Dylan Berens

# Carrying the PyTorch Through the Night

*Personal Energy Project: Estimating Energy Usage of Image Classification with PyTorch*

## I: Introduction

The current technology climate has raised public concerns about the energy usage demands that will result from the significant increase in the use of computational resources as a result of machine learning, not simply in terms of the processing power but also the sheer energy amounts that are needed to run large scale systems. Given my passion for and speciality in Data Science I found a compelling topic for our Personal Energy Project to be tracking the energy usage of the large image classification tasks that I have been working on for the past several months. Using the Python library PyTorch and the computer's built-in terminal, I was able to measure computational usage, extract that data and then plot it. The main alternative I will be comparing is not running such energy-intensive programs, which will be compared with my typical energy usage when not running such a model overnight. I expected to find that the energy and CO2 (Carbon Dioxide) emissions from such tasks were significantly less than the majority of standard tasks by nature of the lack of energy used on output in the form of audio or visual, compared to things such as having a television on. However, media coverage has highlighted the high energy usage of these tasks, and as a result there was a wide range of energy usage that would have been unsurprising. I will be exploring running image classification tasks overnight, the possible alternatives, and evaluating the results I find.

## II. Analyzing Image Classification Energy Usage

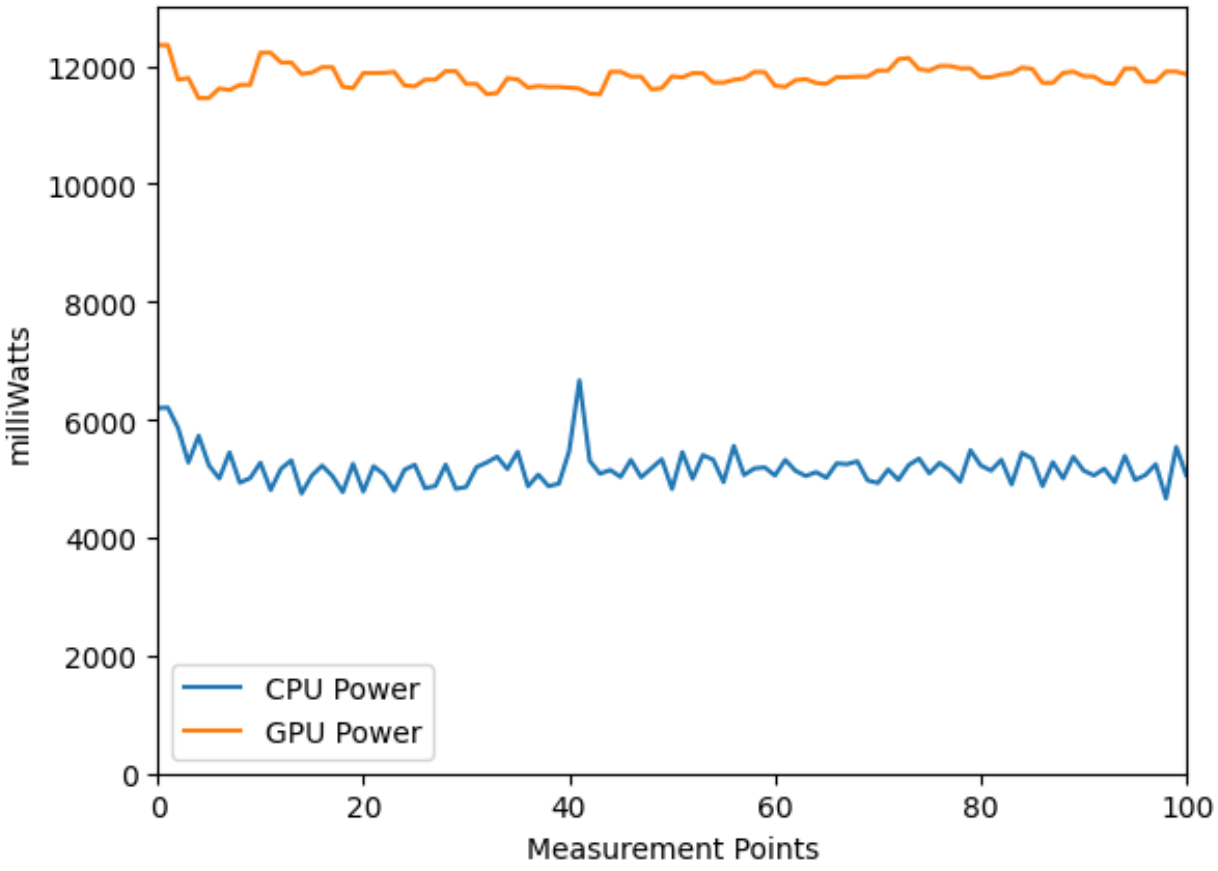
Using the model of ResNet50 in the Interactive Development Environment of Visual Studio Code in conjunction with the Python library PyTorch, I perform image classification tasks several times every week, running the program for a few minutes before going to bed, and hopefully waking up to a completed program 6-10 hours later. The current model uses a publicly available set of 100,000 images of 100 environment categories ranging from an outdoor monastery to a fire station to a volcano. Although I hand typed all of the code and used my own values for parameters and made necessary adjustments including changing the resizing and cropping of the training set and restructuring the program, I did not arrive at the functioning model independently, utilizing resources such as the Computer Vision for Data Scientists LinkedIn course for general structure, PyTorch's website for understanding the process of the model, and TensorSpace.org for visualizing the individual convolutions used in ResNet50.

The longest runtime I have had on one of the earliest iterations of the same type of program was exactly 1,100 minutes (18 hours, 20 minutes). However, I made several adjustments since then to improve performance. Initially, the program was resizing the training images (which had various sizes) to 256 by 256, before taking a crop about 2/3 the size of the image, to then be used later on in the convolutional layers, and would repeat the process for 2 iterations (epochs). After the significant length of runtime, I adjusted these parameters for several runs, first commenting out the resizing, decreasing the crop size, and altering the number of epochs, successfully decreasing the runtime considerably at the cost of model accuracy and prediction capabilities. Ultimately, however, after researching the method being used in the process to use "cuda" as the GPU (Graphics Processing Unit) to decrease the runtime of the

program, I discovered my model of MacBook actually uses “mps” as the equivalent way of using GPU to improve performance as it relates to speed. As a result, I replaced “cuda” in this line of code with “mps”: device = torch.device("mps" if torch.backends.mps.is_available() else “cpu”). This dramatically improved the model’s performance given when it could not find “cuda” it then used the “cpu” which reduced the efficiency of the model until it was corrected. Resulting from the improved performance, I was now free to improve the model’s configuration by reintroducing resizing, increasing the crop size, and increasing the number of epochs considerably to as many as 7.

After experimenting with a few possible methods of measuring the energy usage of running one of these classification programs, I decided to use the built in terminal to output the GPU and CPU usage continuously while in the midst of running the program, and after collecting roughly 250 measurements, I consolidated them into a dataset to perform simple data extraction and mathematical operations to estimate the total energy usage to be extrapolated over the course of a year. After extracting the GPU and CPU usage measurements into individual numbers separated from other characters collected into two lists, I then simply summed each list and divided the lists by the corresponding length of the lists in order to calculate the average energy usages. However, given the energy measurements collected at each instance are measured in milliWatts, I had to then divide by 1,000 in order to convert into the more readable Watts measurement, resulting in an average CPU Power of 5.18 Watts, average GPU Power of 11.84 Watts, and the total average power being used over the course of the collection as 17.02 Watts.

From there, extrapolating the energy usage was a straightforward process. Given the typical runtime of my image classification tasks of around 709 minutes (11.82 hours), that means




the average Watt-hour usage of one of these programs is 17.02*11.82=201.176 Watt-hours in a day. Given I have been running these programs overnight roughly 5 times a week, 201.176*5=1,005.88 Watt-hours per week = 1.006 Kilowatt-hours per week. Extrapolating this action over 52 weeks in a year, 1,005.88*52=52,305.76. Meaning if I run these programs 5 nights a week, every week, I use 52.31 Kilowatt-hours in a year purely from executing image classification models (188,316 Kilojoules).

## III. Analyzing Alternative Behavior

The difficulty with constructing alternatives to an image classification model is that generally, such a task is uniquely computationally-focused. Even the sole exception of the human

brain, capable of surpassing computational performance on smaller datasets, would be incapable of doing image classification on all of ImageNet (roughly 14 million images), or the more common subset of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) which has well over 1 million training images. Certainly one alternative is to not use my computer for such tasks, opting to study business instead. My typical usage of my laptop computer before beginning image classification tasks this semester was roughly 55% of the battery in a day, charging the battery completely overnight, every night. According to the MacBook Pro Tech Specs available on Apple's Support website, my laptop has a 72.4-watt-hour lithium-polymer battery (Apple Support, 2024). 72.4*0.55=39.82. Meaning on an average day studying less computationally demanding subjects, my laptop consumes 39.82 Watt-hours. When extrapolated over an entire year (39.82*365=14,534.3 Watt-hours), this usage amounts to 14.53 Kilowatt-hours (52,308 Kilojoules) from standard, non-programming activities.

The alternative of using my laptop to study things outside of Computer Vision, image classification, and intensive programming would result in using only 27.8% of the same energy from my laptop. Phrased differently, running computationally-demanding programs such as a ResNet50-based image classification task results in 360.3% more energy usage from my laptop.

In calculating the CO2 emissions from such programs rather than strictly energy usage, the answer is less clear-cut. This is a result of the reality that electricity generation for the power grid is a mix of everything from nuclear, solar, wind, hydroelectric, geothermal and even fossil fuels, each with their own amounts of emissions per Kilowatt-hour, and the percentage mix of each of those contributions varies widely depending on the region of the United States and where you are in the local community, amongst other factors such as time of year and even time of day.

That being said, the U.S. Energy Information Administration (EIA) reports that in 2023, the carbon emissions resulting of electricity generation plants was 0.81 pounds of CO2 per Kilowatt-hour (Energy Information Administration, 2023). Being mindful of overcomplicating this project, I will be using the EIA estimate in the CO2 emission calculations.

Running computationally-demanding programs on my laptop for the period of a year results in 42.37 pounds of CO2 emissions (52.31*0.81=42.371). In contrast, standard laptop usage results in 11.77 pounds of CO2 emissions in a year (14.53*0.81=11.769). Again, running these programs results in roughly four times the carbon dioxide emissions compared to standard use, but for a frame of reference we will explore other CO2 emission amounts.

## IV. Conclusion

Despite the Kilowatt-hours being in a reasonable scale of energy usage for my computationally-demanding tasks, the Carbon Dioxide emissions really separate my daily programming from anything directly involved in burning gasoline. The Environmental Protection Agency reports that for the average passenger vehicle, burning one gallon of gasoline results in 8,887 grams of CO2 emissions (Environmental Protection Agency, 2024). Given there are roughly 454 grams in a pound, that means burning one gallon of gasoline results in about 19.57 pounds of CO2 emission. This is a surprising realization, in part because a gallon of gasoline weighs around 6 pounds (JD Power, 4), yet the emissions from burning that gallon are more than triple in weight. My instinct suggested that by nature of the process of combustion, weight would be lost in the equation, not gained. However, due to the complex chemistry that

occurs from burning gasoline, and the interaction with Oxygen, the resulting emission weight differs from my intuition.

At any rate, the understanding that burning just over two gallons of gasoline produces a comparable amount of CO2 emissions as running my computationally-demanding and high energy usage programming tasks for an entire year is both comforting and troublesome. Comforting in that I should not be concerned about how much energy my personal laptop is using under these conditions—as large scale corporate Computer Vision tasks would be far more relevant—and troublesome given that efficiency in one specific area can be entirely negated by driving to campus twice. Reducing my laptop energy usage to 27% of its high energy usage when running continuously overnight is rendered effectively irrelevant as long as I drive a vehicle that burns gasoline. Despite the environmental impact of my programming being far and away negligible as long as I commute, there are two main factors that could influence opting for alternative behavior. If economic conditions were unfavorable for programming skills (which the market currently suggests is the case) I would consider changing the subject of my studies again. Moreso, if there was a subject that engaged my critical thinking and problem solving skills to the same degree, while not needing to be done electronically on screens, I would opt for that. My confident conclusion from this project is that given my concern for the quality of our environment, I should continue to utilize my computationally-demanding programming without reservation, but apply my machine learning knowledge to explore alternative solutions to transportation that are not reliant on gasoline, given a one-way trip to Austin would result in the same amount of CO2 emissions as computationally-demanding programming for three years.

## Works Cited


1. "MacBook Pro (14-Inch, M4, 2024) - Tech Specs." *Apple Support*, support.apple.com/en-us/121552. Accessed 10 Mar. 2025.

2. "Frequently Asked Questions (Faqs) - U.S. Energy Information Administration (EIA)." *Frequently Asked Questions (FAQs) - U.S. Energy Information Administration (EIA)*, www.eia.gov/tools/faqs/faq.php?id=74&t=11. Accessed 10 Mar. 2025.

3. "Greenhouse Gas Emissions Typical Passenger Vehicle." *EPA*, Environmental Protection Agency, 23 Aug. 2024, www.epa.gov/greenvehicles/greenhouse-gas-emissions-typical-passenger-vehicle.

4. Hawley, Dustin. "How Much Does Gasoline Weigh per Gallon?" *JD Power*, JD Power, 18 Dec. 2022, www.jdpower.com/cars/shopping-guides/how-much-does-gasoline-weigh-per-gallon.

The majority of code to estimate energy usage can be condensed into 7 lines:

cpu_powers = [int(m.group(1)) for m in re.finditer(“CPU Power: (\d+) mW”, data)]
gpu_powers = [int(m.group(1)) for m in re.finditer(“GPU Power: (\d+) mW”, data)]
avg_cpu_power = sum(cpu_powers) / len(cpu_powers)
avg_gpu_power = sum(gpu_powers) / len(gpu_powers)
total_avg_power = avg_cpu_power + avg_gpu_power
total_avg_power /= 1000 # this divides total avg power by 1000 to convert milliWatts to watts
print(f”Total Average Power: {total_avg_power:.2f} W”)

Longest Runtime: 1100 minutes
Typical Runtime: 709 minutes =11.82 hours

Average CPU Power: 5.18 Watts
Average GPU Power: 11.84 Watts

Total Average Power: 17.02 Watts

Energy Usage of Running a 5 Epoch Image Classification task with ResNet50 using PyTorch:
17.02 Watts * 11.82 hours = 201.12 Watt-hours in 1 day

Roughly 5 days a week = 201.176*5=1,005.88 Watt-hours per week
52 weeks a year = 1005.88*52=52,305.76 = 52.31 Kilowatt-hours per year

Macbook Pro with M4 Chip has 72.4 Watt-hour battery
72.4 *0.55 =39.82
Normal day use 39.82 Watt-hours

39.82*365=14,534.3 Watt-hours = 14.53 Kilowatt-hours per year under normal conditions

This is the full code written in python to extract the data of computational usage and plot on a graph:

```
import re

data = """
CPU Power: 6196 mW
GPU Power: 12352 mW
GPU Power: 12352 mW
CPU Power: 6206 mW
```

```
… etc for the entire dataset of hundreds of measurements of
usage on my computer running an image classification task for
100,000 images
"""

#extract power values
cpu_powers = [int(m.group(1)) for m in re.finditer(r"CPU Power:
(\d+) mW", data)]
gpu_powers = [int(m.group(1)) for m in re.finditer(r"GPU Power:
\s+(\d+) mW", data)]

# Calculate averages
avg_cpu_power = sum(cpu_powers) / len(cpu_powers) if cpu_powers
else 0
avg_gpu_power = sum(gpu_powers) / len(gpu_powers) if gpu_powers
else 0
total_avg_power = avg_cpu_power + avg_gpu_power

# Convert milliWatts to Watts
avg_cpu_power /= 1000
avg_gpu_power /= 1000
total_avg_power /= 1000

# Show results
print(f"Average CPU Power: {avg_cpu_power:.2f} W")
print(f"Average GPU Power: {avg_gpu_power:.2f} W")
print(f"Total Average Power (CPU + GPU): {total_avg_power:.2f}
W")



import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import rcParams

sns.lineplot(data=cpu_powers, label="CPU Power")
sns.lineplot(data=gpu_powers, label="GPU Power")
plt.ylabel("milliWatts")
plt.xlabel("Measurement Points")
plt.xlim(0, 100)
plt.ylim(0, 13000)
plt.show()
```