POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Profiling and partitioning of Deep Neural Networks on multiple devices

Tesi di Laurea Magistrale in
Mathematical Engineering - Ingegneria Matematica

Author: **Luca Crippa**

Student ID: 970627
Advisor: Prof. Danilo Ardagna
Co-advisors: Abednego Wamuhindo Kambale
Academic Year: 2022-23

Abstract

The majority of current applications in the fields of speech and image processing relies on Deep Neural Networks (DNNs). Recent works showed that, thanks to the more powerful and efficient mobile devices available nowadays, it is possible to split the execution of such DNNs between mobile devices and cloud, with the benefit of reducing the energy consumption of the devices, the total execution time of the neural networks, and the network load towards the servers. In this context, this thesis aims to study the partitioning and the execution of DNNs on multiple devices for Smart Eyewear applications. The considered scenario is the one in which a DNN is executed conjointly by three devices: the smart glasses, a mobile phone or another edge device, and a remote cloud server. Considering the already availabe data obtained by studying the execution of DNNs only on the mobile device and the cloud, this work proves that the use of machine learning to predict the inference time of the layers of unseen networks is not a feasible option, so that the most effective way to accurately understand the performance of a network on a particular device is through profiling. It then moves to study the full scenario in which an initial segment of the DNN is run also on the glasses, and it develops a tool for profiling the excution of DNNs on more than two devices. The tool, named Onnx Multi-Device Profiler (OMDP), is capable of profiling the execution of a network on three or more devices, automatically extracting and running the required submodels. Eventually, in a case study that considered the variants of YOLOv5, it is proven that, by combining the inference times profiled by OMDP with the solutions to the K shortest path problem, it is possible to analyse in detail the optimal partitioning and the execution of DNN on multiple devices, even in the case in which the bandwidths between the devices fluctuate over time, as in a real life scenario.

Keywords: Deep Neural Networks, DNN partitioning, DNN profiling, machine learning, K shortest paths, Shapley values.

Abstract in lingua italiana

La maggior parte delle attuali applicazioni negli ambiti dello speech processing e dell'image processing si basa sull'utilizzo di Deep Neural Networks (DNN). Studi recenti hanno mostrato che, grazie a più potenti ed efficienti dispositivi mobili presenti sul mercato oggi, è possibile partizionare l'esecuzione di tali reti neurali tra dispositivi mobili e cloud, con il vantaggio di ridurre il consumo energetico dei dispositivi, il tempo di esecuzione totale delle DNN e il carico di rete sui server. In questo contesto, questa tesi si pone l'obiettivo di studiare il partizionamento e l'esecuzione di DNN su più dispositivi ai fini dello sviluppo di occhiali smart. Lo scenario considerato è quello in cui una rete neurale è eseguita congiuntamente da tre dispositivi: gli occhiali smart, un dispositivo mobile o un altro edge device e un server cloud remoto. Considerando i dati già disponibili ottenuti studiando l'esecuzione delle DNN solo su dispositivo mobile e cloud, questo lavoro dimostra che attraverso l'uso del machine learning non si è in grado di prevedere il tempo di esecuzione dei layer di reti sconosciute, e quindi il modo più efficace per studiare le prestazioni di una rete neurale su un dato dispositivo è attraverso la sua profilazione. Viene quindi studiato lo scenario completo in cui un segmento iniziale della rete neurale è eseguito anche sugli occhiali, e viene sviluppato uno strumento per la profilazione dell'esecuzione di DNN su più di due dispositivi. Lo strumento, chiamato Onnx Multi-Device Profiler (OMDP), è in grado di profilare l'esecuzione di una rete neurale su tre o più dispositivi, estraendo in automatico ed eseguendo i sottomodelli richiesti dalla configurazione specifica. Infine in un caso studio che considera le varianti di YOLOv5, viene dimostrato che, combinando i tempi di esecuzione profilati da OMDP con le soluzioni al K shortest paths problem, è possibile analizzare in dettaglio il partizionamento ottimale e l'esecuzione delle DNN su più dispositivi, anche nel caso in cui le larghezze di banda delle connessioni tra i dispositivi oscillino nel tempo, come accade in casi reali.

Parole chiave: Deep Neural Networks, reti neurali, partizionamento di reti neurali, profilazione di reti neurali, machine learning, K shortest paths, Shapley values.

Contents

Introduction

The majority of current applications in the fields of speech and image processing relies on Deep Neural Networks (DNNs). While very accurate on their tasks, DNNs are very complex and sophisticated models that require suitable hardware to be run. Until recent years, the status-quo approach was to run the full models on the cloud, due to the low computational power of available mobile devices. A recent work [17] has shown that, thanks to the more powerful and efficient devices available nowadays, it is possible to split the execution of DNNs between mobile devices and cloud, with the benefit of reducing the energy consumption of the devices, the total execution time of the neural networks, and of reducing the pressure and the network load towards the servers. However, the joint execution of a DNN requires to upload to the cloud the output tensor of the first slice of the model, and depending on the size of such tensor and on the network bandwidth between the devices, the data transfer may even worsen the total execution time. As a consequence, an accurate study that considers the size of transferred data, the network bandwidth of the connection between the devices, and the computing power of such devices, is required to find the optimal partition point where to split DNNs in order to minimize the total execution time. In the context of the development of the new smart glasses, in this thesis we meticulously study the partitioning and profiling of DNNs on multiple devices. The goal is to profile and analyse in detail the execution of DNNs on two and especially three devices, also in the case in which the bandwidths between the devices fluctuate over time, as in a real life scenario, in order to understand how to reduce the total execution time and the energy consumption of the future DNN-based applications that will run on the smart glasses.

The thesis is organised as follows: chapter 1 aims to give to the reader a general overview of the neural networks, helping to understand the subsequent chapters. In chapter 2 we will assess the results obtained by [4] that studied the scenario in which the Smart Eyewear (SEW) does not have enough computing capacity or memory to process a complete DNN nor any of its layers, so that the computation of the network is split between a mobile device connected to the glasses and the cloud. The chapter revolves around the use of machine learning models to predict the behaviour of unseen neural networks when run on

the same hardware. From chapter 3 we focus also on a different scenario in which the SEW has some computational power that will enable the device to run all or some DNN layers. Indeed, the chapter develops a new tool, Onnx Multi-Device Profiler (`OMDP`), capable of profiling the execution of DNNs on more that two devices. Eventually, in chapter 4 we look to an additional tool, `network_butcher`, which solves the K shortest paths problem to find the best split nodes in which we may partition the DNNs, and in chapter 5 we provide some concrete results obtained profiling the available variants of YOLOv5 using `OMDP` and we compare them with the paths suggested by `network_butcher`, varying the bandwidths between the considered devices.

6 | Conclusions and future developments

In this thesis, our aim was to study the partitioning and the profiling of DNNs on multiple devices. In this final chapter, let us recall the discussed topics and our conclusions. In chapter 2 we assessed the results in [4], where the author studied the scenario in which the computation of the network is split between a mobile device connected to the glasses and the cloud. Particularly, we enhanced the analysis done in [4] with the aim to use machine learning models to reduce the profiling to only a subset of the layers of a network, and eventually to entirely skip the profiling on unseen networks. We concluded that, even if machine learning may be a reliable approach in some specific contexts, where the layers of the networks are similar in structure and composition, in the more general scenario the use of machine learning is not feasible, since we work with networks with large and complex layers. In chapter 3 we developed `OMDP` in order to profile the execution of a DNN on more than two devices, in particular on three devices, since we were interested to analyse the scenario in which the execution of the network is split among the the glasses, a mobile device and the cloud. To reach our goal, we implemented an ad hoc algorithm that allows us to profile each submodel only once on the considered devices. In chapter 4 we looked at the math behind `network_butcher`, a tool that, given the profiled inference times, solves the K shortest pahts problem to find the best way to partition a network. In chapter 5 we provided some concrete results obtained by profiling the variants of YOLOv5 via `OMDP`, and we compared the profiled data with the paths suggested by `network_butcher`. To deal with the scenarios in which it was not possible to estimate the inference times of a DNN segment with the sum of the inference times of its layers, we proposed a novel approach based on game theory. Additionally, we used the data profiled with `OMDP` and `network_butcher` to analyse the partitioning of YOLOv5l in a realistic scenario in which the bandwidths between the devices fluctuate over time. Chapter 5 showed that, relying on `OMDP` and on `network_butcher`, it is possible to perform a reliable and detailed analysis of the partition and of the execution of a DNN on multiple devices. This analyses can obviously be repeated for different neural networks. For the analysis

reported in chapter 5, we used the odroid, the laptop, and the desktop to simulate the execution of the DNNs respectively on the glasses, on the mobile device, and on the cloud. Since this thesis aimed at studying the profiling and partitioning of DNNs on multiple devices specifically for Smart Eyewear Applications, a natural prosecution would be to reproduce the analysis done in chapter 5 with different hardwares, to better simulate which will be the real functioning of the glasses and their interaction with other devices.

Bibliography

[1] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, A. M. Umar, O. U. Linus, H. Arshad, A. A. Kazaure, U. Gana, and M. U. Kiru. Comprehensive review of artificial neural network applications to pattern recognition. *IEEE Access*, 7:158820–158846, 2019. doi: 10.1109/ACCESS.2019.2945545.

[2] AI-SPRINT. BlurryFaces, 2020. URL `https://gitlab.polimi.it/ai-sprint/scar/-/tree/master/examples/mask-detector-workflow/blurry-faces`.

[3] G. Boracchi. Convolutional neural networks. Slides from the course on Artificial Neural Networks and Deep Learning, 2021.

[4] E. I. Chirica. An approach and a tool for the performance profiling and prediction of partitioned Deep Neural Networks in Computing Continua Environments. Master's thesis, Politecnico di Milano, 12 2022.

[5] I. Corporation. OpenVINO, 2023. URL `https://docs.openvino.ai/latest/home.html`.

[6] L. Crippa. Notes. Notes from the course on Numerical Analysis for Machine Learning given by professor Miglio E., 2022.

[7] U. P. de València (Spain). OSCAR, 2022. URL `https://docs.oscar.grycap.net/`.

[8] G. V. Demirci and H. Ferhatosmanoglu. Partitioning sparse deep neural networks for scalable training and inference. *Proceedings of the ACM International Conference on Supercomputing*, 2021.

[9] D. Eppstein. Finding the k shortest paths. *SIAM Journal on computing*, pages 652–673, 1998.

[10] F. L. Facchinetti. Network butcher, 2023. URL `https://github.com/faccus/network_butcher`.

[11] W. Foundation. Network time protocol, 2023. URL `https://en.wikipedia.org/wiki/Network_Time_Protocol`.

[12] G. Guo and J. Zhang. Energy-efficient incremental offloading of neural network computations in mobile edge computing. *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pages 1–6, 2020.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. doi: 10.1109/CVPR.2016.90.

[14] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017. doi: 10.1109/CVPR.2018.00474.

[15] H.-J. Jeong, H.-J. Lee, C. H. Shin, and S.-M. Moon. Ionn: Incremental offloading of neural network computations from mobile devices to edge servers. *Proceedings of the ACM Symposium on Cloud Computing, SoCC '18*, pages 401–411, 2018.

[16] V. M. Jiménez and A. Marzal. A lazy version of eppstein's k shortest paths algorithm. *International Workshop on Experimental and Efficient Algorithms Springer*, 2003.

[17] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ASPLOS '17: Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 615 – 629, 2017. doi: https://doi.org/10.1145/3037697.3037698.

[18] Z. Li, M. Paolieri, and L. Golubchick. Predicting inference latency of neural architectures on mobile devices. *ICPE '23: Proceedings of the 2023 ACM/SPEC International Conference on Performance Engineering*, pages 99–112, 2023.

[19] F. Martins, C. de Oliveira, and E. Borin. Partitioning convolutional neural networks to maximize the inference rate on constrained iot devices. *Future internet*, 2019.

[20] M. Matteucci. Artificial neural networks and deep learning - from perceptrons to feed forward neural networks. Slides from the course on Artificial Neural Networks and Deep Learning, 2021.

[21] Microsoft. ONNX Runtime, 2023. URL `https://onnxruntime.ai/`.

[22] A. Narayanan, X. Zhang, R. Zhu, A. Hassan, S. Jin, X. Zhu, X. Zhang, D. Rybkin, Z. Yang, Z. M. Mao, F. Qian, and Z. L. Zhang. A variegated look at 5g in the wild: Performance, power, and qoe implications. *SIGCOMM 2021 - Proceedings of the ACM SIGCOMM 2021 Conference*, pages 610–625, 8 2021. doi: 10.1145/3452296.3472923. URL `https://doi.org/10.1145/3452296.3472923`.

[23] Pallets. Flask API Documentation, 2022. URL https://flask.palletsprojects.com/en/2.2.x/api/.

[24] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, 2018. doi: 10.1109/CVPR.2018.00474.

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[26] ultralytics. YOLOv8, 2023. URL https://github.com/ultralytics/ultralytics.

[27] ultralytics. YOLOv5, 2023. URL https://github.com/ultralytics/yolov5.

[28] F. Wilhelmi. [ITU AI/ML Challenge 2021] Dataset IEEE 802.11ax Spatial Reuse, Sept. 2021. URL https://doi.org/10.5281/zenodo.5656866. Test dataset with actual throughput values (solution of the AI Challenge 2021).

[29] Z. Zhao, K. M. Barijough, and A. Gerstlauer. Deepthings: Distributed adaptive deep learning inference on resource- constrained iot edge clusters. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 2348–2359, 2018.