# *Benchmarking GPU using Deep Learning Models*

Sai Kashyap Kurella,
Department of Electrical Engineering

*San Jose State University*,
San Jose, U.S.,
saikashyap.kurella@sjsu.edu

Akash Mattaparthy
Department of Computer Engineering

San Jose State University
San Jose, U.S.
akash.mattaparthy@sjsu.ed

***Abstract*—GPU performance for machine learning tasks requires the need for systematic methodologies to analyze interactions between hardware and software platforms, especially considering the intricate nature of deep learning workloads. Established MLPerf benchmark suites address the challenges posed by diverse ML models and the selection of representative, mature, and community-supported models.**

**The focus shifts to the importance of simple metrics, such as latency, for average users in choosing GPUs, particularly in the absence of detailed hardware descriptions from vendors. We attempt achieve by working out simple metrics for model performance, particularly in the computer vision domain. We try to use of popular model formats and development environments, including ONNX and PyTorch in Google Colab which provide access to Nvidia Tesla GPUs. The results section presents latency comparisons for YOLO v4 and v5 models, emphasizing the importance of model architecture and GPU selection.**

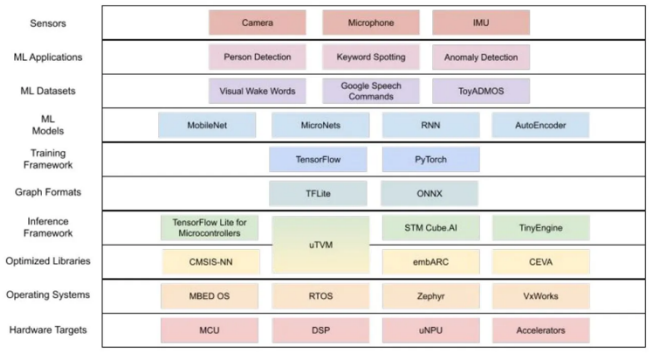
**In conclusion, this work underscores the value of simple metrics for making informed decisions about GPU selection, both for training and inference in deep learning. We acknowledge the preliminary nature of the work and suggests future extensions to include a broader range of problems and formats.**

***Keywords—benchmarking, deep learning.***

## I. Introduction

Benchmarking involves evaluating and comparing the performance of hardware or software components against established standards or other systems. It allowing consumers to make take decisions by comparing related performance metrics. This process involves running a set of standardized tests that measure a GPU's performance under various conditions, providing valuable insights into its strengths and limitations.

Deep learning, a highly impactful field, relies heavily on GPUs to train deep neural networks. Benchmarking in the context of deep learning focuses on evaluating a GPU's performance in tasks such as training and inference for deep neural networks. The intricate nature of deep learning workloads demands specialized benchmarks that go beyond traditional graphics and gaming benchmarks. The presence of several layers in the usual ML workloads like show in the picture below makes benchmarking complicated.




## II. Background

### A. ML Benchmarking

DL-model training is an expensive process, for e.g. involving utilizing hundreds of GPUs over weeks to train the latest Large Language Models (LLM), and has been usually considered the primary bottleneck. But as these LLMs are deployed at a bigger scale, inference has become a critical workload, with models handling trillions of queries daily. To meet growing computational demands, developers focus on optimizing hardware and software for inference performance. Some of the benchmarks considered while evaluating ML systems:

1. *Training Performance*: Benchmarking measures the speed at which a GPU can process and train these models. The training throughput is the number of training samples processed per unit of time, and is dependent on the optimal batch sizes that maximize efficiency.

2. *Inference Performance through latency*: Once trained, deep learning models are used for making predictions. Benchmarking assesses the GPU's ability to perform rapid and accurate inferences on new data. In real-world applications, low latency and high throughput are critical. Benchmarks for inference evaluate how quickly a GPU can process individual predictions and handle multiple requests simultaneously.

3. *Specialized Features*: Modern GPUs often come equipped with specialized hardware, such as tensor cores, designed to accelerate deep learning workloads. Benchmarking takes into account the utilization of these features to achieve faster training times.

4. Parallel Processing: Deep learning tasks are inherently parallelizable. Benchmarking assesses how well a GPU leverages parallel processing capabilities to enhance performance.

5. Framework-specific Benchmarks: Different deep learning frameworks (e.g., TensorFlow, PyTorch) may have unique optimizations for specific GPUs. Benchmarking within the context of these frameworks provides insights into how well a GPU performs with the tools commonly used in deep learning development.

Understanding how GPUs perform in the realm of deep learning is crucial for practitioners and researchers aiming to harness the power of artificial intelligence. As deep learning architectures and models continue to evolve, benchmarking remains an essential practice for staying abreast of advancements and selecting the most suitable GPU for specific deep learning applications. The diversity in ML systems, each with unique approaches to trade-offs like latency, throughput, power, and model quality, makes evaluating inference performance challenging. Over 100 companies target specialized inference chips, compared to around 20 focusing on training chips. The spectrum of ML tasks includes image classification, object detection, machine translation, and more, each with varying quality requirements and real-time processing demands.

### *B. Current Tools*

Both academic and industrial organizations have developed ML inference benchmarks, such as AIMatrix, EEMBC MLMark, AIXPRT, AI Benchmark, TBD, Fathom, and DAWNBench. While each has contributed significantly, the absence of industry-wide input from ML-system designers has led to a lack of consensus on representative ML models, metrics, tasks, and rules within the field until 2018 [1].

The main issue with benchmarking using ML-models requires one to consider the various trade-offs among accuracy, latency, and total cost of ownership (TCO) that are application-specific. A systematic methodology is needed to reveal interactions between hardware and software platforms, considering various model attributes like hyperparameters and have a quick development cycle to adapt rapidly to new platforms and include large models that can stress the limits of emerging platforms.

#### *I) MLPerf*

MLPerf Training and Inference are standardized machine learning inference benchmark suites developed collaboratively by industry and academia, incorporating input from over 200 engineers and practitioners from 30 organizations with the aim to fairly measure the performance of ML hardware, software, and services. They address the challenge of diverse ML models by selecting representative, mature, and community-supported models. This choice enables reproducible measurements and provides an accessible benchmark for the industry[1]. The selected models are open source, contributing to their potential use as research tools.

#### *II) ParaDnn*

ParaDnn was developed because efforts like MLPerf limited themselves by arbitrarily selecting small number of deep learning models to analyse GPUs. New models require months to be added to the suites [2] even when models usually fall behind in the rapid pace of today's development.

This tool generates a diverse set of parameterized multi-layer models, including fully-connected models, convolutional neural networks, and recurrent neural networks. These models exhibit a wide range of parameter sizes, spanning almost five orders of magnitude, surpassing the scope of existing benchmarks. They argue that this analysis in conjunction MLPerf, provides insights that traditional approaches may fail to reveal or adequately explain.

## III. METHODOLOGY

Our work focuses on providing simple metrics like latency to the average users when deciding on the using which GPUs, especially from the multitude of cloud providers. GPU vendors sometimes do not give detailed description of their hardware for commercial reasons and it is up to these simple metrics to get a basic understanding of their performance. Training takes a long time and newer GPUs may have higher throughput but their cost over time might be way larger than undertaking the same workload on an older GPU with lower throughput. Using these simple metrics with cost of using them over time can be used to make that decision.

The first step before embarking on these extensive benchmarking standards (MLPerf, ParaDnn), is to get some simple performance metrics for models (within a class of problems). This gives an idea about the overall performance. We focus on making popular formats and custom developed models to be available to tested on GPUs.

### *A. Model formats*

Most of the current open-source deep learning models are uploaded to platforms like HuggingFace or personal repositories like GitHub. They are mostly developed using frameworks like PyTorch, TensorFlow/Keras, JaX and Julia. The output format of these models may vary according these frameworks and so to address the interoperability, they can be converted to open-source formats like ONNX.

Testing these models on the available hardware show that they vary from the optimized performance. Pruning models to fit on edge devices make them even more susceptible to hardware changes. Hence, getting a first indication of their performance in an easy manner would be helpful for developers before embarking on developing models of their own, by zeroing on the class of models for a particular problem. We attempt to access ONNX and PyTorch models and custom self-developed models (in PyTorch state dictionary 'pth' format).

### *B. Working Environment*

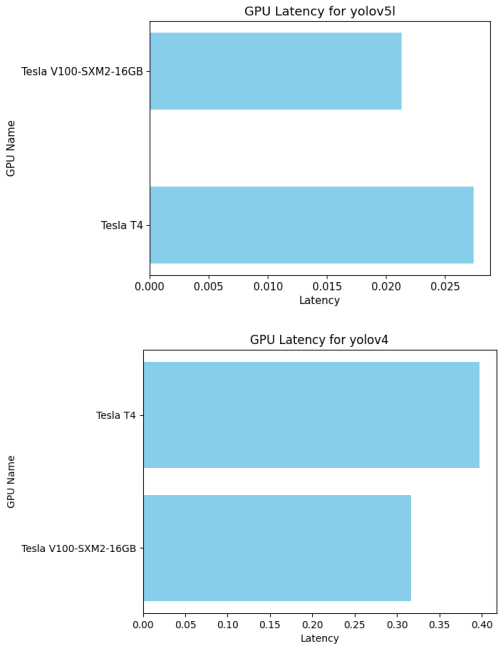
We used Google-Colab environment to test the available GPUs: Nvidia Tesla class V-100 and T4. We concentrate on the class of computer vision problem called object detection using the ubiquitous YOLO models.

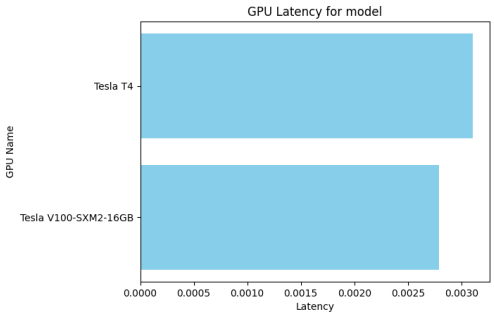
## IV. RESULTS

Latency was calculated by passing a standard input, in this case a representative image repeatedly through the model and clock the average time it took the model to process them. The models chosen are used to perform of common vision task of object detection.

The YOLO v4 used based on the ONNX format and YOLO v5 model used was used from PyTorch Hub Library which loads from the are repository hosted by the creators on GitHub. Though they might seem like different versions of the same model, they are based on different architectures. As it can be seen from the first two graphs, the larger v5 model has lower latency and performs better on an older V-100 GPU. These two GPUs can be accessed at the same rate from Google Colab and if one was to decide on application inference, they can safely choose the larger v5 model on an older GPU. Larger models are more compute intensive than smaller ones.

The last graph shows similar comparison for GPUs based on a custom model we developed, similar to ones most developers have if they are working on developing their own models and want to compare with existing pre-trained state of the art models.




### *Conclusion*

A simple metric for the overall performance can be used to make the important decision of choosing a GPU for either training or inference. But this is only the first step and once a complete model is developed, they can be extensively tested using suites like MLPerf Inference or upcoming ParaDnn which go into performance issues the individual levels of ML development. The work we have done concentrates only on a small subset of problems and format. It can be extended on further development to include other sub-problems and formats.


## ACKNOWLEDGMENT

We extend our sincere gratitude to Prof. Haonan Wang for his guidance and insightful contributions during our repeated interactions.


## REFERENCES


[1] Reddi, V. J. et. al, Cheng, C., Kanter, D., Mattson, P., Guenther Schmuelling, Carole-Jean Wu, Anderson, B., Breughe, M., Charlebois, M., Chou, W., Chukka, R., Coleman, C., Davis, S., Deng, P., Diamos, G., Duke, J., Fick, D., Gardner, J. S., Hubara, I., … Zhou, Y. (2020). MLPerf Inference Benchmark. *ArXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1911.02549

[2] Wang Y., Gu-Yeo W., David B, "A Systematic Methodology for Analysis of Deep Learning Hardware and Software Platforms." In . Third Conference on Machine Learning and Systems (MLSys), 2020.