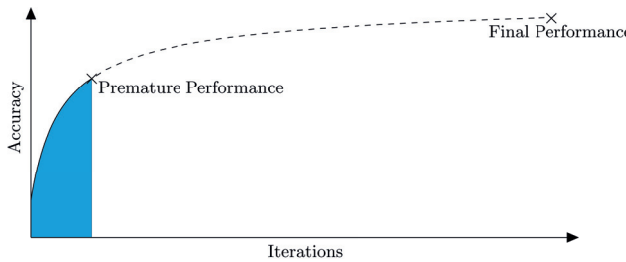


Figure 1.6: Example of early termination of training strategy to accelerate the performance of evaluations.

4. **Single/multi objective optimization.** For some applications, e.g., deploying a network on resource-constrained platforms, it is essential to consider other objectives, even with conflict, besides searching for high accurate networks. For example, the number of model parameters, the number of floating-point operations, and device-specific statistics like the inference time are among the popular objectives considered in some studies [37, 21, 23, 38]. To consider the additional objectives, the neural search problem is considered as a multi-objective optimization problem. In general, multi-objective NAS separates the decision making into two steps. First, a set of candidates is obtained without considering any trade-offs between the different objectives, then the decision for a superior solution is made in the second step.

Here, an imminent question is - which NAS structure is superior? In general, there is no clear answer to this question since it depends on the task, size of dataset, user constraints, search objectives, available computing power, etc. In our studies, we prefer to use neuro-evolutionary based method exploring discrete macro NAS search spaces since neuro-evolutionary methods are faster than RL and need less expertise and easily converge to near-optimal results if we tweak the hyper-parameters and fitness function of these methods. According to our recent study [23], neuro-evolutionary methods also provide

comparable results in contrast to gradient-based methods since gradient-based methods get stuck in local optima in most of the cases and need deep expertise for dynamic learning rate tuning and proper initialization. Table 1.1 summarizes our research contribution in this thesis according the specification of NAS structure.

Table 1.1: Summerizing the contribution regarding the NAS structure.

	Search Space	Search Strategy	Evaluation Strategy	Optimization Objective
Paper A	Discrete / macro NAS	Cuckoo Optimizer	Lower Fidelity Estimation	Accuracy and FLOPS
Paper B	Discrete / macro NAS	SPEA-II	Lower Fidelity Estimation	Accuracy and Network Energy Consumption
Paper C	Discrete / macro NAS	NSGA-II	Lower Fidelity Estimation	Accuracy and Network Parameters
Paper D	Discrete / macro NAS	Genetic Algorithm	Lower Fidelity Estimation	Accuracy
Paper E	Discrete / macro NAS	Simulated Annealing	Lower Fidelity Estimation	Accuracy and FLOPS

1.2 Motivation

Starting with AlexNet's win in the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC), CNNs have changed the landscape by providing superb capabilities in extracting high-dimensional structures from enormous data volume. Meanwhile, mobile embedded platforms such as smartwatches and medical tools are become ubiquitous. Therefore, there is a huge request for on-device deep learning services such as health monitoring, object recognition, and language translation [39, 40, 41]. Encouraged by the superb performance of CNNs in these services, people naturally are motivated to deploy deep learning on their mobile platform [42].

Although CNNs significantly increase the accuracy for image classification, visual recognition, and many other tasks [3, 43, 44], state-of-the-art results are accompanied by increasing complexity of CNNs. Figure 1.7 illustrate the accuracy and complexity of best models winning ILSVRC from 2010 to 2015. There are up to hundreds of millions of floating-point operations (FLOPS) in the advanced CNNs requiring considerable processing throughput and memory resource.

The nature of mobile embedded platforms imposes the intrinsic capacity bottleneck making resource-hungry applications banned. The large scale CNNs exceed the limited on-chip memory of mobile embedded platforms. Hence, they have to be deployed on the off-chip memory leading to consume more energy [21, 7]. As shown in Figure 1.8, there is a strong cor-

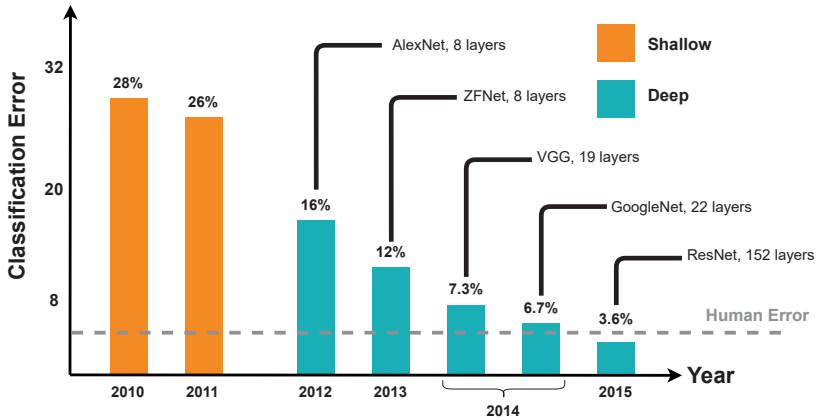


Figure 1.7: The performance and size of the CNNs in ILSVRC'10-15.

relation between GPU energy consumption and complexity of CNNs ($p - value$=0.000149, $Pearson\ Correlation$=0.942). Using cloud infrastructure to overcome the huge energy consumption of cutting-edge CNNs is not feasible since they are not intrinsically real-time solutions, there are privacy concerns about cloud processing paradigm, and permanent access to high-bandwidth Internet is not always guaranteed.

In this thesis, we aim to answer the following questions;

1. What is the best CNN architecture with the highest accuracy that is implementable on a mobile resource-limited (battery and memory) hardware platform?

2. How could we reduce the accuracy degradation of common quantization methods?

3. How could we deal with significant search cost (e.g., [35] needs 3150 GPU-days) of common NAS approaches ?

To answer these questions, we conduct a research on NAS methods for designing energy/performance aware CNN architectures. We leveraged multi-objective neuro-evolutionary search methods within a discrete search space to design both accurate and compact architectures in a short time. Plus, we accomplished a study on quantizing the weight and network activation functions

to achieve a higher level of resource efficiency. The output of our studies are published in five papers (see Section 2).

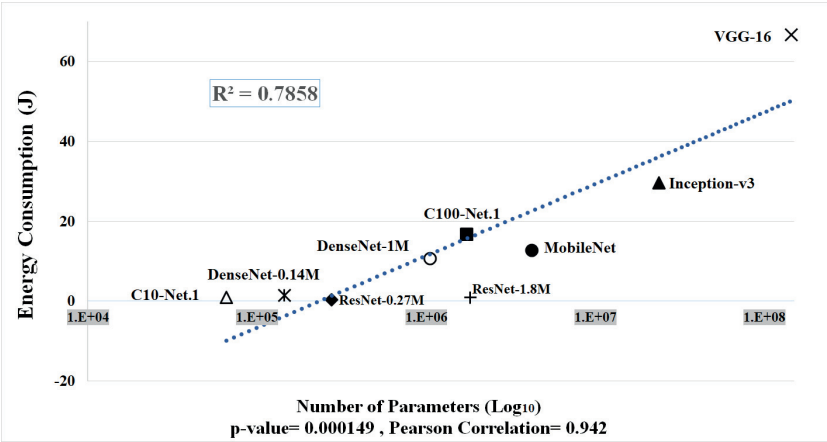


Figure 1.8: Reporting the accuracy vs. computational complexity represented by the number of parameters in the network. Executing a CNN, especially on embedded mobile platforms, can easily kill the whole system energy budget.

1.3 Research Process

For doing a scientific research and walking on the right path toward preparing a concrete thesis, leveraging research methodology is critical. The scientific method [45] provides how to facilitate with new questions and formulate the problems. Holz et al. [46] discuss the four major steps including problem formulation, propose solution, implementation and evaluation. Although, there was not solid research methodology at the beginning period of the Ph.D. study, we tried to follow a similar research methodology proposed by Holz in our research. We start with literature review on similar methods aiming to tackle the problem, then we continued with working on our idea to cover the weaknesses of the proposed solution. The implementation and evaluation phases were the last step in our research journey. Figure 7.2 illustrates the research methodology used in our research.

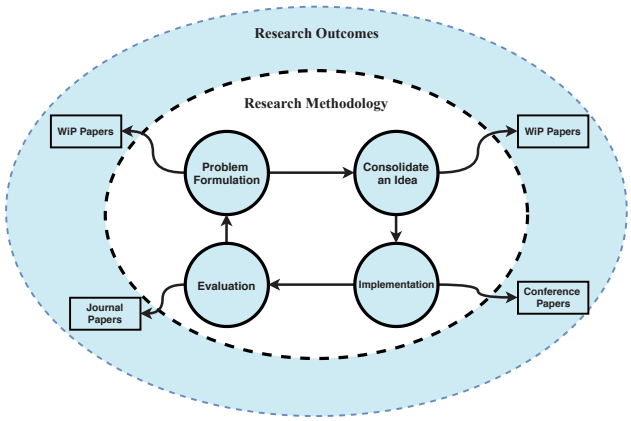


Figure 1.9: Research Methodology.

1.3.1 Problem definition

As the first step, we have done a review of both the state of the art and practice including the reason/problem for initiation of our research. We first investigate computer architecture conferences such as ISCA, DATE, DAC, MICRO, FCCM, FPL, FPGA, ASPLOS, CVPR, ICCV and so on. The referenced papers in the collected papers are included in survey study as well. Then, We discussed with other researchers with overlapping research filed. The research goal(s) are formulated as an outcome of problem formulation step. Plus, we found some ideas for the subsubgoals.

1.3.2 Consolidate an idea

After literature review, we focused on the key papers with remarkable results to consolidate our ideas. Then, we summarized subgoals as a subgoal. In Paper A, and B, we proposed our new method to improve the current state of the art by considering the second optimization objective. In Papers C, we extended the essential idea of paper B. In addition, Paper E extends the idea of paper A. Finally, We have proposed the optimization techniques presented in Paper D to improve the accuracy of quantized CNNs.

1.3.3 Implementation

The practical implementation results on embedded platforms are presented based on either hardware implementation (Paper A, B, C, E) or software implementation (Paper D). Measurements based on practical experience helped us to understand the real impact of our proposed solutions.

1.3.4 Evaluation

Comparison studies using the introduced metrics are considered in the evaluation step. Depending on the results of evaluation step, the problem formulation and proposed solution could be revised and continued with the later steps. This process is iterated until the results are acceptable. The results/outcomes of each step could be presented as papers, reports and presentations in work-in-progress sessions, workshops, conferences and journals.

1.4 Research Goals

The main challenge of the thesis is to accelerate CNNs on the COTS embedded platforms such as Field Programmable Gate Arrays (FPGAs), Graphics Procesisng Unit (GPU), and ARM processor. Due to limited time for Ph.D. study, we focus on the computing performance and energy efficiency aspects of embedded platforms. The overall goal of the thesis is formulated as follows:
Overall goal: Design and implementation of a optimization framework that accelerate CNN inference on COTS embedded devices while maintaining network validation accuracy. For more clarification, the overall goal is divided into the four following subgoals:

- **Subgoal 1:** Analyzing the characteristics of CNNs focusing on computing potential and power consumption in order to identify the bottlenecks of CNNs.
- **Subgoal 2:** Proposing a NAS method to optimize the network architecture at design time in order to improve the energy efficiency and memory-footprint of CNNs.

- **Subgoal 3:** Decreasing the computational cost of CNNs by leveraging network quantization techniques while considering to provide higher level of accuracy and simpler computation units.
- **Subgoal 4:** Evaluating how the proposed solutions save the validation accuracy while decreasing high energy consumption and huge memory-footprint of CNNs.

1.5 Thesis Outline

This thesis is divided into two parts. The first part is a summary of the thesis and is organized in four chapters, which are as follows: Chapter 1 gives an overview of the preliminaries, research challenges, research goals, motivations, and the research process which directed our research. In Chapter 2, we describe the contributions of the thesis to realization of the research goals. Chapter 3 presents an overview of the related work and background concepts. Finally, in Chapter 4, we conclude the first part of the thesis with a discussion on our results as well as possible directions for the future work. The second part of the thesis is given as a collection of the included publications which present the technical contributions of the thesis in detail.

Chapter 2

Research Contribution

In this section, we present our contributions (Paper A-E) in order to achieve the research goals as mentioned in Section 1.4.

2.1 Contributions Addressing the Research Goals

2.1.1 Contribution of subgoal 1

In order to find the processing bottlenecks, we analyzed the characteristics of CNNs. As a result, we figured out CNNs are complex models with huge memory-footprint where the convolutional layers are mainly computational intensive and fully-connected layers are memory intensive. In addition, the results represented in Paper B and Paper C indicate that the total number of network floating-point operations and neural network parameters have strong correlation with network energy consumption and network inference time. CIFAR-10 and CIFAR-100 [47] are the most popular datasets which have been considered in Paper B and Paper C.

2.1.2 Contribution of subgoal 2

Different optimization techniques have been proposed to design the architecture of CNNs such as reinforcement learning, random search, Bayesian optimization, and evolutionary methods. However, the time-consuming search of

design space is main challenge of related studies. Plus, most of the related studies aim to increase the validation accuracy.

Based on the achievements of the subgoal 1 and literature review, we first proposed a multi-objective neuro-evolutionary method to design CNN architecture considering improving validation accuracy and less network complexity as design objectives. The evolutionary-based optimization methods are preferred compared to other methods due to providing a guided search scheme. Next, we tweak the search hyper-parameters and fitness function in order to maximize the search efficiency. Paper A, B, C, and E cover the subgoal 2.

2.1.3 Contribution of subgoal 3

Based on the achievements of the subgoal 1 and literature review, we proposed a novel network quantization method as a potential method for decreasing computation and memory-footprint of CNNs. Recently, many proposed methods (see Section 3) try to address these issues. Although they have significantly decreased computational load of CNNs, they have suffered from accuracy loss especially for large datasets. We propose a ternarized neural network with [-1, 0, 1] values for both weights and activation functions that has simultaneously achieved a higher level of accuracy and less computational load. Moreover, we propose a simple bitwise logic for convolution computations to reduce the cost of multiply operations. As the second contribution, we propose a novel piecewise activation function, and optimized learning rate for different datasets to improve the accuracy of ternarized neural network. Paper D covers the subgoal 3.

2.1.4 Contribution of subgoal 4

Once we figured out the fundamental behavior of CNNs and proposed optimization solutions which are based on network architecture optimization and network quantization, the evaluation between the state-of-the-art and the proposed solutions is conducted. In this subgoal, we consider three essential metrics for comparisons including network compression rate, energy efficiency and computing performance (inference time). All the evaluations are conducted on the COTS embedded platforms including Xilinx Zynq FPGA, Nvidia GPU, and ARM Processor. As a result, we confirm that notable decrease in network com-

putational complexity by using our proposed methods in Papers A, B, C, D and E. In addition, this subgoal shows a trade-off between the network validation accuracy and network complexity.

2.2 Overview of the Included Papers

The main contributions of the thesis are organized and presented as a set of papers which have been included in the thesis. Other papers that have been just listed at the beginning of the thesis and are not included, also strengthen the contributions of the thesis. A summary of the included papers is as follows:

2.2.1 Paper A

Designing Compact Convolutional Neural Network for Embedded Stereo Vision Systems [20].

Abstract. Autonomous systems are used in a wide range of domains from indoor utensils to autonomous robot surgeries and self-driving cars. Stereo vision cameras probably are the most flexible sensing way in these systems since they can extract depth, luminance, color, and shape information. However, stereo vision based applications suffer from huge image sizes and computational complexity leading system to higher power consumption. To tackle these challenges, in the first step, GIMME2 stereo vision system [48] is employed. GIMME2 is a high-throughput and cost efficient FPGA-based stereo vision embedded system. In the next step, we present a framework for designing an optimized Deep Convolutional Neural Network (DCNN) for time constraint applications and/or limited resource budget platforms. Our framework tries to automatically generate a highly robust DCNN architecture for image data receiving from stereo vision cameras. Our proposed framework takes advantage of a multi-objective evolutionary optimization approach to design a near-optimal network architecture for both the accuracy and network size objectives. Unlike recent works aiming to generate a highly accurate network, we also considered the network size parameters to build a highly compact architecture. After designing a robust network, our proposed framework maps generated network on a multi/many core heterogeneous System-on-Chip (SoC). In addition, we have integrated our framework to the

GIMME2 processing pipeline such that it can also estimate the distance of detected objects. The generated network by our framework offers up to 24x compression rate while losing only 5% accuracy compare to the best result on the CIFAR-10 dataset.

Personal Contribution. I am the initiator, the main driver and the author of all parts in this paper. Mr. Amin Majd helped us in designing optimization fitness function and the other co-authors have contributed with valuable reviews.

2.2.2 Paper B

NeuroPower: Designing Energy Efficient Convolutional Neural Network Architecture for Embedded Systems [21].

Abstract. Convolutional Neural Networks (CNNs) suffer from energy-hungry implementation due to their computation and memory intensive processing patterns. This problem is even more significant by the proliferation of CNNs on embedded platforms. To overcome this problem, we offer NeuroPower as an automatic framework that designs a highly optimized and energy efficient set of CNN architectures for embedded systems. NeuroPower explores and prunes the design space to find improved set of neural architectures. Toward this aim, a multi-objective optimization strategy is integrated to solve Neural Architecture Search (NAS) problem by near-optimal tuning network hyperparameters. The main objectives of the optimization algorithm are network accuracy and number of parameters in the network. The evaluation results show the effectiveness of NeuroPower on energy consumption, compacting rate and inference time compared to other cutting-edge approaches. In comparison with the best results on CIFAR-10/CIFAR-100 datasets, a generated network by NeuroPower presents up to 2.1x/1.56x compression rate, 1.59x/3.46x speedup and 1.52x/1.82x power saving while loses 2.4%/-0.6% accuracy, respectively.

Personal Contribution. I am the initiator, the main driver and the author of all parts in this paper. Mr. Ali Zoljodi helped me in preparing the results

of network pruning algorithm and the other co-authors have contributed with valuable discussion and reviews.

2.2.3 Paper C

DeepMaker: A multi-objective optimization framework for deep neural networks in embedded systems [22].

Abstract. Deep Neural Networks (DNNs) are compute-intensive learning models with growing applicability in a wide range of domains. Due to their computational complexity, DNNs demand implementations that utilize custom hardware accelerators to meet performance and response time as well as classification accuracy constraints. In this paper, DeepMaker framework is proposed, which aims to automatically design a highly robust DNN architecture for embedded devices as the closest processing unit to the sensors. DeepMaker explores and prunes the design space to find improved neural architectures. Our proposed framework takes advantage of a multi-objective evolutionary approach, which exploits a pruned design space inspired by a dense architecture. Unlike recent works that mainly have tried to generate highly accurate networks, DeepMaker also considers the network size factor as the second objective to build a highly optimized network fitting with limited computational resource budgets while delivers comparable accuracy level. In comparison with the best result on CIFAR-10 and CIFAR-100 dataset, a generated network by DeepMaker presents up to 26.4 compression rate while loses only 4% accuracy. In addition, DeepMaker maps the generated CNN on the commodity programmable devices including ARM Processor, High-Performance CPU, GPU, and FPGA.

Personal contribution: I am the initiator, the main driver and the author of all parts in this paper. Mr. Ali Zoljodi helped me in preparing the results of network pruning algorithm. Ms. Sima Sinaei helped me with a nice review and reorganizing the presentation structure of the paper.

2.2.4 Paper D

TOT-Net: An Endeavour Toward Optimizing Ternary Neural Networks [24].

Abstract. High computation demands and big memory resources are the major implementation challenges of Convolutional Neural Networks (CNNs) especially for low-power and resource-limited embedded devices. Many binarized neural networks are recently proposed to address these issues. Although they have significantly decreased computation and memory-footprint, they have suffered from accuracy loss especially for large datasets. In this paper, we propose TOT-Net, a ternarized neural network with [-1, 0, 1] values for both weights and activation functions that has simultaneously achieved a higher level of accuracy and less computational load. In fact, first, TOT-Net introduces a simple bitwise logic for convolution computations to reduce the cost of multiply operations. To improve the accuracy, selecting proper activation function and learning rate are influential, but also difficult. As the second contribution, we propose a novel piece-wise activation function, and optimized learning rate for different datasets. Our findings first reveal that 0.01 is a preferable learning rate for the studied datasets. Third, by using an evolutionary optimization approach, we found novel piece-wise activation functions customized for TOT-Net. According to the experimental results, TOT-Net achieves 2.15%, 8.77%, and 5.7/5.52% better accuracy compared to XNOR-Net on CIFAR-10, CIFAR-100, and ImageNet top-5/top-1 datasets, respectively.

Personal Contribution. Ms. Najme Nazari is the initiator and the main driver in this paper. I have done the optimization part with neuro-evolutionary method, obtaining the experiments, and also I was responsible for writing the paper. Other co-authors have contributed with valuable discussion and reviews.

2.2.5 Paper E

DenseDisp: Resource-Aware Disparity Map Estimation by Compressing Siamese Neural Architecture [23].

Abstract. Stereo vision cameras are flexible sensors due to providing heterogeneous information such as color, luminance, disparity map (depth), and shape of the objects. Today, Convolutional Neural Networks (CNNs) present the highest accuracy for the disparity map estimation [49]. However, CNNs require considerable computing capacity to process billions of floating-point operations in a real-time fashion. Besides, commercial stereo cameras produce huge size images (e.g., 10 Megapixels [20]), which impose a new computational cost to the system. The problem will be pronounced if we target resource-limited hardware for the implementation. In this paper, we propose *DenseDisp*, an automatic framework that designs a Siamese neural architecture for disparity map estimation in a reasonable time. DenseDisp leverages a meta-heuristic multi-objective exploration to discover hardware-friendly architectures by considering accuracy and network FLOPS as the optimization objectives. We explore the design space with four different fitness functions to improve the accuracy-FLOPS trade-off and convergency time of the *DenseDisp*. According to the experimental results, DenseDisp provides up to 39.1x compression rate while losing around 5% accuracy compared to the state-of-the-art results.

Personal Contribution. I am the initiator, the main driver and the author of all parts in this paper. I also did the experiments. Dr. Amin Majd helped me in designing optimization fitness function and the other co-authors have contributed with valuable reviews.

2.2.6 Mapping Contributions to Subgoals

Mapping of the research subgoals to the contributed papers are shown in Table 2.1.

Table 2.1: Mapping of the research goals to the contributions.

	subgoal 1	**subgoal 2**	**subgoal 3**	**subgoal 4**
Paper A		✓		✓
Paper B	✓	✓		✓
Paper C	✓	✓		✓
Paper D			✓	✓
Paper E		✓		✓

Chapter 3

Background and Related Work

In this chapter, we first discuss Deep Learning, in particular Convolutional Neural Networks (CNNs) and the role of them in different applications. Next, we present the fundamental of evolutionary optimization. Finally, we review the related work relevant to the contributions of the thesis.

3.1 Deep Learning

Learning is a task that humans are able to perform very well in most of the circumstances, but is difficult for computers to accomplish. Machine learning is the field devoted to the study how computers can learn and/or improve their performance that is of gaining knowledge, making predictions, making intelligent decisions or recognizing complex patterns from a set of data.

Deep Neural Networks (DNNs), aka Deep Learning, are a subset of machine learning algorithms which are proposed to classify multilevel input data. Recently, DNNs spurred interest in many learning tasks such as pattern recognition [50], image processing [51], image classification [52], speech processing [53], Natural language processing [54], and signal processing [55]. Advantages of DNNs against traditional machine learning techniques include that they require less domain knowledge for the problem they are trying to solve. In addi-

tion, DNNs easily scale because accuracy improvement usually is achievable either by augmenting the training dataset and/or the complexity of the network architecture. Shallow learning models such as decision trees and Support Vector Machines (SVMs) are inefficient for many modern applications; meaning that they require a large number of computations during training/inference, large number of observations for achieving generalizability, and imposing significant human labour to specify prior knowledge in the model [52].

In the rest of this section, we briefly present the theory behind neural networks. Afterwards, we introduce convolutional neural network as the target optimization task in this thesis.

3.1.1 Theory behind Neural Networks

Neural Network (NN), so-called Multi-layer Perceptron (MLP), is constructed by artificial neuron(s) grouped in one or more layers. Figure 3.1 pictures the functionality of an artificial neuron. An artificial neuron consists of input values, weights, a bias, and activation function. Each layer is either a input layer, hidden layer or output layer, where the hidden layer(s) extracts features of input data in order to produce the final output (see Figure 3.2.a). In general, the different layers of a MLP have different number of neurons. Regarding the functionality of artificial neurons, the input is multiplied to the weight, then added with the bias, which produces the the activation function input. Together the summation and activation function represent the transfer function defining the neuron output. Hence the characteristics of the NN are defined by the transfer function [56].

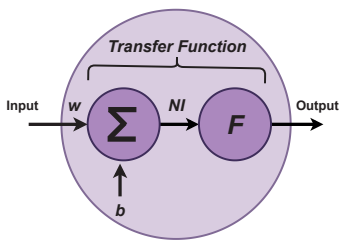


Figure 3.1: Structure of an artificial neuron, where w represents the weight, b the bias, NI the activation function input, and F the activation function.

Transfer Function

As we mentioned in Section 3.1.1, the summation and activation function composes the transfer function. According to [57], the activation functions are categorized in three classes, activation by inner product, distance, or a combination of both. Activation by inner product, also known as weighted activation, is commonly used technique that establishes the base of sigmoidal transfer functions. The base of Gaussian transfer functions is activation by distance defined as the euclidean distance between the input vectors and a reference. The neural network layers or artificial neurons are not required to utilize the same activation functions [58].

With that said, the proper selection of network activation functions significantly influences on the performance of NNs. There are some frequently used activation functions which concepts are presented in following [24, 59].

Sigmoidal activation function. It is a non-linear function which are repeatedly used in subsequent layers, affecting the output according to Equation 3.1. The function transforms the input to a value between zero and one, with the hard indications tendency. In other words, output is more likely to be a high value or a low value rather than a middle value.

$$F(n) = \frac{1}{1 + e^{-n}} \tag{3.1}$$

However, the main disadvantage of this approach is not responding to input values close to the function endpoints causing the vanishing gradients problem. The problem determines whether the neuron activates or not leading to slow down the learning process [56, 60].

Tanh activation function. Tanh is a scaled version of the sigmoidal activation functions. Therefore it inherits sigmoidal properties such as non-linearity. Tanh transforms the input value between minus one to one by using Equation 3.2.

$$F(n) = \frac{2}{1 + e^{-2n}} - 1 \tag{3.2}$$

Tanh suffers from the vanishing gradients problem due to the same reason as sigmoid. The main distinction between the two is their sensitivity to the input data. Tanh is more sensitive than sigmoid since it has sharper derivative [60].

Relu. [61] It is a popular non-linear activation function. The output of Relu produces i as output if i is positive and zero as output if i is negative

(Equation 3.3). Opposed to Tanh, the output of Relu is not shielded by the function which allows the output to be in the range of zero to infinity.

$$F(n) = max(0, n) \tag{3.3}$$

Due to not activating neurons when the input value is negative has with both benefits and drawbacks. Although less activated neurons are Superior for increasing efficiency in deep networks, it gives birth to a negative phenomenon named dying Relu. This phenomenon is the result of a zero gradient, which happens when the input of neuron is repeatedly a negative value causing the neuron to stop learning. leaky Relu [62] is a variation of Relu that attempts to avoid a zero gradient by multiplying the 0.01 value to the Relu negative inputs for minimizing sensitivity to the dying ReLU problem [60].

Neural Network Training

A dataset represents the prospective environment as well as required objects of interest to train the NN. The dataset is initially divided into two sets, train and test. The train data is divided into sets of training and validation. In general, it is optional how to divide the dataset to these three sets. Hagan et al. [56] propose to consider 70% for training, 15% for validation, and the remaining 15% for test. In some scenarios when we deal with huge datasets, we can consider 90% for training, 5% for validation, and the remaining 5% for test. NNs update initial weights which are usually selected at random based on the error made from the training dataset. The backpropagation algorithm alters the network weights in every epoch in order to recognize the optimal weights. Epoch is one iteration of the entire training dataset. For most of the time, the dataset is too large to be able to be processed by the hardware platforms at once. Thus one epoch is divided into batches or mini-batches [63]. The training will performed repeatedly until the model is deemed sufficient, or the learning is stopped [64, 65].

Performance Generalization

The performance of the NN is the model's ability to generalize, which is evaluated by measuring how the model performs on unseen data. To provide a robust model, increasing the generalization performance is critical. However, the generalization performance is affected by presence of errors, such as interpolation

and extrapolation errors [56]. Interpolation errors, known as overfitting, occurs when the prediction accuracy is high for the training dataset, and arbitrary exposed to a new dataset [66, 67, 56]. Extrapolation error, known as underfitting, happens due to a lack of variations in the training dataset. Underfitting causes low prediction accuracy [67, 56]. The concept of methods that can be directly applied to prevent the overfitting and underfitting problems are described below.

Dropout regularization. Dropout is a technique used during the NN training to prevent the over-fitting problem. Dropout removes neurons randomly during training to take samples from different narrowed down architectures. Figure 3.2 shows the difference between a network leveraging dropout at training time and a network that does not. The amount of neurons to drop is determined by the retain probability p. It is recommended to select a high p value for input layers and convolutional layers, while others get a standard probability of 0.5 [68, 69].

Batch normalization. Normalization is applied for each mini-batch in order to address the internal co-variance shift problem. This technique normalize the input to decrease the required training time that results in producing a non-deterministic output. Therefore, the effect or necessity of applying the dropout regularization technique may decrease when applying batch normalization [70].

Transfer learning. The main aim of transfer learning is to reduce the time of finding the optimal NN weights. Transfer learning reuses knowledge from a pre-trained model on a new task. The method may decrease the number of required training epochs [71].

Pre-training. In order to decrease the training error, we train a network on a dataset before re-training and fine-tuning the same NN on another dataset. This technique improves the generalization performance for smaller datasets, while large datasets reap more generalization benefits [72].

3.1.2 Convolutional Neural Network

In recent years, several deep learning models have been proposed to improve the accuracy of different learning tasks. Convolutional Neural Network (CNN) is one of the most popular deep learning architectures attained the state-of-the-art results in many application domains, especially in computer vision tasks

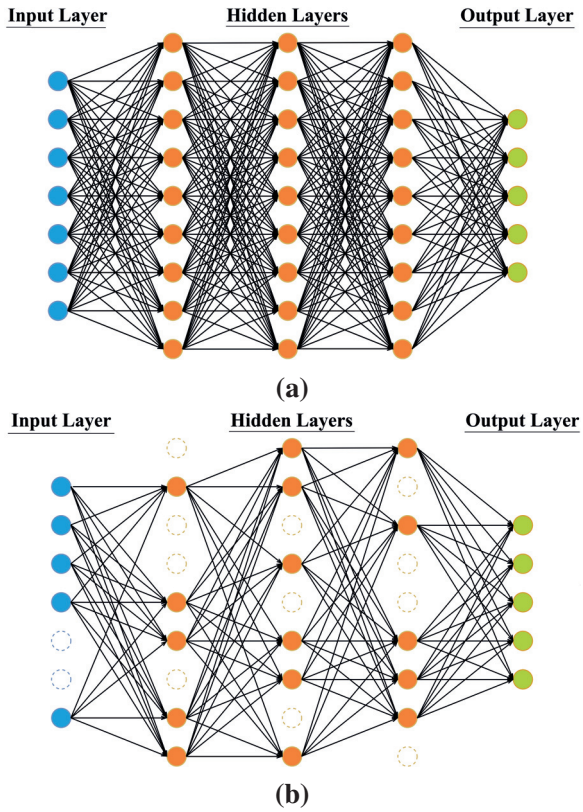


Figure 3.2: Illustrating a) an example of a NN architecture, and b) the same NN architecture during one mini-batch of the dropout regularization.

such as image and video classification, object recognition, and image segmentation [52]. Thus, CNNs have been used in a wide spectrum of platforms from high-performance workstations to mobile embedded devices.

In general, a CNN consists of multiple back-to-back layers connected in a feed-forward manner. The main layers are including: convolutional layer, normalization layer, pooling layer, and fully-connected layer. The convolutional layer is the principal layer of CNNs which extracts high-level abstraction of its inputs called feature map by using various filters. Equation 8.1 demonstrates

the operation of a 3D convolutional layer that convolves the inputs via a filter $W \in R^{C\times X\times Y}$ for each feature map where C, X and Y are the number of input channels and spatial dimensions of the filter, respectively. It is obvious that a lot of multiply and accumulate (MAC) operations are required to just obtain one point of the output feature map.

$$conv3D = f_{act}(\sum_{k=0}^{C-1}\sum_{i=0}^{X-1}\sum_{j=0}^{Y-1} I\,[k]\,[X-i]\,[Y-j] \times W\,[k]\,[i]\,[j]) \quad (3.4)$$

Where $conv3D$, I and W are the output feature maps, input feature maps, and $k\times k$ weight filters, respectively. Pooling layers perform down-sampling on data to decrease the amount of computation. Usually, in CNNs, pooling layers such as max pooling and average pooling are used after some convolutional layers. As demonstrated by their names, max pooling selects the maximum feature map and mean pooling computes the average of feature maps in the pooling window. Mostly, after distinguishing high-level abstraction features, fully-connected layers are applied to the CNN to classify images. A significant portion of computations, over 90%, are performed in the convolutional layers where fully-connected layers are mainly memory-bound [37]. In Fig. 3.3, a general architecture of the convolutional neural network is illustrated.

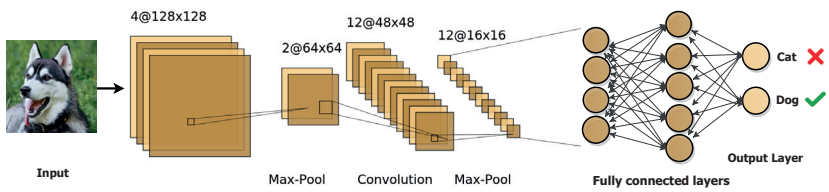


Figure 3.3: The general architecture of CNN.

3.2 Evolutionary Optimization

Optimization algorithms can be divided into 2 categories: heuristic and meta-heuristic methods. Heuristic algorithms are problem dependent and are often

greedy and prone to get stuck in local optima, failing to obtain the global optimum or even a near-optimal solution. Metaheuristic methods such as tabu search, simulated annealing, and genetic or memetic algorithms are problem-independent techniques or frameworks that improve performance of a heuristic search by allowing more thorough exploration of the search space and avoiding local optimum traps [73].

Computability is a significant challenge especially in NP-hard problems; there are no guarantees that such problems can be solved in a satisfactory manner in a limited time. Several techniques have been proposed to improve solving of NP-hard problems. Among these, evolutionary computing (EC) methods are the most prominent and popular. The EC methods are useful for solving various kinds of problems. For instance, well-known genetic algorithms (GA) are very suitable for discrete problems. They are population-based search methods that mimic the process of natural selection and evolution, as some characteristics of this process can be utilized in optimization problems. Simulated Annealing (SA) is another population-based evolutionary method that mimics the flocking behavior of birds when they migrate from a place to another [73].

In the rest of this section, we present two popular EC methods including GA, multi-objective GA, and SA which are used as the main search methods in this thesis.

3.2.1 Genetic Algorithm

GA is an iterative population-based exploration solution mimicking the process of natural selection and evolution where the characteristics of the process can be utilized in solving optimization problems. All GA-based methods have an initial population where selection, crossover, mutation operators are applied to initial population for producing improved population. The operations will be repeated until satisfying user criteria (reaching suitable results) or stopping after a predefined number of iterations. The following subsections explain the basic components of GA.

Step 1. **Generating Initial Population.**

The initial population includes random solutions in the design space, where each solution represented by chromosome is a solution for all the jobs. The size of initial population depends on the size of design space. To check the validity

of solutions in the initial population, each solution is examined by using the fitness function. Invalid solutions will be removed from the population.

Step 2. **Fitness Evaluation.** Objective function (fitness function) is a metric for comparing different solutions that satisfy problem constraints.

Step 3. **Selection.** Obviously the solutions with better fitness function are selected as the next generation and the others will be removed from population set. The goal is to find a solution in design space with lowest or highest fitness function.

Step 4. **Crossover Operator.** Is the most important operator of GA. GA randomly selects two genomes from the population set based on a certain crossover rate. Then two genome strings exchange parts of their corresponding chromosomes to create two new genomes.

Step 5. **Mutation Operator.** The main goal of mutation operator is to increase genetic diversity. Mutation alters one gene value (assigned processor to task) in a chromosome string from its initial state. The solution may be better or even worst solution by using mutation. Mutation forces GA to get rid of local optima. For doing mutation, we need to randomly select one gene in chromosome and modify its assigned value to a new valid number.

After each cycle of selection, crossover and mutation, the newly generated set of solutions is called as new generation. All the generations are evaluated based on the fitness function to determine if they represent a good enough solution to satisfy the fitness function. This determines if the GA can stop searching, or if otherwise, for the GA to continue searching until the predefined stopping criteria is met. The stopping criteria could be the number of generations, or evolution time, or fitness threshold, or fitness convergence, or population convergence. Algorithm 1 represent the Pseudocode of the GA.

3.2.2 Simulated Annealing (SA)

SA iteratively explores for a solution with fewer exploration objective value. Similar to the fitness function in genetic algorithm, exploration objective describes the optimality of a solution. If a reduction in exploration objective is found, the current solution is replaced with the new generated neighbour, otherwise the current solution is maintained. To avoid becoming trapped in a local optimum, SA sometimes accepts a bad solution which increases the value of exploration objective. The acceptance or rejection of a bad solution is de-

Algorithm 1 Pseudo-Code of the Genetic Algorithm (GA)

Input: Exploration space.
Result: Final set of optimized solutions.
GA Function (Input):
t := 0;
initialize ($P(t = 0)$)
evaluate ($P(t = 0)$)
while termination criterion is not satisfied **do**
  $P_{child}(t)$:=crossover ($P(t)$)
  Mutation ($P_{child}(t)$);
  evaluate ($P_{child}(t)$)
  $P(t + 1)$:= selection ($P_{child}(t)$, $P(t)$)
  t += 1;
end while
return solution with maximum fitness

pendent on a sequence of random numbers with a controlled probability. The acceptance probability is set to $exp(-\Delta/(k \times T))$ where T is the controlling parameter. T is the temperature inspired by the physical annealing process which is decreased logarithmic based on the predefined maximum and minimum temperatures (T_{Max} and T_{Min}). Thus, SA starts with a high value of T (T_{Max}) for preventing being prematurely trapped in a local optimum. Most uphill moves will be rejected by approaching T toward T_{Min}. SA proceeds until no further improvements can be found or it will be terminated after a certain amount of iterations. Although SA theoretically may fail to find an optimal solution for a given budget, the experimental results demonstrate that it succeeds to find a near-optimal solution. Algorithm 2 represent the Pseudocode of the SA.

Algorithm 2 Pseudo-Code of the Simulated Annealing (SA)

Input: Exploration space.
Result: Final optimized solution.
SA Function (Input):
j := random initial solution;
Initialize Boltzmann's constant k, reduction factor c, and temperature T
$T_{\text{Factor}} := -log(T_{\text{Max}}/T_{\text{Min}})$;
$Step$:= 0;
while termination criterion is not satisfied **do**
  $Step$ += 1;
  j' := Mutate(j);
  if $Energy(j') < Energy(j)$ **then**
    $j := j'$;
  else
    random $r(0, 1)$
    $\Delta := Energy(j') - Energy(j)$;
    if $r < exp(-\Delta/(k \times T))$ **then**
      $j := j'$;
    end if
  end if
  $T := T_{\text{Max}} \times exp(T_{\text{Factor}} \times (Step/Step_{\text{Total}}))$;
end while
return j

3.3 Related Work

In this section, we review the most significant recent related papers in the area of *automatic design of CNN architectures* and *CNN Quantization techniques*, respectively.

3.3.1 Automatic Design of CNN Architecture

To enable more accurate learning results, selecting the architectural parameters of CNNs are crucial since the network architecture strongly affects the inference time, memory-footprint, the accuracy level, and the network gener-

alization proficiency. However, the hand-crafted designing of CNN parameters is overwhelming due to requiring a lot of trial-and-error and deep expertise since the design space is huge. Therefore, an automatic method for designing CNN architectures has emerged as a significant alternative for decreasing efficiency risk and design cost. There are different automated neural optimization approaches, including random search, Bayesian optimization, RL, neuro-evolutionary methods. Using random search is challenging due to extremely random sampling in the search space [26], while Bayesian-based methods suffer from immense computational cost, suitable only for searching architectures with a fixed-length space and focuses on low-dimensional continuous problems [37]. RL-based methods are mainly slow and require considerable computational resources in both exploration and training steps [26, 37]. Evolutionary algorithms are feasible solutions for optimizing the neural architecture due to exploring improved design space without any prior assumptions. [74] proposed two new activation functions, ELiSH and HardELiSH, for tiny-ImageNet by leveraging an evolutionary method. In this paper, we use a similar strategy to optimize the ternary activation functions.

3.3.2 Neural Network Quantization

Generally, binarized neural networks (BNNs) suffer from the accuracy loss, especially for large datasets. [75] tried to address this issue for BNNs by proposing an efficient training strategy. Some works such as [76] and [77] applied reduced precisions to reduce memory storage and computation. However, they still require computation-intensive MAC operations to perform the convolutional operations, hence, suffer from heavy operations. Bit Fusion [16] demonstrated that 8 bits and less than it is enough for weights and activations in a wide range of CNNs, especially for small datasets. [78] uses bit widths less than 6 bits for quantization and achieves good accuracy compared to the full precision CNNs. LQ-Nets [17] applies proper quantization bits by a learnable quantizer and ReLeQ [18] automates DNN quantization based on a Reinforcement Learning (RL) algorithm, respectively. Ristretto [79] as a CNN approximation framework allows experimental exploration to trade between the classification accuracy and the bit-width of weights and activations. Ristretto also concludes that 8-bit dynamic fixed-point operations are appropriate for large-scale image classifications such as ImageNet.

Some recent researches [80, 81, 82] surpass quantized neural networks and have aggressively reduced precision even till 1 bit. BinaryConnect [82] eliminates multiplication in the forward pass by substituting full precision weights with -1 and 1 values. Ternary weight network [83] achieves more accuracy by applying ternary weights -1, 0, 1. Also, it adds sparsity to network, hence, it is more energy-efficient than binary connect. [84] presents Sparse Ternary Connect (STC) which reduces computation complexity by raising sparsity, and just leads less than 0.5% accuracy loss. Binarized neural network [80] binarizes both activations and weights and substitutes computation-intensive MAC operations with XNOR-bitcount operations. Therefore, the computations are drastically reduced, and also memory-footprint has been intensely decreased. XNOR-Net [81] achievesachieves more accuracy compared to BNN by using scaling factors for both activations and weights. TBN [85] proposes using ternary activations and binary weights, and hence, attains more accuracy for the ImageNet dataset compared to XNOR-Net. Since binarized neural networks can meet the embedded device constraints, hardware implementation of BNN has recently gotten more attention. XNOR Neural Engine [9] and BRein [86] are two samples in this area.

Chapter 4

Discussion, Conclusion and Future Work

In this chapter, we discuss our experimental results and conclude the thesis. Finally, we present a list of potential future research directions.

4.1 Discussion and Conclusion

According to the literature review, we identified room for developing Neural Architecture Search (NAS), in particular multi-objective NAS to help tackle the challenges of deploying a large-scale CNN on embedded mobile platforms.

NAS is a powerful tool to accomplish the intended aims in this area. Nonetheless, there is not a well-detailed solution which gives the maximum accuracy for any unseen task. In other words, many NAS specifications such as fitness function, design space operations, and termination condition depend on the task under study and on the user's constraints. In addition, most of the NAS methods try to improve network accuracy, while they do not consider the network complexity. Finally, the other artifacts such as system for running the NAS algorithm, which usually is a high-performance system, might not be accessible all the time. Network quantization, as the second utilized technique in our studies, is a popular technique for reducing the computational cost and memory footprint of CNNs. Despite providing remarkable computing cost

alleviation, quantization techniques suffer from huge accuracy loss.

Thesis Storyline. In this Section, we present the storyline of included publications.

- To our knowledge, ADONN [37] is the *first* framework aiming to reduce network inference time by compressing the size of architecture at design time. ADONN considers network parameters as the second search objective to balance accuracy and inference time trade-off.

- Next, we proposed NeuroPower framework [21] as the extension of ADONN idea in Paper B. NeuroPower leverages PAES-II as the search engine which finds more diverse architectures compared to NSGA-II utilized in the ADONN. In addition, we consider accuracy and the energy consumption of the network as the search objectives. NeuroPower provides superior results in comparison with ADONN.

- In Paper A, we decided to use a Matrix search space to improve the diversity and flexibility of solutions by finding non-symmetric architectures. We used Cuckoo algorithm as the search engine. Compared to two prior studies, Paper A searches in a huge discrete macro search space with more diverse solutions.

- DeepMaker is a template-based search engine that try to improve the ADONN quality of results by extending the range of hyper-parameters (Paper C). Furthermore, we utilized genetic algorithm in order to perform layer-wised activation tweaking. Finally, we compared the impact of network pruning and on a dense architecture designed by DeepMaker to check if we could achieve further compression rate.

- Paper D proposes TOT-Net framework which is a solution for ternarizing CNNs with [-1, 0, 1] values for both weights and activation functions. Then, TOT-Net introduces a simple bit-wise logic for convolutional layers to reduce the cost of multiply operations. To improve the accuracy, TOT-Net proposes a novel piece-wise activation function, and optimized learning rate for different datasets.

- Finally, DenseDisp tries to reduce huge neural search cost over complex datasets by leveraging multi-objective Simulated Annealing (SA) to dis-

cover hardware-friendly architectures (Paper E). SA is a fast and cost-efficient meta-heuristic search method. The main reason of DenseDisp's performance is the single solution based nature of SA. Although SA is quick search engine, it is sensitive to annealing parameter and its fitness function. The main contribution of DenseDisp is proposing an improved SA fitness function.

Regarding the involved issues in deploying CNNs on embedded platforms using NAS and quantization, a summary of the achievements is concluded as follows:

Performance Evaluation. To evaluate our proposed methods, we consider four popular classification datasets including MNIST [87], CIFAR-10 [33], CIFAR-100 [33], and ImageNet [34]. Plus, we consider to optimize CNN architecture for complex depth estimation task trained on KITTI 2015 [88]. We aim to present our top results compared to other cutting-edge architectures on different datasets. Table 4.1 present the performance efficiency of our proposed methods compared to other NAS approaches. According to the experimental results, DeepMaker strikes better balance between network accuracy and network complexity compare to RL and neuro-evolutionary strategies and hand-crafted designs.

Ternary neural networks provide high compression rate which makes them suitable for low-power embedded platforms. However, there is still a gap between the accuracy of state-of-the-art TNNs and the accuracy of full-precision networks. TOT-Net (Paper D) proposes ternarization on both weights and activation functions with significantly accuracy improvement [24]. In fact, we propose a simple bitwise logic (XOR and AND gates) instead of computation-intensive traditional multiplications with 16$\times$ memory efficiency compared to full-precision networks. According to the results, TOT-Net provides 1.8%, 7.5%, and 5.7% more accuracy compared to XNOR-Net for AlexNet architecture on CIFAR-10, CIFAR-100, and ImageNet datasets, respectively (Table 4.2).

Table 4.1: Error rate and compression rate for different studied datasets.

Dataset	Search Approach	Solutions	#Params ($\times 10^6$)	Error Rate (%)	Compression Rate* ‡
MNIST	Hand-Crafted	Wan et al. [89]	-	0.21	-
	RL	MetaQNN [90]	5.59	0.35	0.023x
	MO^2-EC∓	* ADONN-Arch.3 [37]	0.13	0.41	1.0x
	MO^2-EC	M_Net_1 (NeuroPower) [21]	**0.065**	**0.71**	**2x**
CIFAR-10	Hand-Crafted	* CondenseNet$^{\text{Light}}$ [91]	3.1	3.46	1.0
	Hand-Crafted	SimpleNet [92]	5.48	4.68	0.53x
	Hand-Crafted	DenseNet (k=12)-40 [93]	1.0	7.0	3.1x
	Hand-Crafted	ResNet-20 [94]	0.27	8.75	11.48x
	Hand-Crafted	ResNet-110 [94]	1.7	6.43	1.82x
	Hand-Crafted	Gastaldi et al. [95]	26.4	2.86	0.117x
	RL	Block-QNN-22L [96]	39.8	3.54	0.078x
	RL	MetaQNN [90]	6.92	11.18	0.45x
	RL	NAS-v1/v3 [97]	4.2/37.4	5.50/3.65	0.7x/0.083x
	RL	Block-QNN-S [96]	6.1	4.38	0.5x
	EC	Real et al. [98]	5.4	5.4	0.57x
	MO^2-EC	NSGA-Net [38]	3.3	3.85	0.94x
	MO^2-EC	ADONN-Arch.3 [37]	0.14	14.1	22.14x
	MO^2-EC	Loni et al.[20]	0.56	13.8	5.5x
	MO^2-EC	C10-Net.1 (NeuroPower) [21]	**0.065**	**16.49**	**47.7x**
	MO^2-EC	C10-Net.2 (NeuroPower) [21]	**0.21**	**11.05**	**14.7x**
	MO^2-EC	C10-Net.3 (NeuroPower) [21]	**1.0**	**6.81**	**3.1x**
CIFAR-100	RL	MetaQNN [90]	11.18	27.14	0.28x
	RL	Block-QNN-S [96]	6.1	20.65	0.5x
	Hand-Crafted	* CondenseNet$^{\text{Light}}$ [91]	3.1	17.55	-
	Hand-Crafted	DenseNet (k=12)-40 [93]	1.0	27.55	3.1x
	Hand-Crafted	DenseNet (k=12)-100 [93]	7.0	23.79	0.44x
	Hand-Crafted	SimpleNet [92]	5.48	26.58	0.53x
	MO^2-EC	NSGA-Net [38]	3.3	20.74	0.94x
	MO^2-EC	C100-Net.1(NeuroPower) [21]	**1.1**	**26.63**	**2.82x**
	MO^2-EC	C100-Net.2(NeuroPower) [21]	**1.89**	**24.87**	**1.64x**
KITTI 2015	RL	AutoDispNet-BOHB-C± [99]	37	2.18	1.0x
	Hand-Crafted	Content-CNN [100]	7	4.54	5.28x
	Hand-Crafted	GA-Net-deep [101]	-	1.81	-
	MO^2-SA	DenseDisp [23]	1.03	7.99	**35.9x**

* The baseline for comparing the compressing rate.
‡ The values more than 1.0 indicate improvement. Best results are in **bold**.
∓ Multi-objective evolutionary computing (MO^2-EC).
± The baseline for comparing the compressing rate.

Table 4.2: TOT-Net classification results compared to state-of-the-art quantization methods.

Method	Accuracy %, (Neural Network Architecture)		
	CIFAR-10	**CIFAR-100**	**ImageNet**
Full-Precision [102]	89.67, (NIN)	64.32, (NIN)	80.2/56.6, (AlexNet)† 89.2/69.3, (ResNet-18)†
BC [103]	91.73, (VGG-Net)	NA	66.3/43.1, (ResNet-18)† 61/35.4, (AlexNet)
TWN [104]	92.56, (VGG-Net)	NA	84.2/61.8, (ResNet-18)†
BNN [105]	89.85, (ConvNet)	NA	50.4/27.9, (AlexNet)†
XNOR-Net* [102]	85.74, (NIN)	54.10, (NIN)	62.52/37.47, (AlexNet)*† 73.2/51.2, (ResNet-18)
TNN [106]	87.89, (VGG like)	51.40, (VGG like)	NA
TOT-Net [24]	**87.53**, (NIN)	**61.61**, (NIN)	**68.20/42.99**, (AlexNet) *†
* The experiments have been run by us (trained by 20 epochs). † Presenting both top-5 and top-1 accuracy, respectively.			

4.2 Future Work

In the rest of the research journey, we mainly aim to focus on continuous search spaces due to providing cutting-edge results. In addition, decreasing the search cost of neuro-evolutionary based search methods is one of our interest. We also identifies room for some research directions on optimizing the architecture of time-based classifiers such as Long Short-Term Memory (LSTM). Therefore, we aim to extend the idea of DeepMaker to support designing optimal architecture for extracting temporal features. Language translation, speech recognition, and market analysis are among the applications that can benefit from DeepMaker LSTM optimization functionality.

Bibliography

[1] Hoang Nguyen, Le-Minh Kieu, Tao Wen, and Chen Cai. Deep learning methods in transportation domain: a review. *IET Intelligent Transport Systems*, 12(9):998–1004, 2018.

[2] Daniel W Otter, Julian R Medina, and Jugal K Kalita. A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[3] Zahra Ebrahimi, Mohammad Loni, Masoud Daneshtalab, and Arash Gharehbaghi. A review on deep learning methods for ecg arrhythmia classification. *Expert Systems with Applications: X*, page 100033, 2020.

[4] Mihalj Bakator and Dragica Radosav. Deep learning and medical diagnosis: A review of literature. *Multimodal Technologies and Interaction*, 2(3):47, 2018.

[5] Devashish Shankar, Sujay Narumanchi, HA Ananya, Pramod Kompalli, and Krishnendu Chaudhury. Deep learning based large scale visual recommendation and search for e-commerce. *arXiv preprint arXiv:1703.02344*, 2017.

[6] Hadi Esmaeilzadeh, Emily Blem, Renée St. Amant, Karthikeyan Sankaralingam, and Doug Burger. Power challenges may end the multicore era. volume 56, page 93, 2013.

[7] Hardik Sharma, Jongse Park, Divya Mahajan, Emmanuel Amaro, Joon Kyung Kim, Chenkai Shao, Asit Mishra, and Hadi Esmaeilzadeh. From high-level deep neural models to fpgas. In *The 49th Annual*

IEEE/ACM International Symposium on Microarchitecture, page 17. IEEE Press, 2016.

[8] Yu-Hsin Chen, Joel Emer, and Vivienne Sze. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In *ACM SIGARCH Computer Architecture News*, volume 44, pages 367–379. IEEE Press, 2016.

[9] Francesco Conti, Pasquale Davide Schiavone, and Luca Benini. Xnor neural engine: A hardware accelerator ip for 21.6-fj/op binary neural network inference. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2940–2951, 2018.

[10] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[11] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[12] Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5687–5695, 2017.

[13] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[14] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[15] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[16] Hardik Sharma, Jongse Park, Naveen Suda, Liangzhen Lai, Benson Chau, Vikas Chandra, and Hadi Esmaeilzadeh. Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural networks. In *Proceedings of the 45th Annual International Symposium on Computer Architecture*, pages 764–775. IEEE Press, 2018.

[17] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 365–382, 2018.

[18] Amir Yazdanbakhsh, Ahmed T Elthakeb, Prannoy Pilligundla, and FatemehSadat Mireshghallah Hadi Esmaeilzadeh. Releq: An automatic reinforcement learning approach for deep quantization of neural networks. *arXiv preprint arXiv:1811.01704*, 2018.

[19] Lei Deng, Peng Jiao, Jing Pei, Zhenzhi Wu, and Guoqi Li. Gxnor-net: Training deep neural networks with ternary weights and activations without full-precision memory under a unified discretization framework. *Neural Networks*, 100:49–58, 2018.

[20] Mohammad Loni, Amin Majd, Abdolah Loni, Masoud Daneshtalab, Mikael Sjödin, and Elena Troubitsyna. Designing compact convolutional neural network for embedded stereo vision systems. In *2018 IEEE 12th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, pages 244–251. IEEE, 2018.

[21] Mohammad Loni, Ali Zoljodi, Sima Sinaei, Masoud Daneshtalab, and Mikael Sjödin. Neuropower: Designing energy efficient convolutional neural network architecture for embedded systems. In *International Conference on Artificial Neural Networks*, pages 208–222. Springer, 2019.

[22] Mohammad Loni, Sima Sinaei, Ali Zoljodi, Masoud Daneshtalab, and Mikael Sjödin. Deepmaker: A multi-objective optimization framework for deep neural networks in embedded systems. *Microprocessors and Microsystems*, 73:102989, 2020.

[23] Mohammad Loni, Ali Zoljodi, Daniel Maier, Amin Majd, Masoud Daneshtalab, Mikael Sjödin, Ben Juurlink, and Reza Akbari. Densedisp: Resource-aware disparity map estimation by compressing siamese neural architecture. In *IEEE World Congress On Computational Intelligence (WCCI) 2020*, July 2020.

[24] Najmeh Nazari, Mohammad Loni, Mostafa E. Salehi, Masoud Daneshtalab, and Mikael Sjödin. Tot-net: An endeavour toward optimizing ternary neural networks. In *2019 22st Euromicro Conference on Digital System Design (DSD)*. IEEE, 2019.

[25] Martin Wistuba, Ambrish Rawat, and Tejaswini Pedapati. A survey on neural architecture search. *arXiv preprint arXiv:1905.01392*, 2019.

[26] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377*, 2018.

[27] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1761–1770, 2019.

[28] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.

[29] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

[30] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In *Advances in neural information processing systems*, pages 7816–7827, 2018.

[31] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[32] Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. *arXiv preprint arXiv:1902.08142*, 2019.

[33] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55, 2014.

[34] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[35] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.

[36] Arber Zela, Aaron Klein, Stefan Falkner, and Frank Hutter. Towards automated deep learning: Efficient joint neural architecture and hyperparameter search. *arXiv preprint arXiv:1807.06906*, 2018.

[37] Mohammad Loni, Masoud Daneshtalab, and Mikael Sjödin. Adonn: Adaptive design of optimized deep neural networks for embedded systems. In *2018 21st Euromicro Conference on Digital System Design (DSD)*, pages 397–404. IEEE, 2018.

[38] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. Nsga-net: a multi-objective genetic algorithm for neural architecture search. *arXiv preprint arXiv:1810.03522*, 2018.

[39] Bokai Cao, Lei Zheng, Chenwei Zhang, Philip S Yu, Andrea Piscitello, John Zulueta, Olu Ajilore, Kelly Ryan, and Alex D Leow. Deepmood: modeling mobile phone typing dynamics for mood detection. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 747–755, 2017.

[40] Lichao Sun, Yuqi Wang, Bokai Cao, S Yu Philip, Witawas Srisa-An, and Alex D Leow. Sequential keystroke behavioral biometrics for mobile user identification via multi-view deep learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 228–240. Springer, 2017.

[41] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[42] Nicholas D Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, Lei Jiao, Lorena Qendro, and Fahim Kawsar. Deepx: A software accelerator for low-power deep learning inference on mobile devices. In *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 1–12. IEEE, 2016.

[43] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, 9(4):611–629, 2018.

[44] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. {GAZELLE}: A low latency framework for secure neural network inference. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 1651–1669, 2018.

[45] G Dodig-Crnkovic. Scientific methods in computer science:[ ]/gordana dodig-crnkovic. In *Conference for the Promotion of Research in IT at New Universities and at University Colleges in Sweden, Skövde*, 2002.

[46] Hilary J Holz, Anne Applin, Bruria Haberman, Donald Joyce, Helen Purchase, and Catherine Reed. Research methods in computing: what are they, and how should we teach them? *ACM SIGCSE Bulletin*, 38(4):96–114, 2006.

[47] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 and cifar-100 datasets. *URl: https://www. cs. toronto. edu/kriz/cifar. html*, 6, 2009.

[48] Carl Ahlberg, Fredrik Ekstrand, Mikael Ekstrom, Giacomo Spampinato, and Lars Asplund. Gimme2-an embedded system for stereo vision and processing of megapixel images with fpga-acceleration. In *2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pages 1–8. IEEE, 2015.

[49] S Daniel, S Richard, and H Heiko. Middlebury stereo evaluation-version 3.

[50] Nasser M Nasrabadi. Pattern recognition and machine learning. *Journal of electronic imaging*, 16(4):049901, 2007.

[51] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image processing, analysis, and machine vision*. Cengage Learning, 2014.

[52] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[53] Dong Yu and Li Deng. *AUTOMATIC SPEECH RECOGNITION*. Springer, 2016.

[54] Li Deng and Yang Liu. *Deep Learning in Natural Language Processing*. Springer, 2018.

[55] Li Deng and Dong Yu. Deep learning for signal and information processing. *Microsoft Research Monograph*, 2013.

[56] Howard B Demuth, Mark H Beale, Orlando De Jess, and Martin T Hagan. *Neural network design*. Martin Hagan, 2014.

[57] Wlodzislaw Duch and Norbert Jankowski. Survey of neural transfer functions. *Neural Computing Surveys*, 2(1):163–212, 1999.

[58] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018.

[59] Mina Basirat and Peter M Roth. Learning task-specific activation functions using genetic programming. In *VISIGRAPP (5: VISAPP)*, pages 533–540, 2019.

[60] Avinash Sharma V. Understanding activation functions in neural networks, 2017. accessed: 20 May 2020.

[61] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[62] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[63] Sagar Sharma. Epoch vs batch size vs iterations, 2017. accessed: 20 May 2020.

[64] J.Brownlee. A gentle introduction to the challenge of training deep learning neural network models, 2019. accessed: 20 May 2020.

[65] J.Torres. Learning process of a neural network, 2018. accessed: 20 May 2020.

[66] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

[67] Haotian Zhang, Lin Zhang, and Yuan Jiang. Overfitting and underfitting analysis for deep learning based end-to-end communication systems. In *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–6. IEEE, 2019.

[68] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[69] Sungheon Park and Nojun Kwak. Analysis on the dropout effect in convolutional neural networks. In *Asian conference on computer vision*, pages 189–204. Springer, 2016.

[70] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[71] Lorien Y Pratt. Discriminability-based transfer between neural networks. In *Advances in neural information processing systems*, pages 204–211, 1993.

[72] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pretraining help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

[73] Amin Majd, Golnaz Sahebi, Masoud Daneshtalab, Juha Plosila, Shahriar Lotfi, and Hannu Tenhunen. Parallel imperialist competitive algorithms. *Concurrency and Computation: Practice and Experience*, 30(7):e4393, 2018.

[74] Mina Basirat and Peter M Roth. The quest for the golden activation function. *arXiv preprint arXiv:1808.00783*, 2018.

[75] Wei Tang, Gang Hua, and Liang Wang. How to train a compact binary neural network with high accuracy? In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[76] Yufei Ma, Naveen Suda, Yu Cao, Jae-sun Seo, and Sarma Vrudhula. Scalable and modularized rtl compilation of convolutional neural networks onto fpga. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–8. IEEE, 2016.

[77] Patrick Judd, Jorge Albericio, Tayler Hetherington, Tor M Aamodt, and Andreas Moshovos. Stripes: Bit-serial deep neural network computing. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–12. IEEE, 2016.

[78] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017.

[79] Philipp Gysel, Jon Pimentel, Mohammad Motamedi, and Soheil Ghiasi. Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 29(11):5784–5789, 2018.

[80] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016.

[81] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[82] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

[83] Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

[84] Canran Jin, Heming Sun, and Shinji Kimura. Sparse ternary connect: Convolutional neural networks using ternarized weights with enhanced sparsity. In *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 190–195. IEEE, 2018.

[85] Diwen Wan, Fumin Shen, Li Liu, Fan Zhu, Jie Qin, Ling Shao, and Heng Tao Shen. Tbn: Convolutional neural network with ternary inputs and binary weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 315–332, 2018.

[86] Kota Ando, Kodai Ueyoshi, Kentaro Orimo, Haruyoshi Yonekawa, Shimpei Sato, Hiroki Nakahara, Shinya Takamaeda-Yamazaki, Masayuki Ikebe, Tetsuya Asai, Tadahiro Kuroda, et al. Brein memory: A single-chip binary/ternary reconfigurable in-memory deep neural network accelerator achieving 1.4 tops at 0.6 w. *IEEE Journal of Solid-State Circuits*, 53(4):983–994, 2017.

[87] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998. *URL http://yann. lecun. com/exdb/mnist*, 10(34):14, 1998.

[88] Moritz Menze, Christian Heipke, and Andreas Geiger. Joint 3d estimation of vehicles and scene flow. In *ISPRS Workshop on Image Sequence Analysis (ISA)*, 2015.

[89] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pages 1058–1066, 2013.

[90] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016.

[91] Gao Huang, Shichen Liu, Laurens Van der Maaten, and Kilian Q Weinberger. Condensenet: An efficient densenet using learned group convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2752–2761, 2018.

[92] Seyyed Hossein Hasanpour, Mohammad Rouhani, Mohsen Fayyaz, and Mohammad Sabokrou. Lets keep it simple, using simple architectures to outperform deeper and more complex architectures. *arXiv preprint arXiv:1608.06037*, 2016.

[93] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017.

[94] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[95] Xavier Gastaldi. Shake-shake regularization. *arXiv preprint arXiv:1705.07485*, 2017.

[96] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2423–2432, 2018.

[97] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[98] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2902–2911. JMLR. org, 2017.

[99] Tonmoy Saikia, Yassine Marrakchi, Arber Zela, Frank Hutter, and Thomas Brox. Autodispnet: Improving disparity estimation with automl. *arXiv preprint arXiv:1905.07443*, 2019.

[100] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5695–5703, 2016.

[101] Feihu Zhang, Victor Prisacariu, Ruigang Yang, and Philip HS Torr. Ga-net: Guided aggregation net for end-to-end stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 185–194, 2019.

[102] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

[103] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

[104] Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

[105] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016.

[106] Adrien Prost-Boucle Hande Alemdar, Vincent Leroy and Frédéric Pétrot. Ternary neural networks for resource-efficient ai applications. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 2547–2554. IEEE, 2017.