Efficient Inference at the Edge

George Muraru, Costin Raiciu
University Politehnica of Bucharest
george.muraru@stud.acs.pub.ro, costin.raiciu@cs.pub.ro

1 INTRODUCTION

Beyond higher wireless bandwidth and denser connectivity, 5G networks promise the availability of computation at the edge, either collocated with the mobile base station or in close proximity. Ideally, each edge site should be able to run computation for nearby mobile users and IoT devices (could reach thousand of devices per cell), and the computation should follow the users as they roam through the 5G network. Achieving this vision is difficult because of the scarcity of edge compute resources and the high load variations due to roaming users.

A common denominator of all mobile edge apps is their reliance on machine learning inference tasks at the edge, be it image recognition, speech-to-text, object tracking, and so forth. To realize the promise of edge computing for machine learning, we propose to exploit the accuracy-resource usage tradeoff and make the best out of edge resources by using less precise but cheaper models when load is high, and precise models when computation abounds. Our main contribution is a control algorithm that is independently used by the clients and the edge servers to achieve this goal dynamically.

2 ACCURATE OR CHEAP?

Most DNNs are trained offline and then they are deployed in production where they run inference on new data. In this paper we focus on the inference phase that should be run at the network edge and use MobileNet [3] to understand the constraints when deploying such models. MobileNet has two hyperparameters (depth and size multiplier) which allow users to trade the accuracy of the classification versus the processing cost for each image. To understand this tradeoff, we select different values for these hyperparameters and use the ImageNet dataset to measure the inference, accuracy and energy consumption on two possible edge architectures: an Arm SOC (Exynos Octa 8890 2.6 GHZ) and a AMD Ryzen processor with 6 cores running at 3.6GHZ.

On the x86 machine, the most expensive model takes more than 50ms for inference and has a top-1 accuracy of 71%, whereas the cheapest model takes just 2ms, but has a top-1 accuracy of around 40%. Figure 1 plots the tradeoff between accuracy and CPU cycles used for both the x86 and the Arm machines. The graph shows a linear increase for inference time until accuracy hits 60%; after that significantly more cycles are needed to increase accuracy. The curves are qualitatively similar for the two platforms, but Arm running the standard model with 32 bit weights is five times slower on average than the x86 machine. On Arm we also test a quantized model running with 8 bit weights which halves the inference time with negligible impact on accuracy.

These results show that by carefully selecting the accuracy of the models used, it should be possible to cope with a very wide range of loads on fairly modest hardware—which means great news for edge computing.

For image-based workloads, the results also show that picture resolution has a big effect on CPU consumption: there is a linear dependence between the number of pixels in the image and the resulting inference time.

The same dependence holds for transmission of images over the cellular network from the mobile clients or IoT devices to the edge computing site. In cases when bandwidth and not computation is the resource bottleneck, we again have a tradeoff between transmission time and accuracy. We can use this tradeoff by sending lower resolution images when the network is overloaded.

3 ADAPTIVE EDGE INFERENCE

We want to answer the following question: given a stream of inference requests from users, which models should be used to ensure the highest overall accuracy while ensuring that inference time is bounded?

We are interested in the online version of this problem, where a decision must be taken when an inference request arrives, without knowing subsequent requests.

As both the computation or the bandwidth can be the limiting factor, a solution must involve both the client and the mobile edge; we require that the amount of coordination between these is minimal, to reduce barriers to deployment (i.e. an edge solution should work without client support, and the reverse).

At the edge, we use the following greedy solution: upon request arrival, we estimate the time needed to serve the request using the different models in decreasing order of accuracy. We choose the highest model that provides a latency below the system-wide latency bound T.

The same solution is not applicable at the mobile client, since request latency depends on the unknown load at the edge and on the variable capacity of the network. In this black-box context, we use a similar greedy solution which has the image size and changes it to a lower resolution based on the request latency. The control algorithm is the following, assuming the current resolution of the image is R_i:

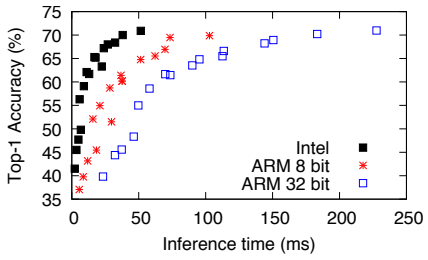

Figure 1: MobileNet accuracy versus cycles tradeoff

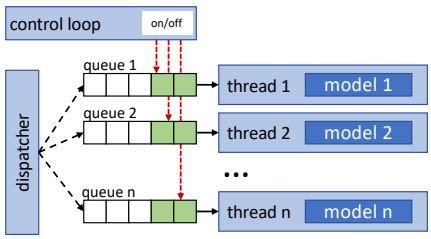

Figure 2: Edge inference architecture

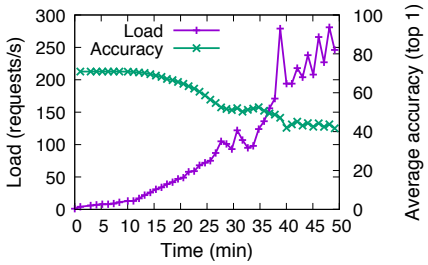

Figure 3: Increasing load gently reduces accuracy at the edge.

(1) If the latency is greater than TC_{upper} and $i < k$ change input image to size R_{i+1}.
(2) If the latency is smaller than TC_{lower} and $i > 1$ change input image size to R_{i-1}.

After a size change, we postpone taking a new decision until the effects of the previous action can be measured. For these two control loops to work well together, the thresholds at the client must be higher than the latency bound used at the edge (i.e. $TC_{upper} > T$, $TC_{lower} > T$).

The architecture of our edge inference implementation is shown in figure 2. The server can be used for any inference task where different models are available that solve the same problem with different accuracy and resource costs. The server loads the models in memory, but is otherwise oblivious to their internal details. The only requirement is that the server is given an ordering of models from highest to lowest accuracy.

Our server accepts client connections and receives images from clients which are then used for inference using one of multiple models pre-loaded in memory. After receiving a request from the client, the dispatcher thread decides the queue for the request which implicitly selects the accuracy and the inference time. To cope with varying load, the latency threshold T is used to decide when a lower accuracy model should be used. When a request arrives, queues are examined starting from the highest accuracy to the lowest one. For each queue, we keep a moving average of the inference time for that model; using this, we estimate the latency of the new request if it is placed in that queue. If the estimated latency is lower than T we enqueue the request, otherwise we continue to the next queue. If we reach the last queue, the request is always enqueued in it, regardless of the latency estimate.

4 PRELIMINARY EVALUATION

For testing purposes we evaluated the control loop by using clients which issues inference requests by sending images to the server in a closed loop. We increase load steadily by adding more clients. Our edge used 16 different preloaded models from MobileNet.

In figure 3, we plot the average accuracy of the system against time: the system starts by using the highest accuracy model, but as more requsts are generated the server switches to models that are less computationally expensive. As a consequence of this, the accuracy starts to fall, reaching a global minima when the requests are placed only in the last queue by the scheduler. The latency of all requests was under 500ms (now shown).

As next steps we intend to test the system using different values for the parameters of the system, like: the latency time, the control loop trigger period (on the client and on the server side), the client timeout period before taking another decision regarding the image size. Another direction is to use more expensive models (e.g. ResNet, Inception) to understand whether memory usage becomes a limiting factor.

5 RELATED WORK

A few works focus on running inference on the mobile or the cloud. Neurosurgeon [8] partitions the neural network workload between the cloud and a mobile device targeting a lower latency or lower energy consumption. MCDNN [4] aims to improve both accuracy and mobile energy consumption by switching between different models. Our approach is similar in that it aims to improve accuracy for compute-constrained settings, but does so in a multi-tenant edge setting.

Mainstream [7] is a video processing system that offers throughput maximization for multiple applications that are sharing the same video stream by using transfer learning to construct models with different level of specialization. Our approach assumes all tenants run the same task, and wants to improve accuracy within the given time-budget.

A series of other works [12, 6, 10, 5, 9, 11, 2, 1, 13, 5] focus on specifics of video processing in a bid to improve accuracy. Understanding how our system can be applied to video inference is our future work.

ACKNOWLEDGEMENTS

This work was partly funded by a grant provided by NEC Research Europe GMBH.

REFERENCES

[1] Ganesh Ananthanarayanan, Paramvir Bahl, Peter Bodik, Krishna Chintalapudi, Matthai Philipose, Lenin Ravindranath, and Sudipta Sinha. "Real-time video analytics: The killer app for edge computing". In: *computer* 50.10 (2017), pp. 58–67.

[2] Utsav Drolia, Katherine Guo, Jiaqi Tan, Rajeev Gandhi, and Priya Narasimhan. "Cachier: Edge-caching for recognition applications". In: *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2017, pp. 276–286.

[3] 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: (Apr. 2017).

[4] Seungyeop Han, Haichen Shen, Matthai Philipose, Sharad Agarwal, Alec Wolman, and Arvind Krishnamurthy. "MCDNN: An Approximation-Based Execution Framework for Deep Stream Processing Under Resource Constraints". In: *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*. MobiSys '16. Singapore, Singapore: ACM, 2016, pp. 123–136. ISBN: 978-1-4503-4269-8. DOI: 10.1145/2906388.2906396. URL: http://doi.acm.org/10.1145/2906388.2906396.

[5] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B Gibbons, and Onur Mutlu. "Focus: Querying large video datasets with low latency and low cost". In: *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. 2018, pp. 269–286.

[6] Samvit Jain, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, and Joseph Gonzalez. "Scaling video analytics systems to large camera deployments". In: *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*. ACM. 2019, pp. 9–14.

[7] Angela H. Jiang, Daniel L.-K. Wong, Christopher Canel, Lilia Tang, Ishan Misra, Michael Kaminsky, Michael A. Kozuch, Padmanabhan Pillai, David G. Andersen, and Gregory R. Ganger. "Mainstream: Dynamic Stem-Sharing for Multi-Tenant Video Processing". In: *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. Boston, MA: USENIX Association, 2018, pp. 29–42. ISBN: 978-1-931971-44-7. URL: https://www.usenix.org/conference/atc18/presentation/jiang.

[8] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. "Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge". In: *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*. ASPLOS '17. Xi'an, China: ACM, 2017, pp. 615–629. ISBN: 978-1-4503-4465-4. DOI: 10.1145/3037697.3037698. URL: http://doi.acm.org/10.1145/3037697.3037698.

[9] Saman Naderiparizi, Pengyu Zhang, Matthai Philipose, Bodhi Priyantha, Jie Liu, and Deepak Ganesan. "Glimpse: A programmable early-discard camera architecture for continuous mobile vision". In: *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*. ACM. 2017, pp. 292–305.

[10] Chrisma Pakha, Aakanksha Chowdhery, and Junchen Jiang. "Reinventing video streaming for distributed vision analytics". In: *10th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 18)*. 2018.

[11] Xukan Ran, Haoliang Chen, Zhenming Liu, and Jiasi Chen. "Delivering deep learning to mobile devices via offloading". In: *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*. ACM. 2017, pp. 42–47.

[12] Ben Zhang, Xin Jin, Sylvia Ratnasamy, John Wawrzynek, and Edward A Lee. "Awstream: Adaptive wide-area streaming analytics". In: *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM. 2018, pp. 236–252.

[13] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J Freedman. "Live video analytics at scale with approximation and delay-tolerance". In: *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. 2017, pp. 377–392.