BrainAI: General AI Hardware Platform

Katheryn Berndt, Michael Montoya, Daniel Price, Etzkiel Martinez

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* – This project goal is to craft a localized learning platform designed to use the power of Large Language Models (LLMs) such as LlaMa. With a primary focus on laying the groundwork for future applications, the platform aims to empower users with self-learning capabilities driven by advanced LLM technology. In addition, it will seamlessly integrate AI functionalities, including computer vision, broadening its scope for various AI applications. Through planning and development, this project strives to create a versatile and adaptable platform poised to catalyze innovations in AI technology, offering transformative solutions across diverse industries and domains.**

Index Terms –Recent advancements in Large Language Models (LLMs), Exploration of small-scale LLMs on single high-end GPU, Project roles: hardware, benchmarking, embedded systems, LLM implementation, democratizing LLM accessibility, Self-learning Robotic Platforms

I. INTRODUCTION

With recent advancements in Large Language Models (LLMs), a growing interest in small-scale large language models has begun to bloom. This project specifically targets the utilization of high-end GPUs, focusing on the powerful RTX 3090. Divided into four key roles—hardware, benchmarking, embedded systems, LLM implementation, and application of AI—the project aims to democratize access to LLM technology, making it more accessible and versatile for researchers or for everyday convenience, designing an AI platform utilizing a robotic arm as the application of the localized models.

An important aspect of the project's vision involves the integration of ESP32, a highly adaptable embedded system, as the central component of a Self-Learning Robotic Platform. This will act as the driver for the robotic platform. As it will communicate with the LLM as that will feed it inputs.

This initiative represents a significant step towards bridging the gap between cutting-edge AI research and practical implementation. By leveraging the computational power of high-end GPUs and the flexibility of ESP32, the project endeavors to pave the way for widespread adoption of LLM technology in various industries and sectors.

In summary, this project goal is to craft a localized learning platform designed to use the power of Large Language Models (LLMs) such as LlaMa. With a primary focus on laying the groundwork for future applications. It represents a collaborative effort to unlock the full potential of LLMs and propel the field of artificial intelligence towards new horizons of discovery and advancement.

II. HARDWARE SYSTEM COMPONENTS

The system details are best introduced by highlighting the hardware components that make it up. Each component was either provided by the sponsor or purchased with funds provided by them. The purpose of this section is to provide an introduction of the physical components utilized in the design.

The system can first be introduced by describing the overall design before showcasing the details of each of the components.

A. *Robotic Platform*

The robotic arm is the main component that physically demonstrates the results of the trained model. It was provided by the sponsor of the project and came prebuilt, eliminating the need to allocate time for design or manufacturing. This made it an ideal addition to the project. The robotic arm consists of four servo motors, each from the brand Hello Maker. They are specifically the HM-MS07. With the four servos, the arm has three rotational axes that are leveraged by the AI models to control the arm. The movement for each arm will be abstracted by the microcontroller. The arm will be powered by a three-amp adjustable power supply, which supplies a variable voltage of 0V – 24V to each of the servos. Each servo has an amp range of 500mA – 900mA and runs at 6V. With these constraints, the power supply will have more than ample power and current to power the robotic arm.

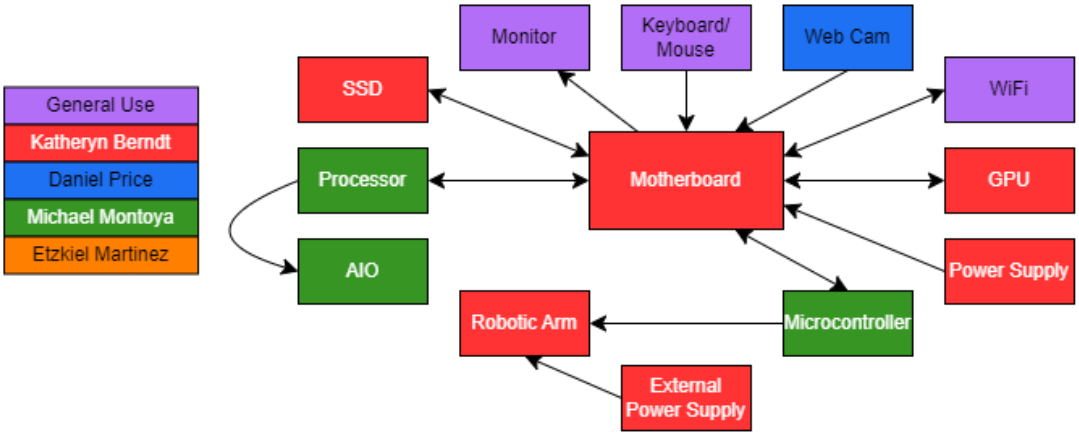

Fig. 1. Block Diagram presenting the major hardware components that make up the system.

B. Microcontroller

As stated in part A, Robotic Arm, the microcontroller is primarily responsible for the movement of the robotic arm. The microcontroller takes in the output provided by the PC and Webcam and converts it into the commands that are used to control the robotic arm movement. The selected microcontroller for this project is the ESP32. It was chosen for its variety of capabilities, such as its very low price and compatibility with the Arduino framework. This component was purchased by the team using the budget provided by the sponsor.

C. GPU

GPU utilization has extended to machine learning and artificial intelligence (AI). Our project necessitates a high-end GPU to support the operation of large language models (LLMs), with a focus on the key considerations in GPU selection include CUDA cores, specialized processors for parallel computing tasks like video rendering and deep learning. These cores, capable of simultaneous task execution, enhance computational speed and memory bandwidth, crucial for time-intensive workloads. Tensor cores, dedicated to deep learning and AI, expedite neural network training by computing multiple operations in a single clock cycle. GPUs with ample tensor cores are preferred for handling heavy machine learning tasks.

Floating Point Operations Per Second (FLOPS) metric gauges a computer's performance based on its floating-point arithmetic capabilities. GPUs represent their FLOPS values in TFLOPS or GFLOPS, with higher values suggesting faster speeds. However, FLOPS alone may not dictate performance, as other factors like tensor cores and CUDA cores play significant roles.

Video Random Access Memory (VRAM), exclusive to GPUs, serves as temporary storage for graphics rendering. Optimal VRAM capacity is crucial for tasks associated with training large language models, as it determines the GPU's ability to handle data and complex computations efficiently. While higher VRAM entails increased costs, it is essential for optimal model performance.

In conclusion, GPU selection for our project is driven by considerations such as CUDA and tensor core counts, FLOPS values, and VRAM capacity. These factors collectively influence the GPU's ability to support the demands of large language model training, ensuring optimal performance and efficiency.

D. CPU

The CPU's role in the PC of large language models (LLMs) is multifaceted and pivotal, serving as the backbone for supporting the GPU while undertaking various essential tasks such as data loading and preprocessing. A deeper understanding of the CPU's significance reveals a complex interplay of factors that contribute to the optimal functioning of LLMs.

One of the key aspects to consider is the importance of high single-threaded CPU performance, particularly during the initial model loading phase. This phase is critical as it sets the foundation for subsequent operations, determining how quickly and efficiently the model can be accessed and utilized. A faster single-threaded CPU

ensures swift model loading, thereby reducing latency and improving overall system responsiveness.

As the model transitions from the loading phase to actual language generation and processing tasks, the GPU emerges as the primary workhorse due to its unparalleled parallel processing capabilities. While the CPU's role becomes less prominent during this phase, it still plays a crucial role in managing data preprocessing, orchestrating the overall workflow, and handling operations that do not heavily rely on the GPU's processing power.

The importance of a multi-core CPU is very essential for the context of LLMs. With the computational demands of these models constantly increasing, a multi-core CPU becomes indispensable for distributing the workload effectively. Tasks such as data preprocessing, input/output operations, and system resource management benefit greatly from the parallel processing capabilities of a multi-core CPU. By offloading these tasks from the GPU, the CPU ensures that the GPU can focus on the more compute-intensive aspects of the language model, thereby enhancing overall efficiency and performance.

The CPU plays a critical role in maintaining system responsiveness, particularly in scenarios where the GPU is under heavy load. Tasks such as managing user interactions, handling background processes, and ensuring error handling and system stability fall within the purview of the CPU. A multi-core CPU ensures that the system remains responsive even under demanding conditions, thereby ensuring a seamless user experience.

Another crucial aspect to consider is the importance of PCIe support in facilitating efficient data transfer between the CPU and GPU. A CPU with a high number of PCIe lanes is essential for minimizing bottlenecks and ensuring that the system can handle the increasing demands of machine learning tasks. This high-speed data transfer capability is vital for optimizing overall system performance and throughput.

Additionally, the integration of graphics capabilities within the CPU offers several advantages in the context of LLMs. When used in conjunction with a dedicated GPU, integrated graphics provide a cohesive and versatile approach, expanding the number of display outputs available and offloading display tasks from the GPU. This not only simplifies system setup but also ensures smoother system operation and resource optimization.

In conclusion, the CPU's role in the realm of large language models is diverse and indispensable, encompassing aspects such as single-threaded performance, multi-core capabilities, PCIe support, and integrated graphics. By effectively managing these.

E. PC

The main purpose of the PC is to run the overall system and process the data provided by the many aspects of the design, including both hardware and software. It consists of the GPU and CPU, previously mentioned in parts C and D respectively. In addition, in includes a motherboard, power supply, RAM and SSD.

The motherboard is the ASUS – Prime B650E-F model. This was selected for its support of PCIe Gen 5 as well as its inclusion of two PCIe slots. This ability two GPUs is ideal in the event of exploring the systems support of two GPUs in the future. It also includes DDR5 RAM, the newest generation provided on the market.

The PC also includes four sticks of 8GB of DDR4 RAM. This is compatible with the motherboard, which supports up to DDR5 RAM, as previously stated. It also includes an SSD that includes up to 2TB of storage. This is more than enough to store all the data associated with the large language model, the datasets used for benchmarking, and any other miscellaneous programs used for the project. The entire system will be powered by a 1000 watt 80 plus power gold power supply.

As mentioned previously, it also includes the AMD Ryzen 9 CPU as well as the NVIDA RTX 3090 GPU. The technical details of these components will be provided in section III, Hardware Detail. Finally, the entire PC design was provided and paid for by the sponsor.

F. Webcam

The purpose of the webcam is to handle the inputs which are given to the AI workload. This includes both voice and video inputs. Voice inputs will be used by both the large language model (LlaMa), and the music analysis model. When being configured for the LLM, the speech will be turned into text. The video will be used for the computer vision model. The chose model for the project is the EMEET 1080P Webcam. This model was selected for its inclusion of both the webcam as well as a microphone. It also is capable of 90 degrees FOV, which is perfect for taking in the input of human movement in front of the camera.

III. HARDWARE DETAIL

Below is a more in depth look at the technical details of the System Components that were outlined previously in section II. This excludes the PC (SSD, Motherboard, Power Supply) and Webcam to avoid providing any redundant information.

A. Robotic Arm

As previously stated, the robotic arm is being utilized to physically demonstrate the results of training the large language model. The robotic arm dimensions are displays in figure 2. The robotic arm standing height depends on the current positions of the servos. The arm length is 270 mm and its base is 85 mm.

The control of the servo motors is provided by commands given by the ESP32 Microcontroller. This connection is made using the three pin connectors on each of the servo motors. One pin is connected to the voltage, one to ground, and the final pin is connected to one of the fifteen control pins provided on the microcontroller. To simplify the connection, a PCB was designed to feed all the connections to a single location. The PCB has fifteen 3 pin male headers, where the servo motors can be connected to. Each of these traces back to two fifteen pin female headers, which are connected to the microcontroller. Power is fed to the servos through a micro-USB located on the PCB. The microcontroller is powered by being directly connected to the PC, since it only required an input voltage of 5 volts.

Each of the robotic arm servos each require four to seven volts to properly run, with an ideal voltage of six volts. This is supplied through a three-amp adjustable power supply, which can provide any voltage within the range of 3 and 24 volts. It is controlled by a dial, which is adjusted to increase the voltage, which is displayed on the front of the supply. The servo motors will utilize a running current of approximately 500 milliamps when running at six volts. To properly account for the potential of the servos hitting peak current, the needed amperage for the servos was multiplied by 1.5x. This brings the required voltage for each of the servos to approximately 750 milliamps. The adjustable power supply provides just enough amperage to control all four servo motors simultaneously.

The signal, managed through Pulse Width Modulation (PWM), precisely guides the servo's angular position

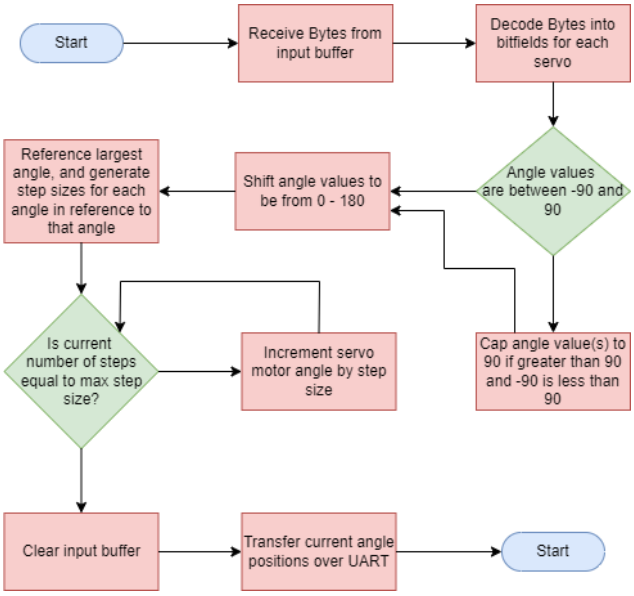

Fig. 2 Microcontroller software flowchart.

within a 0 to 180-degree range. In this project, the ESP32

microcontroller takes on the role of the driver, customizing PWM signals for each servo. This collaboration ensures the arm's movements are executed with precision and agility.

The ESP32's PWM capabilities allow for seamless control over the servos, facilitating their smooth operation in executing complex tasks. By adjusting the width of electrical pulses, the ESP32 dictates the desired angles of rotation for each servo. Demonstrating the transformative potential of modern robotics in real-world applications.

B. Microcontroller

To go into more detail with the ESP32, and the programming of the microcontroller to act as the driver for the arm, the microcontroller and the pc communicate through UART. The esp32 has a UART to USB converter that will help with this.

Underneath the hood, when the ESP32 receives information from the computer, in the form of a bitmap containing 3 bytes of data, is decodes this data to control the robotic arm. Each byte of data refers to a rotational axis that the arm can utilize. Since the robotic arm uses two of the four servos for one rotational axis, we can ultimately treat this as a single servo, and just invert the angle sent from one to the other.

When decoding the three bytes, the ESP32 first splits the incoming data into three separate stored bytes. The individual byte is denoted to which servo(s) it controls. Within each byte, it can be decoded into two values, the direction, and the angle. The MSB (Most Significant Bit) holds the direction bit, and the remaining 7 bits hold the angle value. This angle value is from 0 – 90, and when in tandem with the direction bit, can represent a value from 0 – 180, which is the range of a servo motor. The program encodes as range of -90 – 90 to make it easier for the program to understand left in comparison to right.

Once the bytes have been read, the microcontroller changes the angle value from that of a range -90 – 90 to 0 – 180. During this process, it checks to make sure that no incoming data is outside of the input range. If the data happens to be outside this range, it caps the value at one of the edges of the range. the servo clears the incoming buffer. This is done to make sure the microcontroller is only reading the most recent bitmap, and not backing up with data, causing it to have input lag and be behind any input commands.

During movement, to make the arm move in a more cohesive and fluid way, the value for each angle is written to the servo in increments. This allows the servos to take turns moving, replicating parallel movement. This also ensures that the arm does not follow any rigid movement, moving a single servo at a time. It does this by finding the value that generates the largest movement, and adjusting which servos are incremented based on this largest range. Larger ranges get moved every increment, while smaller ranges get moved only in increments that follow the ratio of the larger ranges.

Once the movements have been completed, the microcontroller replies with its current position. This helps with debugging, validation, and confirmation that a new value can be sent to the microcontroller. The returned values have shown in testing to have a varying degree of error, within a range of -3 to 3 degrees.

C. GPU

The NVIDIA RTX 3090 emerges as the final choice for the e GPU in this build. Priced at an average of approximately $900 for a resale or used model, it represents a cost-effective solution compared to its higher-end counterpart, the RTX 4090, which typically commands nearly double the price.

While the RTX 3090 may have some limitations compared to higher-tier GPUs, such as its lower count of CUDA and tensor cores in comparison to the RTX 4090, it remains a compelling option for this project. With 328 tensor cores, the RTX 3090 may not match the computational capabilities of higher-end models, but it provides a balance between performance and affordability that aligns with our project requirements.

The RTX 3090 offers comparable performance levels with slightly lower power requirements, making it an ideal choice for our build. With its competitive performance and cost-effective pricing, the RTX 3090 fulfills the criteria for our project's GPU selection, promising to deliver the necessary performance without exceeding our budget constraints. While the RTX 3090 may not match the sheer power and capabilities of its higher-tier counterparts, it still presents a viable and cost-effective option GPU in this build. Its competitive performance levels, coupled with its relatively lower power requirements, make it a pragmatic choice for achieving the desired balance between performance and affordability.

D. CPU

The AMD Ryzen 9 3900XT 12-Core Processor stands out as an excellent choice for the localized learning platform. With its high core count and base clock speed of 3.8GHz, the processor comes out as a great CPU that will not bottleneck the system. Thought the CPU will be underutilized when using deep learning frameworks, all tasks use the CPU to some degree, and using a powerful CPU will allow other parts to run as intended with full performance. It can also be noted that some AI models run without GPU acceleration, and any python code created will start on the CPU before being transferred to the GPU.

The Ryzen 9 3900XT features high single-threaded performance, crucial for tasks such as initial model loading and data preprocessing in a localized learning environment. Its multi-core architecture enables efficient parallel processing, facilitating the execution of self-learning algorithms and other AI-driven functionalities seamlessly.

Additionally, the Ryzen 9 3900XT boasts enhanced memory bandwidth, essential for swiftly accessing and manipulating large datasets inherent in LLM applications. This capability ensures smooth operation and faster model inference, contributing to an optimized user experience.

AMD's Ryzen processors are known for their competitive pricing, offering exceptional value for performance. This affordability factor aligns well with the goal of creating a scalable and accessible learning platform for future applications.

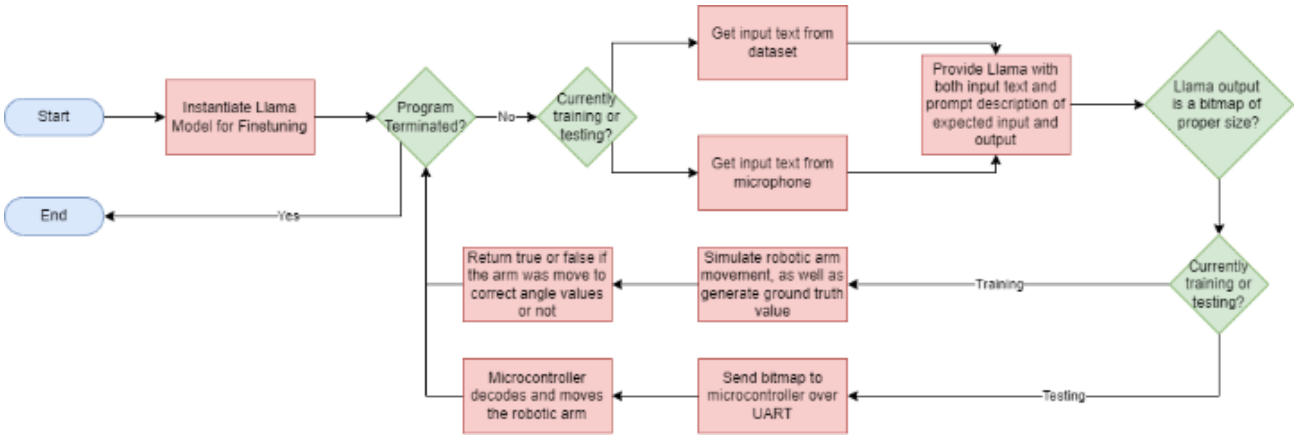

Fig. 3 LlaMa Training and Testing Software Flowchart.

In summary, the AMD Ryzen 9 3900XT 12-Core Processor is an excellent choice for powering the localized learning platform, providing the necessary computational capabilities and efficiency to drive advancements in self-learning and LLM technology.

IV. SOFTWARE DETAIL

A. LLaMa

Artificial intelligence (AI) and machine learning (ML) workloads are reaching unprecedented levels of computational complexity. A primary motivation of this project is to explore consumer-grade computer hardware and see how well it stacks up to the challenges of running AI/ML workloads. The large language model (LLM) that was chosen for evaluating the chosen hardware configuration must fulfill a list of criteria such that the platform can be accurately evaluated. The LLaMa-2 model, an LLM model recently released by Meta, was a great option for the LLM workload as it was open-source, pretrained, variable in size, and well-performing according to widely accepted metrics.

An open-source model was a necessity, as closed-source models restrict access to its architecture, making them impossible to run locally. Having the option of running the model on personal hardware is a strict prerequisite for our project, leaving us with no option except open-source models.

A pretrained model that already performs well was a necessary precondition for the chosen model. Choosing a model that performs well does a better job of reflecting current trends in the AI/ML industry and ensures relevancy in our project structure. The company *Hugging face* provides an excellent community for AI and ML enthusiasts. They have developed several metrics of evaluation for open-source LLMs that are widely accepted as reliable among the AI and ML community. LLaMa consistently performs very well on these metrics, demonstrating its reliability and usability.

With our current configuration, we are attempting to run inference and train LLaMa on single RTX 3090, which contains 24 GB of VRAM. The LLaMa-2 model comes in 3 main sizes: 7 billion parameters, 13 billion parameters, and 70 billion parameters. For running inference on the 7B and 13B parameter versions, 24GB of VRAM are enough to load the model into GPU memory and forward-propagate the model. For training, a value of 7B parameters at full-precision (4 bytes per parameter) results in 28 GB, which is larger than the 24 GB of VRAM provided by the RTX 3090. Parameter efficient fine-tuning (PEFT) techniques are currently being explored by our group to optimize memory utilization and ease the training workload on the RTX 3090.

The process of training an already existing and capable LLM for specific and unique tasks is known as fine-tuning. We aim to fine-tune LLaMa to learn how to control the robotic arm based on a series of user-defined commands to act as a sort of brain for the arm. A reinforcement learning process with supervised fine-tuning has been developed to train the model to learn how to control the arm through a series of interacting with the arm and discovering what works.

For training, LLaMa will be prompt with a task that will contain the generating of the hex code command to control the arm. The model will then attempt to

generate the hex code command. A simulation will be designed to act the robotic arm and will return a pass or fail criteria depending on if the LLM was able to generate a proper command. If the arm generated a correct command, it would be rewarded and trained on a dataset that highlights its successful attempt. If the model fails to generate the correct hex code, then the model will be notified through training that its generated hex code command was incorrect and will be retrained on correct commands.

For evaluating the trained model, we will develop a methodology to prompt LLaMa via voice input (that will be controlled by a separate AI model). The prompt will ask LLaMa to generate the command to control the arm. LLaMa's response will be parsed to extract the command and the command will be sent to the arm. This way, in real time, the user will be able to interact with the model directly and personally. At the same time, the model can be accurately extracted, and such a methodology to incorporate human feedback in the future can be employed.

B. Benchmarking

To adequately test and evaluate the performance of our system on AI workloads, benchmarking performance models have been designed to effectively quantify metrics of GPU during AI inference workloads. The models chosen cover multiple applications to get both an in-depth and broad overview of the performance of our system.

The first AI model selected was Convolutional Neural Networks, as these networks are widely used in AI for a multitude of tasks but are most known for their power in image classification. This stems from the model's convolutional filters and feature sharing attributes, allowing the model to distinctly find specific features within a small window of an image, and share these features with other filters to search through the rest of the image. With this, we aim to utilize a convolutional neural network for image classification tasks, using the widely used ImageNet dataset as the input to the inference test.

With the type of model and dataset outlined, a specific model must be selected to realize the benchmarking test. After comparing numerous models, ResNet stood out for its ease of use and ample community support, being both supported by TensorFlow and PyTorch. Additionally, ResNet is a popular model that is commonly used for image classification tasks and compared to, making the metrics recorded during testing adequately comparable to other benchmarks and tests.

For our second benchmark, a Large Language Model stood out as the next test to test our system and GPU. Not only are LLM's becoming more popular each year, but one the projects stretch goals it be able to train and use LlaMa to control the robotic arm detailed in sections two and three. With this in mind, it would be imperative to test the performance of LLM's on our system for future comparisons of GPUs as the project progresses in future years. For the dataset, we will be using WikiText, as it is widely used within the AI community and will make comparisons of benchmarking more accurate.

Two language models will be used for testing, OPT and T5. Both models were chosen for their ease of use within the Hugging Face Transformers Python library. OPT specifically is used on a wide range of hardware benchmarking utilizing NLP tasks since it is Open Source and has a large range of sizes. T5 is a text-to-text transfer Transformer, which converts all input NLP tasks into a text-to-text format, allowing the model to function on a wide range of tasks.

With the models selected, specific metrics must be outlined to gain relevant information from the benchmarks to effectively test the performance of each model on the GPU. Metrics such as throughput and latency are widely used, as they directly test the model's ability to compute new inputs and the speed at which they do it. Additionally, this project will include metrics such as accuracy and perplexity when running the models. Though these metrics do not directly correlate to the performance of the GPU, there could be a use case in the future of the project, as well as being very common metrics to output when working with AI models.

These models will be built in python and will be scripted to output to a text file detailing the performance of all models. All three of the models, ResNet, OPT, and T5, will adequately benchmark the performance of the system and GPU, and can be referred to benchmark additional GPUs as the project iterates in the future.

V. APPLICATION OF AI

For demonstrative purposes of this project, a platform will be designed to control the robotic arm, using the previously described microcontroller as the driver to it. With this framework, the project can be expanded upon in

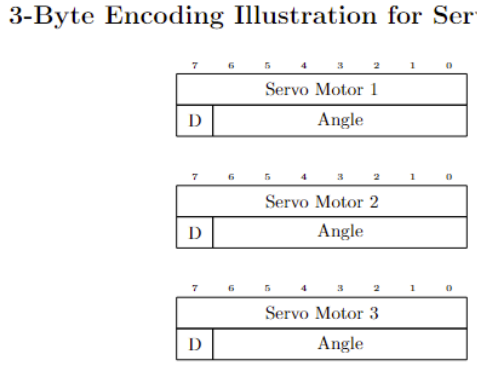

Fig. 4. Bit Encoding for transmission of angle output of AI model to microcontroller.

the future to both more easily communicates with the robotic arm, and to expedite design of the AI models used to control it. To show the use the framework on the arm, as well as demonstrating how the arm could be controlled, this project will leverage multiple AI models to control the robotic arm.

The framework is intended to receive a bitmap from the output of the AI model, where each byte refers to a specific servo. The first bit of the byte is used for direction of the servo, and the other 7 bits contain the angle to move the servo. This gives any movement a range from -90 to 90 degrees. The microcontroller will shift this range to a range from 0 to 180 degrees, which can then directly control the servo motors.

A. Landmark Detection

The first demonstration of an AI model controlling the robotic arm will be a use of landmark detection to send input angles to the arm. Landmark detection is a task that falls under computer vision and utilizes feature detection to label landmarks in real time. In the case of this project, we will be using MediaPipe's hand landmark detection model. There are other landmark detection models that exist, but none come close to the ease of use that is provided by MediaPipe, which can be simply imported into python. The landmarks of the hands outputted by the model are used to create angles between either other landmarks, or in reference to the x-axis created by the image frame. The angles computed can be used to directly control the servo motors.

B. Real-Time Music Analysis

This project's advanced demonstration goal will involve real-time music analysis, so we can play a song, and have the system intently listen and react to the song, making the robotic arm dance to the beat of the song. The model that will be used in this demonstration will be

C. Speech Control and Self-Learning

Finally, as a stretch goal, this project aims to train LlaMa to control the robotic arm through input text, which will come from a microphone. This will involve training the LlaMa 7B parameter model, as it is the smallest model. The training will involve finetuning and self-learning. A dataset has been designed that talks to LlaMa, asking it to move the robotic arm a specific number of degrees a certain direction on one of its rotating axes. The input will be concatenated with a prompt message, detailing to LlaMa the input to analyze, and the desired output of the model. During Training, LlaMa will take in input text, and will output a bit map. This bit map will be simulated as if it were being sent to the robotic arm, moving the simulation, and then returning values and errors depending on if it successfully moved the arm to the right degrees or not.

With these three demonstrations, our project can effectively display the effectiveness of the framework, and a base design of AI utilization to control the robotic arm.

AWKNOWLEDGEMENT

The authors wish to acknowledge the assistance and support of Dr. Lei Wei, the sponsor of this project.

REFERENCES

[1] T. H. D. community, "Wikitext · datasets at hugging face," wikitext · Datasets at Hugging Face, https://huggingface.co/datasets/wikitext (accessed Apr. 4, 2024).

[2] "6 DOF Robot Arm A4," Aliexpress, https://www.aliexpress.us/item/2251832665052148.html?src=google&aff_fcid=860278255624475093309a2ff3dfc568-1708535327049-00562-UneMJZVf&aff_fsk=UneMJZVf&aff_platform=aaf&sk=UneMJZVf&aff_trace_key=8602782556244750933309a2ff3dfc568-1708535327049-00562-UneMJZVf&terminal_id=f3f7eafbd4294254a769ffd6a0e1b2a&afSmartRedirect=y&gatewayAdapt=4itemAdapt#nav-review (accessed Apr. 4, 2024).

[3] "ESP32-C3-DevKitM-1," ESP, https://docs.espressif.com/projects/esp-idf/en/stable/esp32c3/hw-reference/esp32c3/user-guide-devkitm-1.html (accessed Apr. 4, 2024).

[4] “WD_BLACKTM SN770 nvmeTM SSD,” Western Digital, https://www.westerndigital.com/products/internal-drives/wd-black-sn770-nvme-ssd?sku=WDS250G3X0E (accessed Apr. 4, 2024).

[5] “Rog Strix B650E-F gaming WIFI: Motherboards: Rog United States,” @ROG, https://rog.asus.com/us/motherboards/rog-strix/rog-strix-b650e-f-gaming-wifi-model/ (accessed Apr. 4, 2024).

[6] “Nvidia GeForce News,” NVIDIA, https://www.nvidia.com/en-us/geforce/news/rtx-40-series-graphics-cards-announcements/ (accessed Apr. 4, 2024).

[7] “Nvidia GeForce RTX 30 series gpus powered by ampere architecture,” NVIDIA, https://www.nvidia.com/en-us/geforce/graphics-cards/30-series/ (accessed Apr. 4, 2024).

[8] “AMD Ryzen 9 CPU list 2024,” CPU, https://www.cpu-monkey.com/en/cpu_family-amd_ryzen_9 (accessed Apr. 4, 2024).

[9] P. Ruiz, “Understanding and visualizingResNets,”Medium, Oct. 08,2018.https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8 (accessed Dec. 03, 2023).

[10] Meta, “Llama 2,” Meta AI. https://ai.meta.com/llama/ (accessed Dec. 03, 2023).

[11] “Llama 2: Open Foundation and Fine-Tuned Chat Models | Meta AI Research,” ai.meta.com, Jul. 18, 2023. https://ai.meta.com/research/publications/llama-2-open- foundation-and-fine-tuned-chat-models/ (accessed Dec. 03, 2023).

[12] M. Krishnakumar, “Exploring Google’s T5 Text-To-Text Transformer Model,” W&B, Sep. 25, 2022. https://wandb.ai/mukilan/T5_transformer/reports/Exploring-Google-s-T5-Text- To-Text-Transformer-Model--VmlldzoyNjkzOTE2 (accessed Dec. 03, 2023).

[13] Rani Horev, “BERT Explained: State of the art language model for NLP,” Medium, Nov. 10, 2018. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model- for-nlp-f8b21a9b6270 (accessed Dec. 03, 2023).

[14] “Papers with Code - ImageNet Benchmark (Image Classification),” paperswithcode.com. https://paperswithcode.com/sota/image-classification-on-imagenet (accessed Dec. 03, 2023).