Master thesis project proposal

Resource Optimal Neural Networks for Safety Critical Real Time Systems

Student 1

student1@student.gu.se

Suggested Supervisor at CSE:

Supervisor at the Division of Vehicle Safety,
Chalmers University of Technology

Supervisors at Company:

- Supervisor, Company
- Supervisor, Company

Relevant completed courses student 1:

- *DIT869, Deep Machine Learning*
- *DIT370, Discrete Optimization*
- *DIT430, High-Performance Parallel Programming*

January 22, 2020

1 Introduction

Deep neural networks (DNNs) provide state of the art solutions to many computer vision tasks, e.g. object recognition, where they have shown human-level performance on various benchmarks [3]. As a result of this domain excellence, they have become an integral component of automotive systems (e.g. active safety and self driving). Such systems will typically employ multiple DNNs, each trained for a specific task such as pedestrian detection [11] or road segmentation [7], to interpret the surroundings of the vehicle.

One limitation of DNNs is that they typically require storage, energy and computational resources well beyond the budget of most real time embedded systems. This limitation becomes particularly tangible in automotive systems, where many DNNs need to run at the same time. On the other hand, not all of those DNNs are safety critical during the entire driving cycle. This allows for dynamic trade-offs between different requirements (e.g. accuracy, latency, framerate, memory, energy and computational operations) on component level to achieve higher system goals.

2 Problem

Though the modeling capacity of DNNs is mainly attributed to the large number of learnable parameters [4], overparameterization is very common. In particular, studies have shown that the same modeling performance or higher often can be achieved with a smaller number of parameters [2, 5]. This problem seems to be particularly prevalent in *convolutional neural networks* (CNNs) [8], which are commonly used for vision tasks.

A concrete illustation of the overparameterization problem, as well as a proposed solution, is given by *deep compression* [5]. In this paper, a three-stage pipeline of weight pruning, trained quantization and Huffman coding is used to reduce the number of superfluous connections in pre-trained DNNs. Results showed that this compression pipeline is capable of reducing the size of DNNs by up to 49× without losing classification accuracy. The large size reductions allowed the models to fit onto *static random-access memory* (SRAM) caches, resulting in faster and more energy efficient inferences. One serious drawback of this approach is that the model needs extensive fine-tuning after the pruning and quantization stages which inhibits fast, dynamic compression during execution of the host system. It is also questionable whether *indirect* metrics such as compression rate can give an accurate indication of *direct* metrics (e.g. latency and framerate) which require careful monitoring in real time embedded systems. Nontheless, deep compression gives a rough indication of how much compression one could reasonably hope for with alternative algorithms.

3 Context

Two recent compression algorithms, which use direct metrics as a guidance in the compression process, are NetAdapt [12] and MnasNet [10]. Both of these algorithms were benchmarked on MobileNet [6], a CNN specifically designed for mobile devices.

3.1 NetAdapt

NetAdapt defines network compression as a constrained non-convex optimization problem, where the objective is to maximize accuracy given the constraint that the resource consumption must be lower than a prespecified budget. An iterative beam search is used to solve this optimization problem: in each iteration, a number of filters with the lowest L_2-norm are removed from each convolutional layer, creating one candidate network for each layer in the original network. The candidate with the highest accuracy is then selected for further compression in the next iteration. Through this design, NetAdapts generates a whole family of compressed networks with different trade-offs, which allows for a dynamic network selection based on the resources available at runtime. Results showed that NetAdapt is capable of reducing the inference latency of MobileNet by 1.7× while increasing classification accuracy on the ImageNet dataset [9]. One drawback of this design is that fine-tuning is used after each iteration as well as after the full compression. Furthermore, while the algorithm is claimed to work well on multiple resource constraints, it was only evaluated using latency requirements.

3.2 MnasNet

MnasNet defines network compression as a factorized hierarchical search problem. Conretely, a CNN is first factorized into separate blocks and an architectural search space is then defined for each block individually. A reinforcement learning agent is then employed to find multiple Pareto-optimal solutions [1] to the search problem. This agent uses a reward function based on accuracy-latency trade-offs and a set of actions corresponding to architecture altering operations such as filter resizing, layer reductions and insertion of skip operations (e.g. pooling). Results showed that MnasNet is capable of reducing the inference latency of MobileNet by 1.8× while increrasing classification accuacy on the ImageNet dataset. It was shown that factorizing the search problem led to a greater layer-wise diversity, which was hypothesized as a considerable contributor to these state-of-the-art compression results. Similarly to NetAdapt, MnasNet claims to work well on multi-objective trade-offs, but results of using such objectives were not presented in the paper. Alternative approaches to solve the same search problem, e.g. evolutionary optimization, were proposed but not evaluated at all.

4 Goals and Challenges

The high-level goal of this project is to explore methods to incorporate neural network compression in safety critical real time systems. To this end, automotive systems have been selected as a representative demarcation, but general ideas should be extensible to other domains as well. As highlighted in section 3, several algorithms have been proposed for network compression in the context of mobile systems. However, most of them are based on rather low-dimensional resource constraints, often using accuracy-latency trade-offs as the sole evaluation metric. Real time systems, one the other hand, are characterized by their guarantee to provide reasonable response times for certain computational operations. To ensure such guarantees, those systems need the ability to switch models when some part of the system stops working during runtime. As a result, real time systems require careful monitoring of a larger set of resources than what have been taken into account in previous studies. Thus, compression solutions hand-crafted for mobile systems are not necessarily optimal for real time systems. Furthermore, previous research on how to integrate such solutions into a continuous software development process does not seem to exist. To concretize these goals, the top-level research question to be answered is formulated as follows:

> *How can neural network compression be incorporated into safety critical real time systems?*

Since this question is rather large and hence difficult to answer in a definitive way, we split it up into the following subquestions:

1. *What is an appropriate compression objective for safety critical real time systems?*

2. *How does different compression algorithms perform according to the objective determined in (1)?*

3. *How can the algorithms compared in (2) be integrated into a continuous software development process?*

Given the sequential nature of these subquestions, particular emphasis will be put into answering (1) and (2). The main technical challenge will be posed by (2), where several algorithms will be developed and evaluated. Obviously, it is hoped that at least one of those algorithms will be good enough to be implemented in real systems, but this cannot be guaranteed in advance. Instead, the subquestion has been carefully fomulated such that it should be answerable within the given timeframe. Question (3) may be difficult to answer in a definitive way given the time constraints. However, to form a basis for future work, great attempts will be made to shed some light on this issue as well.

5 Approach

The research questions posed in section 4 are of slightly varying characteristics, thus requiring different methods to be answered.

5.1 Compression Objective

The first subquestion asks for an appropriate compression objective for neural networks in safety critical real time systems. Given the engineering nature of this question, an optimization objective will be developed in collaboration with systems engineers at Volvo Cars. Following the most recent paradigm in network compression, this objective will be based on direct metrics. In contrast to previous work, however, the objective will be high-dimensional (i.e. consist of a large set of resource constraints.) Obviously, the objective should give a sound quality indication of a candidate network given some high-level system requirements and a resource budget. Furthermore, given the focus on real time systems, it is preferable if the objective function can be computed with a low runtime overhead. The choice of objective will be presented and motivated thoroughly to allow for future scrutiny and adaption.

5.2 Compression Algorithms

The second subquestion will be answered as a comparative study of several different compression algorithms. First, a set of public, pre-trained neural networks will be selected as a reference set. This set will mainly consist of CNNs trained for vision tasks as those architectures are particularly susceptible to overparameterization and thus more applicable for compression.

A suite of different compression algorithms will then be developed, where the function developed in 5.1 will serve as the optimization objective. Time limits will dictate the size of this suite, but it will be large enough to make a valuable comparison. Furthermore, since the compression objective has yet to be developed, it is not possible to determine the types of algorithms to be tested at this stage. Instead, once the objective is in place, a comprehensive literature survey will be performed to assure an appropriate selection of algorithms that have proven useful for similar optimization problems. Importantly, however, is that the algorithms will not make use of fine-tuning as part of the compression process.

The objective function from 5.1 will then be used to evaluate each algorithm on the reference set under realistic resource constraints. For each network in the reference set, a ranking of the algorithms will be made and in the case of subtle performance differences, appropriate significance tests will be carried out. Performance differences, such as runtime and space consumption, will be measured and presented if the discrepancies are deemed large enough.

There are two sources of data which will be utilized in the development of the compression algorithms. First, each reference network will be represented as a tensor of parameters (i.e. floating point numbers). In this representation, compressing a network simply equates to manipulating the tensor data. There is no inherent notion of quality in this data: intuitively, even a poorly trained network may be applicable to large compression rates by simply tossing a sizable portion of the parameters. Secondly, each reference network will be pre-trained on some public graphics data (e.g. bitmaps) and this data will have some test set associated with it. Since the algorithms to be developed will not require fine-tuning as part of the compression process, the quality of the training set will be completely irrelevant. The test set will only be used for the final evaluation of the models; as such, the quality of this data should not be of any greater significance either.

5.3 Continuous Integration and Delivery

The third subquestion will be difficult to answer in a definitive way within the time that is likely to remain after having answered the first two questions. However, great attempts will be made to shed some light on this issue through a user study at Volvo Cars. In this study, we will investigate if we can automate software optimization to resolve system performance challenges by incorporating the algorithm(s) from 5.2 into a continuous integration and delivery pipeline. In the event that different algorithms require different integration strategies, the one obtaining the best result from the comparison in 5.2 will be selected. This use study will present a different way of working compared to the traditional resource budgeting that is typically being done in systems engineering today. It is hoped that the results and discussion of this study will serve its purpose as a guidance to future research in this area.

References

[1] Kalyanmoy Deb. "Multi-objective optimization". In: *Search methodologies*. Springer, 2014, pp. 403–449.

[2] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando De Freitas. "Predicting parameters in deep learning". In: *Advances in neural information processing systems*. 2013, pp. 2148–2156.

[3] Samuel Dodge and Lina Karam. "A study and comparison of human and deep learning recognition performance under visual distortions". In: *2017 26th international conference on computer communication and networks (ICCCN)*. IEEE. 2017, pp. 1–7.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

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

[6] 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". In: *arXiv preprint arXiv:1704.04861* (2017).

[7] Gabriel L Oliveira, Wolfram Burgard, and Thomas Brox. "Efficient deep models for monocular road segmentation". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 4885–4891.

[8] Siddharth Roheda and Hamid Krim. *Conquering the CNN Over-Parameterization Dilemma: A Volterra Filtering Approach for Action Recognition*. 2019. arXiv: 1910.09616 [cs.CV].

[9] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[10] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. "Mnasnet: Platform-aware neural architecture search for mobile". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2820–2828.

[11] Denis Tomè, Federico Monti, Luca Baroffio, Luca Bondi, Marco Tagliasacchi, and Stefano Tubaro. "Deep convolutional neural networks for pedestrian detection". In: *Signal processing: image communication* 47 (2016), pp. 482–489.

[12] Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. "Netadapt: Platform-aware neural network adaptation for mobile applications". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 285–300.