TOWARDS A GREEN FUTURE: ENERGY EFFICIENT CONVERSATIONAL AI ON THE EDGE

by

Kaylee Williams

HONORS THESIS

Submitted to Texas State University
in partial fulfillment
of the requirements for
graduation in the Honors College
May 2022

Thesis Supervisor:

Apan Qasem

ACKNOWLEDGEMENTS

Thank you to my advisor, Dr. Apan Qasem, for supporting my ideas and guiding me through my research. I am grateful for the opportunity to learn more about energy efficiency in computing. It was a great experience working with you.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

This project aims to show that emerging compute- and data-intensive workloads can be executed in an energy efficient way on low-power edge devices. To this end, I set up a cloud compute cluster consisting of three Raspberry Pis based on the ARM architecture. I then built a conversational AI app, a simple chatbot, to run on this cluster. My proposed framework reduces the energy cost in two ways (i) there is no need to communicate to back-end servers, saving bandwidth and (ii) all computation takes place on a low-power ARM processor, greatly reducing the carbon intensity at the cost of slightly diminished performance. Right now and in the future, complex applications can be created and run efficiently using the proposed framework.

1. INTRODUCTION

1.1 Importance of energy efficient computing

There is a need for energy efficiency in computing due to our current and project energy consumption. In data centers, the energy consumption is immense. They consume about 25% of global energy consumption and will continue to increase in use as data grows exponentially. Computing larger amounts of data takes longer and requires a significant amount of energy. This is a key issue to energy efficiency and as seen in Figure 1, the high use of energy from data centers produce 422,000,000,000 pounds of carbon emissions yearly. This is a significant contribution to the carbon emissions worldwide [5]. Alongside the energy consumption of data centers, Figure 2 shows that in less than two decades computing will no longer be sustainable. It is projected that the world's computers will need more electricity than our global energy production can deliver. Our world's energy production is at a stagnant production, whereas the growth of computing energy is increasing linearly [1], [19]. Both of these contributions affect the energy efficiency of computing and are notable reasons as to why we need to limit our energy consumption as well as find ways to reduce computation.

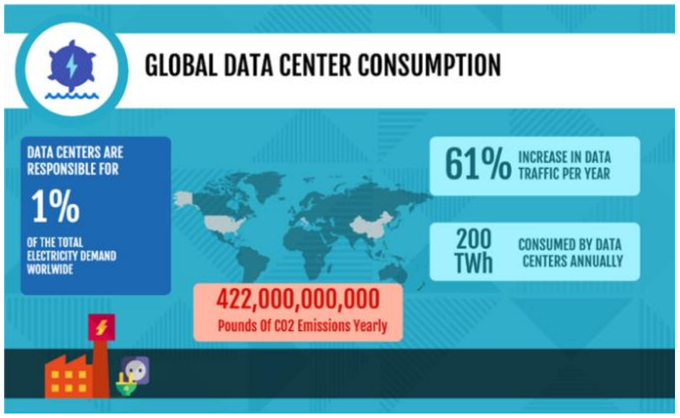

GLOBAL DATA CENTER CONSUMPTION
DATA CENTERS ARE RESPONSIBLE FOR 1% OF THE TOTAL ELECTRICITY DEMAND WORLWIDE
61% INCREASE IN DATA TRAFFIC PER YEAR
200 TWh CONSUMED BY DATA CENTERS ANNUALLY
422,000,000,000 Pounds Of CO2 Emissions Yearly

Figure 1. Data Centers' Global Energy Consumption

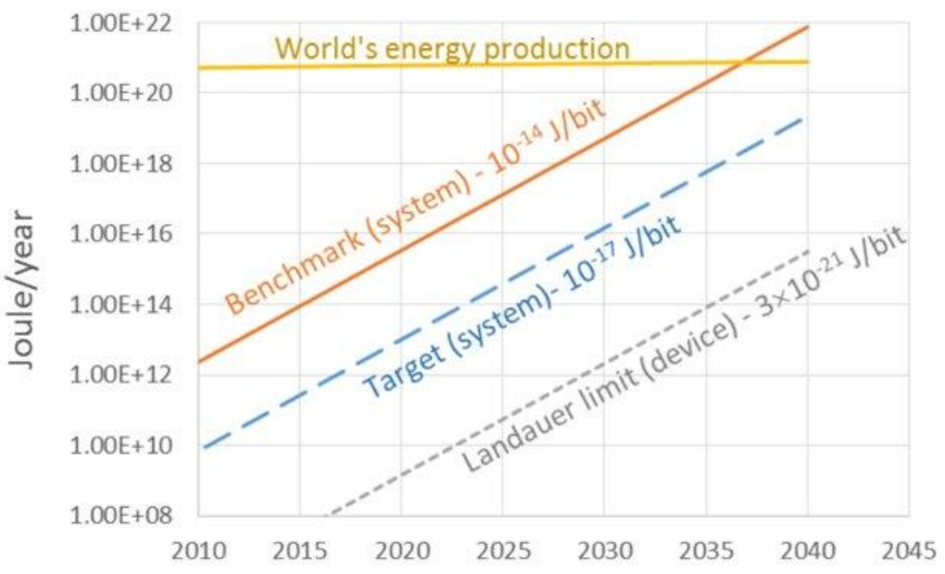


Figure 2. Expected Computer Energy Consumption

1.2 Emerging workloads and Edge

In my project, I chose to replicate an Artificial Intelligence (AI) application, since AI is a relevant topic and a growing field. One of the most common and well-known AI applications is chatbots. Chatbots are software applications that allow for communication between users. They interact with various prompts and answer questions [3]. AI is a high-demand topic and will continue to rise in the future. It is an application that we are seeing more of today and is expected to be even more present in the future. In less than two years, we should see about a 15% increase in AI Software Platforms [Figure 3], [12]. Although with AI, comes data computation. As shown in Figure 4, AI consumes a lot of energy and produces carbon emissions as well, particularly when training on a model. Training is the process where an AI is learning how to interpret data and formulate proper responses. This uses a large amount of energy because most AI handles large amounts of data and is mainly done on the cloud. So not only are AI relevant to being commonly used, but they are important in the discussion of making computing energy efficient [4].

A method of improving energy efficiency in computing is using the Edge Computing paradigm. About 8 years ago, the Edge did not exist. Originally, the internet of things, which are devices and machines, would directly communicate with the cloud or data centers. This uses a lot of bandwidth and consumes a lot of energy since it takes longer to process the data. Edge Computing was created to add another layer to the process and retrieval of data. It allows computation to be done a lot closer to the device itself, which reduces latency, improves efficiency, and consumes less energy than back-end servers [Figure 5], [2]. This paradigm is what I have frame worked my project around.

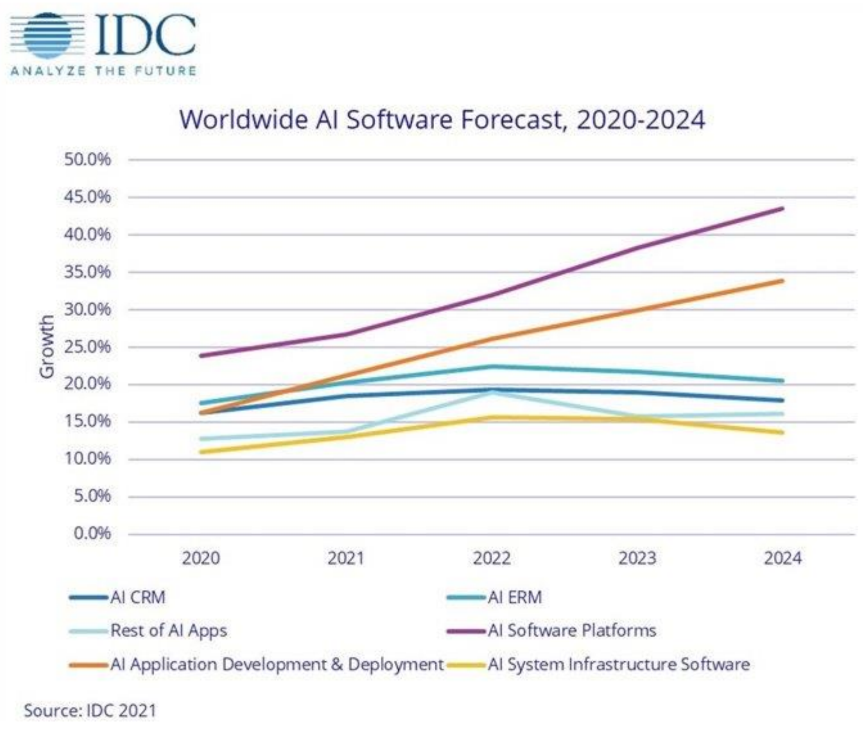


Figure 3. Worldwide AI Software Forecast, 2020-2024

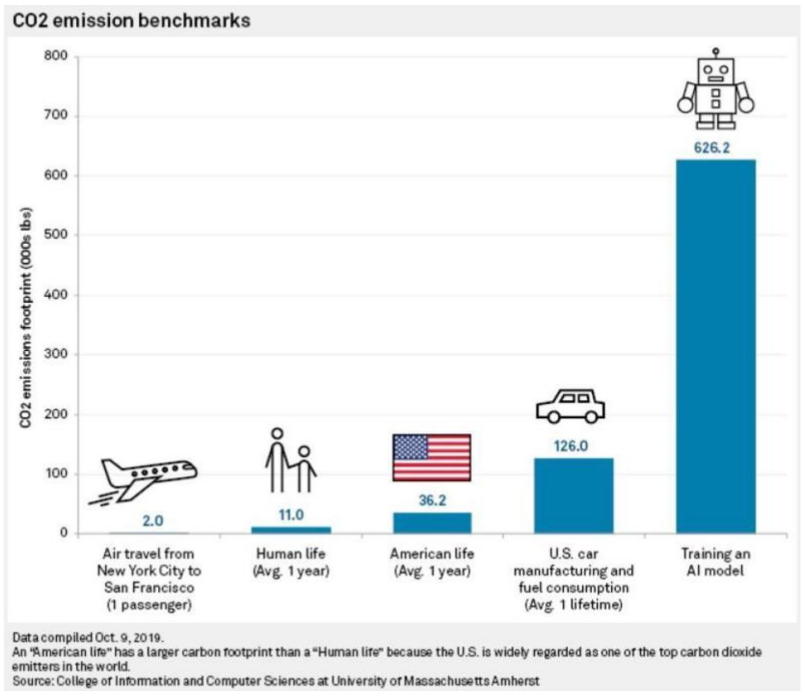


Figure 4. AI Training Carbon Emission Production

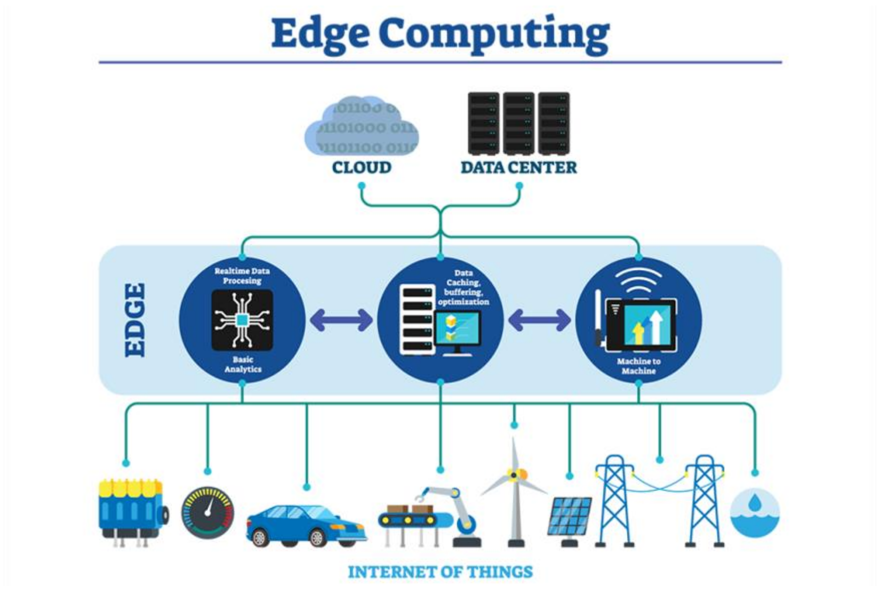


Figure 5. Edge Computing

1.3 Overview

This project aims to show that emerging compute- and data-intensive workloads can be executed in an energy efficient way on low-power edge devices. To this end, I set up a cloud compute cluster consisting of three Raspberry Pis based on the Advanced RISC Machines (ARM) architecture. I then built a conversational Artificial Intelligence (AI) I app, a simple chatbot, to run on this cluster. My proposed framework reduces the energy cost in two ways (i) there is no need to communicate to back-end servers, saving bandwidth, and (ii) all computation takes place on a low-power ARM processor, greatly reducing the carbon intensity at the cost of slightly diminished performance. Right now and in the future, complex applications can be created and run efficiently using the proposed framework.

2. BACKGROUND

2.1 Energy Efficient Computing

Cloud Computing has soared in popularity for storing and handling data, with about 83% of workloads existing on the cloud. The cloud, however, has a number of drawbacks [10]. Energy consumption is very high due to the distance between a device and server, which also makes the performance low. The operational costs continue to grow as well while data increases. The disadvantages to the cloud presented a need for more energy efficient computing. A paradigm that was introduced to address energy efficiency relating to the cloud is Edge Computing [2].

2.2 Edge Computing

Edge Computing was created to combat the issues that are present in the cloud. It brings computation and data geographically closer to the devices that are gathering the information. Edge devices consist of constrained devices, single board computers, and mobile devices [2], [10]. This is the method of my proposed framework by using low-power edge devices, a cluster of Raspberry Pis, which is composed of four small single-board computers. Raspberry Pis are based on the ARM architecture that has low power consumption and lower latency. In this project, a cluster of Raspberry Pis will be used as a server. Creating a cluster of Raspberry Pis increases the efficiency with faster compute times as well as has increased load balance with the distribution of tasks on each Pi [17].

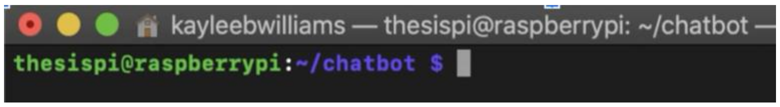
2.3 Conversational AI

Conversational AI is a set of technologies that allows humans and computers to interact. The most well-known form of Conversational AI are chatbot assistants that use conversational dialogue to accomplish a set of tasks. Users tend to use this application to ask several questions or prompt the chatbot for a response. In a conventional model of how a chatbot works, humans communicate with a Conversational AI device and in turn, the device communicates with back-end servers over the internet, which consist of multiple high-power CPUs (Central Processing Units). The servers then interpret the response and send the result back to the device. Lastly, the device sends the result to the user [3], [6]. This method of communication uses a significant amount of bandwidth and energy consumption [4]. In this project, a chatbot will be used as the type of Conversational AI.

3. APPROACH

3.1 Setting up a Raspberry Pi

To initiate this project, I set up the first Raspberry Pi to be run and accessed. First, I connected the microSD card to my MacBook and installed the Operating System (OS). I downloaded the Raspberry Pi Imager and used this to install the Raspbian OS onto the microSD card. After choosing the OS, I went into the advanced settings and set the hostname to be raspberrypi.local as well as enabled Secure Shell (SSH). SSH allows network services to be accessed remotely. To enable this, I connected the Pi to my network and set up a username and password for it. This allows me to access my Raspberry Pi from my MacBook's terminal on my local network. Once I configured the settings, I wrote this information to the microSD card [15], [20]. Then, I placed the card into the Raspberry Pi and plugged in the power source. It is running and connected when the red light is on, and the green light is flickering. In order to confirm that the Raspberry Pi is running correctly, I used the command line "sudo nmap -sn 10.245.147.0/24" on my local device to see the pi's IP address. Then, I connect to the Pi using the line "ssh thesispi@<IP Address>", and it prompts me to input the password [7], [8]. I am successfully connected to the Pi when the terminal displays the username and local hostname as shown in Figure 6.



```
thesispi@raspberrypi:~/chatbot $
```

Figure 6. Terminal view of the Raspberry Pi

3.2 Developing a Chatbot App

The next step was creating the chatbot. I first followed the documentation of a simple node.js application to understand how they work. I then used Node.js and NPM to create my application. Node.js is an event-driven JavaScript runtime that's designed to build scalable network applications. NPM is a package manager for the Node JavaScript platform and puts modules in place so the node can find them. These modules are used to import and export keywords to share with functionalities. Then, I found a simple instruction set of how to create a chatbot to run in the terminal. I used Visual Studio Code to create the program files and intents to create a dialogue between the chatbot and user. The three components of the chatbot consist of intent files, a training program, and an index program. The intents folder consists of multiple JavaScript files containing various prompts and responses. Initially, I created a hello and goodbye intent where I could input a variation of hello or goodbye, and the chatbot would take in my input and respond with an appropriate response, which in this case would be various ways to say hello or goodbye. The training file takes in data and trains a neural network on how to take in prompts and interpret them to give a correct response. Lastly, the index file takes the trained neural network and runs the chatbot allowing me to input a prompt and the chatbot to display a response, which I am implementing on a client-server model. The server side of my project is the Raspberry Pi, which will run the training program and will handle the inputs and outputs of a response [11].

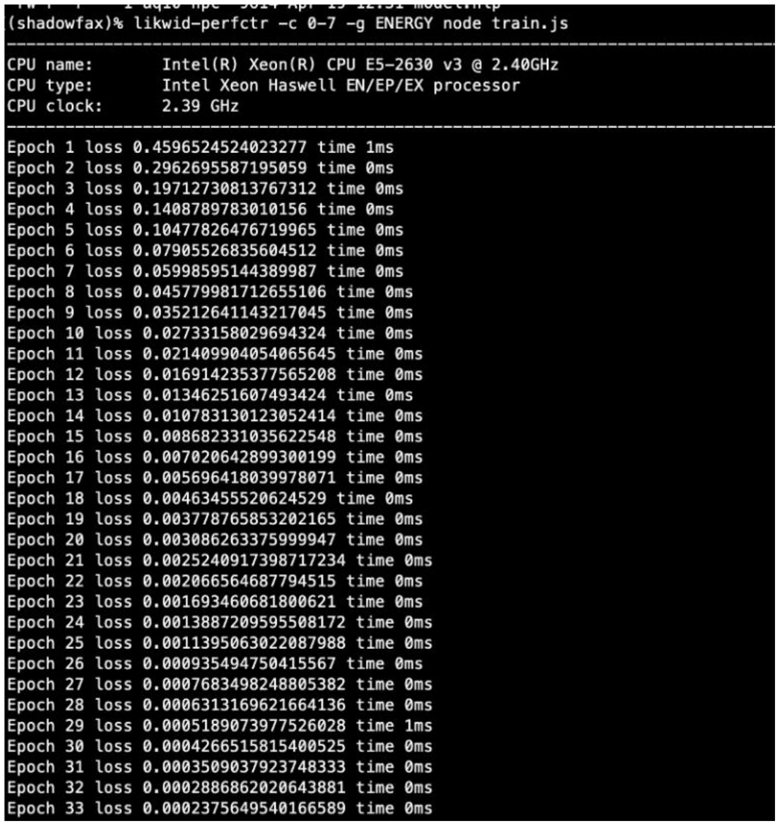
After creating and running the chatbot on my MacBook, I set up the chatbot on the Raspberry Pi. I went through the same environment setup I did on my MacBook; I installed NPM and Node.js as well as followed the chatbot tutorial again to initialize the node modules and JavaScript packages [11]. Since I installed these components on the Raspberry Pi, I could not copy over the entire folder of the application. I had to individually copy the intents folder, index.js, and train.js in order for the transfer to work properly. Next, I trained the chatbot by running “npm run train” in the command line of the terminal. This shows the Epochs running. After training the neural network, I used the command line “npm run start” to run the chatbot. When the chatbot is running “Chatbot started!” is displayed along with the line that prompts the user to input something, which is “>”.

3.3 Setting up a Compute Server for Baseline Comparison

Before measuring the energy consumption of the Raspberry Pi, I used a large-scale Linux Server on Texas State University’s campus, called Shadowfax. It is a High-Performance Computing server that contains 16 cores. This server consumes a large amount of energy. Running the chatbot on this server allows a baseline of results to compare the performance measurements of the Raspberry Pi in order to show efficiency and less energy consumption.

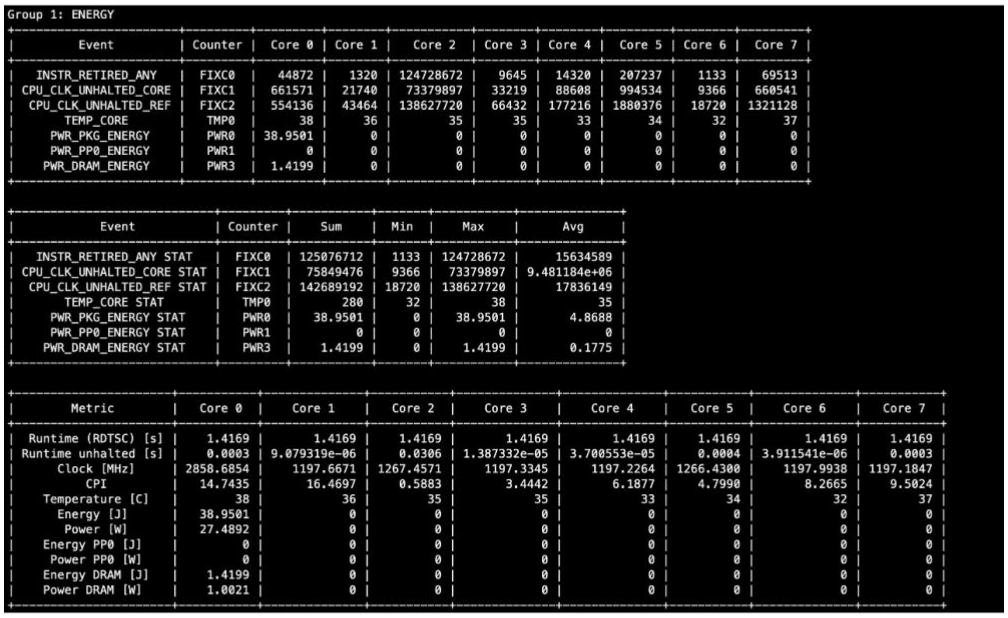
3.4 Energy Measurement

I used a performance tool called LIKWID; it Leverages the Linux RAPL* interface to dynamically measure processor power draw and energy usage. LIKWID also provides more precise readings than external devices hooked up to power outlets [9]. I first ran the chatbot on Shadowfax on 8 of the 16 cores. Using the command line “likwid-perfctr -c 0-7 -g ENERGY node train.js”, LIKWID runs the training program on the Shadowfax server and measures multiple performance information for each core [Figure 7], [16]. There is also a section that provides the minimum, maximum, sum, and average of these measurements from all of the cores [Figure 8]. The middle section is of most interest due to displaying the energy consumption of the chatbot on Shadowfax. LIKWID calculates the energy consumption by taking the measured power consumption and the runtime of the train.js file; then, multiplies the two numbers [Figure 9].



```
(shadowfax)% likwid-perfctr -c 0-7 -g ENERGY node train.js
--------------------------------------------------------------------------------
CPU name:       Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz
CPU type:       Intel Xeon Haswell EN/EP/EX processor
CPU clock:      2.39 GHz
--------------------------------------------------------------------------------
Epoch 1 loss 0.4596524524023277 time 1ms
Epoch 2 loss 0.2962695587195059 time 0ms
Epoch 3 loss 0.19712730813767312 time 0ms
Epoch 4 loss 0.1408789783010156 time 0ms
Epoch 5 loss 0.10477826476719965 time 0ms
Epoch 6 loss 0.07905526835604512 time 0ms
Epoch 7 loss 0.05998595144389987 time 0ms
Epoch 8 loss 0.045779981712655106 time 0ms
Epoch 9 loss 0.035212641143217045 time 0ms
Epoch 10 loss 0.02733158029694324 time 0ms
Epoch 11 loss 0.021409904054065645 time 0ms
Epoch 12 loss 0.016914235377565208 time 0ms
Epoch 13 loss 0.01346251607493424 time 0ms
Epoch 14 loss 0.010783130123052414 time 0ms
Epoch 15 loss 0.008682331035622548 time 0ms
Epoch 16 loss 0.007020642899300199 time 0ms
Epoch 17 loss 0.005696418039978071 time 0ms
Epoch 18 loss 0.00463455520624529 time 0ms
Epoch 19 loss 0.003778765853202165 time 0ms
Epoch 20 loss 0.003086263375999947 time 0ms
Epoch 21 loss 0.0025240917398717234 time 0ms
Epoch 22 loss 0.002066564687794515 time 0ms
Epoch 23 loss 0.001693460681800621 time 0ms
Epoch 24 loss 0.0013887209595508172 time 0ms
Epoch 25 loss 0.0011395063022087988 time 0ms
Epoch 26 loss 0.000935494750415567 time 0ms
Epoch 27 loss 0.0007683498248805382 time 0ms
Epoch 28 loss 0.0006313169621664136 time 0ms
Epoch 29 loss 0.0005189073977526028 time 1ms
Epoch 30 loss 0.0004266515815400525 time 0ms
Epoch 31 loss 0.0003509037923748333 time 0ms
Epoch 32 loss 0.0002886862020643881 time 0ms
Epoch 33 loss 0.0002375649540166589 time 0ms
```

Figure 7. Likwid command on Linux Servers

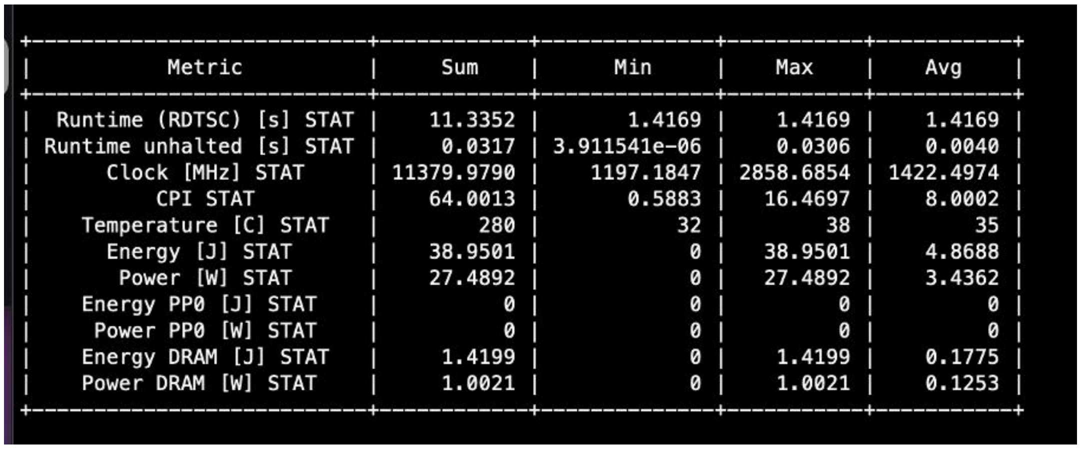


```
Group 1: ENERGY
+-----------------------+---------+---------+-------+-----------+-------+--------+---------+-------+---------+
|         Event         | Counter | Core 0  | Core 1|   Core 2  | Core 3| Core 4 |  Core 5 | Core 6|  Core 7 |
+-----------------------+---------+---------+-------+-----------+-------+--------+---------+-------+---------+
|   INSTR_RETIRED_ANY   |  FIXC0  |   44872 |  1320 | 124728672 |  9645 |  14320 |  207237 |  1133 |   69513 |
| CPU_CLK_UNHALTED_CORE |  FIXC1  |  661571 | 21740 |  73379897 | 33219 |  88608 |  994534 |  9366 |  660541 |
|  CPU_CLK_UNHALTED_REF |  FIXC2  |  554136 | 43464 | 138627720 | 66432 | 177216 | 1880376 | 18720 | 1321128 |
|       TEMP_CORE       |   TMP0  |      38 |    36 |        35 |    35 |     33 |      34 |    32 |      37 |
|     PWR_PKG_ENERGY    |   PWR0  | 38.9501 |     0 |         0 |     0 |      0 |       0 |     0 |       0 |
|     PWR_PP0_ENERGY    |   PWR1  |       0 |     0 |         0 |     0 |      0 |       0 |     0 |       0 |
|    PWR_DRAM_ENERGY    |   PWR3  |  1.4199 |     0 |         0 |     0 |      0 |       0 |     0 |       0 |
+-----------------------+---------+---------+-------+-----------+-------+--------+---------+-------+---------+

+----------------------------+---------+-----------+-------+-----------+--------------+
|            Event           | Counter |    Sum    |  Min  |    Max    |      Avg     |
+----------------------------+---------+-----------+-------+-----------+--------------+
|   INSTR_RETIRED_ANY STAT   |  FIXC0  | 125076712 |  1133 | 124728672 |     15634589 |
| CPU_CLK_UNHALTED_CORE STAT |  FIXC1  |  75849476 |  9366 |  73379897 | 9.481184e+06 |
|  CPU_CLK_UNHALTED_REF STAT |  FIXC2  | 142689192 | 18720 | 138627720 |     17836149 |
|       TEMP_CORE STAT       |   TMP0  |       280 |    32 |        38 |           35 |
|     PWR_PKG_ENERGY STAT    |   PWR0  |   38.9501 |     0 |   38.9501 |       4.8688 |
|     PWR_PP0_ENERGY STAT    |   PWR1  |         0 |     0 |         0 |            0 |
|    PWR_DRAM_ENERGY STAT    |   PWR3  |    1.4199 |     0 |    1.4199 |       0.1775 |
+----------------------------+---------+-----------+-------+-----------+--------------+

+----------------------+-----------+--------------+-----------+--------------+--------------+-----------+--------------+-----------+
|        Metric        |   Core 0  |    Core 1    |   Core 2  |    Core 3    |    Core 4    |   Core 5  |    Core 6    |   Core 7  |
+----------------------+-----------+--------------+-----------+--------------+--------------+-----------+--------------+-----------+
| Runtime (RDTSC) [s]  |    1.4169 |       1.4169 |    1.4169 |       1.4169 |       1.4169 |    1.4169 |       1.4169 |    1.4169 |
| Runtime unhalted [s] |    0.0003 | 9.079319e-06 |    0.0306 | 1.387332e-05 | 3.700553e-05 |    0.0004 | 3.911541e-06 |    0.0003 |
|      Clock [MHz]     | 2858.6854 |    1197.6671 | 1267.4571 |    1197.3345 |    1197.2264 | 1266.4300 |    1197.9938 | 1197.1847 |
|          CPI         |   14.7435 |      16.4697 |    0.5883 |       3.4442 |       6.1877 |    4.7990 |       8.2665 |    9.5024 |
|    Temperature [C]   |        38 |           36 |        35 |           35 |           33 |        34 |           32 |        37 |
|      Energy [J]      |   38.9501 |            0 |         0 |            0 |            0 |         0 |            0 |         0 |
|       Power [W]      |   27.4892 |            0 |         0 |            0 |            0 |         0 |            0 |         0 |
|    Energy PP0 [J]    |         0 |            0 |         0 |            0 |            0 |         0 |            0 |         0 |
|     Power PP0 [W]    |         0 |            0 |         0 |            0 |            0 |         0 |            0 |         0 |
|    Energy DRAM [J]   |    1.4199 |            0 |         0 |            0 |            0 |         0 |            0 |         0 |
|    Power DRAM [W]    |    1.0021 |            0 |         0 |            0 |            0 |         0 |            0 |         0 |
+----------------------+-----------+--------------+-----------+--------------+--------------+-----------+--------------+-----------+
```

Figure 8. Linux Server results from Likwid command



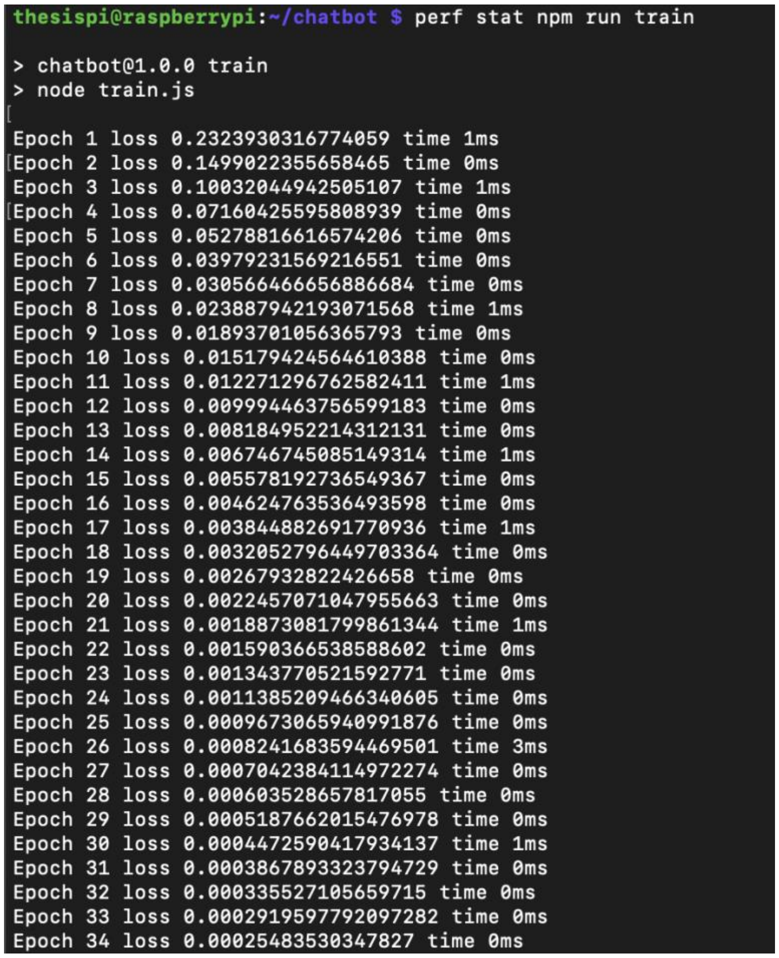
```
+---------------------------+------------+--------------+-----------+-----------+
|           Metric          |     Sum    |      Min     |    Max    |    Avg    |
+---------------------------+------------+--------------+-----------+-----------+
|  Runtime (RDTSC) [s] STAT |    11.3352 |       1.4169 |    1.4169 |    1.4169 |
| Runtime unhalted [s] STAT |     0.0317 | 3.911541e-06 |    0.0306 |    0.0040 |
|      Clock [MHz] STAT     | 11379.9790 |    1197.1847 | 2858.6854 | 1422.4974 |
|          CPI STAT         |    64.0013 |       0.5883 |   16.4697 |    8.0002 |
|    Temperature [C] STAT   |        280 |           32 |        38 |        35 |
|      Energy [J] STAT      |    38.9501 |            0 |   38.9501 |    4.8688 |
|       Power [W] STAT      |    27.4892 |            0 |   27.4892 |    3.4362 |
|    Energy PP0 [J] STAT    |          0 |            0 |         0 |         0 |
|     Power PP0 [W] STAT    |          0 |            0 |         0 |         0 |
|    Energy DRAM [J] STAT   |     1.4199 |            0 |    1.4199 |    0.1775 |
|    Power DRAM [W] STAT    |     1.0021 |            0 |    1.0021 |    0.1253 |
+---------------------------+------------+--------------+-----------+-----------+
```

Figure 9. Linux Server middle column of results

Next, to measure the energy consumption of the chatbot on the Raspberry Pi, I installed a Linux performance analyzing tool called perf. This performance tool is able to do lightweight profiling and displays information about CPU performance counters. I installed this onto the Raspberry Pi and confirmed that it was installed properly by running the command “perf stat ls”. I then ran perf stat on the train program with the command line “perf stat npm run train” [Figure 10], [14]. Figures 11 and 12 display the information based on these two command lines. Using this information, I calculated the power used by the chatbot with the following formula

$$P = Cv^2f$$

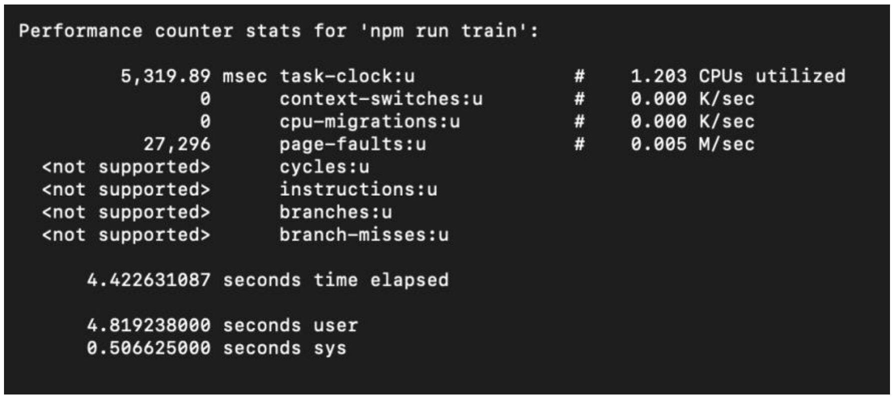
where C is the static power, v is the voltage, and f is the frequency. The frequency is calculated by the clock rate in GHz multiplied by the CPU utilization. After calculating the power, the energy consumed by the Pi was obtained by multiplying the power used and the runtime of the training task.



```
thesispi@raspberrypi:~/chatbot $ perf stat npm run train

> chatbot@1.0.0 train
> node train.js
[
Epoch 1 loss 0.2323930316774059 time 1ms
[Epoch 2 loss 0.1499022355658465 time 0ms
Epoch 3 loss 0.10032044942505107 time 1ms
[Epoch 4 loss 0.07160425595808939 time 0ms
Epoch 5 loss 0.05278816616574206 time 0ms
Epoch 6 loss 0.03979231569216551 time 0ms
Epoch 7 loss 0.030566466656886684 time 0ms
Epoch 8 loss 0.023887942193071568 time 1ms
Epoch 9 loss 0.01893701056365793 time 0ms
Epoch 10 loss 0.015179424564610388 time 0ms
Epoch 11 loss 0.01227129676258241 time 1ms
Epoch 12 loss 0.009994463756599183 time 0ms
Epoch 13 loss 0.008184952214312131 time 0ms
Epoch 14 loss 0.006746745085149314 time 1ms
Epoch 15 loss 0.005578192736549367 time 0ms
Epoch 16 loss 0.004624763536493598 time 0ms
Epoch 17 loss 0.003844882691770936 time 1ms
Epoch 18 loss 0.0032052796449703364 time 0ms
Epoch 19 loss 0.00267932822426658 time 0ms
Epoch 20 loss 0.0022457071047955663 time 0ms
Epoch 21 loss 0.0018873081799861344 time 1ms
Epoch 22 loss 0.001590366538588602 time 0ms
Epoch 23 loss 0.001343770521592771 time 0ms
Epoch 24 loss 0.0011385209466340605 time 0ms
Epoch 25 loss 0.0009673065940991876 time 0ms
Epoch 26 loss 0.0008241683594469501 time 3ms
Epoch 27 loss 0.0007042384114972274 time 0ms
Epoch 28 loss 0.000603528657817055 time 0ms
Epoch 29 loss 0.0005187662015476978 time 0ms
Epoch 30 loss 0.0004472590417934137 time 1ms
Epoch 31 loss 0.0003867893323794729 time 0ms
Epoch 32 loss 0.000335527105659715 time 0ms
Epoch 33 loss 0.0002919597792097282 time 0ms
Epoch 34 loss 0.00025483530347827 time 0ms
```

Figure 10. Perf stat command on Raspberry Pi



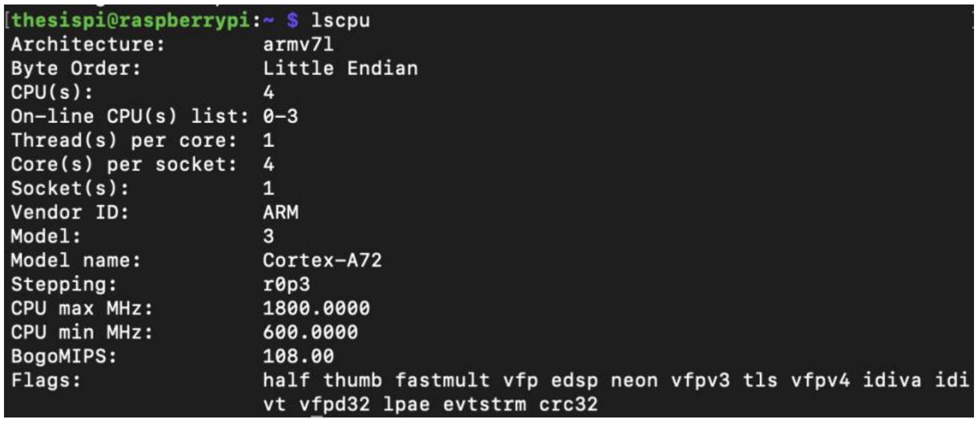
```
Performance counter stats for 'npm run train':

        5,319.89 msec task-clock:u              #    1.203 CPUs utilized
               0      context-switches:u        #    0.000 K/sec
               0      cpu-migrations:u          #    0.000 K/sec
          27,296      page-faults:u             #    0.005 M/sec
  <not supported>      cycles:u
  <not supported>      instructions:u
  <not supported>      branches:u
  <not supported>      branch-misses:u

     4.422631087 seconds time elapsed

     4.819238000 seconds user
     0.506625000 seconds sys
```

Figure 11. Perf stat results on Raspberry Pi



```
thesispi@raspberrypi:~ $ lscpu
Architecture:        armv7l
Byte Order:          Little Endian
CPU(s):              4
On-line CPU(s) list: 0-3
Thread(s) per core:  1
Core(s) per socket:  4
Socket(s):           1
Vendor ID:           ARM
Model:               3
Model name:          Cortex-A72
Stepping:            r0p3
CPU max MHz:         1800.0000
CPU min MHz:         600.0000
BogoMIPS:            108.00
Flags:               half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idi
                     vt vfpd32 lpae evtstrm crc32
```

Figure 12. lscpu output of Raspberry Pi

4. EXPERIMENTS

4.1 Results

I ran the chatbot on Shadowfax as a baseline and then ran the chatbot 10 times on the Raspberry Pi in order to show the difference between energy consumption on both devices. Table 1 shows the data taken to calculate energy as well as the average of the multiple runs on the Raspberry Pi.

Run Number	Runtime (Seconds)	Power (Watts)	Energy (Joules)
1	4.422631087	23.93909181	105.8376223

2	5.609149589	17.10772	95.95977183
3	4.524626147	23.4137079	105.938275
4	4.468746455	23.6922057	105.8744602
5	4.457424235	22.8911327	102.0354897
6	4.475442935	22.6126349	101.2015571
7	4.420373454	23.9707035	105.9594614
8	4.442050708	22.7916692	101.2417503
9	4.455468775	23.0502743	102.6997774
10	4.405979422	23.9707035	105.6144264
		Average Energy	103.2362592

Table 1. Raspberry Pi power and energy consumption

From my measurement of energy consumption, the chatbot on Shadowfax consumes 35.9501 joules of energy and consumes an average of 103.2362 joules of energy on the Raspberry Pi.

4.2 Challenges and Errors

After configuring the settings and attempting to connect to the Raspberry Pi for the first time, I had an issue with the local host and was not able to locate the Raspberry Pi remotely. I tried using the command:

```
ping raspberrypi.local
```

but I ran into the error as follows:

```
kex_exchange_identification: read: Connection reset by peer
ssh error
```

After troubleshooting this issue, I disconnected the Pi and set up its environment again [15]. Sometimes the ping command line would not work, but nmap was a more reliable indication that the Pi was running. I took note of my Raspberry Pi’s IP address and used this each time in order to ssh to it [7], [8].

The second challenge I ran into was when I was first creating the chatbot. The first tutorial I referenced used TypeScript, which I was not familiar with. I ran into some compile errors when trying to run my code because of my lack of knowledge in the programming language [18]. I chose to look into a simpler node.js application instead to understand the basics of the program. Then, I found documentation of a chatbot application using node.js [11], [13]. I referenced this documentation and had success with setting up my chatbot.

Next, I had compilation errors in the chatbot application after I transferred the files from my MacBook to the Raspberry Pi. I removed all of the JavaScript and node module files. I found that after removing these, I needed to set up the environment on the Pi and instead of copying the environment files. Therefore, I followed the documentation again to download the JavaScript packages and node modules [11]. From here, I compiled the chatbot on the Pi and was able to run it successfully.

The last set of errors I ran into were when trying to install the Likwid performance tool on my Raspberry Pi. Since my Raspberry Pi is remote, I wasn’t able to have a quick installation of Likwid. I followed the GitHub Repository containing all of the files, but it was unable to build properly. I attempted to clone the git repository; however, the clone would not download and fail [9], [16]. The next step was to find a way to measure the performance on the Raspberry Pi without using LIKWID, which we solved by installing the perf tool.

5. CONCLUSION

Based on the results, the chatbot on Shadowfax consumed less energy than the Raspberry Pi. This was caused by the throughput being lower on the Pi, meaning the tasks per second were done at a lower rate than the servers. The cloud servers receive many requests simultaneously from different clients. Therefore, at any point in time, they are running many AI applications. The collection of AI applications running on a server within a given time window is called a workload. To measure the performance of cloud workloads, we don't consider the individual running times of an individual application but rather the number of tasks completed in a given time frame. To compute the energy efficiency we would look at the average power consumption over a given unit of time, which is typically one hour. The energy efficiency would be measured as avg power x 60 x 60 joules.

Through my project, it can be seen that the Edge computing paradigm allows data computation to be done a lot closer to devices and is a significant approach to energy efficiency in computing. Using devices such as Raspberry Pis to run applications like AI with large training datasets are in the right direction and could be explored more by using a cluster with more Pis. Along with my project, it is possible to work towards a sustainable future in the tech industry.

WORKS CITED

[1] A. Burgess and T. Brown, “By 2040 there may not be enough power for all our computers,” *The Manufacturer*, 17-Aug-2016. Available: https://www.themanufacturer.com/articles/by-2040-there-may-not-be-enough-power-for-all-our-computers/.

[2] A. Chalimov, “The impact of edge computing on IOT: The main benefits and real-life use cases: Eastern peak,” *Eastern Peak - Technology Consulting & Development Company*, 19-Aug-2020. Available: https://easternpeak.com/blog/the-impact-of-edge-computing-on-iot-the-main-benefits-and-real-life-use-cases/.

[3] A. Freed, “Introduction to conversational AI,” in *Conversational AI*, Shelter Island: Manning Publications, 2021.

[4] C. Adam, “Sustainable AI can be done today,” *Medium*, 07-Jun-2021. Available: https://blog.ml6.eu/sustainable-ai-can-be-done-today-35a85aa0acb0.

[5] C. Piera Garrigosa, “Powering a data center with renewable energy: Dream or reality?,” *Powering a data center with renewable energy: dream or reality? | Blogs La Salle | Campus Barcelona*, 10-Mar-2021. Available: https://blogs.salleurl.edu/en/powering-data-center-renewable-energy-dream-or-reality.

[6] “Conversational AI: What it is and how it works,” *Ada*, 2021. Available: https://www.ada.cx/conversational-ai.

[7] Emmet, “Finding the IP Address of your Raspberry Pi,” *Pi My Life Up*, 30-Jan-2022. Available: https://pimylifeup.com/raspberry-pi-ip-address/.

[8] “How to SSH into the Raspberry Pi,” *The Pi*, 18-May-2017. Available: https://thepi.io/how-to-ssh-into-the-raspberry-pi/.

[9] “Likwid performance tools,” *Erlangen National High Performance Computing Center.* Available: https://hpc.fau.de/research/tools/likwid/.

[10] M. Caprolu, R. Di Pietro, F. Lombardi, and S. Raponi, “Edge computing perspectives: Architectures, technologies, and open security issues,” *2019 IEEE International Conference on Edge Computing (EDGE)*, 2019.

[11] N. Baranwal, “How to Create Chatbot with Node.js,” *Medium*, 16-May-2021. Available: https://medium.com/geekculture/create-chatbot-with-nodejs-cf3d8bc3f302.

[12] Needham, “IDC forecasts improved growth for Global AI market in 2021,” *IDC*, 23-Feb-2021. Available: https://www.idc.com/getdoc.jsp?containerId=prUS47482321.

[13] "Node.js - First Application," *Node.js - first application*. Available: https://www.tutorialspoint.com/nodejs/nodejs_first_application.htm.

[14] P. J. Drongowski, "Performance events on Raspberry Pi 4: Tips," *Sand, software and*
sound, 23-Nov-2020. Available: http://sandsoftwaresound.net/performance-events-on-raspberry-pi-4-tips/.

[15] "Raspberry Pi Documentation," *Getting Started*, 2022. Available: https://www.raspberrypi.com/documentation/computers/getting-started.html#installing-the-operating-system.

[16] "Rrze-HPC/likwid: Performance monitoring and benchmarking suite," *GitHub*. Available: https://github.com/RRZE-HPC/likwid/.

[17] Stoyanka Mollova, Radoslav Simionov, and Kamen Seymenliyski. 2018. "A study of the energy efficiency of a computer cluster." *Association for Computing Machinery*. Available: https://doi-org.libproxy.txstate.edu/10.1145/3278161.3278170

[18] S. Ronce, "Create a universal chatbot in Javascript, for beginners," *Medium,* 17-Sep-2021. Available: https://codeburst.io/create-a-universal-chatbot-in-javascript-for-beginners-cfd4e580680.

[19] U. Gupta *et al.*, "Chasing Carbon: The Elusive Environmental Footprint of Computing," *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021, pp. 854-867. Available at: https://ieeexplore.ieee.org/abstract/document/9407142.

[20] W. Gordon, "Beginner's Guide: How to get started with Raspberry Pi," *PCMAG*, 2019. Available: https://www.pcmag.com/how-to/beginners-guide-how-to-get-started-with-Raspberry-pi.