A Comparative Analysis of CPU and GPU-Based Cloud Platforms for CNN Binary Classification

Taieba Tasnim
Department of Computer Science
Tuskegee University
Tuskegee, Alabama, U.S.A.
Email: ttasnim6386@tuskegee.edu

Mohammad Rahman
Department of Computer Science
Tuskegee University
Tuskegee, Alabama, U.S.A.
Email: mrahman@tuskegee.edu

Fan Wu
Department of Computer Science
Tuskegee University
Tuskegee, Alabama, U.S.A.
Email: fwu@tuskegee.edu

***Abstract*—This study explores how Convolutional Neural Network (CNN) model performs in binary classification tasks, particularly on a cloud platform configured with Central Processing Unit (CPU) and Graphics Processing Unit (GPU) resources. We conducted simulations of the CNN model for binary classification with different parameters to compare the CPU and GPU performances. We analyzed evaluation matrices, emphasizing both binary classification accuracy and training time. This analysis aims to facilitate the selection of a computational platform by considering both budgetary constraints and specific requirements.**

Keywords- CNN; CPU; GPU; Cloud.

I. INTRODUCTION

Convolutional Neural Networks (CNNs) have become the workhorse for image recognition tasks. Their power lies in extracting relevant features from images using pooling layers. This allows the network to make accurate predictions. To achieve this, CNNs are trained on extensive datasets, where they learn to identify features and classify images through backpropagation. This automated training is guided by human decisions in configuring network architecture and parameters, ensuring that the CNN can accurately predict new image labels once trained.

An essential step in effectively utilizing CNNs is verifying their performance on different computing platforms, including CPUs and GPUs. Evaluating performance across these platforms ensures reliability, cost-effectiveness, and adaptability. This process involves understanding each hardware type's strengths and weaknesses, optimizing resource usage, ensuring compatibility, and guiding hardware-specific improvements. It also identifies current limitations and explores new technologies, shaping the future of computer development. A comprehensive evaluation considers metrics beyond processing time, including throughput, latency, memory consumption, and energy efficiency. Standardized benchmark suites like DeepBench, MLPerf, and TensorFlow Benchmark facilitate this process across different setups [1].

This work evaluates CPUs and GPUs for CNNs, emphasizing hardware selection's impact on performance, focusing on binary classification tasks within cloud-based environments. CPUs excel at sequential tasks and offer high clock speeds, making them suitable for smaller tasks and natural language processing on resource-limited devices. Conversely, GPUs' parallel architecture optimizes them for larger, complex CNNs, excelling in high-throughput, low-latency tasks ideal for image recognition. This study extends beyond traditional performance metrics to analyze cloud environments' performance, efficiency, and reproducibility. It addresses challenges such as resource variability and provides insights into how hardware selection impacts the performance and cost-efficiency of cloud-based machine learning. Cloud computing supports this by allowing scalable solutions across diverse CPU and GPU configurations, enhancing CNN deployments' flexibility and potential.

In this research, our main contributions are outlined as follows:

- We provide a detailed empirical analysis comparing the performance of CNN binary classification tasks on CPU and GPU platforms in a cloud environment, highlighting the differences in training efficiency and execution speed.
- Our study offers insights into the computational resource utilization of CNNs, identifying how different parameter settings of batch size and epoch impact the performance of CPU and GPU hardware platforms in binary classification tasks.
- We demonstrate the trade-off between training duration and the performance capabilities of both CPU and GPU hardware.

The paper is structured as follows: Section II covers the literature review, Section III outlines the methodology, including data sourcing, experimental setup, and training environment. Section IV discusses evaluation metrics, Section V analyzes the experimental findings, and Section VI concludes with a summary and future research suggestions.

II. LITERATURE REVIEW

CNNs are pivotal in deep learning, excelling in various applications. This review highlights that GPUs, with their parallel processing capabilities, outperform CPUs by 2 to 24 times in CNN tasks due to CPUs' sequential processing limitations [2]. Several recent studies confirm that GPUs outperform CPUs in CNN tasks, particularly for extensive datasets, due to their superior parallel processing capabilities [3] [4] [5]. However, selecting hardware involves more than speed; power efficiency and cost are also critical factors.

Süzen et al. [6] noted the importance of these considerations. Oh et al. revealed that, in embedded systems, CPUs achieved 65% of a PC's GPU performance while consuming only 2.6% of the power, making CPUs effective for resource-limited tasks [7]. To enhance efficiency, optimizing CNN models through techniques like pruning and quantization can reduce computational complexity. Blott et al. explored these methods, showing they are adaptable across hardware and improve performance [8]. Existing research also investigates benchmarking the performance of CPUs and GPUs for other machine learning tasks, such as Long Short-Term Memory (LSTM) networks, providing a broader understanding of hardware capabilities across different neural network architectures [9]. Machine learning predicts CNN execution time, power, and memory usage, especially on GPUs. Bouzidi et al. presented a model to aid researchers in hardware selection, further illustrating the practical applications of these predictive insights [10].

III. METHODOLOGY

A. CNN Architecture Overview

In this study, we explored the architecture of a CNN as a fundamental framework for image classification tasks, as shown in Figure 1 [11]. This illustration outlines the CNN's evolution, starting with the input layer, followed by consecutive convolution and pooling layers for feature extraction, and culminating with a series of fully connected layers that lead to the final classification output. Such an architecture is adept at recognizing and interpreting the intricate patterns in our dataset, consisting of high-quality images of dogs and cats.

B. Data Acquisition

This research leveraged data from two primary sources: Kaggle and Google, renowned for their comprehensive datasets, including the well-known dog and cat datasets. Kaggle provided a dataset that consists primarily of high-quality images of dogs and cats, complete with detailed metadata and labels. This dataset is particularly useful for studies involving CNNs due to its focus on these animals.

Additionally, we utilized Google to assemble a distinct dataset encompassing a variety of dog and cat breeds. This diverse collection was pivotal in training our machine-learning models. We also curated multiple datasets from these sources to evaluate the machine learning algorithms' performance effectively.

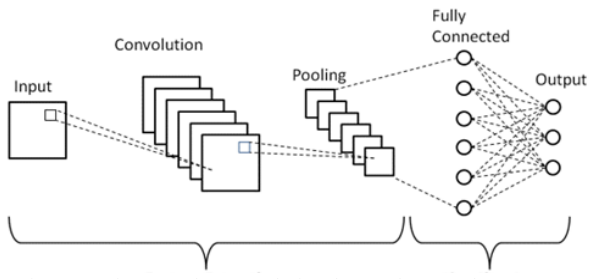

Figure 1. Diagram of a Convolutional Neural Network Structure.

C. Tools and Training Environment

For the experiment setup, we leveraged Google Colab, a Jupyter notebook environment, to train our CNN model [12]. This environment offers excellent support from Keras, allowing for network implementation and training on Google Cloud's GPUs and CPUs [13] [14]. Google Colab enables simultaneous multi-CPU and GPU usage and offers high training speeds, allowing network pruning without losing prediction accuracy. We trained the CNN model on CPU and GPU, benefiting from Colab's easy switching between runtime environments. We trained the same CNN model on both CPU and GPU, enhancing GPU performance with minor code adjustments while keeping the model's structure unchanged. Google Colab's dynamic resource allocation can cause inconsistent performance. Hardware specs and software versions, like TensorFlow or PyTorch, may also vary [1]. To ensure reproducibility, we ran several experiments to reduce resource limits and make the results more reliable.

D. Hardware and Software Integration

Google Colab provides a convenient way to import data from Google Drive. We uploaded our dataset to Google Drive and mounted it on the Colab environment. After completing CPU training, we transitioned to GPU. We improved tensor operations for parallel processing, managed memory better, adjusted GPU settings for efficiency, and changed precision settings for faster and more accurate results (Figure 2) [11]. Once adjustments were made, we resumed GPU training and monitored epoch and batch durations. Despite longer setup times due to model compilation, the enhancements significantly reduced GPU training times compared to CPU, proving the effectiveness of our optimizations.

IV. EVALUATION METRICS

In this study, we employed several evaluation metrics to assess our model's performance on unseen data, particularly in classification tasks. This section briefly describes these metrics, which are paramount for ensuring the practical utility of any binary classification model. The True Positive Rate (TPR) is central to this evaluation.

$$\text{True Positive Rate (TPR)} = \frac{\text{TP}}{\text{TP+FN}} \quad (1)$$

Here TP (True Positives) is the number of correctly predicted positive instances, and FN (False Negatives) is the

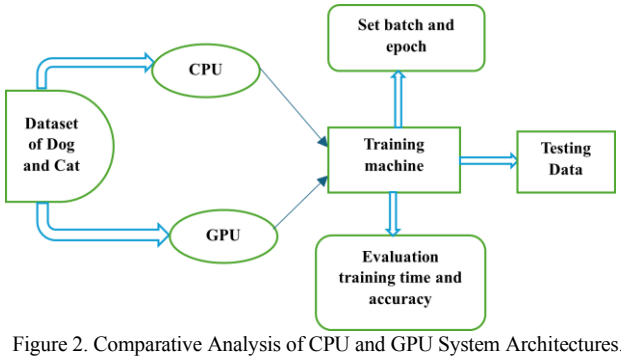

Figure 2. Comparative Analysis of CPU and GPU System Architectures.

number of positive cases incorrectly predicted as negative.

TPR quantifies the proportion of correctly identified positive instances. This metric is particularly crucial in scenarios where class imbalances exist, offering insights into the model's ability to discriminate between classes accurately. Training times were monitored to evaluate model efficiency across different hardware setups, highlighting trade-offs between accuracy and speed.

V. RESULT AND DISCUSSION

To evaluate the performance of GPU and CPU, we embarked on an experiment where the initial step involved training a model using an 8000-image dataset of dogs and cats. Employing various batch sizes (16, 32, 64, and 128) and epochs (1, 2, 3, 4, and 5), we meticulously observed the execution time, noting changes as we adjusted these parameters. An epoch in this context refers to a complete pass through the entire dataset by the learning algorithm. This setup paved the way for a comparative analysis, using a smaller, 1000-image dataset to test the machines.

Building on this groundwork, Figure 3 presents a bar graph comparing True Positive Rates (TPR) for CPU and GPU across five epochs. The TPR fluctuates with each epoch, peaking at the fifth where the GPU slightly outperforms the CPU. However, the difference in TPR performance between CPU and GPU is often negligible, indicating that both platforms can achieve similar accuracy.

The trend of improvement in correctly identifying positive instances is due to effective model learning, fine-tuning, and enhanced feature extraction with each epoch, with GPUs offering a performance edge due to their superior parallel processing capabilities.

Transitioning from TPR to training durations, Figure 4 compares the time taken by CPU and GPU across five epochs with a fixed batch size of 128. The CPU's training times notably increase at the fourth epoch before tapering off, while the GPU shows consistent time expenditure. The GPU generally outperforms the CPU, with nearly equal performance at the third epoch.

Further dissecting the performance dynamics, Figure 5 compares TPR between CPUs and GPUs at varying batch sizes during the fourth epoch. Surprisingly, the CPU outperforms the GPU at a batch size 64 due to its efficiency in managing moderately parallel tasks. This highlights the CPU's strength in handling tasks more effectively than the GPU, where operational overhead can detract from performance. As the batch size increases to 128, the GPU excels by handling larger volumes of parallel operations, delivering peak performance, and surpassing the CPU, showcasing its optimal design for high throughput computing.

The narrative of efficiency continues in Figure 6, which delineates the training durations for CPU and GPU across different batch sizes during the fourth epoch. An interesting trend is observed: while the CPU training time slightly increases at the smallest batch size, it stabilizes as batch sizes escalate, only to surge dramatically at the largest batch size of 128. Conversely, the GPU shows a steady decrease in training time, maintaining its efficiency edge across all tested batch sizes.

Culminating our analysis, Figure 7 introduces a scatter plot showing the relationship between training time and TPR for models trained on CPUs and GPUs. The graph reveals a trend: training time increases with improved accuracy, then plateaus. It shows that GPUs consistently achieve similar TPRs in shorter training times than CPUs, highlighting their superior efficiency.

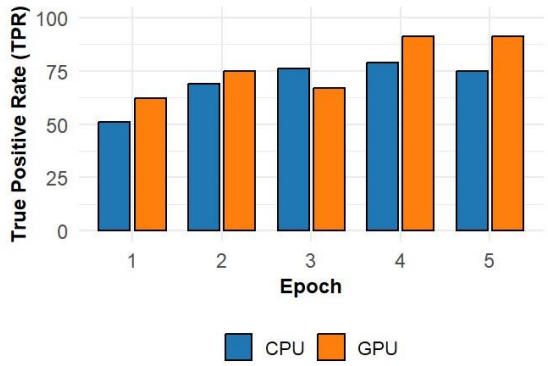

Figure 3. TPR by Epoch for CPU vs. GPU at Batch Size 128.

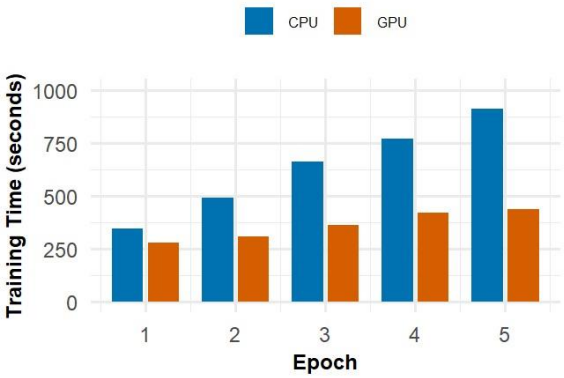

Figure 4. Training Time vs. Epoch for CPU and GPU at Batch sizes 128.

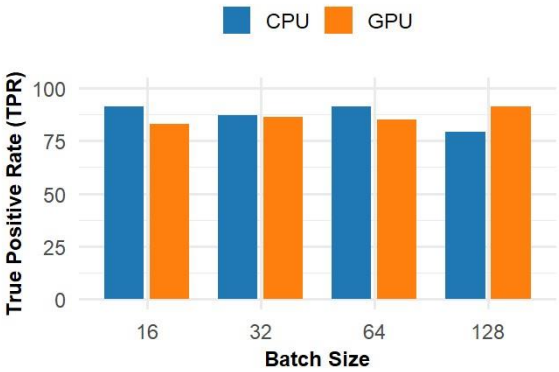

Figure 5. TPR Comparison by Batch Size for CPU and GPU at Epoch 4.

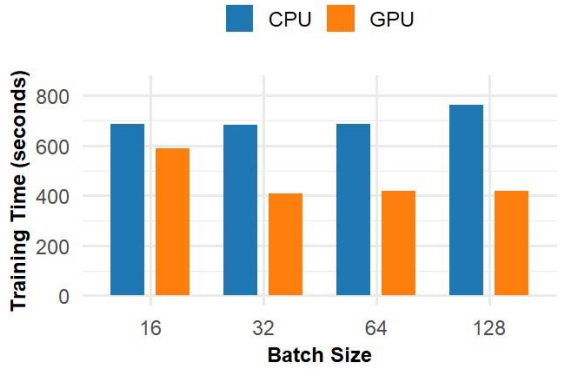

Figure 6. Training Time vs. Batch Size for CPU and GPU at Epoch 4.

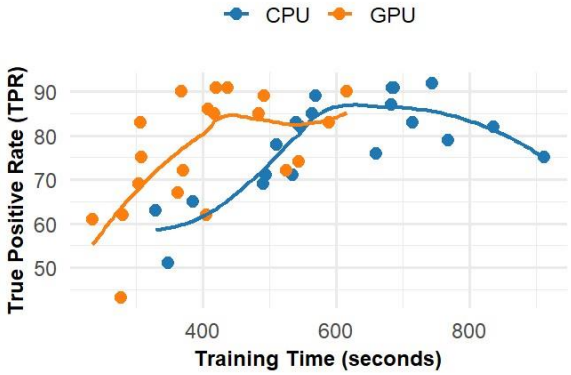

Figure 7. TPR vs. Training Time for CPU and GPU.

VI. CONCLUSION

Our investigation into the performance dynamics of GPUs and CPUs for CNN applications yielded insightful findings. Utilizing a dataset comprising 8,000 images of dogs and cats for training and an additional 1,000 for testing, we methodically analyzed the impact of various batch sizes and epochs on the system's performance. Empirical data showed GPUs consistently outperformed CPUs in training efficiency and speed, achieving higher or comparable TPRs. GPUs excelled with larger batch sizes, demonstrating superior performance for extensive CNN tasks, offering significant speed and accuracy advantages over CPUs.

Future research should include more hardware models, like NVIDIA's Tesla and RTX series, to better understand CPU and GPU performance differences. This will help select optimal hardware for CNN tasks and enhance our findings with cost and performance analysis.

ACKNOWLEDGMENT

The work is partially supported by the National Science Foundation (NSF) under NSF Awards #2019561, #2234911, #2209637, and #2100134. The opinions, findings, and recommendations in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] S. Verma *et al.*, "Demystifying the MLPerf Training Benchmark Suite," in *2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2020, pp. 24-33.

[2] D. Strigl, K. Kofler, and S. Podlipnig, "Performance and Scalability of GPU-Based Convolutional Neural Networks," in *2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, 2010, pp. 317-324.

[3] E. Buber and B. Diri, "Performance Analysis and CPU vs GPU Comparison for Deep Learning," in *2018 6th International Conference on Control Engineering & Information Technology (CEIT)*, 2018, pp. 1-6.

[4] E. Cengil, A. Çinar, and Z. Güler, "A GPU-based convolutional neural network approach for image classification," in *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*, 2017, pp. 1-6.

[5] M. U. Yaseen, A. Anjum, O. Rana, and R. Hill, "Cloud-based scalable object detection and classification in video streams," *Future Generation Computer Systems,* vol. 80, pp. 286-298, 2018/03/01/ 2018.

[6] A. A. Süzen, B. Duman, and B. Şen, "Benchmark Analysis of Jetson TX2, Jetson Nano and Raspberry PI using Deep-CNN," in *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, 2020, pp. 1-5.

[7] S. Oh, M. Kim, D. Kim, M. Jeong, and M. Lee, "Investigation on performance and energy efficiency of CNN-based object detection on embedded device," in *2017 4th International Conference on Computer Applications and Information Processing Technology (CAIPT)*, 2017, pp. 1-4.

[8] M. Blott *et al.*, "Evaluation of Optimized CNNs on FPGA and non-FPGA based Accelerators using a Novel Benchmarking Approach," *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays,* pp. 317-317, 2020.

[9] A. Saha, M. Rahman, and F. Wu, "Evaluating LSTM Time Series Prediction Performance on Benchmark CPUs and GPUs in Cloud Environments," 2024, pp. 321-322.

[10] H. Bouzidi, H. Ouarnoughi, S. Niar, and A. A. E. Cadi, "Performance Modeling of Computer Vision-based CNN on Edge GPUs," *ACM Transactions on Embedded Computing Systems,* vol. 21, no. 5, pp. 1-33, 2022.

[11] V. H. Phung and E. J. Rhee, "A deep learning approach for classification of cloud image patches on small datasets," *Journal of information and communication convergence engineering,* vol. 16, no. 3, pp. 173-178, 2018.

[12] *Google Colaboratory (2024). https://colab.research.google.com. Last Accessed 10 Mar 2024*.

[13] *Keras (2024). https://keras.io. Last Accessed 10 Mar 2024*.

[14] V. Sharma, G. K. Gupta, and M. Gupta, "Performance Benchmarking of GPU and TPU on Google Colaboratory for Convolutional Neural Network," in *Applications of Artificial Intelligence in Engineering*, Singapore, 2021, pp. 639-646: Springer Singapore.