Assignment A1: Roofline Plots and Hardware Benchmarking

1st William Daugherty-Miller
M Eng. Operations Research and Information Engineering
Cornell Tech
New York City, United States
wld37@cornell.edu

Abstract—**This paper aims to explore and understand the intersection between hardware and software. By understanding how the model properties, we can gain insights into how to better design hardware or allocating resources to execute a model.**

DICLAIMER: I am using two late days for this assignment. In the spirit of transparency, I worked with, talked to and helped Ajan, Anastasia, Nick, Naman and Bill. Additionally, I used code from stack overflow and had improvements and suggestions given by Tabine.ai, Chatgpt, Github Copilot and balckbox.

Background

Do online research and coding to complete the following tasks. You will need to submit (1) the code used to plot and calculate the requested metrics and plots, and (2) a detailed assignment writeup describing your methodology, showing your calculations and citing any reference or online material that you used. The submitted code should be executable (for example, a Jupyter Notebook) and it should be sufficient to reproduce all of the results in your assignment write-up. Please make sure to clearly denote the different sections in your notebook(s), either by using markdown headers in text fields or by using separate files for each section.

In this assignment, we will focus on understanding and comparing the computation and memory capabilities of different computer chips under different workloads. By benchmarking popular DNNs, we will also understand the performance, efficiency and hardware utilization of important DNNs on common hardware. Note that throughout this assignment, you can simply use random tensors when measuring the performance of DNNs.

Assignment

1 Chip Analysis
   1.1 Find the peak FLOPs/s and memory bandwidth of at least 10 different chips. Please choose a diversity of hardware platforms including at least a CPU, a GPU, an ASIC and an SoC. Map and label them on a roofline plot. Note that you may use more than one (optionally log-scale) plot if that makes your visulization clearer, for example, one plot can be used to visualize data-center chips and the other for mobile-phone chips. Include how you computed the peak FLOPs/s and memory bandwidth of each chip and/or cite any sources that you have used. Comment briefly on how different chips compare to each other.

2 DNN Compute and Memory Analysis
   2.1 Pick at least 10 different DNNs to analyze. You may use torchvision, torchaudio, pytorchcv or any available model definitions.
   2.2 Find the number of FLOPs and memory footprint of each model and calculate the operational intensity. Note that you may use any PyTorch-compatible libraries to compute these values (for example thop, torchsummary, ptflops or others. Visualize the results using bar charts or other chart types as you see fit. The idea is to visually compare the models in an easy way.
   2.3 Overlay the operational intensity on the roofline plots of the CPU and GPU that you have access to on Colab.
   2.4 Comment on which workloads are memory/compute bound on different hardware targets.

3 DNN Performance Benchmarking
   3.1 Plot the inference latency (at batch sizes = 1,64,256) vs. FLOPs of each model from part 2 on both the CPU and GPU that you have access to on Colab. You may find it useful to color the points on the plot based on batch size and perhaps label the points on the plot with the model name. It is important to produce an easily-readable plot.
   3.2 Repeat the point above but put parameters on the x-axis instead of FLOPs.
   3.3 Compute the rank correlation coefficient between (1) CPU latency, (2) GPU latency, (3) FLOPs and (4) parameters. Visualize this as a 2D matrix of correlation values. You may use either Spearman's or other rank correlation coefficients.
   3.4 Comment on whether FLOPs or parameters are a good estimate of how fast the model runs. Explain, with examples, why FLOPs/parameters could in some cases be an accurate indicator of latency, and why in other cases it isn't.
   3.5 Plot the throughput of your DNNs on CPU/GPU at different batch sizes.

4 Hardware Utilization and Peak Performance
  4.1 Go back to the CPU/GPU rooflines from part 2 and plot the actual performance (at different batch sizes) of each model on the roofline.
  4.2 Comment on any gaps between peak and achieved performance.
5 Inference vs training
  5.1 We would like to experimentally determine the ratio of computation that goes to the forward pass vs. the backward pass of a DNN. Compute the FLOPs and measure the runtime so some of your selected DNNs from part 2 when performing a backward pass. Compute the ratio of forward:backward pass, both in terms of FLOPs and latency. Choose a reasonable training batch size to complete this part. Comment on this ratio for different models.
  5.2 For both the forward and backward passes, plot pie charts depicting the breakdown of latency, flops and memory footprint for each layer type in your chosen DNNs.

Chip Analysis

To answer this question, the first step is to research and correctly identify what different chips will be used for the comparative analysis. In making this analysis, we compare traditional CPUs, GPUs, TPUs, SOC and FPGA chips. To begin, we either the found the FLOPs and Memory Bandwith on the manufacturer's website or benchmarking sites. If this information was not available, we then used the formulas presented in class. Additional information was required for FPGA chips to convert from GMACs to GFLOPS which was a 1:2 ratio.

GFLOPs Calculation GMACs to GFLOPS

With the specifications for the chips found(see Chip References section), we then turn our attention towards comparing them in a roof-line plot (pictured below) for all chip types.

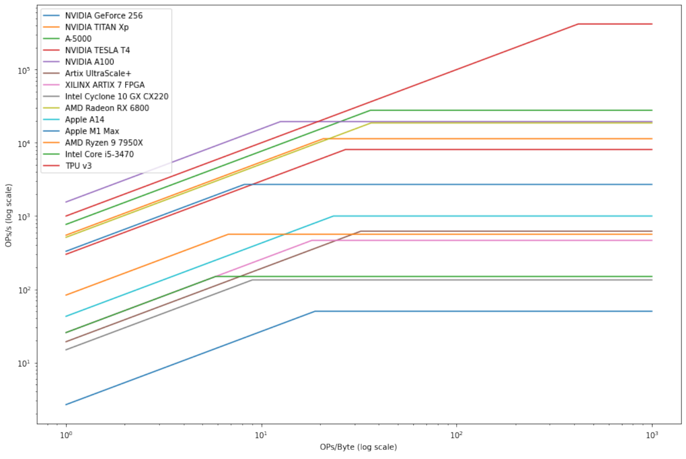

Fig. 1: Mixed chip type roof-line plot

This plot tells us some information but is not ideal given the variety of chips types we have selected, therefore we will break down the chips by designation and show the results in supplementary information section.

In the further breakdown, we are able to more closely examine the stratification of the various generations and capabilities of the chip types. A curious point to note is that while some SoC chips have a lower plateau when compared to traditional CPU and GPU counterparts the newer generation of these chips, especially the Apple Silicon line, is starting to become comparable. Another interesting point, was the inclusion of higher-end computing devices like the TPU which had the most FLOPs and memory bandwidth of the devices tested.

This section introduces us to roof-line plots however there are still interesting comparisons to be made for comparing "generations" of hardware like the NVIDIA lines, Apple Silicon lines and others. Additionally, a more thorough study/inclusion of more FPGA and various degrees of high-end computing would be useful for looking at more trends.

DNN Compute and Memory Analysis

To complete this section, 13 neural network models were selected from the PyTorch library, specifically the torchvision and torchaudio libraries. Sample PyTorch Documentation

Shown later is a table containing all the models selected for this assignment.II

With the models picked, a function was written to find the number of FLOPs and memory footprint. Below is a bar chart depicting the results of these calculations. It is worth noting that there is a significant difference in the operational intensity of the various models selected which will appear later when the various larger batch sizes are used.

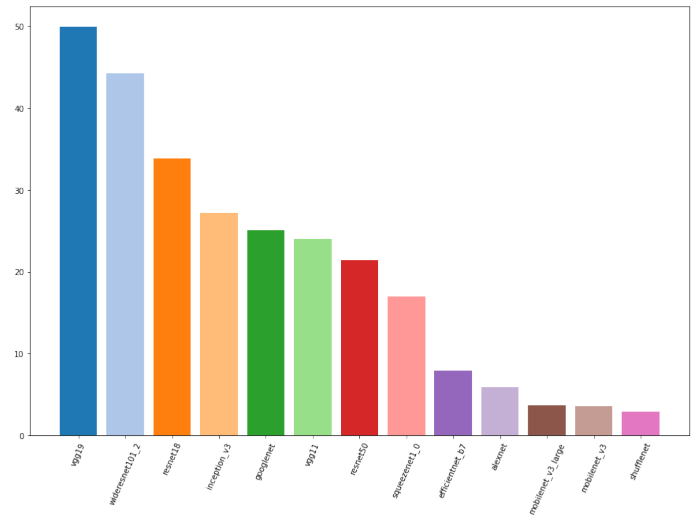

Fig. 2: DNN Model Comparison

With the models now having calculations for GFLOPS, memory footprint, and operational intensity(oi). We show the results in the previously mentioned table.

Name	GFLOPS	Bandwidth	OI
googlenet	3.01	119.95	25.094
mobilenet v3 large	0.47	126.9	3.70
resnet18	3.65	107.96	33.81
vgg11	15.2	632.78	24.02
resnet50	8.22	384.62	21.37
alexnet	1.43	242.03	5.91
mobilenet v3	0.45	126.9	3.55
vgg19	39.29	787.31	49.90
squeezenet1 0	1.65	97.14	16.99
efficientnet b7	10.53	1330	7.92
shufflenet	0.09	30.85	2.92
wideresnet101 2	45.59	1030	44.26
inception v3	5.69	209.48	27.16

TABLE I: Summary Model Calculations

Next, we overlay these results onto a roofline containing just the CPU and GPU used in Google Colab. Where we note that it appears most models have similar bounds whether it is compute or memory for both the CPU and GPU, however some like squeezenet are compute bound for bandwidth for the GPU and compute bound for the CPU. This is curious given that we expect to have most of the models being compute bound for the CPU and memory bound for the GPU, expectations come from lecture.

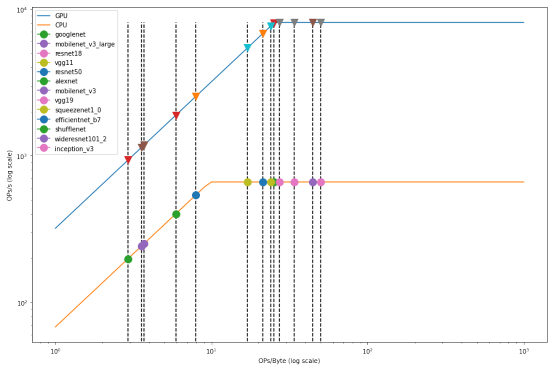

Fig. 3: Overlay of OI and Roofline

With this analysis it is worth noting that more models and CPUs/GPUs could be tested to observe trends and that there may be a difference in numbers found/calculated to what has been reported.

DNN Performance Benchmarking

For DNN model benchmarking, we create a function to run the model for CPU and GPU, storing the results in a dictionary. From this dictionary we are able to write functions that plot parameters versus latency and FLOPs versus latency. We note that these are the original plots however more information can be obtained by labeling the various models on the grapg which does get crowded at times so a non-labeled version is also included.

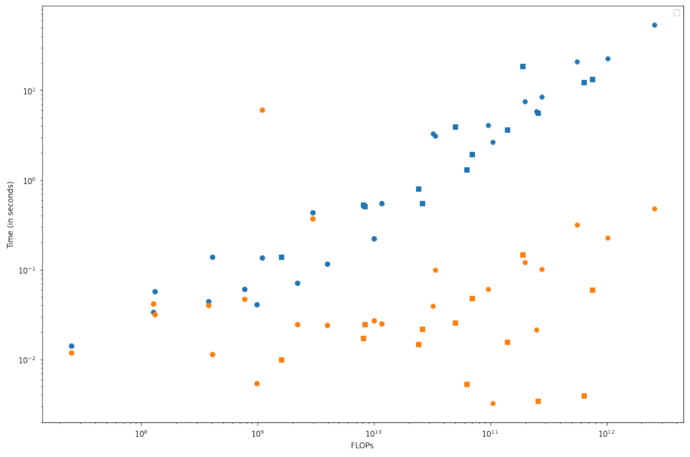

Fig. 4: FLOPs versus Latency

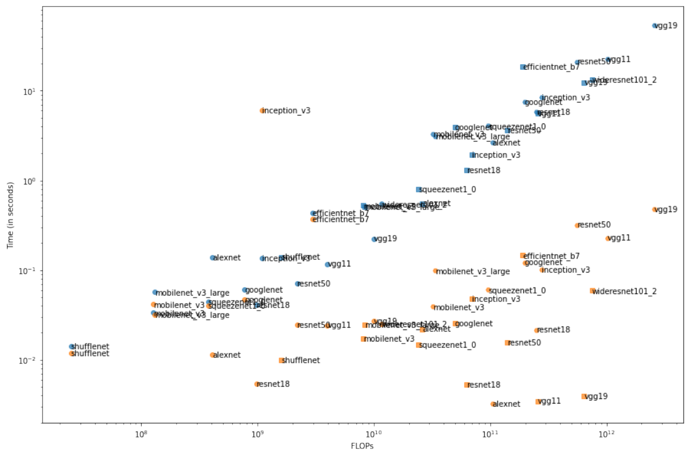

Fig. 5: FLOPs versus Latency with model labels

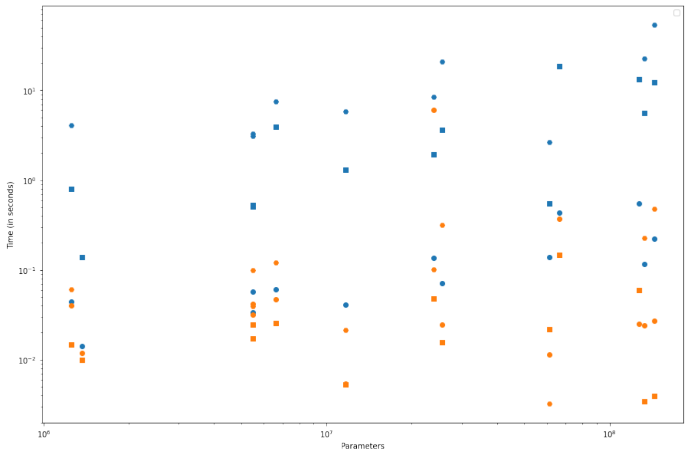

Fig. 6: Parameters versus Latency

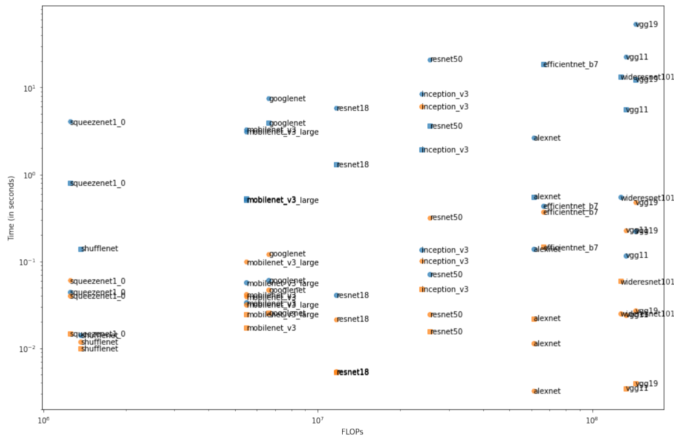

Fig. 7: FLOPs versus Latency with model labels

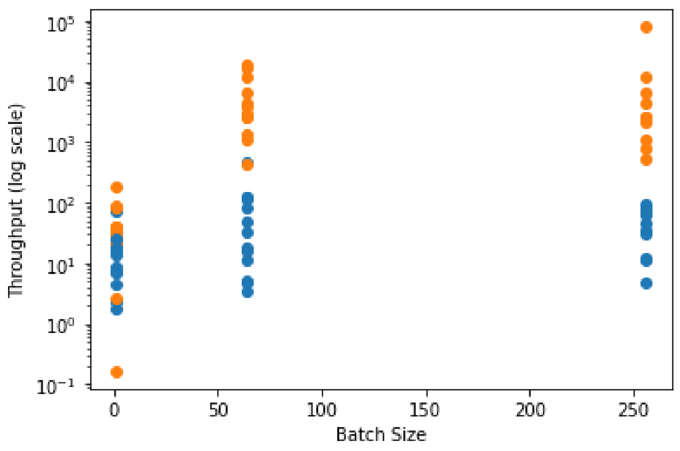

Fig. 9: Parameters versus Latency

After this examination of FLOPs and Parameters versus latency, we then examine the correlation between CPU/GPU latency, FLOPs and Parameters. Initially this was done using one correlation matrix, however after talking to Anastasia Sorokina I made three additional plots that more closely examined batch size and how this correlates the the already specified matrix(Figures attached in the supplementary data). The results of this correlation matrices show that there is a strong correlation between FLOPs and CPU latency but not with GPU latency. This may be becuase of how the two are designed, parallelization for a GPU versus sequentially for a CPU in regards to batches. It is interesting to note that parameters so not seem suited for latency in the overall spearman correleation matrix, however closer inspection of the various batch sizes reveals that it might be especially at the higher batch sizes such as 256. Source for parallelization Source for sequentially

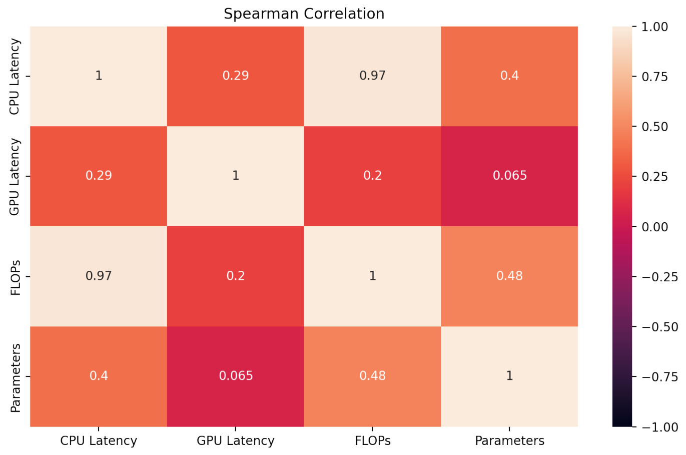

Fig. 8: Parameters versus Latency

We then come to throughout for the CPU and GPU. This plot is not surprising given what we have previously seen with CPUs and GPUs.

Hardware Utilization and Peak Performance

For this section, we plot the actual performance of the the various models on the roof-line plot for the Google Colab CPU and GPU. It is important to note that this original plot has been split into multiple for clarity and overcrowding of data points.

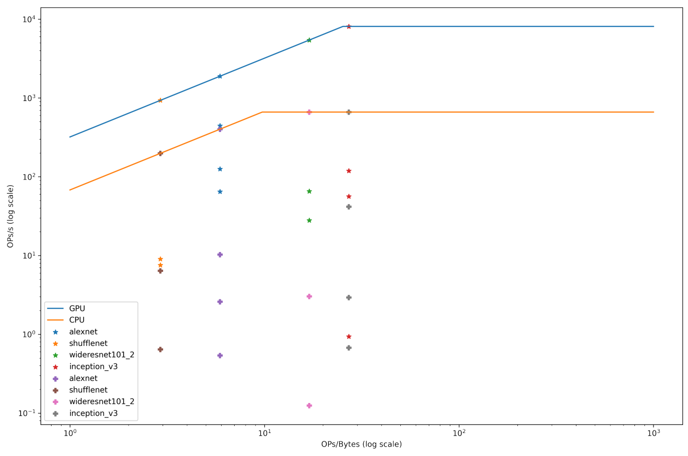

Fig. 10: Roofline plot with DNN models

We observe from the three graphs that depending on the batch, there are no bottlenecks. However when the batch size does increase we can see the respective compute and memory bottlenecks mentioned in class. However the model vgg19 seems to have some reporting errors either inherent to what has been documented or what was been calculated in the assignment as it exceeds the plot limitations.

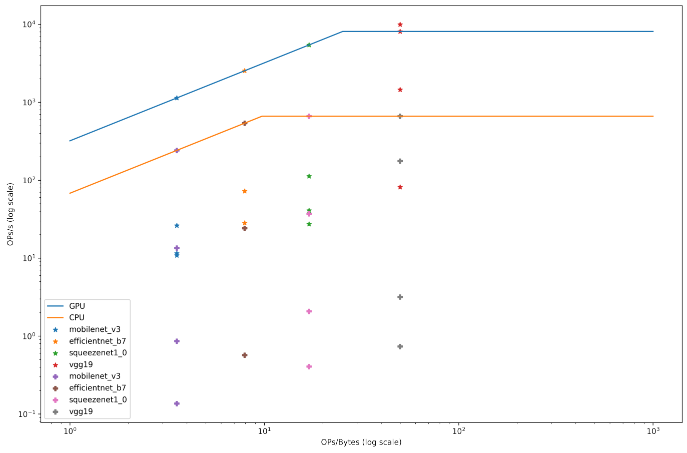

Fig. 11: Roofline plot with DNN models

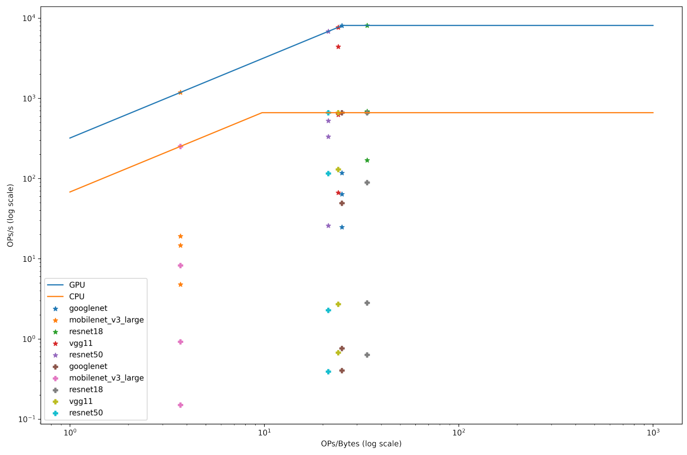

Fig. 12: Roofline plot with DNN models

Inference vs training

We then come to the final section which experimentally determines the ratio of computation for forward versus backward passes for FLOPs and run-time. There are several ways to find these ratios however the easiest method is to use a profiler. Inspiration came from reading documentation for ptflops, deep speed and various articles such as PyTorch Documentation, Medium Articles, Github repos and talking with classmates. With this information a function was created to look at the FLOPs/ latency ratios using a batch size of 32 as it is not too large and does not previously appear in batch size comparisons from prior parts. Below is a table for some of the models used previously in the homework.

Name	latency Ratio	FLOPs Ratio
googlenet	0.52	425.38
mobilenet v3 large	0.73	425.38
resnet50	0.56	425.38
alexnet	0.48	425.38
vgg19	0.41	425.38

TABLE II: Summary Model Calculations

It is worth noting that this table is most likely not accurate as the FLOPs Ration is the same for each entry. Source regarding torch profiler error. Additionally, it is worth noting that we expect a 2:1 ration however this was not observed, but other sources Source on ratio for nerual networks. Another method for finding this ratio could be to plot and empirically observe the plots as some of my other classmates have done. To do this, I coded and used chatgp to create a quick plots which did not work that well.

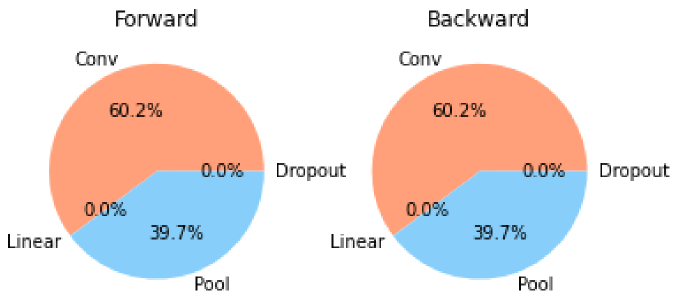

Fig. 13: Pie Chart for Googlenet

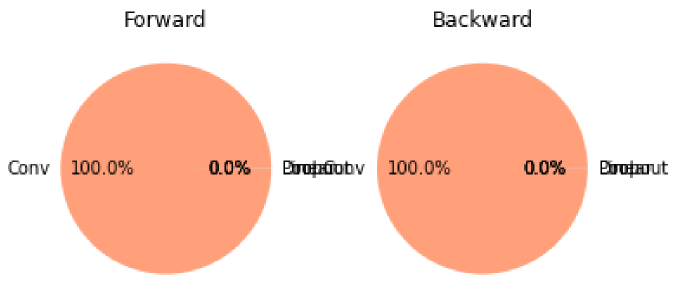

Fig. 14: Pie Chart for Mobilenet v3 large

For the second part, I found a lot of inspiration from this github repo GitHub Repo and from talking to classmates/emulating what we discussed and talked about. For the first approach, I tried two different approaches to layers, one based on the type of layer (linear, convolution, pooling, etc ...).

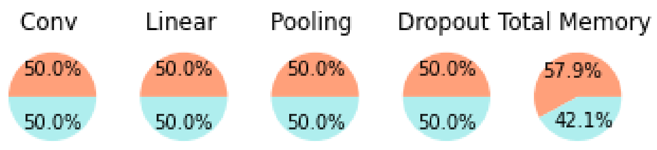

Fig. 15: Different layer types for GoogleNet

For the other approach I modelled it based on operation as seen in the linked github repository and in the profiler used previosly in the homework.

We note that both methods are not quite as detailed as they could be and so there is room for improvement here.

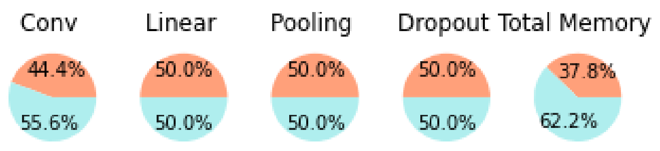

Fig. 16: Different layer types for MobileNet

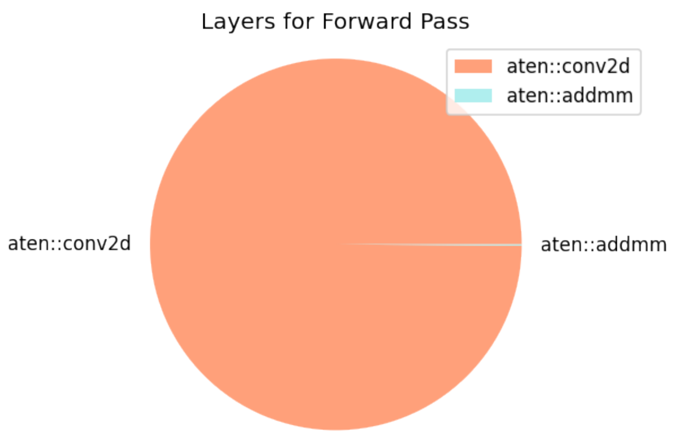

Fig. 17: Forward Pass for GoogleNet

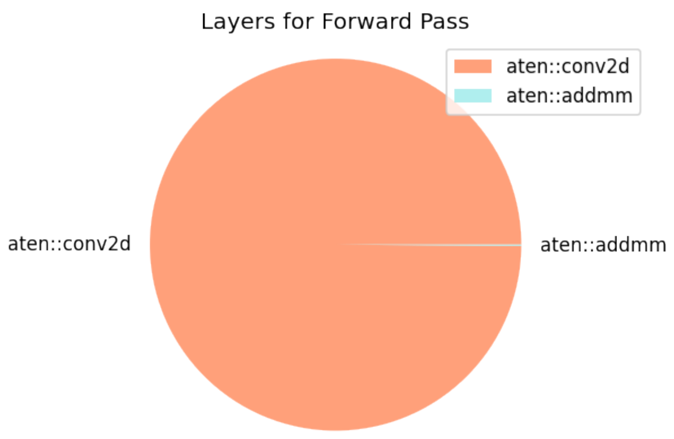

Fig. 18: Forward Pass for MobileNet

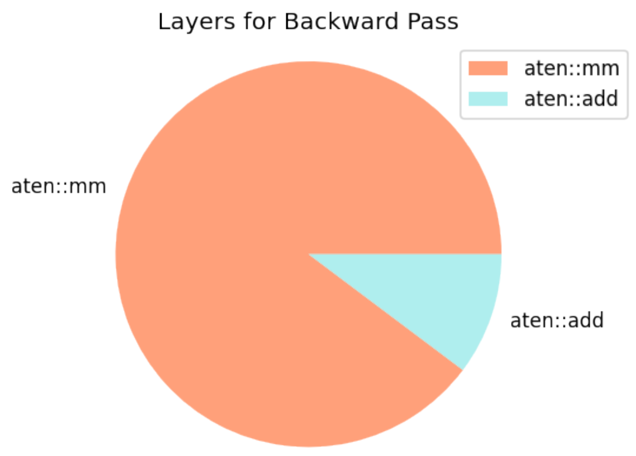

Fig. 19: Backward Pass for GoogleNet

Future Considerations

As previously mentioned in the various sections, we note that there is much more in depth analysis and research to done outside of the scope of this paper as more chips, models and methods for more accurately counting flops for both backward and forward passes. This latter point could be a new profiler or to fix current issues within existing profiler in the open source and developer communities.

Chip References

Contains all links to where information was found about the various chips for part one and other sources used for completing this assignment.

References

[1] NVIDIA GeForce 256 SDR https://www.techpowerup.com/gpu-specs/geforce-256-sdr.c731
[2] NVIDIA TITAN Xp https://www.nvidia.com/en-us/titan/titan-xp/
[3] NVIDIA RTX A5000 https://www.leadtek.com/eng/products/workstation_graphics(2)/NVIDIA_RTX_A5000(40914)/detail
[4] NVIDIA TESLA T4 https://www.techpowerup.com/gpu-specs/tesla-t4.c3316
[5] NVIDIA A100 https://www.techpowerup.com/gpu-specs/a100-pcie-40-gb.c3623
[6] TPU v3 https://cloud.google.com/tpu/docs/system-architecture-tpu-vm
[7] NVIDIA A100 SXM4 40 GB https://www.techpowerup.com/gpu-specs/a100-sxm4-40-gb.c3506
[8] NVIDIA TITAN RTX https://www.nvidia.com/en-us/deep-learning-ai/products/titan-rtx/
[9] NVIDIA GeForce RTX 3090 Ti https://www.nvidia.com/en-us/geforce/graphics-cards/30-series/rtx-3090-3090ti/
[10] Intel core i7 920 https://www.intel.com/content/www/us/en/products/sku/37147/intel-core-i7920-processor-8m-cache-2-66-ghz-4-80-gts-intel-qpi/specifications.html
[11] Dual Intel Xeon E5-2697 https://www.cpubenchmark.net/cpu.php?cpu=Intel+Xeon+E5-2697+v2+40+2.70GHz&id=2009&cpuCount=2
[12] AMD Ryzen 5 3600 https://www.techpowerup.com/cpu-specs/ryzen-5-3600.c2132
[13] Intel Core i9-13900K https://gadgetversus.com/processor/intel-core-i9-13900k-specs/
[14] Intel Core i5-13500 https://www.techspot.com/review/2612-intel-core-i5-13500/
[15] AMD FX-8350 https://www.techpowerup.com/cpu-specs/fx-8350.c1099
[16] AMD Ryzen 9 7950X https://www.techpowerup.com/cpu-specs/ryzen-9-7950x.c2846
[17] Intel Core i5-3470 https://www.cpu-world.com/CPUs/Core_i5/Intel-Core%20i5-3470.html
[18] Intel Core i7-13700K https://www.techpowerup.com/cpu-specs/core-i7-13700k.c2850
[19] AMD Ryzen 7 5800X https://www.amd.com/en/products/cpu/amd-ryzen-7-5800x
[20] Apple A14 https://www.notebookcheck.net/Apple-A14-Bionic-Processor-Benchmarks-and-Specs.494578.0.html
[21] Apple A15 https://www.cpu-monkey.com/en/igpu-apple_a15_5_gpu_cores-275
[22] Apple A16 https://www.cpu-monkey.com/en/igpu-apple_a16_5_gpu_cores-344
[23] Apple M1 Max https://www.notebookcheck.net/Apple-M1-Max-32-Core-GPU-Benchmarks-and-Specs.579797.0.html

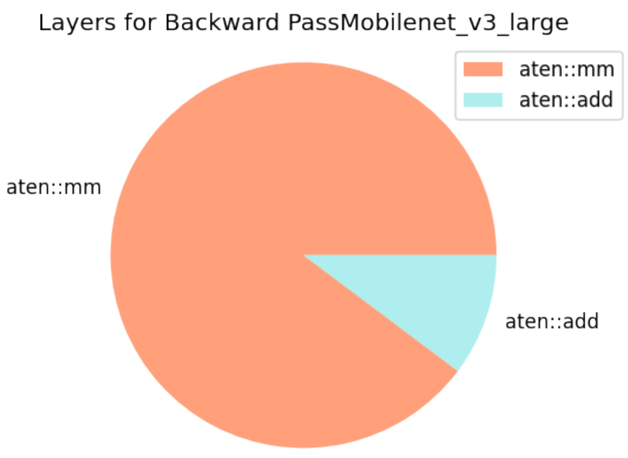

Fig. 20: Backward Pass for MobileNet

[24] Apple M1 Ultra https://www.cpu-monkey.com/en/igpu-apple_m1_ultra_64_core-317

[25] Apple M2 https://nanoreview.net/en/cpu-compare/apple-m2-vs-apple-m1-max

[26] Qualcomm Snapdragon 888 https://nanoreview.net/en/soc/qualcomm-snapdragon-875

[27] Qualcomm Snapdragon 865 https://nanoreview.net/en/soc-compare/qualcomm-snapdragon-875-vs-qualcomm-snapdragon-865

[28] Qualcomm Snapdragon 720G https://nanoreview.net/en/soc/qualcomm-snapdragon-720g

[29] MediaTek Dimensity 9200 https://nanoreview.net/en/soc/mediatek-dimensity-9200

[30] Artix UltraScale+ https://www.xilinx.com/products/silicon-devices/fpga/artix-ultrascale-plus.html
ARTIX 7 FPGA XILINX ARTIX 7 FPGA https://www.xilinx.com/products/silicon-devices/fpga/artix-7.html

[31] Intel Cyclone 10 GX CX220 https://ark.intel.com/content/www/us/en/ark/products/series/141555/intel-cyclone-10-gx-fpga.html

Supplementary Data

All code and other resources used for this assignment can be found on the A1 github link and the referenced items below.

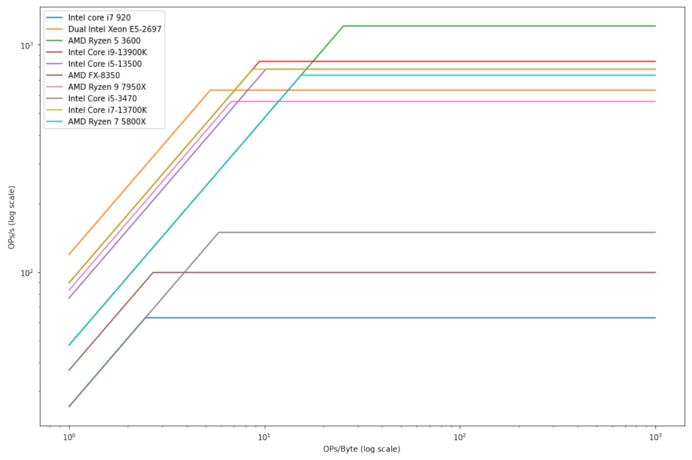

Fig. 21: CPU Chip Analysis

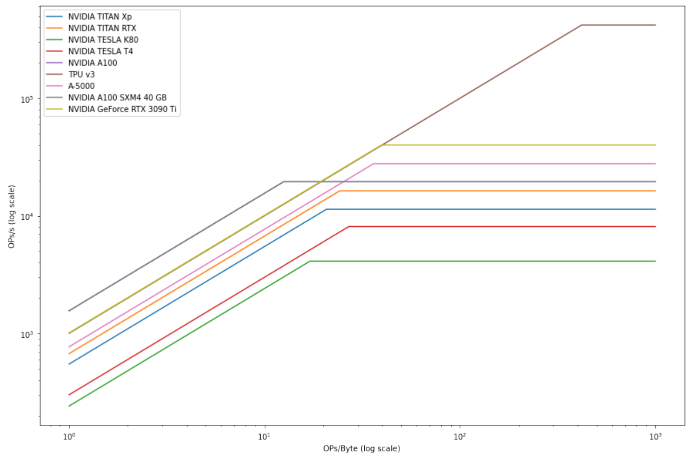

Fig. 22: GPU Chip Analysis

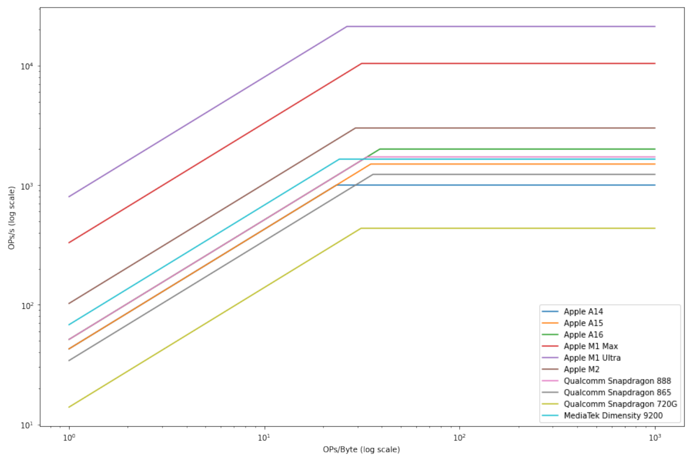

Fig. 23: SoC Chip Analysis

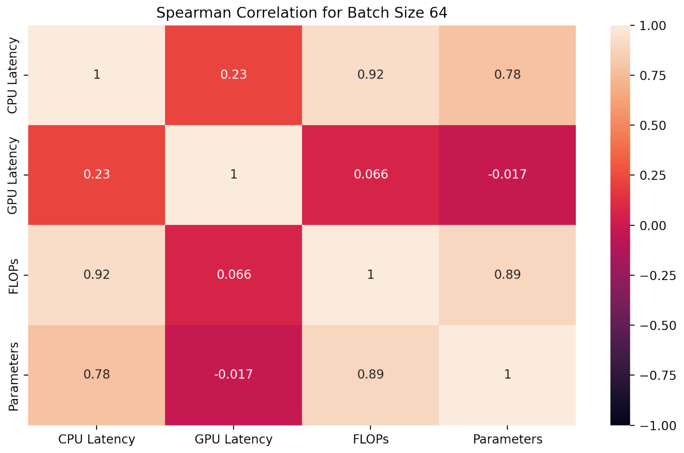

Fig. 26: Parameters versus Latency

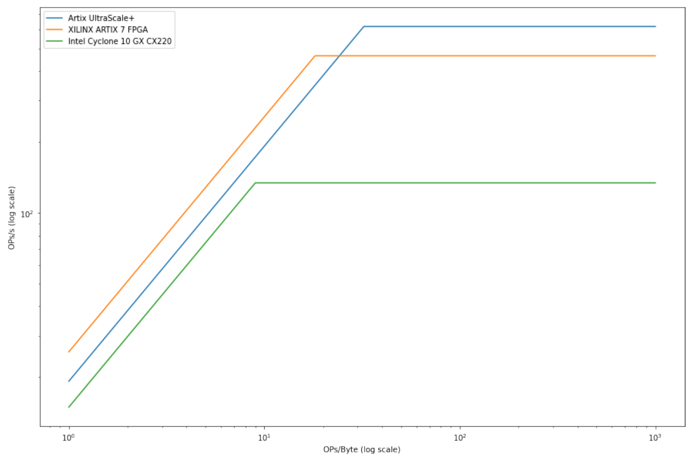

Fig. 24: FPGA Chip Analysis

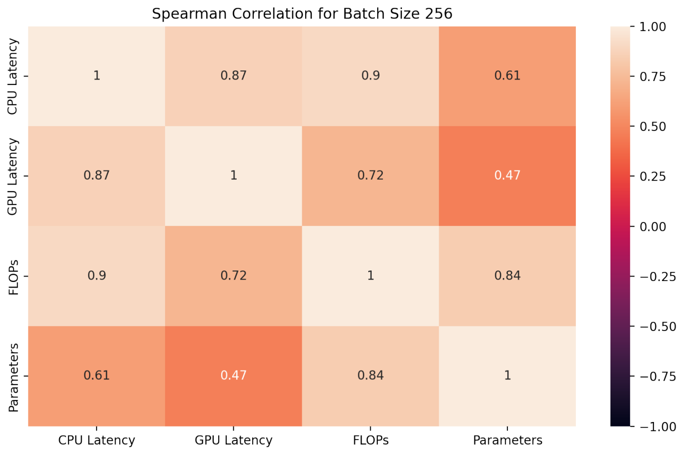

Fig. 27: Parameters versus Latency

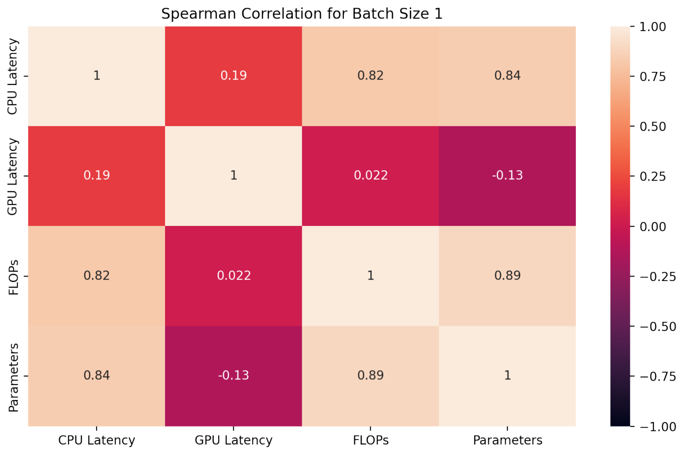

Fig. 25: Parameters versus Latency