CIS 655 Course Project Final Report

A principled Approach for Analysing MAC Optimized Computers for AI driven Applications

Mandar Angchekar

Brijesh Tiwari

Roshan Khandelwal

Piyush Mishra

1. Introduction and Background

The motivation behind the proposed project is the growing demand for real-time AI applications and the need for optimized computer architectures that can handle the complex computations required to process vast amounts of data. AI has become an integral part of many industries, including healthcare, finance, and transportation, and its applications continue to grow. Convolutional Neural Networks (CNNs) are a common type of AI algorithm used in computer vision tasks such as image and video recognition. These networks require billions of Multiply-and-Accumulate (MAC) operations to be performed, which can be a bottleneck for traditional computer architectures. To address this issue, specialized hardware and software architectures have been developed to optimize these computations. However, developing optimized architectures for AI-driven programs is a complex task that requires a deep understanding of the specific requirements of these applications. There are various approaches that have been taken in the development of these architectures, and it is important to analyse their effectiveness and understand the trade-offs between different optimization strategies. By providing a principled approach for analysing MAC-optimized computers for AI-driven applications, this project aims to contribute to the field of AI by providing a framework for the development of optimized hardware and software architectures.

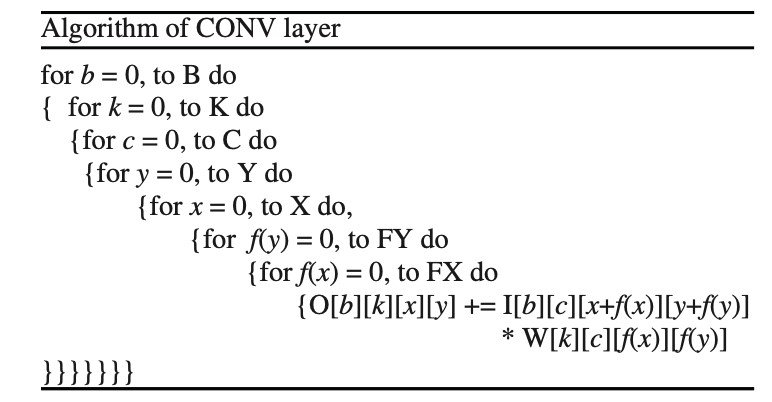
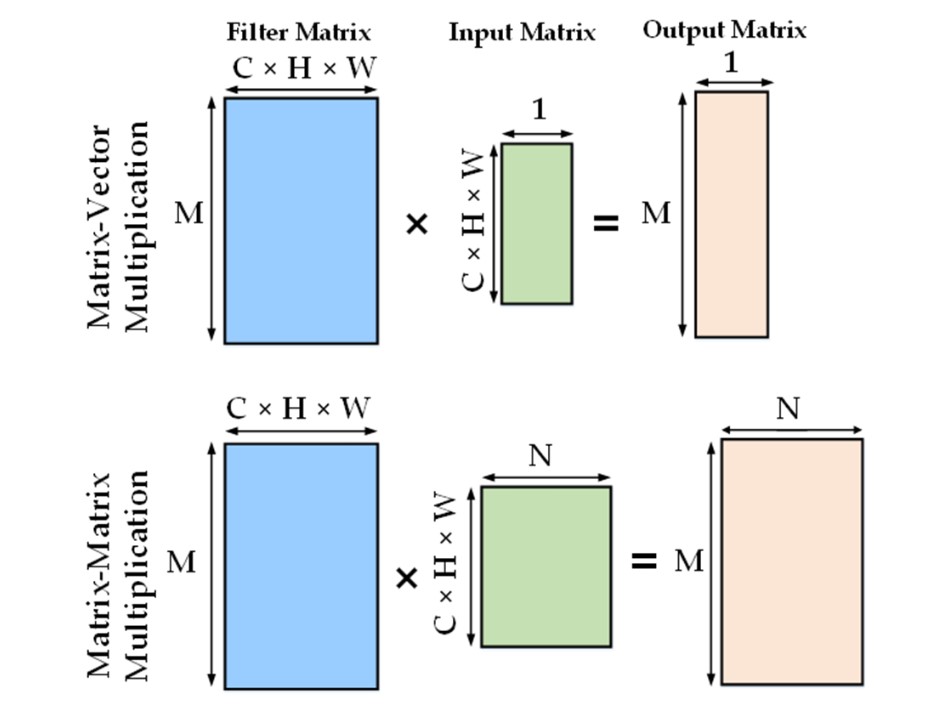
Diagram

Description automatically generated

*Fig. 1. Taxonomy of AI Algorithms*

This will allow for efficient processing of vast amounts of data and support the growth of the AI industry. Additionally, this project has the potential to enable the development of real-time AI applications such as self-driving cars, which require optimized architectures to handle the high computational and memory requirements. Overall, the proposed project is motivated by the need to advance the field of AI and facilitate the development of efficient and effective computer architectures for AI-driven programs.

*Fig. 2. Computation for the Convolution layer of a CNN*



*Fig. 3. Mapping of fully connected layers onto Matrix Multiplication*

Artificial Intelligence (AI) has become an increasingly important field in recent years, with applications in a variety of industries such as healthcare, finance, self-driving cars, and recommendation systems. One of the strengths of AI lies in its ability to learn patterns from data and make informed decisions based on that data. However, this requires processing large amounts of data in real-time, which can be a bottleneck for traditional computer architectures. This is especially true for Convolutional Neural Networks (CNNs), which are commonly used in computer vision tasks like image and video recognition, and require billions of Multiply-and-Accumulate (MAC) operations. To address this issue, specialized hardware and software architectures have been developed to optimize these computations. However, the development of such optimized architectures for AI-driven programs is a complex task that requires a deep understanding of the specific requirements of these applications. This is where the proposed project comes in - it aims to provide a principled approach for analyzing MAC-optimized computers for AI-driven applications. The project will begin by providing an overview of AI algorithms and their unique requirements in terms of computation and memory. This will include an understanding of the different methods used in AI, such as Regression Algorithms, Artificial Neural Networks, Deep Learning, Decision Trees, Clustering, and Fuzzy logic. The focus will then shift towards MAC-optimized computers, including the hardware and software architectures used to optimize computations, such as SIMD architecture and software libraries like Open BLAS, Math Kernel Library (MKL), and Intel DL Boost. The project will then analyze the various approaches taken in the development of MAC-optimized computers, including the use of specialized hardware like Graphics Processing Units (GPUs), Field Programmable Gate Arrays (FPGAs), and Application-Specific Integrated Circuits (ASICs). The project will also explore the trade-offs between these different approaches, such as cost, power consumption, and flexibility. Additionally, the project will analyze the impact of real-time requirements on AI applications, with a particular focus on latency-sensitive applications like self-driving cars. This analysis will require an understanding of the different levels of optimization, including hardware optimization, algorithm optimization, and software optimization. Finally, the project will conclude by providing a set of guidelines for the development of MAC-optimized computers for AI-driven applications. These guidelines will take into account the specific requirements of different AI algorithms, as well as the trade-offs between different optimization strategies. This will be a valuable contribution to the field of AI, as it will provide a framework for the development of optimized hardware and software architectures that can efficiently process vast amounts of data and support the growth of the AI industry. In summary, the proposed project aims to provide a principled approach for analyzing MAC-optimized computers for AI-driven applications. The project will begin with an overview of AI algorithms and their unique requirements, before analyzing the various approaches taken in the development of MAC-optimized computers. The project will also explore the impact of real-time requirements on AI applications and provide guidelines for the development of MAC-optimized computers. This will be a valuable contribution to the field of AI, as it will provide a framework for the development of optimized hardware and software architectures that can support the growth of the AI industry.

Diagram

Description automatically generated

*Fig. 4. Single Instruction Multiple Data (SIMD) Operation*

1. Analysis

2.1 LENET

In the field of artificial intelligence (AI), the development of optimized computer architectures for AI-driven programs is of paramount importance. One of the earliest and most basic AI machines is LeNet, which was developed in 1989 by Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. LeNet was designed to recognize handwritten zip code numbers provided by the US Postal Service. LeNet architecture consists of two main parts: a series of convolutional and pooling layers, and three fully connected layers. The input to the network is a 28 x 28-pixel image. The first convolutional layer, C1, applies six 5 x 5 convolution kernels to the input image. The output of C1 is fed into the first pooling layer, S2, which applies a 2 x 2 pooling window to produce 6 channels of 14 x 14-pixel images. The second convolutional layer, C3, applies sixteen 5 x 5 convolution kernels to the output of S2. The resulting 16 feature images are then fed into the second pooling layer, S4, which applies a 2 x 2 pooling window to produce 16 channels of 5 x 5-pixel images. The third convolutional layer, C5, applies 120 5 x 5 convolution kernels to the output of S4, producing 120 feature images of size 1 x 1. The output of C5 is fed into the first fully connected layer, F6, which consists of 84 neurons. Finally, the output of F6 is fed into the output layer, which consists of 10 neurons corresponding to the 10 possible classes of digits (0-9). While LeNet is a relatively simple AI machine by today's standards, it still provides a good understanding of AI architectures from an algorithmic perspective. It is important to note that even in this basic AI machine, there are a large number of calculations required to produce the final output. With the increasing demand for real-time AI applications, such as self-driving cars, the need for optimized architectures that can handle the huge amounts of data and complex computations required is becoming more pressing. The study of AI programs and their unique requirements, as well as the various software technologies and hardware architectures developed to optimize these computations, is crucial to the development of efficient AI machines. By providing a principled approach for analyzing MAC optimized computers for AI-driven applications, this project will contribute to the advancement of the field and the development of more efficient and effective AI machines. The optimization of computer architectures for AI-driven programs has far-reaching implications for a variety of industries, including healthcare, finance, and transportation. Therefore, research in this area is critical to the advancement of technology and to the betterment of society as a whole.

Diagram

Description automatically generated

*Fig.5. LeNet Architecture. Image source :* [*Source*](https://medium.com/mlearning-ai/lenet-and-mnist-handwritten-digit-classification-354f5646c590)

* 1. Multiply and Accumulate

The multiply-accumulate (MAC) operation is a common computation used in many computer applications, particularly in Neural Networks. It involves multiplying two numbers and adding the result to an accumulator. In modern computers, there are specialized MAC units that consist of a combinational logic multiplier, an adder, and an accumulator register. These units are capable of computing the product more quickly than older computers that use shifting and adding techniques. Percy Ludgate was the first to conceptualize a MAC in his Analytical Machine in 1909, and he used it for division by employing multiplication seeded by reciprocal via the convergent sequence 1/(1+x). Reduced precision arithmetic is a technique that uses fewer bits to represent numbers, and it can be used to implement MAC operations. This technique can significantly reduce the hardware complexity and energy consumption of MAC operations while maintaining a high level of accuracy. By reducing the precision of the numbers used in MAC operations, fewer hardware resources are required to perform the computation. This can lead to significant energy savings, especially in applications where large numbers of MAC operations are performed, such as in Neural Networks. However, it is important to note that reducing precision can also result in loss of accuracy, and it is necessary to carefully evaluate the trade-off between accuracy and hardware complexity when using reduced precision arithmetic in MAC operations.

Diagram

Description automatically generated

*Fig. 6. RPR Multiplier*

In Fig. 6, a Reduced Precision Redundant (RPR) multiplier is shown, which is a common implementation of using reduced precision arithmetic in MAC operations. The RPR multiplier consists of two redundant multipliers with lower accuracy in addition to the original multiplier with full precision. To reduce the precision of the inputs, the least significant bits are truncated. The RPR multiplier also includes logic to detect and correct faults by calculating the difference between the output with full accuracy and one of the copies with decreased precision. If the difference is greater than a specified threshold, an error is assumed to have occurred. To identify the error, the outputs of the two reduced precision copies are compared. If they are different, the redundant copies contain an error. On the other hand, if they are equal, the error has affected the full precision multiplier. By using RPR multipliers, the hardware complexity and energy consumption of MAC operations can be significantly reduced while maintaining a high level of accuracy.

Diagram

Description automatically generated

*Fig. 7. Two stages reduced precision redundancy multiply accumulate*

Two Stages Reduced Precision Redundancy Multiply Accumulate (TSRPR MAC) is a method for implementing multiply-accumulate (MAC) operations using reduced precision arithmetic. The TSRPR MAC method utilizes two stages of redundancy to improve the accuracy of the MAC operation while reducing the hardware complexity and energy consumption. As shown in Fig. 7, the TSRPR MAC method consists of two stages. In the first stage, the input data is quantized to a reduced precision format using a quantization function. The quantization function maps the input data to the nearest value in the reduced precision format, which reduces the number of bits required to represent the data. The quantized data is then used as input to the second stage. In the second stage, three MAC operations are performed using the quantized data and a reduced precision format. The first MAC operation is performed using the full precision multiplier and accumulator. The second and third MAC operations are performed using two reduced precision multipliers and accumulators. The outputs of the three MAC operations are then compared to identify any errors. If the outputs of the three MAC operations are equal, then the result of the full precision MAC operation is correct. However, if the outputs are not equal, then an error has occurred. The error is corrected using a correction function, which adjusts the output of the full precision MAC operation to match the outputs of the two reduced precision MAC operations. Finally, the corrected result is rounded to the desired precision using a rounding function. The TSRPR MAC method allows for the implementation of MAC operations using fewer bits, which reduces the hardware complexity and energy consumption of the MAC operation. At the same time, the use of the quantization and rounding functions allows for a high level of accuracy to be maintained in the MAC operation. Overall, the TSRPR MAC method is a promising approach for improving the performance of MAC operations in DSP systems using reduced precision arithmetic.

Diagram

Description automatically generated

*Fig. 8. One stage reduced precision redundancy multiply accumulate*

The One Stage Reduced Precision Redundancy Multiply Accumulate (OSRPR MAC) method (Fig. 8) involves quantizing the input numbers to a reduced precision format through a quantization function. The quantized numbers are multiplied and accumulated using the reduced precision format, and the result is rounded to the desired precision through a rounding function. This approach combines the quantization and rounding stages into a single stage, simplifying the hardware implementation of the MAC operation and reducing its complexity and energy consumption. The OSRPR MAC method is similar to the TSRPR MAC method, but with a simplified implementation. These methods show promise for improving the performance of MAC operations.

2.3 Memory Organization:

The speed difference between a computer's processor and its memory has been a longstanding bottleneck in computer architecture. However, there is an architecture called CIM that aims to take advantage of this bottleneck by integrating computing elements directly into the machine's memory. By doing this, the CIM architecture reduces the amount of data movement between the computer's off-chip memory and its computing logic, which helps to decrease overhead[2]. The idea behind CIM technology was inspired by advances in CMOS transistors, machine learning, and big data. Today, AI computing relies heavily on multi-dimensional convolutions that require processing numerous channels. To address this, 3D stacking technology (such as SRAM or DRAM) is used to enable near-memory computing in the hardware of AI processors.

Top of Form

Diagram

Description automatically generated

*Fig.9. CIM architecture based on Look-up tables*

The differences in speed between a computer's processor and memory have long been a bottleneck in computer architecture. To overcome this issue, a new architecture called CIM has been developed, which places computing elements within the memory of a machine. CIM reduces the amount of data movement between the off-chip memory and the computer's processing unit. CIM technology is particularly useful for AI computing, which involves processing large amounts of data. CIM hardware is typically designed using SRAM and is capable of processing multiple bit inputs into WL signals. This is done using a completely parallel input structure. First, precharge is applied to the bit line pair, then the WLS is driven to produce the matching bit cell current. The information stored in the bit cell determines which current is applied to BL or BLB, resulting in a weight of either -1 or +1, depending on the case[2]. The currents of all bit cells on BL/BLB are aggregated, and the comparator produces a threshold symbol. To detect bit line current, an 8-bit ADC may be used in the design, but this will result in a power overhead. To overcome this, a 1-bit comparator is suggested, and the number of R1 units is altered in each step to ensure that positive and negative BLs have the same amount of current. Multi-bits are input in order, and positive and negative BLs are compared by bit serial by adjusting the CD signal.

The analog-digital hybrid CIM design has several challenges, including the large space it occupies, its sensitivity to process, voltage, and temperature conditions, and the need for the ADC to perform computations[2]. Furthermore, verifying the mixed signal-based SoC devices on a large scale is difficult due to the lack of accurate prototype models. To overcome these challenges, advanced CIM architectures have been developed that move beyond the mixed solution. An example of such an architecture is the lookup-table based compute-in-memory.

The lookup-table based compute-in-memory (CIM) architecture is a solution to the issues found in mixed analog-digital CIM designs, while taking advantage of their benefits. This type of CIM has two memory rows devoted to the lookup table, which can be used for both storage and computation. During storage mode, the lookup table array is separated from the memory array, resulting in lower power usage. In computation mode, the lookup table enables multiplication, division, and activation operations by storing critical data inside it. By converting an even number of calculations into a shift operation, the LUT-based CIM design reduces the required table size from 225 to 49 bits during multiplication [2]. To improve the parallelism of matrix multiplication, different phases of the operation are rebuilt as multilayer pipelines.

To reduce resource usage and energy consumption, the LUT-based CIM architecture stores complex 1/dividend values during division operations, eliminating the need for complex division hardware. The piecewise linear approximation approach is used for functions such as counting and accumulating partial sums, and the value of each iteration is stored in the LUT for the activation function. The suggested LUT-based CIM architecture outperforms both the neural cache and L3 cache in terms of overall performance and energy consumption, with a performance increase of 1.72 times and a reduction in energy consumption of 3.14 times [2].  
Diagram

Description automatically generated

*Fig. 10. (a) Resistive RAM (b) phase-change RAM (c) Spin-transfer torque MRAM (d) Spin-orbit torque MRAM*

he use of emerging non-volatile memory (NVM) technologies, such as RRAM, PCRAM, and MRAM, is a promising approach for CIM design. RRAM devices consist of a metal-oxide layer that switches between high and low resistance states when an external voltage surpasses a threshold[2]. The resistance of RRAM cells can range from a few K to tens of M. PCRAM devices use chalcogenide glass as a switching element, which undergoes regulated heating and cooling cycles to switch between amorphous and crystalline forms. The resistance of PCRAM cells can exceed 1 M. MRAM devices use magnetic tunnel junctions (MTJs) as their main storage core, where the parallel and antiparallel magnetization states of the free layer represent low and high resistance states, respectively. Two major MTJ structures are used for programming MRAM, namely spin transfer torque and spin orbit torque. NVMs offer advantages over conventional SRAM such as low leakage power, high density, and increased energy efficiency[2].

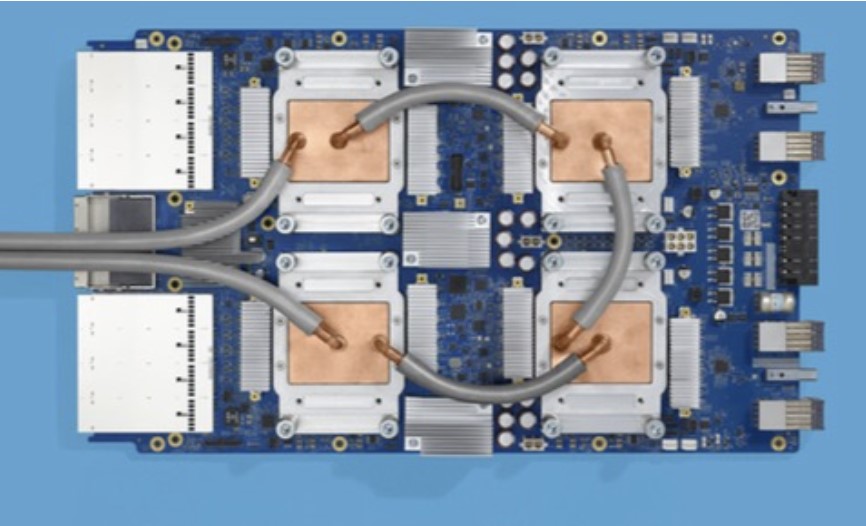
The NVM-based CIM architecture includes mathematical procedures such as MAC operations, matrix transposition, and bitwise logic operations. In contrast to conventional CIM macro, the multi-state resistive cell provides more accuracy and density by allowing continuous adjustments of resistance. For MAC operations, several memory rows are activated, and each memory cell contributes a portion of the total bitline current to generate voltages that are compared to various reference cells, which are then identified by the sense amplifier to produce the calculation result. To increase performance, MAC results can be calculated in parallel. For NN acceleration, feature maps are transformed into SL voltage for each matching word line, while filter weights are kept in memory cells. An RRAM-based CIM solution uses an analog-to-digital converter to distinguish between different resistance values, allowing for high-precision MAC operations. Additionally, with extra wires and switches, NVMs can be used to transpose data quickly by modifying the memory column and peripheral circuit.

2.4 GPU and TPU

GPU and TPU are types of processors that are designed to accelerate the performance of certain types of computing tasks. GPUs, or graphics processing units, are specialized processors that are designed to handle the complex calculations needed to render images and video. TPUs, or tensor processing units, are specialized processors that are designed to handle the complex mathematical operations needed to train and run machine learning models[1]. Both GPUs and TPUs are used in a variety of applications, including gaming, video editing, and machine learning.

The history of GPUs and TPUs can be traced back to the early days of computer graphics and the development of the first graphical user interfaces (GUIs). In the late 1970s and early 1980s, researchers at companies like Xerox and Apple began working on ways to improve the performance of graphics processing on personal computers. This led to the development of the first GPUs, which were used to accelerate the rendering of images and video on early personal computers.

TPUs, on the other hand, have a more recent history. The concept of a tensor processing unit was first proposed by Google in 2016, with the goal of creating a specialized processor that could handle the complex mathematical operations needed to train and run machine learning models. The first TPUs were released in 2017, and they quickly gained popularity among machine learning researchers and developers due to their impressive performance and efficiency. GPU and TPU are two different types of hardware that are commonly used to accelerate machine learning and other computationally intensive tasks.



*Fig 9. GPU and TPU:* [*Source*](https://serverguy.com/comparison/cpu-vs-gpu-vs-tpu/)

A GPU, also known as a graphics processing unit, is a type of processor specifically designed to handle complex calculations needed for rendering images and video. It is ideal for tasks that involve parallel processing, like training deep learning models, as it has multiple cores that can run simultaneous calculations, making it faster than a regular CPU. On the other hand, a TPU, or tensor processing unit, is a specialized hardware accelerator designed solely for deep learning tasks, particularly matrix operations used in deep learning like matrix multiplication and convolution[1].

Both GPUs and TPUs are specialized types of processors used to handle complex calculations, but they differ in their design and purpose. While GPUs are versatile and can handle multiple tasks, including gaming, video editing, and machine learning, TPUs are optimized specifically for deep learning. GPUs can access data from a high-bandwidth memory system, while TPUs can access data directly from the processor's memory, which accelerates deep learning workloads. TPUs are designed to handle a wider range of data types than GPUs, making them more efficient for deep learning tasks. In summary, the main difference between GPUs and TPUs is that GPUs are general-purpose processors, whereas TPUs are specialized accelerators designed for deep learning tasks.

2.4 Software Perspective

This section will examine how software frameworks like CUDA and Intel Deep Learning Boost can affect AI processors from a software perspective.

**CUDA:**

CUDA is a high-performance computing platform developed by NVIDIA, which enables developers to speed up compute-intensive tasks by leveraging the massive parallelism of GPUs. It is a programming model and software development kit that enables developers to write parallel programs using a C++-like syntax to execute across multiple GPUs and CPUs. The main advantage of CUDA is its ability to harness the power of GPUs, which have many more cores than CPUs and are specifically designed for parallel processing. GPUs can perform many calculations in parallel, making them ideal for applications that require high computational power, such as machine learning, data analytics, and scientific computing. With CUDA, developers can write programs that can be optimized for parallel processing, allowing them to run much faster than on a CPU alone. They can take advantage of the multiple cores in a GPU to perform calculations in parallel, which can lead to significant performance gains. CUDA is supported on many of NVIDIA's GPUs, as well as on other hardware platforms through third-party partnerships. To use CUDA, developers must first install the CUDA Toolkit, which includes the CUDA driver and other tools needed to develop and run CUDA applications. The toolkit also includes the CUDA runtime, which allows developers to run their applications on any system with a CUDA-enabled GPU. Once the toolkit is installed, developers can use the included CUDA compiler to compile their CUDA code into binaries that can run on the GPU. They can then use the CUDA runtime to execute their applications and take advantage of the parallel processing capabilities of the GPU. In addition to its support for parallel computing, CUDA also provides a range of tools and libraries that can help developers optimize their applications. For example, the cuDNN library provides optimized routines for deep neural networks, while the Thrust library provides a collection of parallel algorithms and data structures. Overall, CUDA is a powerful platform for accelerating compute-intensive tasks, enabling developers to leverage the massive parallelism of GPUs to achieve significant performance gains in their applications. By providing a range of tools and libraries, CUDA makes it easier for developers to optimize their applications and take advantage of the parallel processing capabilities of GPUs.

**Intel Deep Learning Boost (DL Boost) :**

Intel Deep Learning Boost (DL Boost) is a technology developed by Intel to enhance the performance of deep learning algorithms on Intel CPUs. Deep learning is a subset of machine learning that involves training neural networks on vast amounts of data, which can be computationally demanding. Intel DL Boost uses Vector Neural Network Instructions (VNNI), which are specialized hardware instructions that can optimize deep learning algorithms to improve their performance and reduce their power consumption on Intel CPUs. The use of VNNI instructions in Intel DL Boost can improve the performance of key deep learning operations, such as matrix multiplication and dot products, which are essential to many deep learning algorithms. By making these operations more efficient, Intel DL Boost can help deep learning algorithms to run faster and more efficiently on Intel-powered devices. This technology can also reduce the power consumption of deep learning workloads, allowing them to be run more efficiently on devices with limited power, such as laptops and tablets. In addition to improving the performance of deep learning workloads, Intel DL Boost can enable the deployment of deep learning models on a wider range of devices. This can facilitate the development of new applications and innovations in the field of AI, expanding its capabilities and potential impact. Intel DL Boost is supported by many of Intel's latest CPU models, including the Xeon and Core processors, and is integrated with popular deep learning frameworks such as TensorFlow and PyTorch. Developers can use these frameworks to write code that utilizes Intel DL Boost to improve the performance of their deep learning models. Intel also provides development tools, such as the Intel AI Analytics Toolkit, to help developers optimize their deep learning algorithms for use with Intel DL Boost. Overall, Intel DL Boost is a powerful technology that can significantly enhance the performance and efficiency of deep learning algorithms on Intel CPUs. Its integration with popular deep learning frameworks and availability on a wide range of Intel processors makes it a valuable tool for developers and researchers in the field of AI. By enabling the deployment of deep learning models on a wider range of devices, Intel DL Boost can help advance the capabilities of AI and drive new breakthroughs in the field.

3. FUTURE WORK AND CONCLUSIONS

Artificial intelligence (AI) algorithms have revolutionized every domain and infrastructure. This has been made possible due to the advancements in mathematics, software libraries, and the processor architectures of systems capable of running these programs. However, there is still much more room for improvement from both the software and hardware perspectives that were not covered in this paper. In the last decade, AI's capabilities and applications have expanded rapidly. It has impacted every industry and individual, directly or indirectly. Many real-world applications have incorporated AI into their functionalities to provide outstanding benefits. However, AI is still in its infancy in many real-world applications. Current trends suggest that AI algorithms will continue to explore the neural structure to match the human brain's capacity to accurately handle critical tasks. Big tech companies, such as Apple and Google, have significant budgets set aside for the research and advancement of AI, and academic institutions are recognizing AI as a separate field of study. AI is expected to develop in new and innovative ways, driving disciplines like IoT, big data, autonomous vehicles, and intelligent robotics. One of the primary challenges for the development of AI has been computing power. The focus is on hardware systems that can handle enough resources to meet the increasing demands of AI. Although Moore's Law has been responsible for the progress made in microprocessors, above a certain clock limit, thermal problems become inescapable, and the performance path has already been diverted. However, theories suggest that structuring microchips in three dimensions could make them more efficient. While high performance is a benefit, this idea presents several thermal and interconnection problems that need to be resolved before it can be successfully applied. Therefore, microprocessors are expected to continue to rule for another ten years. Nonetheless, cutting-edge methods like quantum and molecular computing may alter how microchips are made in the future. In conclusion, AI has transformed every sector and is poised to continue doing so. The future of AI will be determined by both software and hardware advancements. AI algorithms will continue to evolve, making machines more intelligent, efficient, and human-like. The continued improvement of hardware systems will be critical to meeting the demands of AI applications. It is clear that the AI revolution is just beginning, and we can expect to see significant developments in the near future.

Artificial Intelligence (AI) has been transforming various domains and infrastructures due to the advancements in mathematics, software libraries, and processor architectures that support these programs. However, there is still ample scope for improvement from both software and hardware perspectives. In the past decade, AI has expanded its capabilities and applications, impacting every industry and individual either directly or indirectly. Although AI has been incorporated into various real-world applications, it is still in its infancy in many areas. AI algorithms will continue to develop and delve deeper into neural structures to match the human brain's capacity to handle important tasks. The development of AI has always been hampered by computing power, with a focus on hardware systems that can handle enough resources to meet the increasing demands of AI. Moore's Law has been responsible for the progress made in microprocessors up to this point. Still, beyond a certain clock limit, thermal problems become inescapable, and the performance path has already been diverted. High performance is a benefit, but this idea also presents a number of thermal and interconnection problems that need to be resolved before it can be successfully applied. Therefore, microprocessors will continue to rule for another ten years, and new methods like quantum and molecular computing could alter how microchips are made in the future. Microprocessors are critical to AI, and their architectural improvements have significantly increased performance and efficiency. The computational capacity of microprocessors increased exponentially with the development of RISC architecture, superscalar, deep pipelining, and multicore architectures. AI developers began to make progress with machine learning, deep learning, and related datasets after becoming aware of the processing power available. Microprocessors have also made changes to their design to effectively carry out the intricate algorithms of AI for a variety of jobs, such as SIMD, GPUs, and TPUs. As microprocessors have adopted new ways to meet the rising needs of complex AI models, the two areas of computer architecture and artificial intelligence have complemented one another. AI is now accessible in every home and industry thanks to PCs, smart devices, and other embedded platforms due to the convergence of these two technologies. In the future, the next generation of computers will be based on microprocessors working in tandem with highly specialized accelerators dedicated to executing power-hungry AI algorithms. However, the limits of parallel hardware in microprocessors and the deviation from Moore's Law make it necessary to develop precisely calibrated AI accelerators. The present area of research on AI accelerators will continue to expand and change with more cutting-edge trends. AI and microprocessor design will continue to develop in tandem, with new topologies and specialized accelerators to address issues like data security and the complexity of information in data-intensive applications like big data. The integration of AI into various industries will continue to create benefits and transform the way we do things, with a better adoption of the AI revolution expected in the near future.

4.REFERENCES

# Jouppi N, Young C, Patil N, et al. In-datacenter performance analysis of a tensor processing unit. In: Proceedings of ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA), 2017

# Chang L, Li C L, Zhang Z M, et al. Energy-efficient computing-in-memory architecture for AI processor: device, circuit, architecture perspective. Sci China Inf Sci, 2021, 64(6): 160403, https://doi.org/10.1007/ s11432- 021- 3234- 0

# Khan, F.H.;Pasha,M.A.; Masud, S. Advancements in Microprocessor Architecture for Ubiquitous AI—An Overview on History, Evolution, and Upcoming Challenges in AI Implementation. *Micromachines* **2021**, *12*, 665. https://doi.org/10.3390/mi12060665

# Liu L, Qu Z, Deng L, et al. Duet: boosting deep neural network efficiency on dual-module architecture. In: Proceedings of the 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2020. 738–750

# Zairi H, Talha M K, Meddah K, et al. FPGA-based system for artificial neural network arrhythmia classification. Neural Comput Appl, 2019, 32: 4105–4120

# Albericio, J., Judd, P., Hetherington, T., Aamodt, T., Jerger, N.E. and Moshovos,

# A., 2016 Cnvlutin: Ineffectual-Neuron-Free Deep Neural Network Computing. Proc. Int’l Symp. on Computer Architecture.

# Contreras, Iván & Vehí, Josep. (2018). Artificial Intelligence for Diabetes

# Management and Decision Support: Literature Review. Journal of Medical

# Internet Research. 20. 10.2196/10775.

# Le Cun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989, January). Handwritten digit recognition with a back-propagation network. In *Proceedings of the 2nd International Conference on Neural Information Processing Systems* (pp. 396–404)

# Chi P, Li S, Xu C, et al. Prime: a novel processing-in-memory architecture for neural network computation in ReRAM-based main memory. In: Proceedings of

# ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), 2016. 27–39

# 10.Cook, S. (2012). CUDA Programming: A Developer’s Guide to Parallel Computing with GPUs (1st ed.). San Francisco, CA, USA: Morgan Kaufmann

# Publishers Inc.

# 11.K. Chen, L. Chen, P. Reviriego and F. Lombardi, "Efficient Implementations of Reduced Precision Redundancy (RPR) Multiply and Accumulate (MAC)" in IEEE

# Transactions on Computers, vol. 68, no. 05, pp. 784-790, 2019. doi:

# 10.1109/TC.2018.2885044

This project is done in combination with the CS229 Machine Learning final project. The CS231A Computer Vision primary component is the hand and finger segmentation using 3D camera.