OpenVINO Deep Learning Workbench: Comprehensive Analysis and Tuning of Neural Networks Inference

Alexander Demidovskij
Intel Corporation
Higher School of Economics
alexander.demidovskij@intel.com

Yury Gorbachev
Intel Corporation
yury.gorbachev@intel.com

Mikhail Fedorov
Intel Corporation
mikhail.fedorov@intel.com

Iliya Slavutin
Intel Corporation
iliya.slavutin@intel.com

Artyom Tugarev
Intel Corporation
artyom.tugarev@intel.com

Marat Fatekhov
Intel Corporation
marat.fatekhov@intel.com

Yaroslav Tarkan
Higher School of Economics
yatarkan@yandex.ru

Abstract

A task of maximizing deep learning neural networks performance is a challenging and actual goal of modern hardware and software development. Regardless the huge variety of optimization techniques and emerging dedicated hardware platforms, the process of tuning the performance of the neural network is hard. It requires configuring dozens of hyper parameters of optimization algorithms, selecting appropriate metrics, benchmarking the intermediate solutions to choose the best method, platform etc. Moreover, it is required to setup the hardware for the specific inference target. This paper introduces a sophisticated software solution (Deep Learning Workbench) that provides interactive user interface, simplified process of 8-bit quantization, speeding up convolutional operations using the Winograds minimal filtering algorithms, measuring accuracy of the resulting model. The proposed software is built over the open source OpenVINO framework and supports huge range of modern deep learning models.

1. Introduction

The OpenVINO Toolkit [7] provides the developers and data scientists with various tools for accelerating their computer vision solutions. However, there is also a number of improvements that can dramatically optimize neural model inference on various Intel hardware: CPU, Integrated Graphics, Intel Neural Compute Stick. In addition to that, measuring performance, exploring model bottlenecks, tuning the model and measuring accuracy are exhaustive tasks that requires deep knowledge in the intersection of engineering and Data Science domains. In order to keep the primary focus on the performance and not on the infrastructure configuration aspects the OpenVINO Deep Learning Workbench (Deep Learning Workbench) was created. Deep Learning Workbench is also designed to be an umbrella over the range of tools in the OpenVINO Deep Learning Development Toolkit (DLDT) [6] with the intention to simplify their usage and streamline the workflow of taking the pretrained neural model till the deployment to the target platform and user facing application. The key problem is that the neural model taken as is after training can be not optimally executed on the Intel hardware and should be tuned and optimized first.

This paper is organized as follows. Section 2 gives a short overview of existing software solutions. Section 3 contains information on the exposed metrics that are used to compare performance of neural models. Then, Section 4 considers visualization aspects of neural models. Two key optimization techniques that empower the Deep Learning Workbench are highlighted in Section 5. Section 6 contains key implementation aspects of the system. Finally, conclusions and directions of further research are given in Section 7.

2. Related Work

Due to the huge development of the field, there are numerous works devoted to understanding of the neural networks mechanics. One of the most popular topics is visualization of activations distributions and high level features [11], [12], [18]. Such an approach lets the scientist perceive how the network learns to adopt its weights for the given input. It is important to note that such visualizations are extremely helpful in case of supervised learning in a computer vision field.

On the other hand, there are multiple contributions to different ways of representing the topology structure so that a scientist or an engineer could identify key features of a topology: examine residual connections, input shape propagation, dimensionality reduction etc. Usually those tools are organized as Web applications: Tensorboard [8], Netscope [5], Netron [4], programming libraries: ANNVisualizer [1] or desktop applications: Net2Vis [9]. Network graph visualization provides access to layers attributes, however, does not explain network dynamics.

Alongside with the learning properties and model structure visualization there are several software solutions that allow to investigate the dataflow of a model during the training and inference process [17]. Tensorboard is one of the most proficient tools in the field and it allows to track hardware metrics: computational load, power, peak memory consumption etc.

In one of the recent research the interesting idea was proposed as a concept of NTP [10] that is formulated as a tool providing benchmarking of neural models across frameworks and hardware. However, to the best of our knowledge, there is no publicly available piece of software as well as any details on supported workflows are absent. Other tools are either focused on a kernel level benchmarking (DeepBench [2]) or are limited to specific workloads (MLPerf [3]). That implies that they do not provide other important functionality like model visualization and accuracy measurements.

However, the fundamental gap in such tools is absence of the option to benchmark the neural model during the inference, apply one of the available optimizations options and analyze the speedup or a slowdown. Inference efficiency is the most important aspect after the training is done. Every enterprise level workload puts a strong requirement for a neural model to deliver optimal throughput and latency. This gap is addressed by the software solution that is proposed in the current paper.

3. Neural Model Metrics

The preliminary survey of experts in the field of Deep Learning revealed the huge need in collecting, visualizing and aggregating both topology-level and primitive-level benchmarking metrics. The proposed software solution addressed that demand and exposes the metrics described below. The important fact is that the inference on different hardware as well inference of an original model and its derived tuned models can be compared among each other on each of the following metrics and create a room for an application engineer to either select the particular configuration for the deployment or continue exploring the optimization options.

3.1. Theoretical analysis

GFLOP GFLOP metric stands for quantity of Floating Point Operations measured in billions of operations. This information is required for a reasearcher to understand if the given model can be computed on the target hardware.

GIOP GIOP metric stands for quantity of Integer Operations and is similar to the aforemnetioned GFLOP metric. For OpenVINO framework models integer operations appear in the quantized model. An application engineer ensures what ratio of the operations is moved to the 8-bit integer mode and can compare it to the GFLOP of the original model.

Number of parameters This metric reflects number of weights in the model.

Minimum and maximum memory consumption This metric depends on the precision of the model weights and reflects the theoretical range of memory amount required for the inference.

3.2. Inference analysis

Configuration of the benchmarking part of the software solution is one of its core parts and provides a user with a set of settings: batch and inference streams (Fig. 4).

Throughput Throughput of the model is measured in FPS (frames per second) by dividing the number of images that were processed by the total execution time. This metric is highly used in the intense applications where are lots of images being processed simultaneously.

Latency Latency of the model is measured in milliseconds and represent time spent for processing of a single image (single forward inference of a network).

Per-layer statistics Apart from the high level benchmark metrics such as Latency and Throughput, statistics is collected for each layer, so that it is possible to compare performance of a particular layer on different hardware, inspect the selected kernel and computational precision, also there is the execution order of the primitive.

4. Neural Model Visualization

There is also a growing demand in representing the model graph as a whole to inspect its building blocks before and during the inference phase. Inside the OpenVINO framework there are two main stages of the neural network acceleration: hardware-agnostic general purpose optimizations of a model with Model Optimizer component and hardware-specific optimizations performed during the inference. The latter optimizations are often hidden from the application engineer although they could bring insights on further optimizations of the model.

Inside the Deep Learning Workbench the rendering engine is Netron [4] that allows to visualize both the original model structure (Fig. 2) and the runtime one (Fig. 3). It is represented as a Directed Graph in a vertical layout with arrows representing directed data flow between layers. Each layer has a code color that makes them easily distinguishable on the large graph.

5. Neural Model Optimizations

8-bit quantization A lot of investigation was made in the field of deep learning with the idea of using low precision computations during inference in order to boost deep learning pipelines and gather higher performance. For example, one of the popular approaches is to shrink the precision of activations and weights values from 32-bit floating point precision to smaller ones, for example, to 11-bit floating point or 8-bit integer precision [16], [13], [15].

8-bit computations offer better performance compared to the results of inference in higher precision (for example, 32-bit floating point precision), because they allow to load more data into a single processor instruction. Usually the cost for significant boost is a reduced accuracy. However, it is proved that the drop in accuracy can be negligible and depends on task requirements, so that the application engineer can set up the maximum accuracy drop that is acceptable.

From the usability perspective, there is a single parameter that represents the maximum accuracy drop value. However, quantizing the model requires access to the dataset, proper configuration of images pre-processing and benchmarking toolset that brings huge problems to an application engineer.

Faster Convolutions with Winograds minimal filtering algorithms There is a known approach to accelerate convolutional layers with the Winograd's minimal filtering algorithms [14]. This optimization requires quite a powerful hardware with AVX512 instructions. Using the Winograd's algorithms for convolution execution can provide increased performance compared to common implementation. However, it can be difficult to understand which algorithm would be faster due to dependency on convolution layer parameters and hardware configuration. In particular, several heuristic approaches are used that take into account the performance of each particular convolution layer and analyze changes in the execution graph that can lead to performance overhead.

6. Application Design

The proposed tool is designed as a Web application that is wrapped in a Docker image and available for use on premise [1]. The application can be launched either on Windows, Linux or macOS OSes. Inside the Docker image there is an OpenVINO package with all the required tooling. For accuracy measurement the Open Model Zoo Accuracy Checker is used, Deep Learning Deployment Toolkit Calibration tool performs neural networks quantization while the Benchmark application collects inference metrics and aggregated reports on the target hardware. A typical user workflow consists of following steps:

1. conversion of a framework-specific model to the OpenVINO IR format;

2. import of a model: either uploading of the OpenVINO IR model or selecting one from the OpenVINO Open Model Zoo;

3. import of a dataset: uploading of the archive with images in a layout of ImageNet or Pascal VOC datasets;

4. selecting the target hardware available for inference on the given machine: CPU, GPU or the Intel Neural Compute Stick and executing inference;

5. analyzing the theoretical information about the model: GFLOP, memory consumption etc.

6. performing various experiments to find a configuration that provides the highest throughput with the minimal latency value

7. performing quantization of the neural model to 8-bit within the accepted accuracy drop

8. comparing performance of the quantized model with the original one

9. comparing performance of original, quantized models on different hardware

10. selecting the best configuration and apply them in the customer application

From the design perspective the software solution can be considered as a SaaS (Software-as-a-Service) and when

[1] `https://hub.docker.com/r/openvino/workbench`

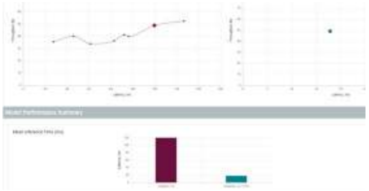


Figure 1. Example of inference comparison. There are two set of points representing different configurations of the benchmarking engine that can be compared.

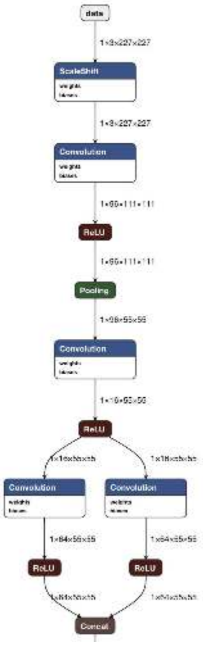


Figure 2. Example of network representation after the training.

needed can be adopted to the this scenario with slight modifications. To this moment the expected use case is running this tool on the local host of the server where all the measurements are conducted.

7. Conclusion

In this paper a novel software solution is described that allows to holistically analyse the model performance on various hardware and not only compare it across the different targets but also to estimate the effect of tuning the model. Availability of the theoretical analysis, quick visualization techniques, accuracy measurement, rich set of benchmarking settings help an application engineer identify performance bottlenecks and optimize the model inference accordingly.

The current state of the tool is limited to the support of only two types of datasets: ImageNet and VOC. Also, there is more data that can be captured and analyzed as well as there is a huge need in extending the list of supported targets.

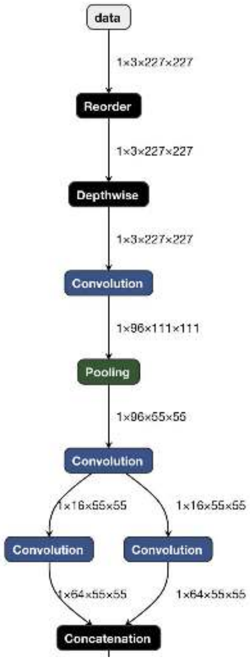
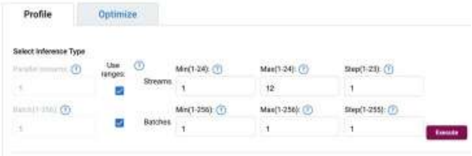


Figure 3. Example of network representation during the inference process.



Profile | Optimize

Select Inference Type

Parallel streams: 1

Batch(1-256): 1

Use ranges:

☑ Streams — Min(1-24): 1 — Max(1-24): 12 — Step(1-23): 1

☑ Batches — Min(1-256): 1 — Max(1-256): 1 — Step(1-255): 1

Execute

Figure 4. Benchmarking settings panel.

OpenVINO Deep Learning Workbench should be considered as a robust solution that is designed to cover the end-to-end flow starting at obtaining the trained model and finishing at helping the users deploy the model in the custom application.

References

[1] ANNVisualizer: model visualizer, 2019. https://github.com/Prodicode/ann-visualizer.

[2] DeepBench: model benchmarking tool, 2019. https://github.com/baidu-research/DeepBench.

[3] MLPerf: model benchmarking tool, 2019. https://mlperf.org/.

[4] Netron: model visualizer, 2019. https://lutzroeder.github.io/netron/.

[5] Netscope: Caffe model visualizer, 2019. https://ethereon.github.io/netscope/quickstart.html.

[6] OpenVINO Deep Learning Deployment Toolkit, 2019. https://github.com/opencv/dldt.

[7] OpenVINO toolkit, 2019. https://software.intel.com/en-us/openvino-toolkit.

[8] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[9] Alex Bäuerle and Timo Ropinski. Net2vis: Transforming deep convolutional networks into publication-ready visualizations. *arXiv preprint arXiv:1902.04394*, 2019.

[10] Raghavendra Bhat, Pravin Chandran, Juby Jose, Viswanath Dibbur, and Prakash Sirra Ajith. Ntp: A neural network topology profiler. *arXiv preprint arXiv:1905.09063*, 2019.

[11] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

[12] Fred Matthew Hohman, Minsuk Kahng, Robert Pienta, and Duen Horng Chau. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE transactions on visualization and computer graphics*, 2018.

[13] Kyuyeon Hwang and Wonyong Sung. Fixed-point feedforward deep neural network design using weights+ 1, 0, and-1. In *2014 IEEE Workshop on Signal Processing Systems (SiPS)*, pages 1–6. IEEE, 2014.

[14] Andrew Lavin and Scott Gray. Fast algorithms for convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4013–4021, 2016.

[15] Naveen Mellempudi, Abhisek Kundu, Dipankar Das, Dheevatsa Mudigere, and Bharat Kaul. Mixed low-precision deep learning inference using dynamic fixed point. *arXiv preprint arXiv:1701.08978*, 2017.

[16] Vincent Vanhoucke, Andrew Senior, and Mark Z. Mao. Improving the speed of neural networks on cpus. In *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*, 2011.

[17] Kanit Wongsuphasawat, Daniel Smilkov, James Wexler, Jimbo Wilson, Dandelion Mane, Doug Fritz, Dilip Krishnan, Fernanda B Viégas, and Martin Wattenberg. Visualizing dataflow graphs of deep learning models in tensorflow. *IEEE transactions on visualization and computer graphics*, 24(1):1–12, 2017.

[18] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.