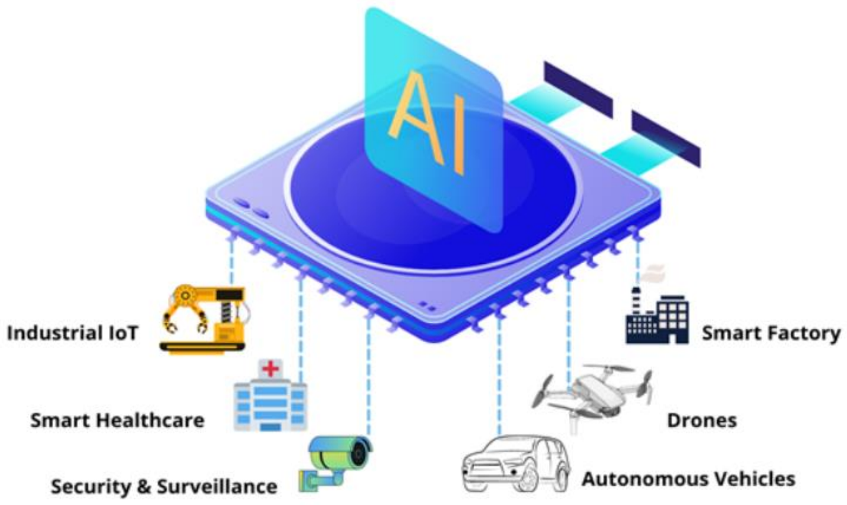
How can deep learning models be optimized using OpenVINO to improve efficiency on CPU-based devices in preparation for the rise of AI?



Student:	Thomas van Egmond (1038854)
Teachers:	Sandra Hekkelman
	Wouter Volders
Deadline:	Resubmission
Date:	30-06-2024

Table of content

Summary

This report looks at how deep learning models can be optimized using OpenVINO (Open Visual Inference and Neural Network Optimization, an optimization software for Intel-based CPUs) to improve efficiency on CPU-based devices in preparation for the rise of AI (Artificial Intelligence). This will be done by answering two sub-questions.

“How can deep learning models be optimized for a CPU-based device” a literature review in which we will take a look at different methods of optimizing deep learning models for CPU-based devices.

“What efficiency improvements does OpenVINO yield on Intel-based CPUs?” an experimental benchmark in which we will compare performance of the standard deep learning model format to OpenVINO and a OpenVINO alternative.

The results show that there are various options for optimizing deep learning models, however none of them are as effective as OpenVINO which uses Intel exclusive hardware optimizations to create a up to 300% more efficient model. The OpenVINO alternative can match the optimization on smaller models but will stray further away from OpenVINO’s performance the bigger the model becomes.

We concluded that if OpenVINO or any other optimization mentioned in this report is available to you, you should greatly consider using it. However with the rapid expansion of AI it is not realistic that future models will be running on CPU-based computers. Nevertheless OpenVINO and its alternatives can be of great importance to expanding the lifetime of AI on CPU-based devices or scenarios where cost and efficiency play a significant role.

Introduction

This year, my school participated in the Self Driving Challenge (SDC), a competition for colleges and universities that involves building and programming an autonomous vehicle to complete a series of predetermined challenges. Our team consists of nine students from three different programs: four from Technical Computer Science, three from Mechanical Engineering, and two from Automotive Engineering. My colleagues from Technical Computer Science and I are tasked with finding solutions to complete these challenges.

Part of these challenges will be tackled using a deep learning model for object recognition, Ultralytics YOLOv8 [1]. Running a deep learning model requires substantial computing power. Traditional Central Processing Units (CPUs) present in modern laptops and desktops are often not powerful enough to run these models in real time. Graphics Processing Units (GPUs), in comparison, offer a significant increase in the efficiency of deep learning models [2].

However, we do not have access to a computer with a GPU. Our computer, equipped only with a CPU, has proven insufficient for running our model in real time without optimization. This issue is not unique to us, with the increasing integration of Artificial Intelligence (AI) in daily life [3], the demand for running deep learning models on-device instead of in the cloud has surged [4]. This demand is driven by privacy concerns about user data being sent to the cloud and the potential negative effects thereof [4]. Consequently, new devices are incorporating GPUs or Neural Processing Units (NPUs) [5] to improve the efficiency of on-device deep learning models.

Nevertheless, older devices equipped only with a CPU, which is not powerful enough to run these models, must offload computing to the cloud, making them more vulnerable to privacy concerns.

The desired outcome of this paper is to find a solution for our lack of computing power that will enable our deep learning model to run in real time on our CPU-based computer. Additionally, we aim to gain insights into optimizations for deep learning models that could enhance the performance of AI applications on CPU-based hardware.

Therefore, I, Thomas van Egmond, will address the main research question: "How can deep learning models be optimized using OpenVINO to improve efficiency on CPU-based devices in preparation for the rise of AI?"

The main research question will be answered by exploring the following sub-questions: "How can deep learning models be optimized for a CPU-based device?" and "What efficiency improvements does OpenVINO yield on Intel-based CPUs?"

Theoretical

The problem of optimizing deep learning models for CPU-based devices is not new, but it has gained increased attention due to the growing demand for AI applications. The computational power required for running deep learning models has traditionally been solved with GPUs. However, as AI is becoming more popular there is a significant increase in demand to run these models on devices that only have a CPU, due to either cost constraints or for an increase in user data privacy.

This problem is present in many organizations, especially in sectors where privacy and cost are critical. Companies in healthcare, automotive, and consumer electronics all face the challenge of running AI models efficiently on CPUs. Solutions such, GPUs, NPUs and software optimizations like OpenVINO have been developed. These solutions help increase the efficiency on existing CPUs but are not always sufficient for highly complex models without proper optimizations [6].

A similar problem to optimizing deep learning models for CPU-based devices is optimizing software for embedded systems. These systems have limited processing power, memory, and energy resources. Solutions such as code optimization and efficient memory management techniques like static memory allocation [7] have been used. These solutions parallel the strategies used for optimizing deep learning models to run efficiently on CPUs.

Some solutions for our problem already exist, as stated by YOLOv8 documentation using ONNX (Open Neural Network Exchange) or OpenVINO can result in a up to 3x efficiency improvement [8] on CPU-based devices. ONNX is an open format used to represent machine learning models, the format is supported by many frameworks and tools [9]. Nevertheless, ONNX is still is not perfect. Depending on the model's architecture and implementation, converting complex models to ONNX may be difficult and necessitate modifying the code [9].

Method

We will be using a combination of experimental and literature review methodologies. The first sub-question "How can deep learning models be optimized for a CPU-based device?" will be done using a literature review focusing on scientific- and website articles.

The second sub-question "What efficiency improvements does OpenVINO yield on Intel-based CPUs?" will be done using an experiment. This experiment will be measuring the average inference time per image in milliseconds (ms/im) and accuracy in mAP50-95 (mean Average Precision from 0.50 to 0.95) [10] of different sized deep learning models benchmarking the same dataset on a Intel CPU.

These benchmarks we will be done using YOLOv8's [1] built in benchmark tool. We will test the default model used in YOLOv8, PyTorch, the alternative to OpenVINO, ONNX, and OpenVINO.

To reproduce the our benchmarks perform the following steps.

1. Install Python 3.10
2. Install the Ultralytics package using `pip install ultralytics'
3. Create a Python file with the following code

   ```
   from ultralytics import YOLO

   # Load a PyTorch model

   model = YOLO("yolov8n.pt")

   # Benchmark model speed and accuracy on the COCO8 dataset for all all export formats

   results = model.benchmarks(data="coco8.yaml")
   ```

4. Run the Python file, you will get a table with the results.
5. Run the Python file for every model by changing the model in YOLO("yolov8n.pt") to every model shown in the results section.

Results

How can deep learning models be optimized for a CPU-based device

A deep learning model consists layers of nodes that mimic neurons in a human brain. These nodes are connected to and influence each other. How much influence nodes have on other nodes is determined by a parameters called weights [11].

Model Pruning

Model pruning is technique that filters and removes parameters from a model that will not decrease the precision of it beyond a preset value when removed. This reduces the size of the model while still maintaining its accuracy. Making a model smaller results in a decrease in required compute power to run the model [12].

Quantization

The weights of a not optimized deep learning model are stored with a high precision which is expressed in a bit count, for example 32-bits. Quantization converts the weights to a lower precision value which takes less bits to store, for example 8-bits, without a substantial drop in accuracy. This greatly reduces the size of the model decreasing the required compute power to run the model. Quantization can be applied during model training (quantization-aware training) or post model training [13].

Knowledge Distillation

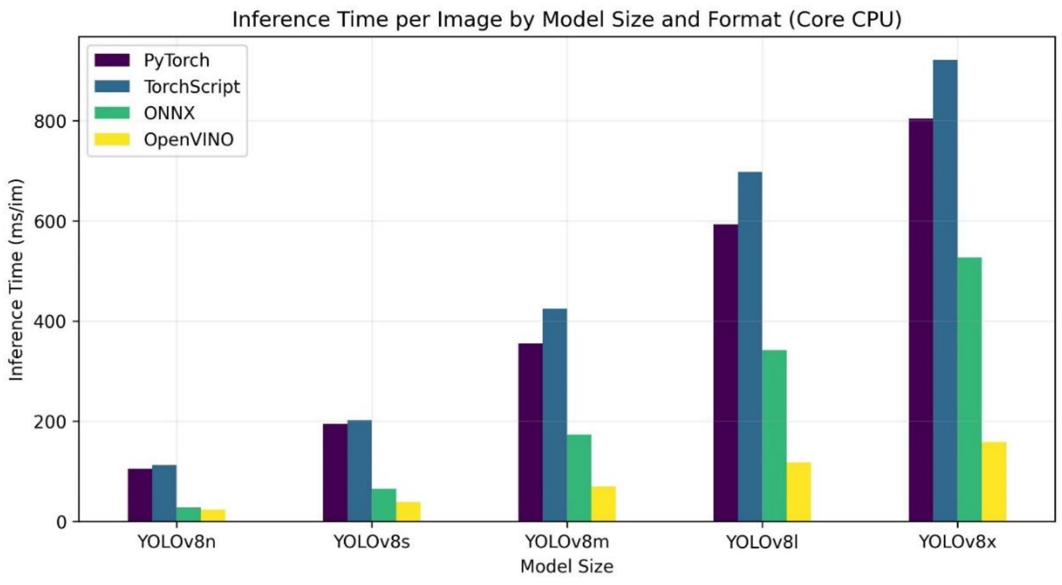
After having successfully trained a model (parent), a new smaller model (child) is trained to mimic the results of the parent model. The result is a child model that is comparable in accuracy to the parent model but has a reduced size decreasing the required compute power to run the model [14].

OpenVINO

OpenVINO (Open Visual Inference and Neural Network Optimization) is a toolkit developed by Intel, it aims to optimize deep learning models for efficiency on Intel hardware. It uses software techniques such as model pruning and quantization [15]. Additionally, OpenVINO can convert a model to an intermediate representation (IR: the code used internally by a compiler to represent source code) specifically optimized for inference (the prediction phase of the model, running the model) performance [16], it does this by applying multiple low-level and memory optimizations which are only possible on Intel hardware [17]. This means OpenVINO is only compatible with Intel hardware.

What efficiency improvements does OpenVINO yield on Intel-based CPUs?

Model	Format	Size (MB)	mAP50-95(B)	Inference time (ms/im)
YOLOv8n	PyTorch	6.2	0.4478	104.61
YOLOv8n	ONNX	12.2	0.4525	28.02
YOLOv8n	OpenVINO	12.3	0.4504	23.53
YOLOv8s	PyTorch	21.5	0.5885	194.83
YOLOv8s	ONNX	42.8	0.5962	65.74
YOLOv8s	OpenVINO	42.9	0.5966	38.66
YOLOv8m	PyTorch	49.7	0.6101	355.23
YOLOv8m	ONNX	99.0	0.6120	173.39
YOLOv8m	OpenVINO	99.1	0.6091	69.80
YOLOv8l	PyTorch	83.7	0.6591	593.00
YOLOv8l	ONNX	166.8	0.6580	342.15
YOLOv8l	OpenVINO	167.0	0.0708	117.69
YOLOv8x	PyTorch	130.5	0.6651	804.65
YOLOv8x	ONNX	260.4	0.6650	526.66
YOLOv8x	OpenVINO	260.6	0.6619	158.73



Conclusion

In this paper the primary research question was, "How can deep learning models be optimized using OpenVINO to improve efficiency on CPU-based devices in preparation for the rise of AI?". This question arose from the need to run a deep learning model on a CPU-based computer. To address this, we explored how deep learning models can be optimized for CPU-based devices and what efficiency improvements OpenVINO offers on Intel-based CPUs.

How can deep learning models be optimized for a CPU-based device

Luckily for us there exist multiple methods of optimizing a deep learning model. These methods all come down to decreasing the size and complexity of the model while retaining the accuracy of it. Of course applying these optimization take time, but if you're only interested in inference time and tolerate a miniscule accuracy decrease it would be a waste not to apply these optimizations.

What efficiency improvements does OpenVINO yield on Intel-based CPUs?

A minimal 300% improvement in inference time while hardly compromising on accuracy. The results show the 300% improvement also stays the same with the increase of model size, meaning that with increasing model size the difference between OpenVINO and the default format PyTorch grows bigger and bigger. As shown in the graph 'Inference Time per Image by Model Size and Format (Core CPU)' we can see that the OpenVINO competitor ONNX can't keep up with the optimizations OpenVINO delivers for larger sized models. Not using OpenVINO when running an deep learning model on any sort of Intel hardware would be a great mistake.

How can deep learning models be optimized using OpenVINO to improve efficiency on CPU-based devices in preparation for the rise of AI?

It seems there various great options for optimizing deep learning models which support al CPU brand. However these optimizations do not come close to the efficiency increase OpenVINO can deliver using its Intel hardware specific optimizations. Luckily Intel has a 60% + market share for windows computers [18] meaning that more than half of all people running windows can profit from OpenVINO. If companies developing AI applications wanted to add support for CPU-based computers it's greatly recommended that they look into OpenVINO or a similar software for their CPU brand.

So it is definitely possible to develop software such as OpenVINO for all brands of CPUs as a preparation for the rise of AI, however most companies will probably try to sell consumers new hardware. Instead it's recommended that software such as OpenVINO is developed and used for specific scenarios where cost and efficiency play a significant role.

Sources

1. Kadirhan Akin, “YoloV8 en easyOCR.docx”, June 19, 2024. https://github.com/ThomasvanEgmond/Self-Driving-Challenges/blob/main/Modules/YoloV8%20en%20easyOCR.docx
2. R. Merritt, “Why GPUs are great for AI | NVIDIA blog,” NVIDIA Blog, May 23, 2024. https://blogs.nvidia.com/blog/why-gpus-are-great-for-ai/
3. Tprestianni, “131 AI Statistics and Trends for (2024) | National University,” National University, May 30, 2024. https://www.nu.edu/blog/ai-statistics-trends/
4. “Introducing Apple’s On-Device and server foundation models,” Apple Machine Learning Research. https://machinelearning.apple.com/research/introducing-apple-foundation-models
5. G. Peru, “What is an NPU? Here’s why everyone’s suddenly talking about them,” Digital Trends, Dec. 27, 2023. https://www.digitaltrends.com/computing/what-is-npu/
6. “Intel® distribution of OpenVINOTM toolkit,” Intel. https://www.intel.com/content/www/us/en/developer/tools/openvino-toolkit/overview.html
7. J. Ganssle, The art of designing embedded systems, Second Edition. Elsevier, 2008. [Online]. Available: https://theswissbay.ch/pdf/Gentoomen%20Library/Misc/Art%20of%20Designing%20Embedded%20Systems~tqw~_darksiderg.pdf
8. Ultralytics, “Export,” Ultralytics YOLO Docs, Jun. 10, 2024. https://docs.ultralytics.com/modes/export/#why-choose-yolov8s-export-mode
9. A. Czapski, “Bridge tools for machine learning frameworks - Jit Team - Medium,” Medium, Sep. 06, 2022. [Online]. Available: https://medium.com/jit-team/bridge-tools-for-machine-learning-frameworks-3eb68d6c6558
10. Ultralytics, “YOLO Performance Metrics,” Ultralytics YOLO Docs, Jun. 02, 2024. https://docs.ultralytics.com/guides/yolo-performance-metrics/#class-wise-metrics
11. “What is a Neural Network? | IBM.” https://www.ibm.com/topics/neural-networks
12. M. Neo, “A comprehensive guide to neural network model pruning.” https://www.datature.io/blog/a-comprehensive-guide-to-neural-network-model-pruning
13. Deci, “Model Quantization & Quantization-Aware Training: Ultimate Guide,” Deci, Mar. 19, 2024. https://deci.ai/quantization-and-quantization-aware-training/
14. P. Potrimba, “What is Knowledge Distillation? A Deep Dive.,” Roboflow Blog, Apr. 09, 2024. https://blog.roboflow.com/what-is-knowledge-distillation/
15. “Model Optimization Guide — OpenVINOTM documentation — Version(2024).” https://docs.openvino.ai/2024/openvino-workflow/model-optimization.html
16. “Running Inference with OpenVINOTM — OpenVINOTM documentation — Version(2024).” https://docs.openvino.ai/2024/openvino-workflow/running-inference.html
17. “Further Low-Level Implementation Details — OpenVINOTM documentation — Version(2024).” https://docs.openvino.ai/2024/openvino-workflow/running-inference/optimize-inference/optimizing-low-level-implementation.html
18. Statista, “Share of Intel and AMD x86 computer CPUs worldwide 2012-2024, by quarter,” Statista, Apr. 23, 2024. https://www.statista.com/statistics/735904/worldwide-x86-intel-amd-market-share/

Wordlist

Artificial intelligence (AI): is intelligence exhibited by machines, particularly computer systems.

Deep learning: is the subset of machine learning methods based on neural networks with representation learning.

Intermediate representation (IR): is the code used internally by a compiler to represent source code.

Intersection over Union (IoU): IoU is a measure that quantifies the overlap between a predicted bounding box and a ground truth bounding box. It plays a fundamental role in evaluating the accuracy of object localization.

Machine learning: is a field of study in artificial intelligence concerned with the development and study of statistical algorithms that can learn from data and generalize to unseen data and thus perform tasks without explicit instructions.

mAP50-95: The average of the mean average precision calculated at varying IoU thresholds, ranging from 0.50 to 0.95. It gives a comprehensive view of the model's performance across different levels of detection difficulty.

Neural network: is a model inspired by the structure and function of biological neural networks in animal brains. It consists of connected units or nodes called artificial neurons, which loosely model the neurons in a brain.

Neural Processing Unit (NPU): is a class of specialized hardware accelerator[1] or computer system[2][3] designed to accelerate artificial intelligence and machine learning applications

OpenVINO (Open Visual Inference and Neural Network Optimization): is a toolkit developed by Intel, it aims to optimize deep learning models for efficiency on Intel hardware.

Optimized for inference, it means that the model has been specifically adjusted to run efficiently during the prediction phase (inference phase).