Comparison study of GPU and CPU for deep learning methods

[1] **Charulatha.K**
PG Scholar, Department of Computer Science and Engineering,
Panimalar Engineering College, Chennai, India.

[2] **Rajendiran.M**
Professor, Department of Computer Science and Engineering,
Panimalar Engineering College, Chennai,India

***Abstract*—** Applications for deep learning and machine learning have grown significantly in recent years. The application of ML and DL algorithms can produce relevant results from the vast quantity of information being generated online. It has been simple for us to implement these algorithms due to hardware resources and opensource libraries. One of the most popular frameworks for implementing ML projects is Tensorflow with PyTorch. With the help of such frameworks, we can track the operations that are carried out on the CPU and GPU to examine resource allocations and use. The time and memory allocation of the CPU and GPU during Pytorch-based deep neural network training is discussed in this research. The analysis of this research demonstrates that the GPU runs deep neural networks faster than the CPU. Few are the notable GPU gains over CPU for simpler networks.

Keywords—CPU, GPU, Memory usage

I. Introduction

Making a computer system learn without explicit programming is known as machine learning. To produce useful information and make conclusions, enormous amounts of data are needed. "A computer program is said to learn from experience E with regard to some class of task T and performance P," according to the official and scientific definition of machine learning. if it becomes more effective at activities in T as measured by P as it gains experience E." The form of machine learning called deep learning frequently uses neural networks to examine data and make decisions. Artificial neural network models replicate how the human brain functions. Due to the massive quantity of matrix and algebraic operations, deep neural networks require a lot of calculations and hardware resources to function properly.

The two processing units that are frequently utilized to process ML and DL models are the central processing unit (CPU) and the graphics processing unit (GPU). While CPU is not utilized for parallel calculation, GPU is specifically designed for it [7]. There are two frameworks that provide abstraction for difficult mathematical computations in the field of deep learning they are Pytorch and Tensorflow. When enormous amounts of data are created, the requirement for GPUs for deep learning network training increases.

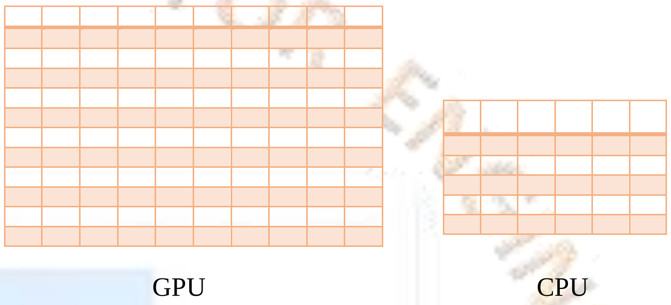


Figure 1: GPU and CPU

The normal number of cores in a conventional CPU is 4–5, which limits the number of threads that may be processed. As seen in Figure 1, GPUs, on the other hand, contain a huge number of tiny cores that can support many parallel computing threads. NVIDIA A6000, as an example, has 10752 CUDA cores [11]. Millions of computations are made as part of a deep learning system to train and infer, which requires a GPU for quick processing [16].

II. Problem Statement

In this paper, we will examine the time and memory requirements for mathematical operations on CPU and GPU. Running-time analysis of time and memory helps to optimise network activities, resulting in quicker execution and inference. We are able to see network graphs and determine the execution times of individual operations using the Pytorch Profiler API and NVIDIA commands, as well as the current state of memory consumption.

III. Scope

The main goal of this paper is to find out how CPU and GPU [8] activities occur during training machine learning and deep learning models. The deep learning models will not be the main focus; rather, the CPU and GPU time and memory profiling will. The profiling will be produced using a deep learning model that makes use of the Tensorflow and Pytorch profiler. Each training stage involves tracing in order to collect data on memory and timing. Then, the resource usage of the CPU and GPU are compared.

IV. BACKGROUND

A. Tensorflow:

TensorFlow is an free-source software library for computation numerical operations and developed by the Google Brain team.
TensorFlow allows developers and researchers to create and train machine learning models using a variety of high-level APIs in Python.
It is a flexible for building and deploying machine learning models at scale, on a variety of platforms including CPUs, GPUs, and TPUs.

B. Pytorch:

PyTorch is an free-source machine learning framework which was developed by Facebook's AI research team. Like TensorFlow, PyTorch provides a platform for building and training ML models, with a focus on flexibility and ease of use.
It features a dynamic computational graph that allows developers to build and modify their models on the fly, as well as a powerful autograd system that can automatically compute gradients for any differentiable function[9].

C.Profiler

A profiler is a tool that helps developers to measure and analyze the performance of their software code
A profiler works by monitoring the execution of a program and collecting data about the code's performance, including metrics such as execution time, CPU usage, memory usage, and more.

D.CUDA

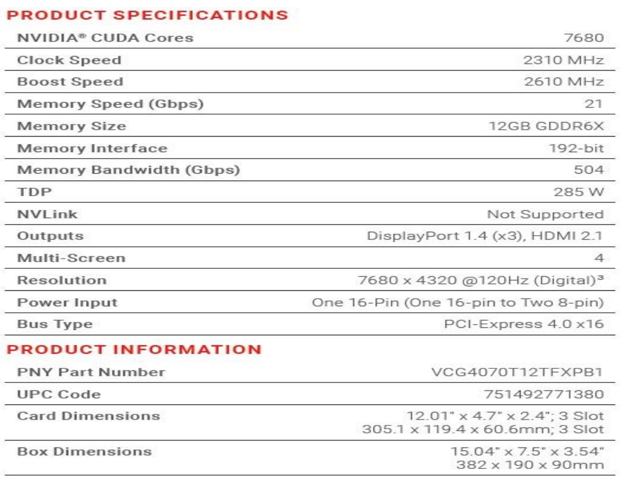
CUDA is widely used in fields such as scientific computing, machine learning, and high-performance computing[19], where the massive parallelism of GPUs can be harnessed to accelerate computation[5]. It has become a popular choice for building deep learning models using frameworks such as TensorFlow, PyTorch, and Keras.

E.CPU

A CPU (Central Processing Unit) is responsible for executing instructions that control the operation of the computer, including running applications, managing memory, and handling input/output operations[18].

F.GPU

A GPU (Graphics Processing Unit) is a dedicated processor mainly developed to handle the complex mathematical operations essential for rendering images, video, and other graphical content. GPUs are used in many applications[10] which includes gaming, scientific computing, and machine learning. Various GPUs models are available. In this research, we have used 2 NVIDIA 7680 [20] GPU named GPU:0 and GPU:1 in this experiment whose specification is shown is Figure 2.[17]



PRODUCT SPECIFICATIONS	
NVIDIA® CUDA Cores	7680
Clock Speed	2310 MHz
Boost Speed	2610 MHz
Memory Speed (Gbps)	21
Memory Size	12GB GDDR6X
Memory Interface	192-bit
Memory Bandwidth (Gbps)	504
TDP	285 W
NVLink	Not Supported
Outputs	DisplayPort 1.4 (x3), HDMI 2.1
Multi-Screen	4
Resolution	7680 x 4320 @120Hz (Digital)[3]
Power Input	One 16-Pin (One 16-pin to Two 8-pin)
Bus Type	PCI-Express 4.0 x16
PRODUCT INFORMATION	
PNY Part Number	VCG4070T12TFXPB1
UPC Code	751492771380
Card Dimensions	12.01" x 4.7" x 2.4"; 3 Slot 305.1 x 119.4 x 60.6mm; 3 Slot
Box Dimensions	15.04" x 7.5" x 3.54" 382 x 190 x 90mm

Fig 2: NVIDIA specifications

F.Neural Network

A neural network is a type of machine learning algorithm, that works in an organized form to perform a exact task, such as image recognition or natural language processing. Neural networks are mainly well-suited for works that involve recognizing patterns in huge amounts of data.

G.Artificial Intelligence:

Artificial intelligence (AI) is a field of computer science that focuses on the development of algorithms which leads to the system can perform tasks without human intervention, such as perception, reasoning, learning, and decision-making. The goal of AI is to build machines that can function autonomously, adapt to changing environments, and improve their performance over time. AI systems can process vast amounts of data, learn from their experiences, and adapt to changing circumstances.

The aim of AI is to build machines that can work and make decisions autonomously, without human intervention. AI systems use machine learning algorithms to learn from data given to the system and improve their performance over time. These algorithms are trained on bulky amounts of data to identify the relationships between data that are used to make predictions or decisions. Other AI techniques include rule-based systems, expert systems, and evolutionary algorithms.

H.Machine Learning

Machine learning is a subfield of artificial intelligence which develops algorithms that learns from the data and make predictions or decisions based on data [13]. The main aim of machine learning is to build systems that can automatically improve their performance on a specific task over time, without being explicitly programmed to do so.

In machine learning, a model will be trained on a set of input training data with respective output labels or predictions. The model then uses this training data to learn patterns and relations within the data that can be used to make decisions on new, unseen data (testing data).

I.Deep Learning

Deep learning is a subset of machine learning that uses neural networks with multiple layers, allowing for the learning of increasingly abstract features of the data. The term "deep" refers to the depth of the neural network, which can have many layers and millions of parameters.

Deep learning models are trained with huge amount of data and it can learn features from the data that are relevant to the task at hand. The layers in the neural network are arranged in a hierarchy, with each layer learning features that are increasingly abstract and complex. The output of the final layer is used to make decisions based on the input data[3].

Deep learning requires significant computational resources, and is typically trained on powerful GPUs or specialized hardware such as TPUs (Tensor Processing Units).

J.TPU

TPU stands for Tensor Processing Unit, which is a specialized hardware accelerator designed by Google specifically for deep learning tasks. TPUs are designed to work with TensorFlow, a popular deep learning framework, and are optimized for high-speed matrix operations, which are commonly used in deep learning algorithms.

TPUs are built using custom-designed chips that are improved for matrix multiplication and other mathematical operations commonly used in deep learning. They are able to complete these operations much faster and more efficiently than CPUs or GPUs, which can significantly speed up the training of deep learning models.

In addition to their high performance, TPUs are also designed to be highly scalable, allowing for the parallel processing of huge amounts of data. This makes them ideal for training large, complex models, such as those used in natural language processing or image recognition.
Google offers access to TPUs through its cloud computing platform, Google Cloud Platform, making them accessible to researchers and developers around the world.

K.CPU

CPU stands for Central Processing Unit. It is responsible for performing the computational tasks. It is frequently denoted to as the "brain" of a computer[15].

The CPU is responsible for execution of instructions and commands stored in memory and processing data. CPUs can also have numerous cores, which allow them to complete several tasks instantaneously and improve overall performance.

V. METRICS

A.GPU Utilization

GPU utilization refers to the amount of work that a GPU is doing at a given time. GPUs are commonly used for computationally intensive tasks, such as deep learning, image processing, and scientific simulations, because they can perform many calculations in parallel, which can significantly speed up the computation time compared to a CPU[2].

GPU utilization is typically measured as a percentage, with 100% utilization indicating that the GPU is fully utilized and all of its processing resources are being used[12]. Lower utilization percentages indicate that the GPU is not being fully utilized and may be able to perform additional work.
Monitoring GPU utilization is important for optimizing performance and identifying potential bottlenecks in a system.

If a GPU is not being fully utilized, it may indicate that there is a bottleneck elsewhere in the system, such as in the CPU or memory[4]. On the other hand, if a GPU is consistently at 100% utilization, it may indicate that the workload is too heavy and additional GPUs may be needed to handle the workload.

GPU utilization can be monitored using various tools, such as the NVIDIA System Management Interface (nvidia-smi) or GPU-Z. These tools provide real-time information on GPU usage, including GPU temperature, memory usage, and power consumption

B.GPU memory access

GPU memory access and utilization refer to how a GPU uses its memory to store and access data during computations. GPUs have their own dedicated memory called Graphics Random Access Memory (GRAM) or Video Random Access Memory (VRAM), which is used to store data and instructions for processing.

GPU memory access refers to how the GPU accesses and reads/writes data from/to its memory during computations. The GPU reads data from memory and performs calculations on that data in parallel [6]. After the computation is complete, the results are written back to the GPU memory.

GPU memory utilization refers to how much of the GPU memory is being used at a given time. Just like GPU utilization, memory utilization is measured as a percentage, with 100% utilization indicating that the GPU memory is fully used.

High GPU memory utilization can be a problem because it can lead to slower performance or even crashes if there is not enough memory available for the computation. This is especially true for deep learning models, which often require large amounts of memory to store the neural network weights and data.

To optimize GPU memory utilization, developers can use techniques such as data batching, which involves breaking up the data into smaller batches that can fit into the GPU memory, and data parallelism, which involves splitting the data and processing it simultaneously on multiple GPUs[14].

Monitoring GPU memory utilization is important for optimizing performance and preventing memory-related issues. Developers can use tools such as nvidia-smi or GPU-Z to monitor GPU memory usage in real-time and adjust their code or system configurations accordingly.

C.GPU power usage

GPU power usage refers to the amount of power consumed by a GPU during its operation. GPUs are known for their high power consumption due to their heavy computational workload and parallel processing capabilities.

The power consumption of a GPU depends on various factors such as the number of CUDA cores, clock speed, memory bandwidth, and the workload being executed. Higher-end GPUs typically consume more power than lower-end ones due to their higher computational capabilities.

GPU power usage can be monitored using tools such as NVIDIA System Management Interface (nvidia-smi), which provides real-time information on GPU power usage, temperature, and other metrics. High GPU power usage can lead to increased heat generation, which can cause thermal throttling or even damage to the GPU if it is not properly cooled.

To optimize GPU power usage, developers can use techniques such as data batching, which involves breaking up the data into smaller batches that can be processed more efficiently, and precision tuning, which involves selecting the appropriate data type for the computation to reduce memory usage and power consumption.

Additionally, using energy-efficient GPUs or implementing power management strategies, such as dynamic voltage and frequency scaling, can help reduce GPU power consumption and improve the overall energy efficiency of a system

D.Bias update

In ML and DL, bias update refers to the process of adjusting the bias term in a neural network during training. A bias term is a scalar value that will be added to the output of a neuron before it is passed through an activation function.

During training, the neural network learns to adjust the weights and biases of its neurons to minimize the error between its predicted outputs and the actual outputs. The bias term is one of the parameters that can be adjusted to improve the performance of the network.

The bias update process typically involves calculating the gradient of the loss function with respect to the bias term using backpropagation. The gradient indicates the direction and magnitude of the change required to reduce the loss, and it is used to update the bias term using an optimization algorithm such as stochastic gradient descent (SGD).

The frequency of bias updates and the learning rate used to update the bias term can significantly impact the performance of the network. Updating the bias term too frequently or with a huge learning rate can cause the network to converge to a suboptimal solution, while updating it too infrequently or with a small learning rate can result in slow convergence and longer training times.

Bias update is an essential component of neural network training, and optimizing this process is critical to achieving high performance and accuracy in machine learning and deep learning applications.

E.Accurarcy

In machine learning (ML) and deep learning (DL), accuracy is a measure of how well a model is able to correctly predict the outcomes of a task. It is a commonly used metric to evaluate the performance of a model on a specific task, such as image classification or natural language processing.

Accuracy is typically defined as the ratio of the number of correct predictions made by the model to the total number of predictions made. It is expressed as a percentage, with a value of 100% indicating that the model has made all the correct predictions.

For example, in image classification, accuracy is calculated by comparing the predicted labels to the actual labels of a set of images. The accuracy of the model is the percentage of images that were classified correctly.

While accuracy is a main important metric for calculating the performance of a model, it is not constantly the finest metric to use, especially in cases where the classes are imbalanced. In such cases, a model that predicts the majority class for all instances may attain high accuracy, even though it is not very useful in practice.

F.Throughput

In machine learning and deep learning applications, throughput refers to the rate at which a model can process input data and generate output predictions. It is a critical performance metric for applications such as real-time image and speech recognition, where the system needs to process large volumes of data quickly and efficiently.

The throughput of a machine learning or deep learning model can be pretentious by numerous issues such as the model architecture, input data size, and hardware infrastructure. The amount of operations essential to process each input, the amount of memory required to store the model and input data, and the speed of the processors and other hardware components all contribute to the throughput of the system.

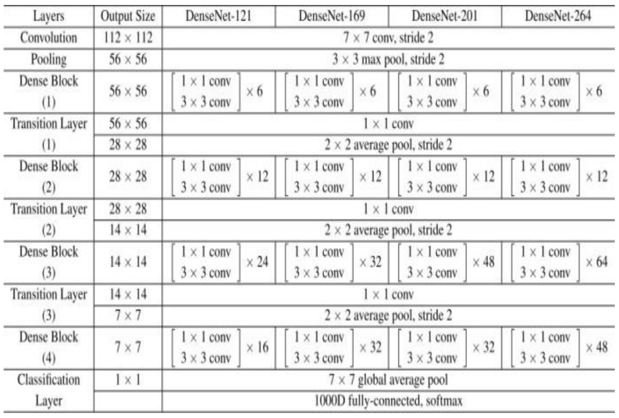
To optimize the throughput of a machine learning or deep learning system, developers often use techniques such as model quantization, which involves reducing the precision of the model parameters to reduce the memory requirements and speed up the processing time. They may also use specialized hardware such as GPUs or TPUs to speed up the computation and improve the throughput.

Overall, the throughput of a machine learning or deep learning system is a critical factor in determining its performance and suitability for real-world applications. By optimizing the throughput, developers can ensure that the system can process large volumes of data quickly and accurately, and meet the demands of modern data-intensive applications.

VI. Simulation Setup

The main goal of the paper is finding out how much resources is utilized by CPU and GPU for training a deep learning and machine learning model. Tensorflow and pytorch together constructs a convolutional neural network model and the model is pre-trained using denseNet and has changed the last layer of the neural network to train out dataset. The neural network has been trained on CPU, GPU [1] and Pytorch model. In our model Profiler had been used to visualize the performance. Many frameworks has been used along with that NVIDIA has been used to evaluate the GPU performance. The experimental conditions that are configured to carry out this experiment specifically for hardware are listed below.

- ✓ Operating System: Ubuntu 20.04/Windows 11
- ✓ Manufacturer: Acer
- ✓ CPU: Intel core i9 -1 to 900
- ✓ GPU: NVIDIA 7680
- ✓ Total GPU Memory: 12 GB
- ✓ Clock Rate: 1.55 GHZ
- ✓ Total RAM: 32 GB
- ✓ Total Disk: 4 TB SSD



Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Fig 3 : DenseNet architecture

The neural network and the dataset used in our research are, the dataset has been downloaded from Kaggle. There are a total of 1000 training set and 502 testing set of dog and cat images. The architecture is defined using a pretrained CNN network called "densenet121". Transfer learning is utilized to train the network since it speeds up the learning process and frees up time for evaluation. In order to determine the response of CPU and GPU resource utilization, various hyper parameters are adjusted.

The settings for the convolutional neural network model are listed below.

- ✓ Model: DenseNet121
- ✓ Total Training set: 1000
- ✓ Total Testing set: 502
- ✓ Total classes: 2
- ✓ Total epochs: 10
- ✓ Optimizer: Adam
- ✓ Loss Function: Cross-entropy
- ✓ Batch size: 32 and 64
- ✓ Learning Rate: 0.001 and 0.01

VII. Discussion and Result

CPU and GPU metrics has been carefully evaluated and the CNN model is trained for 10 epochs.

A.Throughput

The inference time is used to calculate the throughput. Inference takes more time on the CPU than on the GPU. A single image tests in the CPU in around 5 seconds, while a single image tests in the GPU in about 2-3 seconds, which is better than the CPU. This demonstrates that GPU also contributes significantly to the inference time that affects network throughput.

B.Accuracy

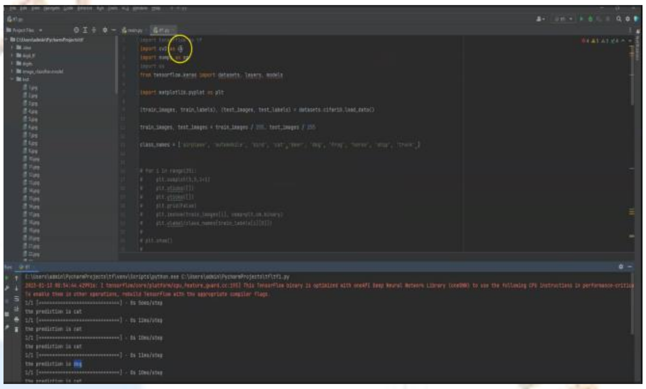
On both the CPU and the GPU, the test accuracy appears to be comparably equal. Even though training the model on a CPU takes a long time, there are no appreciable changes in test accuracy between CPU and GPU testing. Both the CPU and GPU test accuracy ranges from (98-99)%.

C.Utilization

The computations are split between two GPUs with a learning range of 0.001 and a batch size of 32, only 3% of the GPU:0 and 14% of the second GPU:1 were used to train the model. It displays the GPU model's minimal use. The task was first handled by the CPU, which used 62% of the total resources.

There is greater GPU usage than previously after raising the batch size by 32, making the total batch size 64. GPU:0 uses 69% of the system's total RAM, whereas GPU:1 uses 14%. This demonstrates how understanding GPU utilization metrics enables us to adjust the hyper parameters inside neural networks, which are always helpful for network optimization.

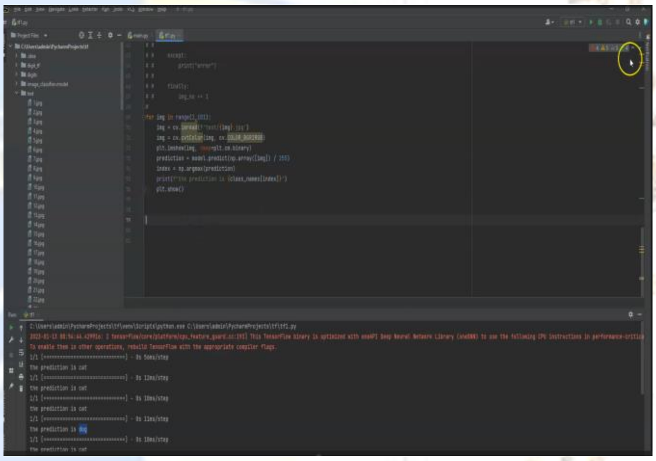
However, compared to GPU, the model training time when utilizing only CPU is considerably greater. It demonstrates that in order to train deep neural networks quickly and effectively, GPU use must be maximized.



```
import tensorflow as tf
import cv2 as cv
import numpy as np
import os
from tensorflow.keras import datasets, layers, models

import matplotlib.pyplot as plt

(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()

train_images, test_images = train_images / 255, test_images / 255

class_names = ['airplane', 'automobile', 'bird', 'cat','deer', 'dog', 'frog', 'horse', 'ship', 'truck']


# for i in range(16):
#     plt.subplot(4,4,i+1)
#     plt.xticks([])
#     plt.yticks([])
#     plt.grid(False)
#     plt.imshow(train_images[i], cmap=plt.cm.binary)
#     plt.xlabel(class_names[train_labels[i][0]])
#
# plt.show()
#
```

```
C:\Users\admin\PycharmProjects\tf\venv\Scripts\python.exe C:\Users\admin\PycharmProjects\tf\tf1.py
2023-01-13 08:54:44.429916: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critic
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
1/1 [==============================] - 0s 50ms/step
the prediction is cat
1/1 [==============================] - 0s 12ms/step
the prediction is cat
1/1 [==============================] - 0s 10ms/step
the prediction is cat
1/1 [==============================] - 0s 11ms/step
the prediction is dog
1/1 [==============================] - 0s 10ms/step
the prediction is cat
```

Fig 4: Tensorflow implementation in simulations



```
#    except:
#        print("error")
#
#    finally:
#        img_no += 1
#
for img in range(1,101):
    img = cv.imread(f"test/{img}.jpg")
    img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
    plt.imshow(img, cmap=plt.cm.binary)
    prediction = model.predict(np.array([img]) / 255)
    index = np.argmax(prediction)
    print(f"the prediction is {class_names[index]}")
    plt.show()
```

```
C:\Users\admin\PycharmProjects\tf\venv\Scripts\python.exe C:\Users\admin\PycharmProjects\tf\tf1.py
2023-01-13 08:54:44.429916: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critic
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
1/1 [==============================] - 0s 50ms/step
the prediction is cat
1/1 [==============================] - 0s 12ms/step
the prediction is cat
1/1 [==============================] - 0s 10ms/step
the prediction is cat
1/1 [==============================] - 0s 11ms/step
the prediction is dog
1/1 [==============================] - 0s 10ms/step
the prediction is cat
```

Fig 5: predicting results

VIII. Conclusion

We have run tests to demonstrate how the deep learning model's CPU and GPU affect the amount of time and memory they use. The training of artificial neural networks is influenced by numerous factors. We have investigated the effectiveness of various metrics for CPU and GPU consumption. We also draw the conclusion from the experiment that some neural network model parameters are also in charge of CPU and GPU resource utilization. After profiling, we can see how the use of the CPU and GPU for the optimization process affects performance.

REFERENCES

[1] Raju, K., Niranjan N. Chiplunkar, and Kavoor Rajanikanth. "A CPU-GPU cooperative sorting approach." 2019 Innovations in Power and Advanced Computing Technologies (i-PACT). Vol. 1. IEEE, 2019.

[2] Yudha, Ardhi Wiratama Baskara, et al. "A simple cache coherence scheme for integrated CPU-GPU systems." 2020 57th ACM/IEEE Design Automation Conference (DAC). IEEE, 2020.

[3] Sourouri, Mohammed, et al. "CPU+ GPU programming of stencil computations for resource-efficient use of GPU clusters." 2015 IEEE 18th International Conference on Computational Science and Engineering. IEEE, 2015.

[4] Yang, Yi, et al. "CPU-assisted GPGPU on fused CPU-GPU architectures." IEEE International Symposium on High-Performance Comp Architecture. IEEE, 2012.

[5] Valerievich, Bakulev Aleksandr, et al. "The implementation on CUDA platform parallel algorithms sort the data." 2017 6th mediterranean conference on embedded computing (MECO). IEEE, 2017.

[6] Buber, Ebubekir, and D. I. R. I. Banu. "Performance analysis and CPU vs GPU comparison for deep learning." 2018 6th International Conference on Control Engineering & Information Technology (CEIT). IEEE, 2018.

[7] Thomas, Winnie, and Rohin D. Daruwala. "Performance comparison of CPU and GPU on a discrete heterogeneous architecture." 2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA). IEEE, 2014.

[8] Arora, Manish, et al. "Redefining the Role of the CPU in the Era of CPU-GPU Integration." IEEE Micro 32.6 (2012): 4-16.

[9] China, Tianjin. "The face detection system based on GPU+ CPU desktop cluster."

[10] Power, Jason, et al. "gem5-gpu: A heterogeneous cpu-gpu simulator." IEEE Computer Architecture Letters 14.1 (2014): 34-36.

[11] Bakhoda, Ali, et al. "Analyzing CUDA workloads using a detailed GPU simulator." 2009 IEEE international symposium on performance analysis of systems and software. IEEE, 2009.

[12] Harvey, Jesse Patrick. "Gpu acceleration of object classification algorithms using nvidia cuda." (2009).

[13] Sarker, Iqbal H. "Machine learning: Algorithms, real-world applications and research directions." SN computer science 2.3 (2021): 160.

[14] Rai, Siddharth, and Mainak Chaudhuri. "Improving CPU performance through dynamic GPU access throttling in CPU-GPU heterogeneous processors." 2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). IEEE, 2017.

[15] Buber, Ebubekir, and D. I. R. I. Banu. "Performance analysis and CPU vs GPU comparison for deep learning." 2018 6th International Conference on Control Engineering & Information Technology (CEIT). IEEE, 2018.

[16] Chandrashekhar, B. N., H. A. Sanjay, and Tulasi Srinivas. "Performance Analysis of Parallel Programming Paradigms on CPU-GPU Clusters." 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS). IEEE, 2021.

[17] Wezowicz, Matthew, and Michela Taufer. "On the cost of a general GPU framework: the strange case of CUDA 4.0 vs. CUDA 5.0." 2012 SC Companion: High Performance Computing, Networking, Storage and Analysis (SCC). Vol. 1. IEEE Computer Society, 2012.

[18] Hukerikar, Saurabh, and Nirmal Saxena. "Runtime Fault Diagnostics for GPU Tensor Cores." 2022 IEEE International Test Conference (ITC). IEEE, 2022.

[19] Kumakura, Kota, et al. "CPU Usage Trends in Android Applications." 2022 IEEE International Conference on Big Data (Big Data). IEEE, 2022.

[20] Choquette, Jack, et al. "Nvidia a100 tensor core gpu: Performance and innovation." IEEE Micro 41.2 (2021): 29-35.