Übung 3

Repository: https://github.com/Raphael90K/HCA3

1 Projektbeschreibung

Als dritte Übung wurde die Arbeit mit einem KI-Beschleuniger gewählt. Als Hardware wird ein Raspberrry Pi 5 mit einem Hailo8L[1] Chip verwendet. Dieser ist im Stande Berechnungen mit 13 tera-operations pro Sekunde (TOPS) zu berechnen und ist für die Inferenz von KI-Modellen vorgesehen [Hail24a]. Abbildung 1 zeigt ein Bild des verwendeten Chips auf dem Raspberry Pi.

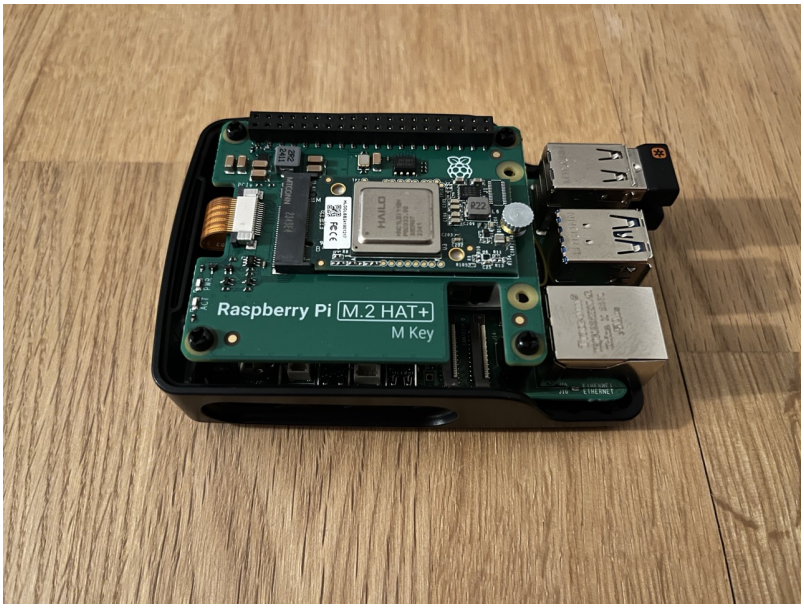

Abbildung 1: Hailo8L Chip auf dem Raspberry Pi

Außerdem steht ein Notebook mit Intel i9 13900HX[2] und eine Nvidia GeForce RTX 4070[3] zur Verfügung. Gemäß den Spezifikationen kann die Nvidia Grafikkarte 321 TOPS berechnen [Nvid24]. Die Aufgabe besteht nun darin ein einfaches Modell auf beiden Endgeräten zu implementieren,

[1] https://hailo.ai/products/ai-accelerators/hailo-8l-m-2-ai-acceleration-module-for-ai-light-applications/

[2] https://www.intel.de/content/www/de/de/products/sku/232171/intel-core-i913900hx-processor-36m-cache-up-to-5-40-ghz/specifications.html

[3] https://www.nvidia.com/de-de/geforce/laptops/

auf verschiedenen Hardwarekomponenten (Prozessor (CPU), Grafikkarte (GPU) und Hailo-Chip) auszuführen und die Performance zu vergleichen. Da auf allen Hardwarekomponenten das gleiche Modell verwendet wird, soll es nicht um die Performance des Modells an sich, sondern um einen Laufzeitenvergleich der verschiedenen Hardware gehen.
Aufgrund der technischen Gegebenheiten sollte die Grafikkarte um einen Faktor 24 schneller sein. Zudem soll auch ein Vergleich mit den Hauptprozessoren zeigen, wie sich diese im Vergleich zur Grafikkarten und dem KI-Beschleuniger verhalten. Es soll daher geprüft werden, ob erwartungsgemäß ein deutlicher Speedup der Grafikkarte im Vergleich zu den übrigen Komponenten zu verzeichnen ist.

2 Konzept und Implementierung

Um einen Vergleich auf allen Geräten zu ermöglichen, soll bei allen Komponenten das gleiche Modell verwendet werden. Da die Leistungskapazität des Hailo8L, der auf deinem Raspberry Pi 5 betrieben wird, nicht vollumfänglich bekannt ist, soll als KI-Modell ein einfaches Neuronales Netz für eine Klassifizierungsaufgabe verwendet werden. Das Neuronale Netz soll für alle Systeme die gleiche Struktur haben. Das Projekt wird in Python umgesetzt und grundlegende Funktionalitäten werden mit Hilfe von Funktionen des `PyTorch`[4] Frameworks umgesetzt. Dieses ermöglicht das einfache Arbeiten mit Neuronalen Netzen und das Ausführen auf Nvidia Grafikkarten[PGML+19]. Zudem können `PyTorch` Modelle in ausführbare Binärdateien für den Hailo KI-Beschleuniger umgewandelt werden [Hail24b].
Abhängig von der Hardware auf der das Modell ausgeführt wird, wird ein unterschiedliches `Python` Modul aufgerufen. Die Funktionen zum Ausführen des Modells auf dem Prozessor und Grafikkarte sind in der Datei `notebook_run.py` implementiert. Wenn das Modell auf dem Hailo8L Chip ausgeführt werden soll, wird die Datei `hailo_run.py` verwendet. Die Methoden zur Durchführung der Inferenz mit Hilfe eines Neuronalen Netzes auf dem Hailo8L wurden vom Grundsatz aus dem *Python inference tutorial* von der Hailo Website übernommen und auf die konkrete Anwendung angepasst [Hail24d].

[4] https://pytorch.org/get-started/locally/

Die Messung der Performance erfolgt mit Hilfe des `profiler.py` Moduls. Diese misst die Laufzeit für eine übergebene Funktion in mehreren Iterationen, gibt die einzelnen Laufzeiten aus und bildet im Anschluss das arithmetische Mittel über die gemessenen Laufzeiten. Die übergebene Funktion ist bei den beiden hardwarespezifischen Modulen jeweils die Methode, die den Inferenzschritt ausführt. Die Vorverarbeitung erfolgt jeweils außerhalb dieser Methode. Die in diesem Abschnitt genannten Module wurden mit Hilfe von ChatGPT[5] in der kostenlosen Version erstellt. Das Starten der Anwendung ist über die Kommandozeile über `main.py` möglich.

2.1 Das Modell

Bei der Recherche konnte mit dem MobileNetV2 ein einfaches Neuronales Netz identifiziert werden, welches für die Klassifizierung von Bildern geeignet ist [SHZZ+18]. Das Modell ist in der `PyTorch` model library bereits enthalten und muss daher nicht neu erstellt werden[6]. Dieses Modell wird mit den Bildern des `CIFAR-10` Datensatzes trainiert, einem Satz von 60000 kleinen Bildern, die in zehn Klassen gelabelt sind[7]. Da Ziel dieses Projekts nicht die Qualität des Inferenzmodells ist, wird der gesamte Datensatz zum Training verwendet und kein Testdatensatz erzeugt. Der Code hierzu befindet sich in der Datei `create_and_train_model.py` und wurde mit der Hilfe von ChatGPT in der kostenlosen Version erstellt. Der Trainingsablauf entspricht dem Vorgehen, der bei derartigen Verfahren auf der Website von `PyTorch` beschrieben ist [PyTo24]. Das Training erfolgt in 5 Durchläufen und das trainierte Neuronale Netz wird im Anschluss auf der Festplatte gespeichert. Mit diesem Schritt ist das Modell für die weiteren Experimente erstellt.

2.2 Portierung für Hailo8L

Damit das erstellte Modell auf dem Hailo8L Chip funktioniert, muss das Modell für diesen umgewandelt werden. Hierzu muss das Modell im `ONNX`[8] Format vorliegen. Bei dem `ONNX` Format handelt es sich um einen offenen Standard für Anwendungen des Maschinellen Lernens zur Speicherung

[5] https://chatgpt.com/
[6] https://pytorch.org/vision/stable/models/mobilenetv2.html
[7] https://www.cs.toronto.edu/ kriz/cifar.html
[8] https://onnx.ai/

von Neuronalen Netzwerkmodellen [ONNX24]. Die Umwandlung des gespeicherten Modells in das `ONNX` Format erfolgt mit Hilfe des Skripts `transform_model_to_onnx.py`. Hierzu werden Funktionen von `PyTorch` verwendet. Der Code wurde mit Hilfe von ChatGPT in der kostenlosen Version erzeugt. Ausgehend von diesem `ONNX` Modell kann die weitere Umwandlung in mehreren Schritten mit Hilfe des Hailo Dataflow Compilers (DFC) in den folgenden Schritten über die Kommandozeile erfolgen [Hail24c]:

1. Umwandlung des `ONNX` Modells in das Hailo Archive (`HAR`) Format.
2. Umwanldung des `HAR` Modells in ein optimisiertes `HAR`.
3. Umwandlung des optimisierten `HAR` in die Binärdatei Hailo Executable File (`HEF`) Format, welches vom Hailo8L Chip verwendet werden kann.

Die verwendeten Befehle wurden in der Textdatei `steps` dokumentiert.

3 Experiment

Im nächsten Schritt soll die Laufzeit auf den verschiedenen Hardwarekomponenten getestet werden. Zur Klassifizierung wird ein Bild einer Katze verwendet. Die Laufzeit wird in fünf Durchläufen auf allen in Abschnitt 1 genannten Komponenten getestet. Die Messergebnisse und Durchschnittswerte sind im Ordner `results` gesichert. Zusätzlich werden die Durchschnittswerte in Abbildung 2 dargestellt.

Dass der Prozessor des Raspberry Pi mit dem geringsten Leistungsvermögen die längste Durchschnittslaufzeit aufwies ist hierbei erwartungsgemäß. Der Hailo8L Chip weist die beste Laufzeit im Durchschnitt auf, wobei auch die einzelnen Iterationen eine ähnliche Laufzeit von ungefähr 0,01 Sekunde aufweisen. Die Ausführung auf der GPU dauerte im Schnitt 0,0542 Sekunden, wobei man deutlich erkennen konnte, dass die erste Iteration mit 0,2530 Sekunden eine sehr hohe Laufzeit aufweist. Die weiteren Iterationen haben in der Folge nur noch eine Laufzeit zwischen 0,0040 - 0,0050 Sekunden. Hier ist davon auszugehen, dass Start- oder Initialisierungsvorgänge für die lange Laufzeit im ersten Durchlauf gesorgt haben dürften. Die längere Laufzeit in der ersten Iteration

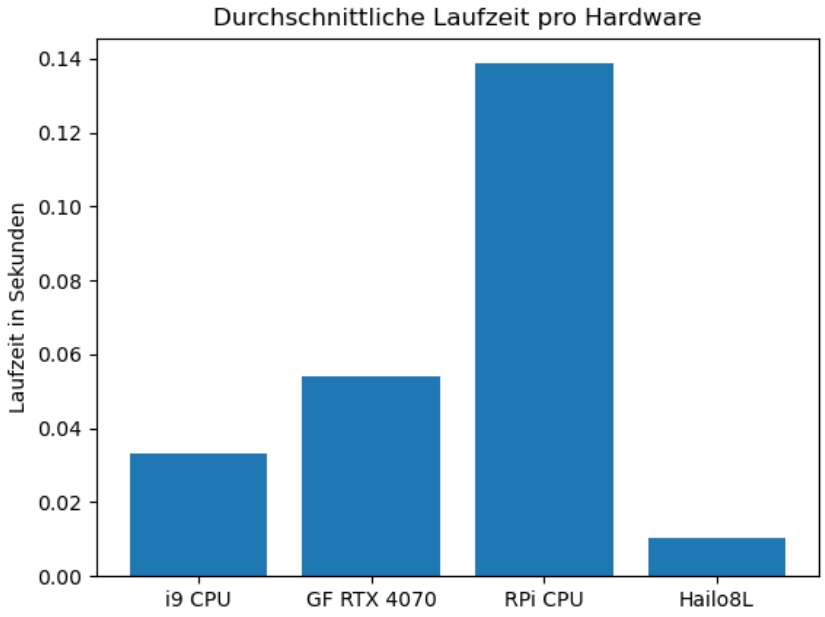

Abbildung 2: Durchschnittliche Laufzeit pro Hardware

ist auch bei den anderen Komponenten festzustellen, wobei die Zeitdifferenz in der Folge bei der Grafikkarte am höchsten ausfällt. Wenn man die erste Iteration unberücksichtigt lässt, hätte die Grafikkarte mit einem Faktor von ungefähr 2,26 gegenüber dem Hailo8L Chip die beste Laufzeit.

4 Projekthürden

Im Rahmen dieser Übung zeigte sich, dass einzelne Codekomponenten in häufig verwendeten Standardframeworks wie `PyTorch` mit der guten Dokumentation und ChatGPT schnell und unkompliziert erstellt werden können. Die Schritte des Trainings und der Nutzung entsprechen hierbei auch der Dokumentation auf der Website. Eine größere Hürde stellte die Nutzung des Hailo8L Chips dar. Auf der Website und in GitHub sind mehrere Beispielmodelle für die Anwendung zu finden, hierunter auch Modelle für Klassifizierungsaufgaben[9]. Diese sollten aber in dieser Übung nicht

[9] https://github.com/hailo-ai/hailo_model_zoo/blob/master/docs/public_models/HAILO8L/HAILO8L_classification.rst

verwendet werden, da sich diese bereits in einem `HEF` Format befinden und daher in erster Linie für den Hailo8L Chip konzipiert wurden. Um das gleiche Modell auf mehreren Komponenten auszuführen, wurde daher ein Modell erstellt und trainiert. Die Umwandlung in den in Unterabschnitt 2.2 beschriebenen Schritten war dabei nicht trivial, da der Compiler erst im Juli 2024 der Öffentlichkeit zugänglich gemacht wurde [gila24] und die Dokumentation etwas spärlich erscheint. Zudem war der benötigte Dataflow Compiler nur unter Linux verwendbar, weshalb während des Projekts ein Windows Betriebssystem, das Subsystem für Linux und das Betriebssystem des Raspberry Pi verwendet werden musste. Die in der Folge notwendigen Paketinstallationen, das Herausfinden der richtigen Schrittreihenfolge und Parameter kostete bei der Umwandlung die meiste Zeit.

5 Fazit

Das Experiment in Abschnitt 3 zeigt, dass ein einfaches Neuronales Netz zur Verwendung bei einer Klassifizierungsaufgabe auf den Komponenten CPU, GPU und Hailo8L Chip bei der Erkennung eines Katzenbildes verwendet wurde. Hierzu wurde in allen Fällen das gleiche Modell verwendet. Im Rahmen des Experiments wurde die Laufzeit auf den Hardwarekomponenten Rapsberry Pi CPU, Notebook CPU, Grafikkarte und Hailo8L Chip getestet. Überraschend zeigte sich hierbei, dass der Hailo8L Chip im Mittel und die Grafikkarte in einzelnen Iterationen die beste Laufzeit erreichte. Der in Abschnitt 1 theoretische Speedup der Grafikkarte konnte durch das Experiment nicht bestätigt werden.

Es wurde ein simples Modell verwendet. Dies zeigt allein die Größe der erstellten Modelldateien. In weiteren Experimenten sollte untersucht werden, wie sich die Laufzeit bei komplexeren Modell verhält und bei welchen Modellen der Versuchsaufbau mit dem Raspberry Pi aufgrund des geringeren Leistungsvermögens im Vergleich zum leistungsstärkeren Notebook an die Belastungsgrenze stößt.

Wenn nicht nur die Leistung der Hardware, sondern auch der Modelle verglichen werden soll, sollten zukünftige Experimente im Hinblick auf die Modellqualität durchgeführt werden. Hier kann insbesondere untersucht werden, ob sich die in Unterabschnitt 2.2 beschriebenen Umwandlungsschritte auf die Qualität des Ausgangsmodells auswirken.

Literatur

[gila24] giladn. Dataflow Compiler (DFC) Availability. https://community.hailo.ai/t/dataflow-compiler-dfc-availability/1476, 2024. Stand: 02.07.2024, Aufruf am: 25.09.2024.

[Hail24a] Hailo. Hailo-8L M.2 Entry-Level Acceleration Module. https://hailo.ai/products/ai-accelerators/hailo-8l-m-2-ai-acceleration-module-for-ai-light-applications/, 2024. Stand: 23.09.2024, Aufruf am: 23.09.2024.

[Hail24b] Hailo. Hailo AI Software Suite. https://hailo.ai/de/products/hailo-software/hailo-ai-software-suite/, 2024. Stand: 24.09.2024, Aufruf am: 24.09.2024.

[Hail24c] Hailo. Hailo Dataflow Compiler Overview. https://hailo.ai/developer-zone/documentation/v3-28-0/?sp_referrer=overview/overview.html, 2024. Stand: 25.09.2024, Aufruf am: 25.09.2024.

[Hail24d] Hailo. Python inference tutorial. https://hailo.ai/developer-zone/documentation/hailort-v4-18-0/?sp_referrer=tutorials_notebooks/notebooks/HRT_0_Inference_Tutorial.html, 2024. Stand: 25.09.2024, Aufruf am: 25.09.2024.

[Nvid24] Nvidia. Laptops der GeForce RTX 40-Serie. https://www.nvidia.com/de-de/geforce/laptops/, 2024. Stand: 23.09.2024, Aufruf am: 23.09.2024.

[ONNX24] ONNX. Open Neural Network Exchange. https://onnx.ai/index.html, 2024. Stand: 25.09.2024, Aufruf am: 25.09.2024.

[PGML+19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai und S. Chintala. *PyTorch: an imperative style, high-performance deep learning library.* Curran Associates Inc., Red Hook, NY, USA. 2019.

[PyTo24] PyTorch. Training with PyTorch. https://pytorch.org/tutorials/beginner/introyt/trainingyt.html, 2024. Stand: 24.09.2024, Aufruf am: 24.09.2024.

[SHZZ+18] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov und L.-C. Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, S. 4510–4520.