GPU vs CPU Performance

By: Yash Kamoji, Abhishek Phaltankar, Atif Abedeen, and Isaiah Baker

Introduction

The advancement of Machine Learning (ML) has increased the need for powerful computing hardware like Graphics Processing Units (GPUs), which excel in handling parallel computations required for ML tasks. However, GPUs are often expensive and not always accessible, leading to a need for optimizing ML algorithms for Central Processing Units (CPUs) as well. This project focuses on comparing the performance of GPUs and CPUs in ML tasks using the CIFAR-10 dataset for image classification and a hate speech dataset for text classification. We aim to identify performance bottlenecks and analyze them across these platforms.

Motivation

The motivation for this project stems from the need to optimize machine learning (ML) efficiency using GPUs, known for their superior parallel processing capabilities. As demand for advanced ML applications grows, understanding the performance differences between GPUs and CPUs is essential. This study aims to validate the advantages of GPUs in effectively running ML algorithms, providing insights that can enhance algorithm performance and resource allocation. This knowledge will help ensure the ML community continues to leverage the best available technologies for innovation.

Design Description

Introduction:

- ML advancement demands powerful hardware like GPUs for parallel computations.
- This project focuses on comparing the performance of GPUs and CPUs in different ML tasks

Motivation:

- Given the increasing demands of ML applications, understanding the performance disparities between GPUs and CPUs is imperative.

- Demonstrating the advantages of GPUs and understanding the bottlenecks in the CPUs will enhance ML algorithm efficiency and optimize resource utilization, driving innovation in the field.

Overall Project Design

Goals: To analyze system differences in two tasks; Image Classification and Text Classification using various CNN and Transformer models using PySpark.

Compare different metrics achieved when using a CPU vs GPU

Tune the Hyper-parameters (# of epochs, batch-size, etc) for various models and identify the bottlenecks when training on a CPU vs GPU.

Develop a small-scale app for image classification task using PySpark streaming

Yash and Isaiah were tasked with obtaining results from image classification modelling.

Abhishek and Atif were tasked with obtaining results from text classification modelling.

Image Classification

Design Descriptions:

- **Goal:** To compare image classification performance between multiple different models.
- **Idea:** Have a design structure that can support various model design comparisons.
    - CNNs
        - Smaller Models
        - Larger Models
    - Transformers
        - Smaller Models
        - Larger Models
- Utilizing pretrained models, CNNs and Transformers were fine-tuned on CFAR-10 dataset.

Image Classification (models used)

CNNs (#: implies not utilized due to computational constraints)

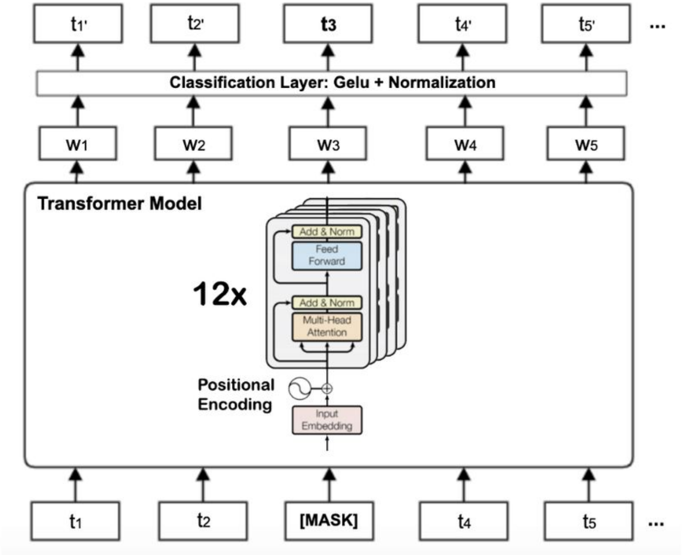
```
conv_model_list = [
    "facebook/convnext-tiny-224" ,
    "facebook/convnext-small-224" ,
    #"facebook/convnext-base-224",
    #"facebook/convnext-large-224",
    "facebook/convnextv2-nano-22k-224" ,
    "facebook/convnextv2-tiny-22k-224" ,
    #"facebook/convnextv2-base-22k-224",
    #"facebook/convnextv2-large-22k-224",
    #"facebook/convnextv2-huge-1k-224"
]
```

Transformers (#: implies not utilized due to computational constraints)

```
vit_model_list = [
    "google/vit-base-patch16-224" ,
    #"google/vit-large-patch16-224",
    "facebook/deit-tiny-patch16-224" ,
    "facebook/deit-small-patch16-224" ,
    "facebook/deit-base-patch16-224" ,
    "facebook/deit-tiny-distilled-patch16-224"
    "facebook/deit-base-distilled-patch16-224"
]
```

Text Classification Task (Transformers)

Design Descriptions: We chose BERT (Bidirectional Encoder Representations from Transformers) as the Transformer model for the Text Classification task



Used the following versions of BERT:

GoogleBERT

HuwaeiBERT

MicrosoftBERT

RoBERTa (for profiling)

Tests

Tests: Image Classification

Training:

- Performed ConvNet and ViT model training on Linux (2 Core, 15 GB GPU) and Mac GPU (8 Core, 30 GB GPU).
- Used the same hyperparameters on both environments to have same baseline for comparing performances.
- Avoided CPU training since it is too slow and crashes frequently due to memory resources.
- Pyspark distributed model training requires multi GPU support and so our training is done on a single GPU.

Inference:

- We performed distributed data inference using PySpark cluster giving faster evaluations and analyse performance for CPU vs GPU.
- Performed Pyspark transformations on model input images (integrated with Pytorch) and during result analysis.
- The accuracies from the fine-tuned models on different systems are same giving credibility to the system analysis.

Pyspark Streaming:

- We run the image classification task on a spark cluster to perform real time image prediction.
- The user (simulated spark client) is pushing images which is continuously being steamed another spark client to predict the image.
- Tested on different models and environment to understand the various underlying system improvements and bottlenecks.

Tests: NLP

Training:

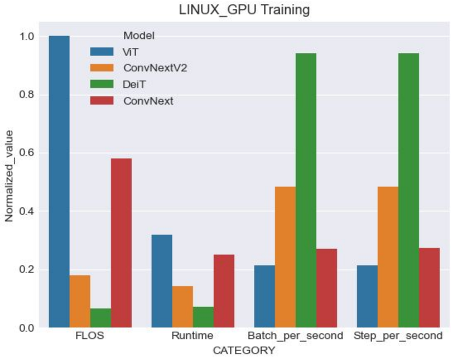
- Utilized Google Colab's T4 GPU and a Mac M2 Pro with a 12-core CPU for text classification model training.
- Employed consistent hyperparameters in both environments for performance comparison.

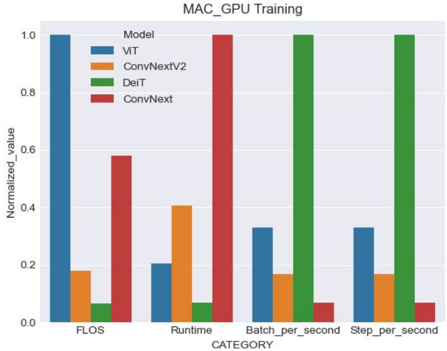
Inference:

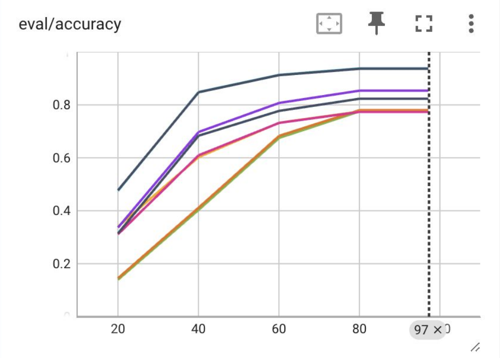
- Executed distributed data inference to enhance evaluation speeds on GPU.
- Analyzed CPU vs GPU performance by comparing execution time and accuracy across different hardware setups.

Experimental Results

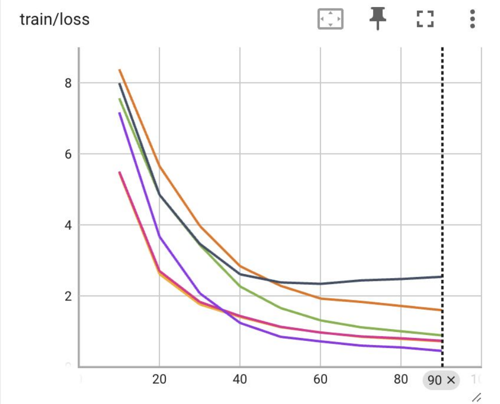
Experimental Results - Image Classification





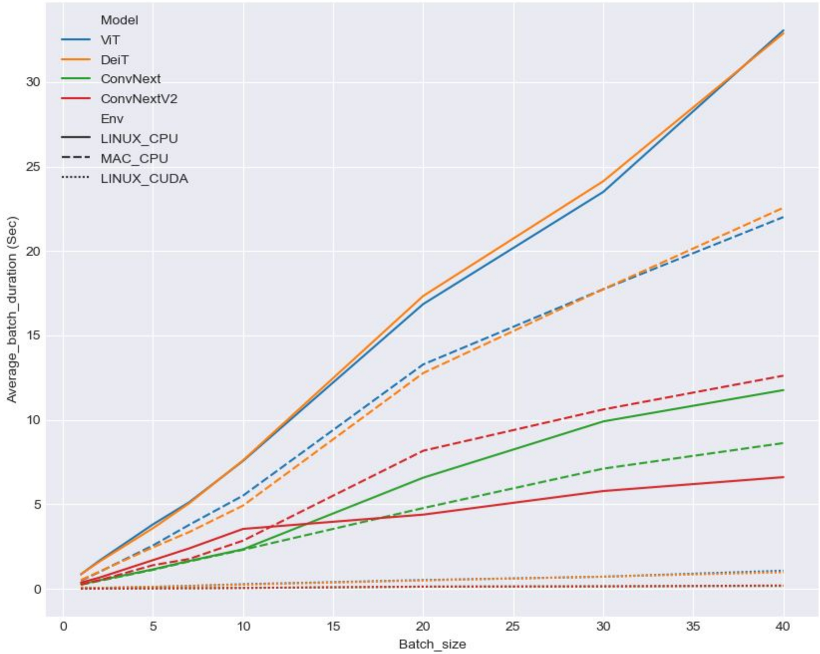


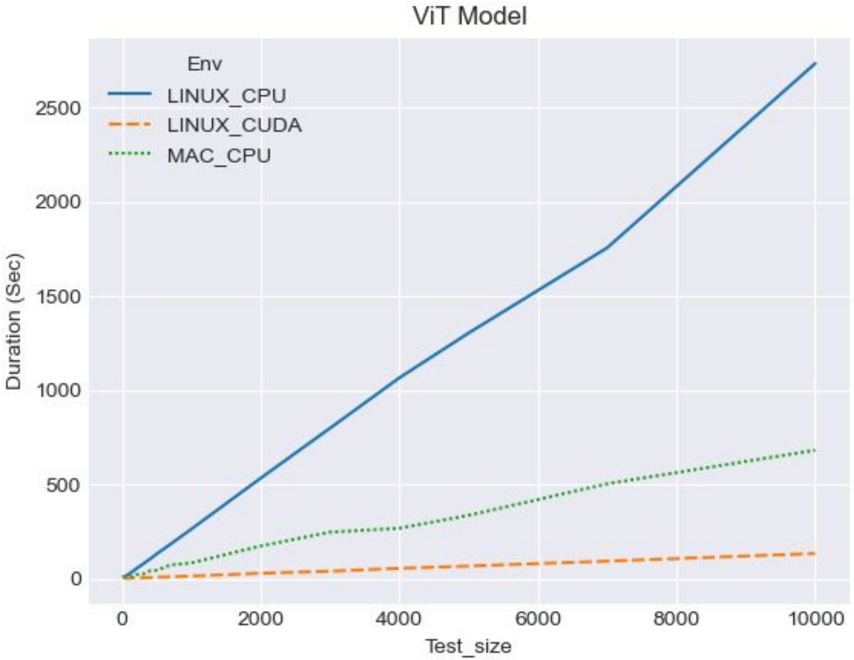
Run ↑	Value	Step	Relative
ConvNext-S:LINUX_GPU/logs	0.7755	97	11.51 min
ConvNext-S:MAC_GPU/logs	0.7804	97	40.9 min
ConvNextV2-N:LINUX_GPU/logs	0.8538	97	6.458 min
ConvNextV2-N:MAC_GPU/logs	0.8233	97	16.75 min
DeiT-T:Linux_GPU/logs	0.7749	97	3.418 min
DeiT-T:MAC_GPU/logs	0.7737	97	3.111 min
google/ViT-B:LINUX_GPU/logs	0.9363	97	14.55 min
google/ViT-B:MAC_GPU/logs	0.9379	97	9.379 min



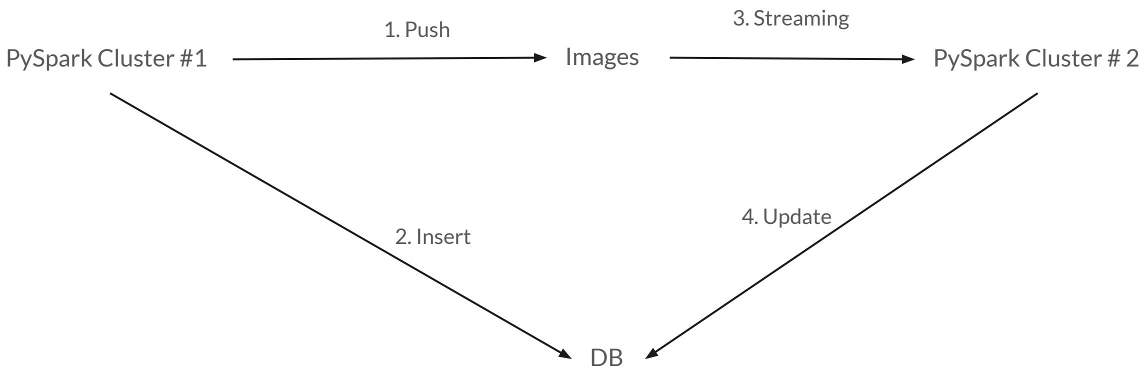
Run ↑	Value	Step	Relative
ConvNext-S:LINUX_GPU/logs	0.8901	90	11.61 min
ConvNext-S:MAC_GPU/logs	1.5977	90	42.3 min
ConvNextV2-N:LINUX_GPU/logs	0.452	90	6.556 min
ConvNextV2-N:MAC_GPU/logs	2.5377	90	17.33 min
DeiT-T:Linux_GPU/logs	0.7202	90	3.403 min
DeiT-T:MAC_GPU/logs	0.7378	90	3.2 min

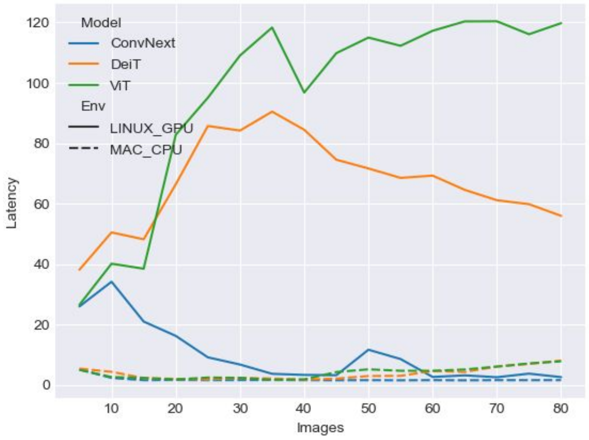
Experimental Results - Image Classification

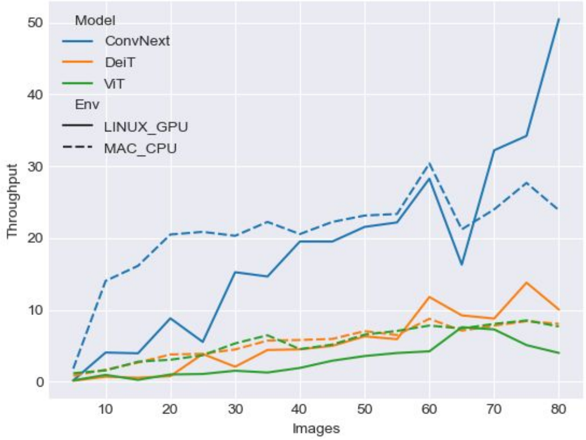




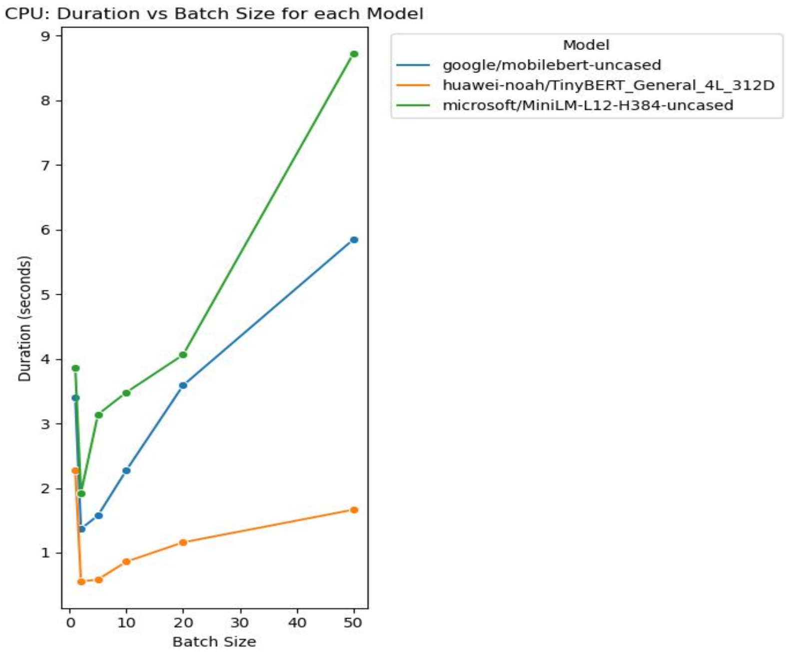
Experimental Results - Pyspark Streaming

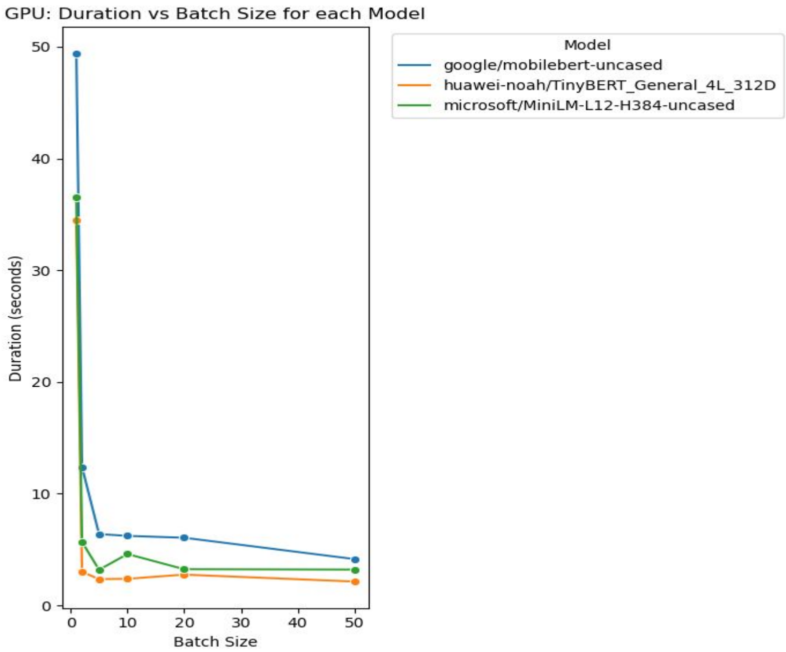




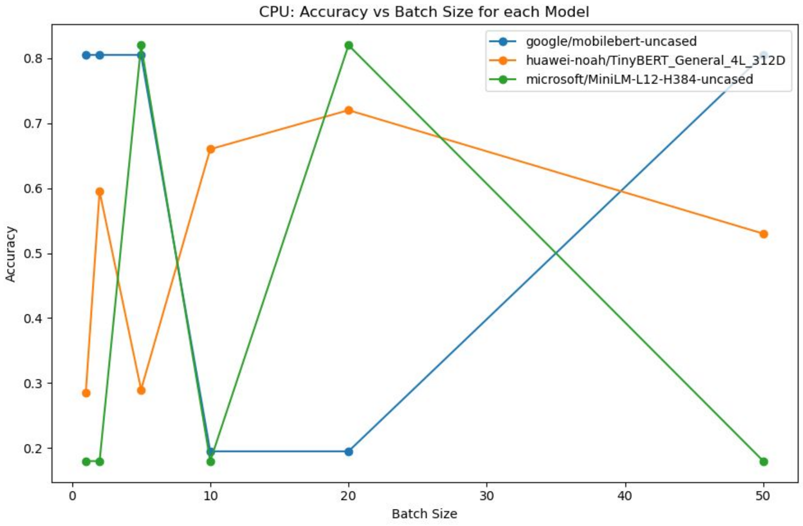


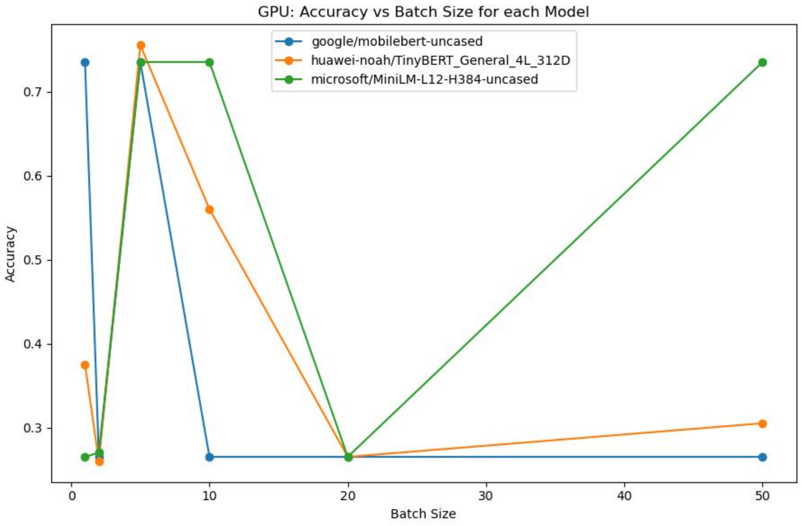
Experimental Results - NLP





Experimental Results - NLP





Analyzing bottlenecks in CPU and GPU

RoBERTa on GPU		
Operator	**Self-CUDA**	**CUDA Total**
atten::copy_	845.07ms	845.07ms
aten::addmm	257.53ms	257.53ms
aten::mm	66.02ms	66.02ms
aten::bmm	47.14ms	47.14ms
Optimizer.step#AdamW.step	191.69ms	191.69ms
Self CUDA time total: 1.794s		

RoBERTa on CPU		
Operator	**Self-CPU**	**CPU Total**
atten::addmm	23.06s	24.54s
aten::bmm	6.17s	6.17s
aten::copy	4.61s	4.61s
aten::mm	4.19s	4.19s
aten::sqrt	3.43s	3.43s
Self CPU time Total: 60.21s		

ViT GPU		
Operator	**Self-CPU**	**CPU total**
aten::copy_	102.826ms	102.898ms
aten::_mps_convolution	37.347ms	37.353ms
aten::linear	14.376ms	14.432ms
aten::native_batch_norm	8.188ms	8.246ms
aten::empty_strided	7.909ms	7.909ms
Self CPU time total: 209.432ms		

ViT CPU		
Operator	**Self-CPU**	**CPU total**
aten::bmm	671.137ms	671.708ms
aten::copy_	352.227ms	352.227ms
aten::addmm	195.439ms	446.854ms
aten::add	157.512ms	157.512ms
aten::gelu	135.115ms	135.115ms
Self CPU time total: 1.757s		

ConvNet GPU		
Operator	**Self-CPU**	**CPU total**
aten::copy_	75.894ms	75.963ms
aten::native_batch_norm	24.288ms	24.330ms
aten::_mps_convolution	22.567ms	22.586ms
aten::linear	21.855ms	21.874ms
aten::add	18.908ms	18.913ms
Self CPU time total: 251.478ms		

ConvNet CPU		
Operator	**Self-CPU**	**CPU total**
aten::_slow_conv2d_forward	2.918s	2.929s
aten::thnn_conv2d	339.266ms	2.932s
aten::addmm	219.056ms	298.109ms
aten::gelu	148.391ms	148.391ms
aten::copy_	122.295ms	122.295ms
Self CPU time total: 3.628s		

Goals

Goals

Use different model architectures (CNNs and Transformers) of different sizes to analyze performance on CPU and GPU - **ACHIEVED**

Compare key metrics for different tasks while training and Inferencing - **ACHIEVED**

Understand the different bottlenecks in a CPU and GPU - **ACHIEVED**

Build a streaming process for images using PySpark and compared its performance on the CPU and GPU - **ACHIEVED**

Possible Improvements

Possible Improvements

Computation Limitations:

- Distributed data inferencing with PySpark involves multi-threaded RDD architecture for optimal performance during distributed model inferencing.
- Running experiments on systems with more resources (higher cores / GPUs), we can perform faster inference, save model evaluation time and analyse different model performance.

Pyspark Streaming Synchronization:

- The two Pyspark clusters: one for publishing data and the other for processing them, are currently independent of each other leading to intermittent data leaks in the pipeline.
- By adding thread locking mechanism, the two clusters will communicate much efficiently giving persistent logging and accurate results.