Deep Learning Models Benchmarking for the Image Recognition

Bohdan Blagitko, Dmytro Myronyuk
Dept. of Radiophysics and Computer Technologies
Ivan Franko National University of Lviv
Lviv, Ukraine
blagitko@gmail.com, myronyukdmytro@gmail.com

Igor Zajachuk,
Pidstryhach Institute for Applied Problems of Mechanics and Mathematics National Academy of Sciences of Ukraine
Lviv, Ukraine
igorzajach@gmail.com

Yurii Bodakovskyi
Ukrainian Catholic University Faculty of Applied Sciences
Lviv, Ukraine

Порівняльний Аналіз Моделей Глибинного Навчання для Розпізнавання Зображень

Богдан Благітко, Дмитро Миронюк
кафедра радіофізики та комп'ютерних технологій
Львівський національний університет імені Івана Франка
Львів, Україна
blagitko@gmail.com, myronyukdmytro@gmail.com

Ігор Заячук
Інститут прикладних проблем механіки і математики імені Я. Підстригача НАН України
Львів, Україна
igorzajach@gmail.com

Юрій Бодаковский
Український Католицький університет
факультет прикладних наук
Львів, Україна

***Abstract* — Machine learning has gained popularity for image recognition on edge devices. This study evaluates the performance of three devices, Nvidia Jetson Nano, Raspberry Pi 4 Model B, and Raspberry Pi 5 at accuracy, latency, and other metrics. The proprietary neural image recognition network is installed on each device, and their activity is analyzed. The study identifies the pros and cons of different image recognition methods. The various image recognition methods' advantages and disadvantages are identified.**

***Анотація* — Машинне навчання на мобільних пристроях стало дуже популярним для розпізнавання зображень. Описано продуктивність при вимірюванні точності, затримки та інших показників на Nvidia Jetson Nano, Raspberry Pi 4 Model B і Raspberry Pi 5. Власна нейронна мережа для розпізнавання зображень установлена на кожному з цих трьох пристроїв. Проведений аналіз її діяльності. Визначено переваги та недоліки різних методів розпізнавання зображень.**

Keywords — deep learning; image recognition; computer vision; Raspberry Pi

Ключові слова — глибинне навчання; розпізнавання зображень; комп'ютерний зір

I. Introduction

Machine Learning tasks on edge devices have become very popular in recent years. There are a lot of developers that create or use existing models on these platforms. Single-board microcomputers, such as Raspberry Pi, gained popularity due to their efficiency, affordability, and power, and are used for both production and education. Deep Neural Networks (DNNs) are widely used for different tasks now. However, they can need a lot of memory and computation resources since edge devices have their constraints - limited memory and resources, so different optimization techniques are used to meet these constraints. There are some papers aiming at evaluating performance lot that use different edge devices, frameworks, and models. Such architectures of neural networks do not make it possible to realize image recognition with sufficient speed. In 2018 Xingzhou Zhang et al. [1] compared some of the at that time best ML frameworks for edge devices, such as Tensorflow, Caffe2, MXNet, PyTorch, and Tensorflow Lite. This work makes recommendations on which framework to choose for your needs and the platform to use. In 2021 Stephan Patrick Baller et al. [2] presented their benchmark for measuring inference time, power consumption, and accuracy, called DeepEdgeBench. They compared the performance of four SoCs: Asus Tinker Edge R, Raspberry Pi4, Google Coral Dev Board, and Nvidia Jetson Nano. One microcontroller: Arduino. Nano 33 BLE, using only CPU and ML Accelerators. In their case Nvidia Jetson Nano with GPU, and Google Coral Dev Board with Edge TPU.

II. Network operation principles for image recognition on mobile platforms

A benchmarking tool for measuring the performance of Deep Learning models on single-board microcomputers with ARM architecture is described in this article. The supported devices are Raspberry Pi 4 Model B [3], Raspberry Pi 5 [4, 6], and Nvidia Jetson Nano [5]. Also, the benchmark can run inference using the following frameworks: PyTorch, ONNX, TensorRT, and Tensor-Flow Lite with or without the full-integer quantization and delegate usage. The frameworks are tools, libraries, or interfaces used to help developers build their models or take existing ones, train, use, and deploy them.

TensorFlow is a framework for building and deploying models developed for the Google Brain team. It has an ecosystem of tools that will make it easy to create solutions for different needs. TensorFlow Lite is an open-source library for machine learning models on mobile deploy, embedded, and edge devices. It was released in 2017 and developed by the Google Brain team. In order to do this models built with TensorFlow are converted into TensorFlow Lite models. It is represented in FlatBuffers format that are smaller and more efficient. PyTorch is an ML framework for building, training, and deploying deep learning models developed by Facebook AI Research lab. ONNX Runtime is a cross-platform inference engine for ONNX (Open Neural Network Exchange) models developed in Microsoft. It is designed to accelerate machine learning models across different platforms and devices. It can used with models for PyTorch, TensorFlow/Keras, TFLite, scikit-learn, and other frameworks. TensorRT is a deep learning high-performance optimizer and runtime library developed by NVIDIA. It is designed specifically for NVIDIA GPUs to accelerate deep learning inference. Applications using TensorRT have significant improvements in inference speed compared to CPU-only devices. Different optimizations are used, such as quantization, layer and tensor fusion, kernel tuning, and others on NVIDIA GPUs.

III. Learning the Image Recognition Models, Data, and Learning Process Parameters

The two models cover the whole supported range. Classification and regression for this image recognition experiment were used. We chose: ResNet50 [7] - object classification model trained on ImageNet dataset. It takes a 224x225 RGB image as input and classifies it as one of the 1000 available classes; MobileNetV2 [8] - this is also an object classification model that was trained on the ImageNet dataset. It has the same input and output as ResNet50. However, it is smaller in cost of minor accuracy loss, so it is suitable for edge devices with limited resources.

The latency and accuracy are in the best interest of developers while building and testing their models. Latency (or inference time) is the time it takes for a model to make predictions based on input data. A time module in Python measures it. Fig.1 demonstrates the measured latency for all devices and frameworks using the ResNet50 model.

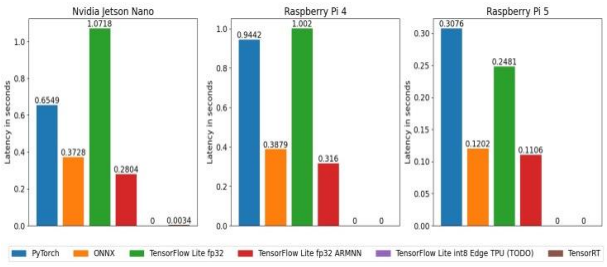

Fig.I. Measured latency for all devices and frameworks using ResNet50 model

Fig.2 demonstrates of the measured latency for all devices and frameworks using MobileNetV2 model.

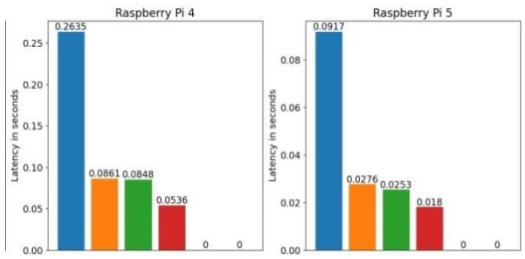

Fig.2. Measured latency for all devices and frameworks using MobileNetV2 model

Accuracy - ACC [9] - calculated (1). It ranges from 0 to 1 and then is multiplied by 100 to get a percentage (1).

$$ACC = \frac{TP + TN}{TP + TN + FN + FP} \quad (1)$$

Where

- TP - predicted elements as positive and are positive;
- FP - predicted elements as positive but are negative;
- TN - predicted elements as negative and are negative;
- FN - predicted elements as negative but are positive.

Accuracy of the Classification models ResNet50 and MobileNetV2 were measured. There is no loss in accuracy when converting between PyTorch, ONNX, TensorRT, and TensorFlow Lite without any optimizations. However, it has a small accuracy drop if using full-integer quantization, which is justified by latency improvements.

IV. Learning Results

The best performance for CPU-only inference has TensorFlow Lite with ARMNN usage. It is twice as good as running the same model without a delegate. ONNX performance is also really good and close to ARMNN, even without any optimizations. PyTorch has the worst latency, which is expected. It is not optimized for running on edge devices. Raspberry Pi 5 results are much better than Raspberry Pi 4. It is because of better CPU and RAM speed.

Conclusion

The test results of the trained models are satisfactory, although they are far from real-time recognition. It can be considered an improvement of image recognition systems, which have become a fast and fairly accurate solution for working on less powerful platforms.

References

[1] Xingzhou Zhang, Yifan Wang, and Weisong Shi. “PCamp: Performance comparison of machine learning packages on the edges”. In: USENIX Association, 2018. URL: https://www.scopus.com/ inward/ record.uri?eid=2- s2.0-85084162686&partnerID=40&md5= ded7b4ff3b308f00b320e7c849deb014.

[2] Stephan Patrick Baller et al. “DeepEdgeBench: Benchmarking Deep NeuralNetworks on Edge Devices”. In: Institute of Electrical and Electronics Engineers Inc., 2021, 20–30. DOI: 10.1109/IC2E52221.2021.00016. URL: https://doi.org/10.1109/ IC2E52221.2021.00016.

[3] Raspberry Pi Foundation. Raspberry Pi 4 Model B - Specifications. Accessed: 2024-04-22. URL: https://www.raspberrypi.com/ products/raspberry-pi-4-model-b/specifications.

[4] Raspberry Pi Foundation. Raspberry Pi 5. Accessed: 2024-04-22. URL: https: //www.raspberrypi.com/products/raspberry-pi-5..

[5] NVIDIA. Jetson Nano Developer Kit. Accessed: 2024-03-24. URL: https://developer. nvidia.com/embedded/jetson-nano-developer-kit.

[6] Salem Ameen et al. “Optimizing Deep Learning Models For Raspberry Pi”. Salem Ameen, Kangaranmulle Siriwardana, Theo Theodoridis. School of Science, Engineering and Environment University of Salford Manchester, United Kindom. arXiv preprint arXiv:2304.13039. 2023/4/25.

[7] Kaiming He et al. “Deep Residual Learning for Image Recognition”. 2015. arXiv:1512.03385

[8] Mark Sandler et al. “MobileNetV2: Inverted Residuals and Linear Bottlenecks”. 2019. arXiv: 1801.04381.

[9] Željko D. Vujovic. “Classification Model Evaluation Metrics” In: International Journal of Advanced Computer Science and Applications 12.6 (2021), 599 – 606. DOI: 10.14569/IJACSA. 2021.0120670. URL: https://doi.org/10.14569/IJACSA. 2021.0120670.