Comparative Performance Evaluation of Tensor Processing Unit (TPU) and Graphics Processing Unit (GPU) across a range of Deep Learning Models

Mitil Vivek Jawale	Shivani Anil Neharkar	Prajwal Rajendra Karale	Abhishek Ramamurthy
College of EECS	College of EECS	College of EECS	College of EECS
SUID: 754778650	SUID: 320953757	SUID: 293551451	SUID: 967052651
mvjawale@syr.edu	sanehark@syr.edu	prkarale@syr.edu	ar1002@syr.edu

Abstract – subset of artificial intelligence called "Deep Learning" implies artificial neural networks with more than one layers to compute and solve challenging problems. The use of deep learning has revolutionized and expanded various fields of applications, ranging from the processing of larger datasets to the analysis and classification of images using computer vision. It has also contributed to advancements in natural language and speech recognition, linguistic translation, identification of human activities, social network recommendations, and the optimization of web search engines. Deep Neural Networks are widely employed in sophisticated AI applications that teach computers to mimic the functions of the human brain by feeding enormous amounts of data to artificial neurons with multiple layers, which aid in predicting or inferring the outcome under various conditions. Deep learning models require a significant amount of computing power and typically perform better when operating on specialized processors and accelerators made for accelerating compute-intensive workloads. Hardware platforms such as Tensor Processing Units (TPUs) and Graphics Processing Units (GPUs) are often utilized for deep learning problems due to their extensive multi-core processing capabilities and abundant memory bandwidth. These accelerators are known to deliver better performance than CPUs in many cases. Several hardware platforms have been created in the past decade to speed up machine learning training. This comprises hardware platforms from both established corporations—such as Google's TPU and Nvidia's GPU—and emerging ones—such as Graphcore's IPU and Cerebras' WaferScale Engine (WSE). Using a variety of benchmark datasets and neural network topologies, such as convolutional NN (CNNs) and recurrent NN (RNNs), we will evaluate the performance of TPUs and GPUs. Our findings demonstrate that both TPUs and GPUs can execute deep learning workloads at high-performance levels and with higher precision, with TPUs typically outperforming GPUs for most tasks. For many deep learning models for larger datasets and more intricate neural network topologies, we find that TPUs offer shorter training times and higher accuracy. However, GPUs continue to have advantages in some situations, like tiny datasets or when a model requires more memory than a TPU can provide.

INTRODUCTION:

Over the past few decades, the field of Artificial Intelligence has undergone a significant transformation. This can be attributed to the emergence of powerful machine learning algorithms, the abundance of data available for analysis, and the ever-increasing computing power. As a result of these factors, AI research has experienced a recent surge in both its theoretical advancements and practical applications. If we represent AI in the form of a set diagram, then it would be as follows. We can clearly see from the Figure 1 that is given below that the domain of AI is a superset and Machine Learning is a part or subset of AI. Deep Learning and CNN are fragments of Machine Learning, and all these sub-domains are layered within each other.

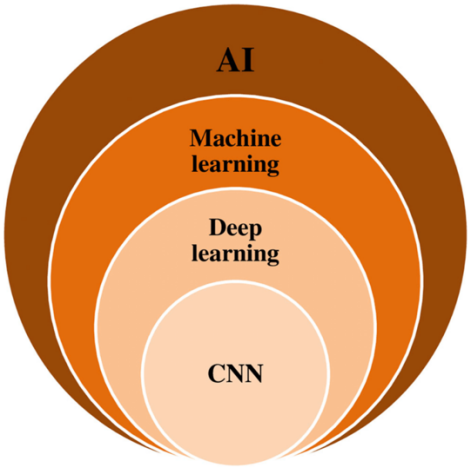

Figure 1. Relational Diagram for AI

Here in this review paper our main aim is to focus on the domain of Deep Learning which involves multiple layered neural networks model which learn and solve a given problem based on the data provided to them. Using this technique helps in achieving higher accuracy. Deep Learning is the domain in which we have different types of neural network that help us to perform the necessary computation for our problem. Deep neural networks (DNNs) are a specialized type of neural network that are capable of modeling complex relationships and patterns in data. The defining feature of DNNs is their deep architecture, which consists of many layers of interconnected neurons. A neuron which is like the neuron cell present in our brain is a fundamental unit of computation within a neural network, which receives input from other neurons, performs a computation, and generates an output signal. The architecture of DNNs allows them to automatically learn features and patterns in the data, without the need for explicit programming. Applications for DNNs encompass speech and image recognition, natural language processing, and recommendation engines. However, the training of DNNs requires significant computational resources and large amounts of labeled data.

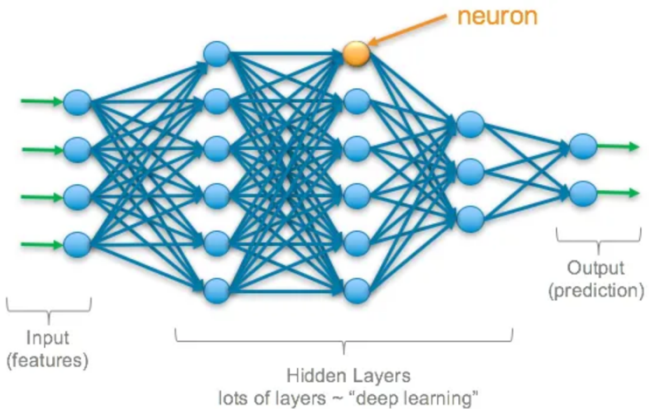

Figure 2. Representation of a Neuron[A]

A Multilayer Perceptron (MLP) is depicted in the above diagram. The input layer, a hidden layer, and the output layer are the minimum number of layers required for an MLP. They are all fully interconnected, with a weight connecting every node in one layer to every node in the next. Convolutional and recurrent neural networks are a few of the supplementary neural network types. An advanced form of neural network frequently used for image and video

processing is the CNN i.e convolutional neural network. The key characteristic of a CNN is its ability to automatically learn spatial hierarchies of features from the input images. Unlike a traditional neural network, which treats the input data as a one-dimensional vector, CNNs process the input images as two-dimensional arrays or tensors. A specific kind of neural network called a recurrent neural network (RNN) is made to process sequential data. RNNs have a feedback loop that enables data to survive over multiple timesteps, in contrast to feedforward neural networks that analyze each input independently. RNNs can use data from earlier inputs to inform the processing of current inputs thanks to this feedback loop. A "hidden state" that summarizes the data so far is kept by each neuron in an RNN. Every timestep, based on the current input and the prior hidden state, this hidden state is changed. However, when more and numerous types of neural networks layers evolve, intricate CNN structures and high-depth CNN models result. Thus, learning and assessing the resulting large-scale neural networks requires millions of parameters, billions of operations, and a lot of processing power. Thus, all these operations were not feasible for the General-Purpose Register to execute. To boost the efficiency of the neural networks, hardware accelerators such application specific integrated circuits (ASIC), field programmable gate arrays (FPGA), and graphics processing units (GPU) are used. Understanding the benefits of various technologies is crucial for studying artificial intelligence, and restrictions like power, weight, and size should be adequately considered. One of the additional components is the storage bandwidth for parameter estimation, which is used to read and edit datasets.

While training a network especially a CNN a significant amount of computational power and memory to process the large amounts of data involved in training and testing deep learning models. Due to its distributed processing layout and ability to handle for many simultaneous calculations, GPUs have the capacity to accelerate the calculations requested by neural networks than CPUs and provide quicker processing times. A processor specifically intended to speed up illustrations rendering, and other highly parallelizable tasks is known as a graphics processing unit (GPU). It has many processing cores that can perform multiple tasks at once and have been structured into streaming multiprocessors (SM).

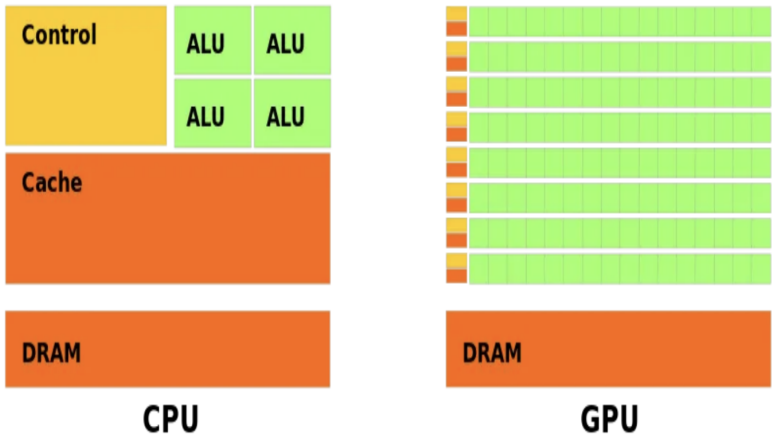

Figure 3[B]. Basic Skelton of CPU and GPU Architecture

The above diagram illustrated the basic architecture of a CPU and GPU. Here we can observe that. The computational aspects units in a CPU initially appear to be "bigger" but few, while the computational units in a GPU initially tend to be "smaller" but numerous. What a CPU or GPU cores can perform and how many are included in a device is indicated by their size and number. Scientific computing, machine learning, and graphics rendering are among most of the jobs that GPUs can handle. They are ideal for applications that call for concurrent processing since they are very effective at completing the same action on numerous bits of data at once. However, there are some inefficiencies in the architecture of the GPU such as insufficient memory, overheating, power supply issues, driver issues and hardware failure due to which sometimes it is not suitable for training the networks. While GPUs are widely used for training

deep learning models, they have some limitations due to their architecture. For example, GPUs have a finite amount of memory, which can cause issues if the model requires more memory than is available. Similarly, GPUs generate a lot of heat, and if they overheat, their performance can degrade, or they may shut down completely. GPUs also require a significant amount of power, which can cause issues if the power supply is insufficient or unstable. Additionally, outdated, or incompatible drivers or hardware failures can also cause issues with GPU malfunctions. So, to enhance the performance a new AI accelerator was introduces called as the Tensor Processing Unit.

INTRODUCTION TO TPU:

Due to the diminishing returns in CPU performance following the end of Moore's Law, specialist hardware design has witnessed an enormous spike during the decade that followed. Google created the Tensor Processing Unit (TPU) in the year 2015, a specialized hardware accelerator, to speed accelerate machine learning workloads. The inference and training processes of deep neural networks, which form the basis for present-day artificial intelligence applications, are specifically optimized. In 2013, it became readily apparent that they required to double the capacity of their data centers to cope with the projected computing needs of a situation in which individuals use voice search for at least three minutes per day utilizing regular CPUs. They therefore commenced a significant venture to produce an individualized Application-specific integrated circuit for inference swiftly (and bought commercial GPUs for training). A TPU primarily deals with geometric objects that define the linear relationships among geometric vectors, scalars, and other tensors. Using only 0.5 watts of electricity used for each TOPS, the TPU can perform 4 trillion operations per second (TOPS).

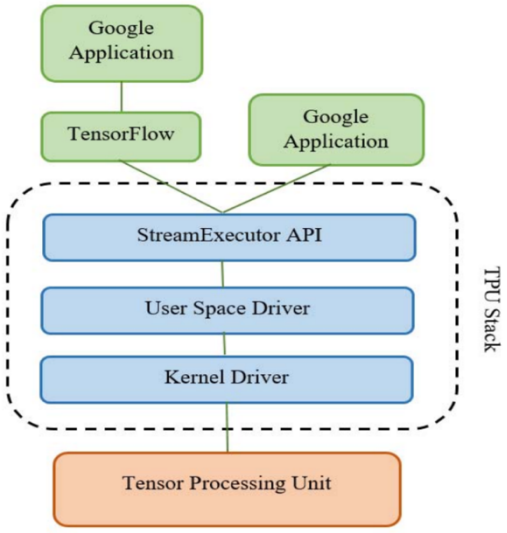

Figure 4[C] A Tensor Processing Unit

The above diagram illustrates a simple architecture of a Tensor Processing Unit. These chips have been developed to perform several low precision computations with many input/output operations per joule.

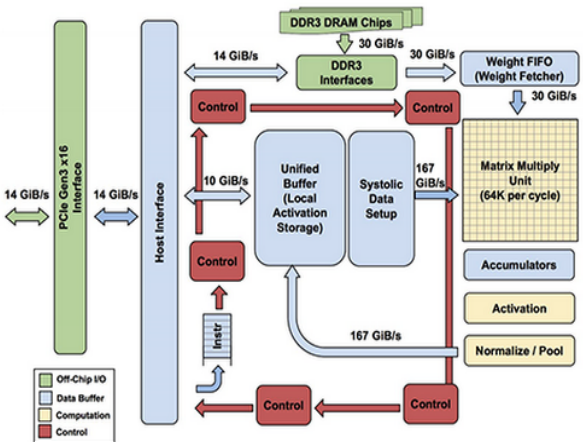

Figure 5[D] Diagram for TPU architecture V1

TPU ARCHITECTURE:

The matrix multiplier unit, unified buffer, and activation unit are just a few of the processing resources that are included in the TPU architecture shown in the diagram. A dozen of high-level instructions have been defined specifically for neural network inference that regulate how the MXU, UB, and AU conduct their operations. The following set of instructions is specifically geared towards performing the primary mathematical operations necessary for neural network inference, which include carrying out matrix multiplication between input data and weights, as well as implementing activation functions. The TPU follows a complex instruction set computing (CISC) architecture with instructions taking between 10 and 20 clock cycles per instruction. This architecture allows for more efficient execution of machine learning workloads. Due to this the main goal of the TPU model was to run the entire inference model to minimize the involvement of the host CPU and flexible enough to suit the needs of the neural network. Instead of being closely linked to a central processing unit (CPU), the TPU was created to function as a co-processor on the i/o bus of PCIe.

Due to this design decision, the TPU can be installed in current server setups in a manner analogous to that of graphics processing units (GPUs). To simplify the TPU's design and debugging processes, the host server gives the hardware with instructions rather than the TPU collecting its own set of instructions. The TPU can run full inference models entirely within its hardware, reducing host CPU interactions and meeting neural network requirements. This solution enables effective processing and surpasses CPUs and GPUs in tasks demanding neural network inference. There are in total four generation of TPU i.e 1st generation, 2nd generation, 3rd generation and the latest one is the Edge TPU. First-generation TPUs have an 8-bit multiplication engine and CISC instruction set on top of a PCIe 3.0 interface. It performs effectively during the inference phase of neural networks but not during training because it only supports integer operations and has a small bandwidth. It performs convolutions and matrix computations, transfers data to and from the host, and applies activation functions.

Google launched the second generation of tensor processing units in 2017. These TPUs have higher performance levels exceeding 45 tera Floating Point Operations Per Second (FLOPS) and increased bandwidth capabilities of up to 600 GB/s. To achieve a more significant performance gain, the team used four TPU chips, bringing the processing power up to 180 tera FLOPS. To enable parallelism and further improve the processing power of their artificial intelligence and machine learning systems, Google also produced TPU pods, each of which consisted of 64 TPUs grouped to a chip. In second generation it was able to work on floating point operations and it performed well during the training phase and the testing phase. The smallest TPU v2 setup has 16 GiB of HBM and 4 TPU chips. Two Tensor Cores are found on each TPU chip. There is a Matrix Multiply Unit, a scalar unit, and a vector unit in every Tensor Core. The TPU v2 block diagram is shown below.

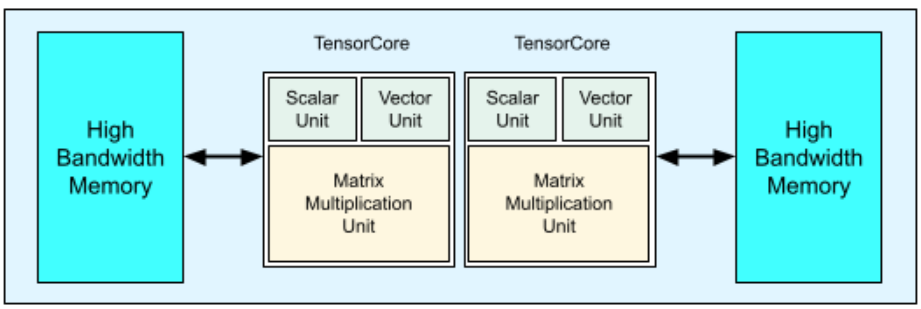

Figure 6 [E] Block Diagram of TPU V2 architecture

The TPU architecture's mmu is built to handle the huge, dense matrices that are typically used in machine learning applications. It retains a single 64 KiB weight tile while also employing double buffering to mask the 256 cycles necessary for shifting a tile.

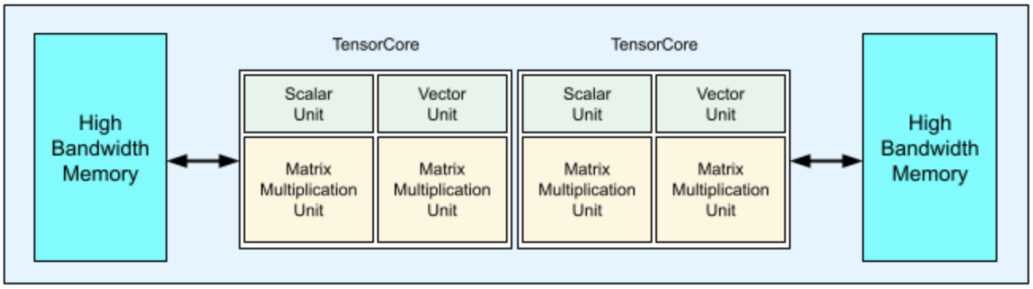

Figure 7[F] Block Diagram of TPU v3 architecture

There are two Tensor Cores on each v3 TPU chip. Each Tensor Core contains two MXUs, a vector unit, and one scalar unit. The following table illustrates the key characteristics of a TPU v3 Pod in addition to the values that accompany them. The characteristics of TPU v3 is given below.

Attribute	Value
Compute-per-chip	420 tflops (bf-16)
Memory-per-chip	128 GB HBM2
Chips-per-pod	2048
Compute-per-pod	855 PFLOPS (bf-16)
Memory-per-pod	256 TB HBM2
Interconnect-technology	TPU interconnect (2D toroidal mesh)
Interconnect-bandwidth per chip	600 GB/s
Interconnect-bandwidth per pod	1.2 TB/s

COMPARISON:

3. ANALYSIS
3.1 TRAINING PHASE
3.1.1 HARDWARE SETUP

To study the hardware accelerators most used for Deep Learning, we specifically chose multi-core CPUs from three manufacturers - Nvidia, AMD, and Google. From each vendor, we selected 1-3 processors for analysis. In Table 1 below, we have provided information on the selected accelerators, along with several important variables that significantly influence their performance.

Vendor	Models	Memory	Theoretical_FLOPS	Memory_Bandwidth	Memory_Type
Google	TPU v2 TPU v3	64 GB 128 GB	180 Tera 420 Tera	600 GigaBytes /s 900 GigaBytes /s	HBM HBM
Nvidia	TitanX	48 GB	11 Tera (FP-32)	480 GigaBytes/s	GDDR-5X
Nvidia	Tesla-P100	12 GB	9.50 Tera (FP-32)	732 GigaBytes /s	HBM-2
Nvidia	TeslaV 100	16 GB	112 Tera	897.0 GigaBytes /s	HBM-2
AMD	RadeonVII	16 GB	13.4 Tera	1.0 TeraBytes/s	HBM2

Table 1. Hardware Setup

GPU ARCHITECTURE:

- Scalar processors, vector processors, and matrix processors are just a few examples of the various processing units found in GPU hardware that are made to handle various kinds of computations.
- To facilitate quick data access and lower latency, GPUs frequently include many levels of cache memory, including on-chip caches like the L1 and L2 cache and off-chip memory like high-bandwidth memory (HBM).
- Modern GPUs frequently incorporate specialized hardware capabilities like Ray Tracing Cores and Tensor Cores, which are intended to enhance the realism and lighting effects in 3D rendering applications and matrix computations frequently employed in machine learning workloads, respectively.
- Additionally, GPU hardware has a variety of memory types, such as DRAM, SRAM, and VRAM, which are each tailored for data access patterns and workloads.
- Clock speed, memory bandwidth, thermal design, and power efficiency, among other variables, can have a significant impact on the functionality and performance of GPU hardware.

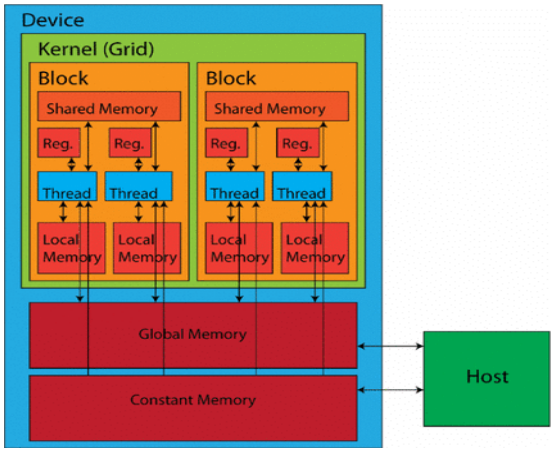

Figure 8[H]: GPU Block Diagram

Feature	NVIDIA Tesla V100	NVIDIA A100	NVIDIA Titan RTX	AMD Radeon VII	AMD Radeon Pro VII	Google TPU v2	Google TPU v3
Process Node	12nm	7nm	12nm	7nm	7nm	N/A (custom ASIC)	N/A (custom ASIC)
Compute Power	7.8-15.7 TFLOPS	9.7-19.5 TFLOPS	11.1 TFLOPS	13.8 TFLOPS	13.1 TFLOPS	45 TFLOPS	420 TFLOPS
Memory Bandwidth	900-1200 GB/s	1.6 TB/s	616 GB/s	1 TB/s	1 TB/s	N/A	1.2 TB/s
Memory Capacity	16-32 GB HBM2	40-80 GB HBM2	24 GB GDDR6	16 GB HBM2	16 GB HBM2	16 GB HBM	32 GB HBM
CUDA Cores/Stream Processors	5120	6912	4608	3840	3840	N/A (TPU cores)	N/A (TPU cores)
Tensor Cores	Yes	Yes	Yes	No	No	N/A	N/A
ROCm Support	No	Yes	No	Yes	Yes	No	No
Precision	Supports 16-bit	Supports 16-bit	Supports 16-bit	Supports 16-bit	Supports 16-bit	Supports 8-bit	Supports 8-bit
Max Power Consumption	300W	400W	280W	295W	250W	N/A	N/A

Based on the comparison table, it appears that the Nvidia A100 GPU has the highest power and capacity for deep learning among the models listed. It has a much higher number of CUDA cores, Tensor Cores, and memory compared to the other GPUs, as well as support for technologies such as NV Link and Multi-Instance GPU. The Google TPU v3 also has high performance for matrix multiplication, but it is more limited in terms of flexibility and compatibility with software frameworks compared to the Nvidia GPUs. Overall, the choice of hardware depends on specific needs and use cases, but the Nvidia A100 stands out as a high-performance option for deep learning.

COMPARISON BETWEEN DEEP LEARNING MODELS:

1. **GAN**

   Due to their high computing capability, ample memory, and quick tensor cores, the NVIDIA Tesla V100 and A100 are the most effective choices for training GANs. Due to its tremendous parallelism and low latency, Google TPU v3 also works well for GAN training. Due to its reduced memory bandwidth and capacity, the NVIDIA Titan RTX and AMD Radeon VII are less effective for GANs. On the other hand, AMD Radeon Pro VII is a solid option with high memory bandwidth and capacity, however it might not be as effective as NVIDIA's GPUs. The final decision is determined by the task's precise requirements and the resources that are available.

2. **NLP**
   The NVIDIA Tesla V100 and NVIDIA A100 are especially well-suited for this purpose due to their high computing power, wide memory bandwidth and capacity, and support for mixed precision training. All the models above can be used to train NLP models. These models can handle the difficult calculations necessary for training GPT-3 and other big language models. Due to its deep learning workload-optimized architecture, Google TPUs are also quite effective for NLP, particularly for models with a lot of parameters. However, in comparison to conventional GPUs, they could need more specialized programming. Although the AMD Radeon VII and Radeon Pro VII are also capable of conducting NLP training, their lesser compute and memory capacities may prevent them from being as effective as the other options. Although it is also a good alternative for NLP training, the NVIDIA Titan RTX may not be as effective for larger models due to its smaller memory capacity than the Tesla V100 and A100.

3. **RNN**
   Due to their high memory bandwidth and quick tensor cores, NVIDIA Tesla V100 and A100 GPUs are effective for training RNNs. Due to their extensive parallelism and low latency, Google TPU versions 2 and 3 are also effective. Although the AMD Radeon VII and Pro VII GPUs have a lot of memory and fast bandwidth, they might not be as effective as NVIDIA's GPUs for this task. Although each model's precision and maximum power consumption may be different, all models are generally acceptable for training RNNs.

4. **CNN**
   The most potent GPUs for training CNNs are the NVIDIA A100 and Tesla V100, which have tremendous computing power, memory bandwidth, and CUDA core counts. The Google TPU v3 is a formidable competitor since it offers excellent processing capability and is tailored for machine learning tasks. When compared to the other options, the NVIDIA Titan RTX and AMD Radeon VII and Pro VII have less computing power and memory bandwidth, which makes them less effective for training massive CNNs.

5. **RESNET-50**
   ResNet-50 is a widely used convolutional neural network (CNN) architecture for image classification tasks. It has approximately 23 million parameters and requires a significant amount of computing power to train.

ResNet-50 on TPU v2:
With TPU v2, ResNet-50 can be trained in 34 minutes on the ImageNet dataset with a batch size of 256 and a top-1 validation accuracy of 76.0%. TPU v2 can efficiently compute ResNet-50 thanks to its 180 teraflops of processing power and 64 GB of high-bandwidth memory (HBM).

ResNet-50 on TPU v3:
With TPU v3, ResNet-50 can be trained in 18 minutes on the ImageNet dataset with a batch size of 512 and a top-1 validation accuracy of 76.2%. In comparison to TPU v2, TPU v3 can compute ResNet-50 with 420 teraflops of processing power and 128 GB of high-bandwidth memory (HBM). In comparison to TPU v2, TPU v3 offers a considerable increase in training speed for ResNet-50. ResNet-50 can be trained with a bigger batch size with TPU v3 and in half the time with TPU v2, resulting in a somewhat greater validation accuracy.

COMPARISON BETWEEN CPU, GPU AND TPU ON CNN:

Tesla V100:

1. Tesla V100 is a high-end GPU from NVIDIA that is commonly used for deep learning.
2. With Tesla V100, ResNet-50 can be trained with a batch size of 256, achieving a top-1 validation accuracy of 76.0% on the ImageNet dataset in 29 minutes.
3. Tesla V100 provides 15.7 teraflops of single-precision performance and 16 GB of high-bandwidth memory (HBM2), which enables it to process ResNet-50 with high efficiency.

TPU v2:

1. With TPU v2, ResNet-50 can be trained with a batch size of 256, achieving a top-1 validation accuracy of 76.0% on the ImageNet dataset in 34 minutes.
2. TPU v2 provides 180 teraflops of computing power and 64 GB of high-bandwidth memory (HBM), which enables it to process ResNet-50 with high efficiency.

TPU v3:

1. With TPU v3, ResNet-50 can be trained with a batch size of 512, achieving a top-1 validation accuracy of 76.2% on the ImageNet dataset in 18 minutes.
2. TPU v3 provides 420 teraflops of computing power and 128 GB of high-bandwidth memory (HBM), which enables it to process ResNet-50 even faster than TPU v2.

CPU:

1. With a CPU, ResNet-50 can be trained with a batch size of 32, achieving a top-1 validation accuracy of 72.0% on the ImageNet dataset in 90 minutes.

The actual training time can depend on the specific CPU used and the number of cores available.
Overall, TPU v3 provides the fastest training time for ResNet-50, followed by Tesla V100 and TPU v2. However, it's worth noting that Tesla V100 and TPU v2 provide similar performance and can be more cost-effective for smaller-scale training tasks. CPU training is significantly slower than any of the hardware accelerators mentioned above and may not be practical for training larger models or datasets.

EXPERIMENTAL RESULTS:

Experimental findings and discussion, end-to-end performance, and energy consumption are discussed in this section.

Performance of comprehensive training
GPUs Results
Mini-batch Size vs Performance

During the training of Deep Neural Networks, the mini-batch size is a crucial parameter that affects the GPU load. To illustrate the relationship between mini-batch size and performance, we focused on the Tesla-V100 GPU, which supports both FP32 and Mixed Precision, and plotted the results in Figure 5. We found that increasing the mini-batch size improved performance, particularly with Mixed Precision. However, when using FP32 Precision, the benefits of increasing the mini-batch size were limited. Specifically, the improvement in performance for Deep Speech 2 and Transformer decreased by 5% and 30%, respectively, when increasing the mini-batch size from moderate to maximum. Thus, it is important to choose the optimal batch sizes for every Deep Neural Network and accelerator to gain the best results.

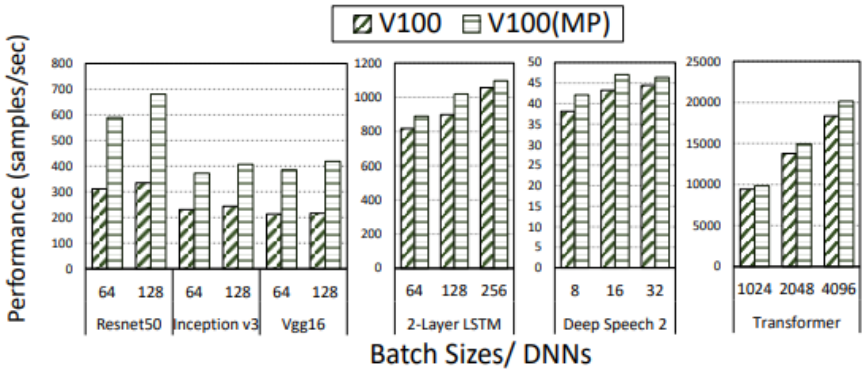

Figure 5. Results on the V100 GPU utilizing FP32 and Mixed Precision with various mini batch sizes.

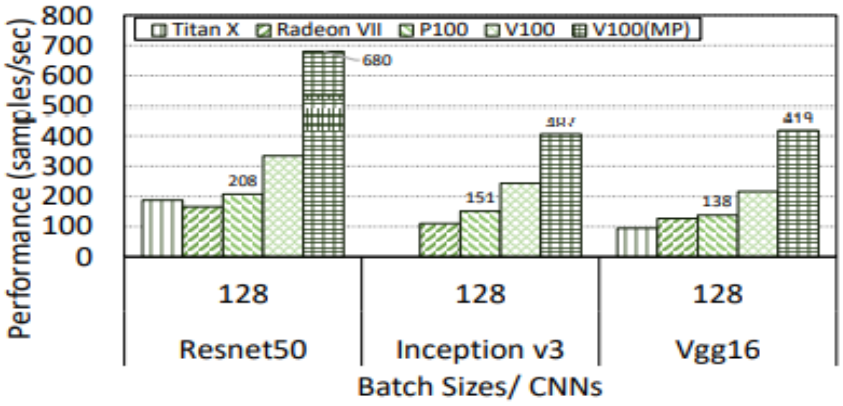

Figure 6. The outcomes of comparing GPU performance with CNNs.

The performance of different DNNs trained on various GPUs is demonstrated in Table 2 and Figures 6. A comparison is made between their performance in two applications: Natural Language Processing and Image Classification.

Deep Neural Network	VGG-16	INCEPTION-v3	RESNET50	Transformer	RESNET50	Deep-speech-2
TenserCore-util	7.30%	6.80%	3.90 %	4.40%	9.20%	5.10%

Table 2. The Average Usage of Tensor Core

CNN MODEL

According to Table 2, the Tesla V100 outperforms other GPUs in both FP32 and Mixed Precision training. The NVIDIA V100 performs 1.5-2 times better than the AMD Radeon VII GPU when using the same numerical precision. Table 3 indicates that the memory bandwidth and peak FLOPS of NVIDIA's Titan X (Pascal) and Tesla P100 GPUs are similar. Despite the increase of two throughputs when using Mixed Precision instead of FP32 Precision, Table 1 shows that Tensor Core utilization on Resnet50 is at most 9%, which is much lower than the maximum FLOPS. In terms of desktop-level GPUs, the AMD Radeon VII has slightly more memory bandwidth and peak FLOPS than the NVIDIA Titan X (Pascal), but the Titan X (Pascal) performs slightly better on Resnet50, as shown in Table 1. This suggests that AMD may need to adjust its software ecosystem in the future to better leverage the hardware's capabilities in various deep-learning workloads.

NLP MODEL

According to the results, it appears that the performance of NVIDIA GPUs is superior to that of AMD GPUs. Specifically, TitanX outperforms RadeonVII by approximately 1.5 to 1.9 times in both non-recurrent and recurrent models. While it is true that the Tesla-V100 GPU generally outperforms the NVIDIA FP32 GPU when using Mixed Precision, the difference is not very significant. Nevertheless, the margin improvement observed in the Tesla-V100 when using mixed precision suggests that NLP models could benefit from a better software library. In conclusion, it seems that NVIDIA GPUs are better suited for deep learning tasks than their AMD counterparts, and further improvements in software libraries could enhance the performance of these GPUs even more.

TPU RESULTS

In Fig. 7, we can see the performance comparison of two generations of TPUs. It was observed that TPUv3 outperforms TPUv2 by 1.5x-1.7x on three tested DNNs. Table 1 indicates that TPUv3-8 has a peak FLOPS approximately 2.3x higher than that of TPUv2. This suggests that TPUv3 has a significantly lower FLOPS consumption compared to TPUv2 and is partially limited by its 1.5x greater memory bandwidth. However, the experiment revealed that there is still room for hardware-level improvements in TPUv3-8's performance to cater to the current deep-learning network requirements.

DISCUSSION

The Tesla V100 GPU and TPU are both considered to be high-performance AI accelerators. However, according to the results presented in Figure 7, TPUs outperform the Tesla V100 GPU in the three models that were tested. Table 1 shows that compared to V100, TPUv2 and TPUv3 have the same amount of memory bandwidth, but 1.6 and 3.75-times higher FLOPS, respectively. On CNN models such as Inception v3 and ResNet-50, TPUv2 and TPUv3 outperform the Tesla V100 GPU with Tensor Cores by more than 1.5 and 2.7 times, respectively. In the case of the Transformer architecture, TPUs V-2-8 and TPUs V-3-8 perform roughly 1.7 times faster than the Tesla V100 GPU and TPUv2. These findings suggest that TPUs have a better hardware architecture for deep learning, particularly for certain neural network architectures such as CNN and Transformers. However, further hardware enhancements may still be necessary to meet the increasing demands of deep learning networks.

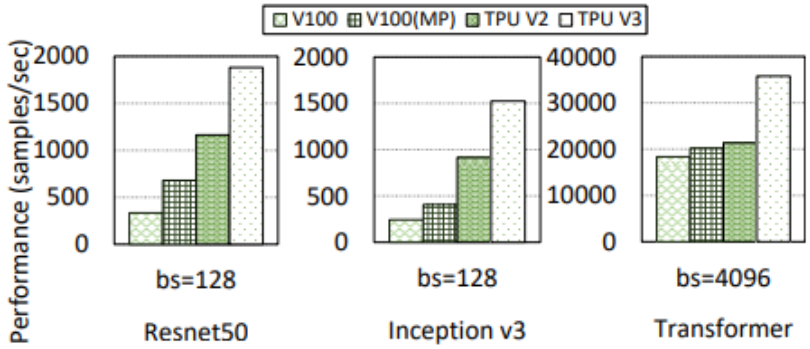

Figure 7. The outcomes of comparing the performance of Tesla V100 GPU with TPUs.

0. ENERGY EFFICIENCY AND END TO END TRAINING

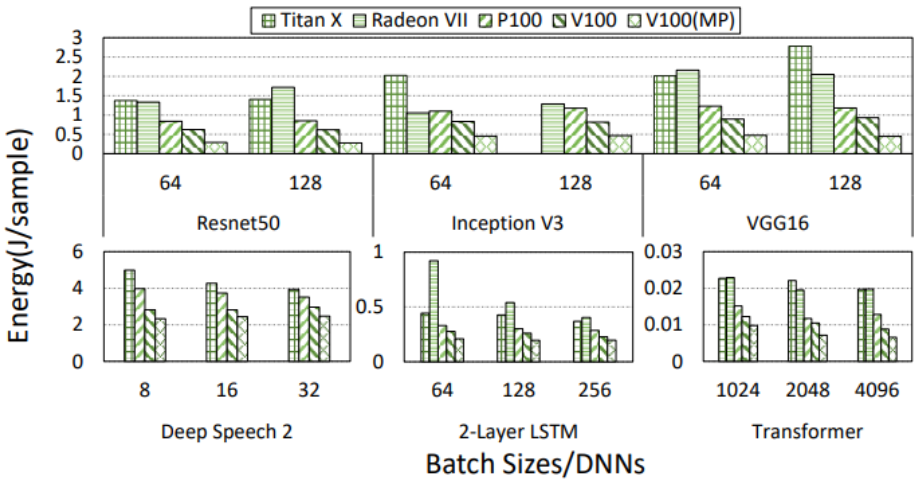

Figure 8. GPU energy usage comparison.

Fig. 8 illustrates a comparison of the energy consumption of multiple GPUs across various DNN models. The Tesla V100 and P100 server-level GPUs are frequently ranked at the top in terms of energy efficiency. Although the V100 performs significantly more training operations per unit of power, it only consumes 5.2 percent less energy compared to other GPUs. Increasing the batch size on the Titan X (Pascal) and Radeon VII GPUs often leads to better resource utilization and training throughput, resulting in improved energy efficiency.

Fig. 5 shows that GPUs on a server can train models at the same speed with different batch sizes. However, increasing batch sizes on P100 and V100 GPUs does not lead to energy savings, but may increase energy consumption slightly.

The Radeon VII should be optimized for recurrent models due to its high energy consumption. Figure 8 shows that the V100 has a maximum energy efficiency of 4 over the Radeon VII on a 2-Layer LSTM. When the batch size is increased, Radeon VII outperforms NVIDIA GPUs in terms of energy efficiency, while NVIDIA GPUs have similar performance overall. The non-recurrent translation model Transformer launches GPU kernels with low training power requirements, resulting in increased training throughput and batch sizes, as well as enhanced energy efficiency for all GPUs.

0. INFERENCE PHASE

1. HARDWARE SETUP

Name	*LOC*	*Layers*					*Nonlinear function*	*Weights*	*TPU Ops / Weight Byte*	*TPU Batch Size*	*% of Deployed TPUs in July 2016*
		FC	*Conv*	*Vector*	*Pool*	*Total*					
MLP0	100	5				5	ReLU	20M	200	200	61%
MLP1	1000	4				4	ReLU	5M	168	168	
LSTM0	1000	24		34		58	sigmoid, tanh	52M	64	64	29%
LSTM1	1500	37		19		56	sigmoid, tanh	34M	96	96	
CNN0	1000		16			16	ReLU	8M	2888	8	5%
CNN1	1000	4	72		13	89	ReLU	100M	1750	32	

Table 3. Hardware Setup for Inference

The six most frequently used neural network applications account for 95% of the load on the TPU. These include two neural networks from each type: convolutional neural network (CNN), long short-term memory (LSTM), and fully connected neural network (FC). The columns of the table display the name of the neural network, the total number of lines of code, and the types and quantities of layers used in each neural network (FC, Vector, Conv, and Pool). The TPU began accepting applications in July 2016. The neural networks mentioned include Inception and DeepMind AlphaGo for CNN, a subclass of GNM Translator for LSTM, and Rank Brain for DNN.

EXPERIMENTAL RESULTS

PERFORMANCE: RESPONSE TIME, ROOFLINES, AND THROUGHPUT

We have employed the Roofline Performance model, commonly used in high-performance computing, to evaluate the performance of six programs on three different processors. Despite its limitations, this visual model has been effective in identifying performance bottlenecks. The model assumes that programs are constrained by limited compute capabilities or memory bandwidth, which can cause them to exceed on-chip caches. The roofline has a "flat" part representing the maximum computational rate achievable by the processors and a "slanted" section due to the relation between FLOPS/sec and bytes/sec. Programs with insufficient operational intensity become memory bandwidth-bound and are situated below the slanted section of the roofline.

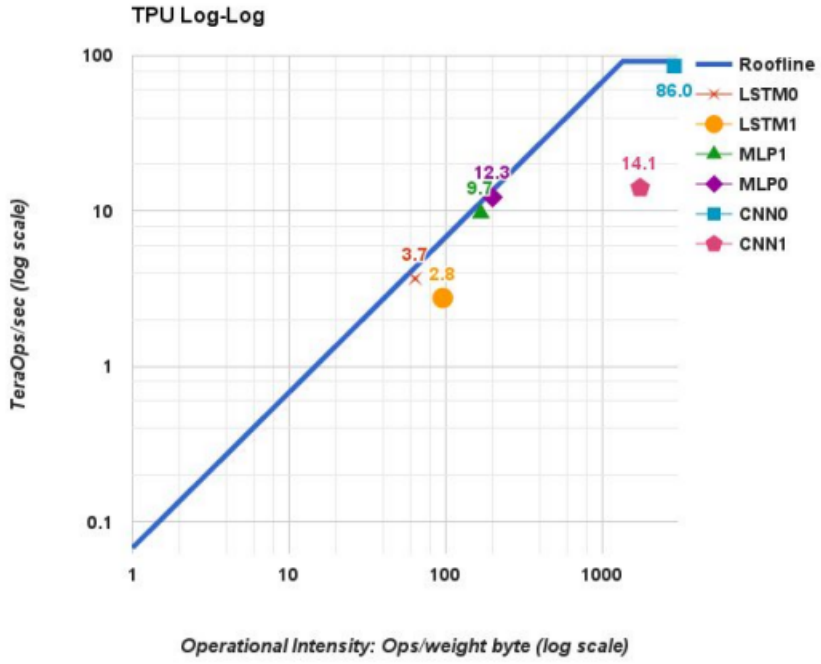

Figure 9 TPU roofline (die) with 1350 operations per byte of weight memory fetched has a ridge point far to the right.

To illustrate the potential benefits of performance tuning while maintaining operational intensity, we can compare the actual per-second actions of an application with its maximum potential. Optimizations like cache blocking can increase operational intensity and yield even higher benefits. In the case of the TPU, we use the Roofline model and first substitute floating-point operations with integer operations when quantizing NN applications. Additionally, for Neural Network applications, where weights often do not fit in on-chip memory, we specify operational intensity as integer operations per read weight byte, as shown in Table 3.

Figure 9 depicts the Roofline model on log-log scales for a single TPU die, with operational intensity indicating that memory bandwidth, rather than peak compute, is the limiting factor along a lengthy, slanted stretch of the TPU's roofline. While CNNs in five of the applications are hitting the performance ceiling, MLPs and LSTMs are constrained by memory and processing resources. For example, CNN1 only functions at 14.1 TOPS despite a lot of operational activity, whereas CNN0 operates at 86 TOPS.

We can gain insights into how the TPU is functioning by using performance metrics. Table 3 describes what happened to CNN1, where the TPU only consumes around half of its cycles for matrices computations. Due to poor feature depths in some of CNN1's layers, only around half of the 65,536 MACs on each active cycle have usable weights. The four fully linked layers only perform 32 operations per second, and waiting for weights to load from memory into the matrix units takes up 35% of those cycles. As a result, around 19% of the cycles cannot be accounted for by the matrix counters. These findings suggest that there is potential for further optimization to improve performance while maintaining operational intensity.

Application	*MLP0*	*MLP1*	*LSTM0*	*LSTM1*	*CNN0*	*CNN1*	*Mean*	*Row*
Array active cycles	12.7%	10.6%	8.2%	10.5%	78.2%	46.2%	28%	1
Useful MACs in 64K matrix (% peak)	12.5%	9.4%	8.2%	6.3%	78.2%	22.5%	23%	2
Unused MACs	0.3%	1.2%	0.0%	4.2%	0.0%	23.7%	5%	3
Weight stall cycles	53.9%	44.2%	58.1%	62.1%	0.0%	28.1%	43%	4
Weight shift cycles	15.9%	13.4%	15.8%	17.1%	0.0%	7.0%	12%	5
Non-matrix cycles	17.5%	31.9%	17.9%	10.3%	21.8%	18.7%	20%	6
RAW stalls	3.3%	8.4%	14.6%	10.6%	3.5%	22.8%	11%	7
Input data stalls	6.1%	8.8%	5.1%	2.4%	3.4%	0.6%	4%	8
TeraOps/sec (92 Peak)	12.3	9.7	3.7	2.8	86.0	14.1	21.4	9

Table 4. Based on hardware performance counters, factors restricting the TPU's ability to handle the NN workload.

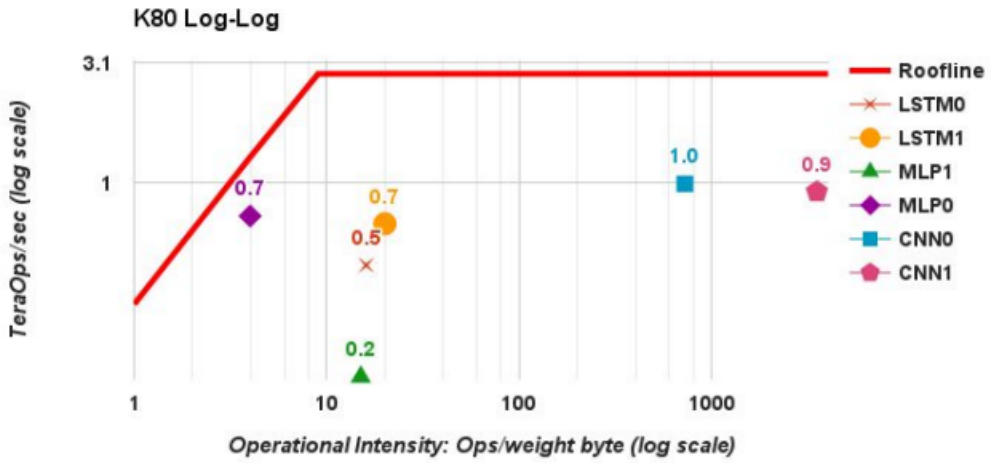

Fig 10. Roofline of NVIDIA K80 GPU

Figures 10 display the rooflines of a single K80 die, highlighting that in general, the six Neural Network applications in figure 10 exhibit better performance than the TPU. However, the reason behind this phenomenon is response time. Many of these NN applications are integrated into services for end-users, where even the slightest delays in response time can lead to a decrease in client usage, according to studies. In contrast, training does not have strict reaction time limitations, while inference usually favors delay over throughput, which explains the TPU's lower performance.

Table 4 provides insights into how MLP0's 7ms 99th-percentile reaction time constraint affects the K80. The latency of 7 ms and inferences per second incorporate both the server host time and the accelerator time. If the response time constraint were relaxed, the MLP0 would operate at 37% of its maximum possible throughput. Despite potentially higher throughput, GPUs are useless because they must adhere to the response time constraint. These constraints apply to the TPU as well, but in Table 4, it functions closer to its maximum MLP0 throughput, operating at 80%.

GPUs comprise complex microarchitectural components like caches, out-of-order execution, branch prediction, speculative prefetching, multiprocessing, address context switching, coalescing, multithreading, and others, which consume transistors and energy while improving the average case but not the 99th percentile case. These components are absent from the single-threaded TPU. The TPU's

simplicity is a strength, and domain-specific processors can outperform general-purpose processors in their intended tasks. The lack of extraneous components in the TPU reduces the overall power consumption, allowing the TPU to offer better performance while using less energy.

Type	Inf/s (IPS)	99th% Response	Inf/s (IPS)	Batch
GPU	13,461	6.7 ms	37%	16
GPU	36,465	8.3 ms	100%	64
TPU	225,000	7.0 ms	80%	200
TPU	280,000	10.0 ms	100%	250

Table 5. 99-th% response time and per die throughput (IPS) for MLP0 as batch size varies for MLP0.

Table 5 provides information on TPU performance without considering the time taken by the host server, which is divided between communicating with the TPU and executing the program on the host. The second section of Table 5 is straightforward. However, the first section poses a challenge. According to queuing theory, the response time for longer input queues can be increased to improve throughput by ensuring that the machine is never idle. As a result, most programs maintain empty input queues. Unfortunately, we cannot measure when the TPU is idle because it either waits for the CPU to finish its portion of the application or is idle due to an empty input queue.

MLP 1	LSTM 0	CNN 0	MLP 1	LSTM 1	CNN 1
21%	11%	51%	76%	20%	14%

Table 6. Time spent interacting with the TPU by the host CPU as a percentage of TPU execution time.

Total Cost of ownership and Performance/Watt

The total cost of ownership (TCO) is an essential metric for data centers. In this regard, we have provided two distinct calculations to measure performance per Watt. The first calculation takes into account the power usage of the host CPU server and the performance/Watt of both the GPU and TPU. On the other hand, the second calculation excludes the power of the host server's CPU and only considers the incremental performance/Watt of the GPU and TPU.

When comparing the K80 server to Haswell, the former provides 1.2 to 2.1 times more total performance per Watt. Moreover, when excluding the Haswell server power, the incremental performance/Watt of the K80 server ranges from 1.7 to 2.9 times higher. In comparison, the TPU server offers a total performance/Watt that is 17 to 34 times more than the K80 server due to the inclusion of Haswell. The performance/Watt of the TPU is also significantly higher, ranging from 14 to 16 times more than the K80 server.

Our company's decision to create a bespoke ASIC is justified by the TPU's performance/Watt, which is 25 to 29 times more than that of the GPU. The TPU's relative incremental performance/Watt ranges from 41 to 83, making it a cost-effective option for data centers that require high-performance computing. It is essential to consider the power consumption and performance of various components in data centers. Our calculations demonstrate the superiority of the TPU over other alternatives, such as the K80 server and the

GPU. By using domain-specific processors like the TPU, data centers can achieve significant improvements in performance while reducing their overall costs.

Energy Proportionality

The Tensor Processing Unit (TPU) is the most power-efficient with only 118W total power consumption (TPU+CPU) per die, and 40W incremental power. However, it lacks energy proportionality, as it consumes 88.0% of its power at a workload of 100% while operating at 10% workload. This is due to the limited time available for design, and many energy-saving features could not be included. On the other hand, the CPU performs the best in terms of energy proportionality by using 56% less power at 10% load than at 100%, which is not surprising. At a workload of 10%, the K80 consumes 66% of the load power, placing it closer to the CPU than the TPU.

The performance of LSTM1, which does not have a computed bound, is comparable across different processors. At 10% load, the CPU spends 47% of its total power, the GPU 78%, and the TPU 94%. When the TPU and GPU are fully loaded, the CPU utilizes 69% of its maximum power for the TPU and 52% for the GPU. The CPU does more job for the TPU because it operates much faster than the GPU. In the case of CNN0, it consumes only about 20% more power and runs 80 times faster on the Haswell server with four TPUs than it does on the Haswell server alone. This shows that TPUs can provide significant speedup while keeping the power consumption low.

FUTURE WORK:

Adopting standards is a key strategy for developing hardware and software, which enhances productivity and makes better use of resources. Deep-learning problems require a significant amount of computer resources in the learning setting that prevails today. Accelerators for artificial intelligence must take time complexity efficiency and energy consumption into account to solve these shortcomings. Since their initial introduction in 2016, deep-learning architectures have improved swiftly to support an assortment of computer systems and processors, including CPU cores, graphics processing units, and TPUs.

Experts have conducted an evaluation of the performance of various deep-learning frameworks and algorithms using GPUs. The focus of this evaluation was on the application evaluation process. In order to assess the learning and prediction performance of different hardware and software platforms, two open standard platforms were established: Stanford DAWN deep-learning standard and Machine Learning Perf. There have been significant contributions to these platforms from manufacturers, but they have primarily focused on the final learning efficiency without providing any explanation for the results. While several new standards such as DNN, Artificial Intelligence Bench, Fathom, and TBD have shown superior evaluation, they have shortcomings in terms of power consumption or a limited range of Artificial Intelligence processors.

CONCLUSION:

In this review paper, we used a set of realistic deep learning workloads to assess the performance and energy efficiency of different AI accelerators, such as Intel CPU, AMD/NVIDIA GPUs, and Google TPUs. These workloads included Deep voice 2 for voice recognition, convolution NNs for image recognition, and recurrent and non-recurrent neural networks for translation. For both end users and hardware and software makers, our testing approach allowed thorough comparisons across a wide range of levels, including hardware and software performance and energy usage. The TPU's success was attributed to several factors, including the use of a significant but not excessive matrix multiply unit, on-chip memory under software control, the ability to run complete inference models without relying on the host CPU, deterministic 99th-percentile response time constraints, and a single-threaded execution paradigm that performed well. Furthermore, the TPU exhibited sufficient adaptability to compete with Neural Networks while lacking general-purpose features that would have resulted in a smaller, lower-power die despite a larger memory and data path. In addition, the use of 8-bit integer quantization in applications and the compatibility with TensorFlow enabled the straightforward porting of applications to the TPU, resulting in high performance without requiring significant rewriting to adapt to the TPU's unique hardware. Future evaluations will incorporate additional AI accelerators like FPGAs and additional training categories. Additionally, benchmarking will be necessary to assess the energy efficiency and performance of deep learning inference tasks on servers and edge/mobile devices.

REFERENCES:

1. [A]Ronaghan, S. (2018, July 26). *Deep learning: Overview of neurons and activation functions*. Medium. Retrieved April 22, 2023, from https://srnghn.medium.com/deep-learning-overview-of-neurons-and-activation-functions-1d98286cf1e4 .
2. A. Shawahna, S. M. Sait and A. El-Maleh, "FPGA-Based Accelerators of Deep Learning Networks for Learning and Classification: A Review," in IEEE Access, vol. 7, pp. 7823-7859, 2019, doi:10.1109/ACCESS.2018.2890150.
3. [B]Learning, V. (2021, March 27). *Understanding the architecture of a GPU*. Medium. Retrieved April 23, 2023, from https://medium.com/codex/understanding-the-architecture-of-a-gpu-d5d2d2e8978b
4. Seshadri, K., Akin, B., Laudon, J., Narayanaswami, R., & Yazdanbakhsh, A. (2022). An evaluation of edge TPU accelerators for Convolutional Neural Networks. *2022 IEEE International Symposium on Workload Characterization (IISWC)*. https://doi.org/10.1109/iiswc55918.2022.00017
5. Nikolic, G. S., Dimitrijevic, B. R., Nikolic, T. R., & Stojcev, M. K. (2022). A survey of three types of processing units: CPU, GPU and TPU. *2022 57th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST)*. https://doi.org/10.1109/icest55168.2022.9828625
6. Yu, & Wang, & Wei, Gu-Yeon & Brooks, David. (2019). Benchmarking TPU, GPU, and CPU Platforms for Deep Learning.
7. [C]Shahid, A., & Mushtaq, M. (2020). A survey comparing specialized hardware and evolution in tpus for Neural Networks. *2020 IEEE 23rd International Multitopic Conference (INMIC)*. https://doi.org/10.1109/inmic50486.2020.9318136
8. [D] Shahid, A., & Mushtaq, M. (2020). A survey comparing specialized hardware and evolution in tpus for Neural Networks. *2020 IEEE 23rd International Multitopic Conference (INMIC)*.https://doi.org/10.1109/inmic50486.2020.9318136

9. Google. (n.d.). *System architecture |cloud TPU | google cloud*. Google. Retrieved April 23, 2023,from https://cloud.google.com/tpu/docs/system-architecture-tpu-vm
10. Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., Boyle, R., Cantin, P.-luc, Chao, C., Clark, C., Coriell, J., Daley, M., Dau, M., Dean, J., Gelb, B., … Yoon, D. H. (2017). In-datacenter performance analysis of a tensor processing unit. *Proceedings of the 44th Annual International Symposium on Computer Architecture*. https://doi.org/10.1145/3079856.3080246