<https://cotscomputers.com/blog/pcie-lanes/>

<https://www.intel.sg/content/www/xa/en/gaming/resources/how-to-choose-a-motherboard.html>

<https://en.wikipedia.org/wiki/Southbridge_(computing)>

<https://en.wikipedia.org/wiki/Northbridge_(computing)>

<https://ngc.nvidia.com/catalog/resources/nvidia:resnet_50_v1_5_for_tensorflow/performance>

<http://www.vassox.com/linux-general/ubuntu/removing-gui-from-ubuntu/>

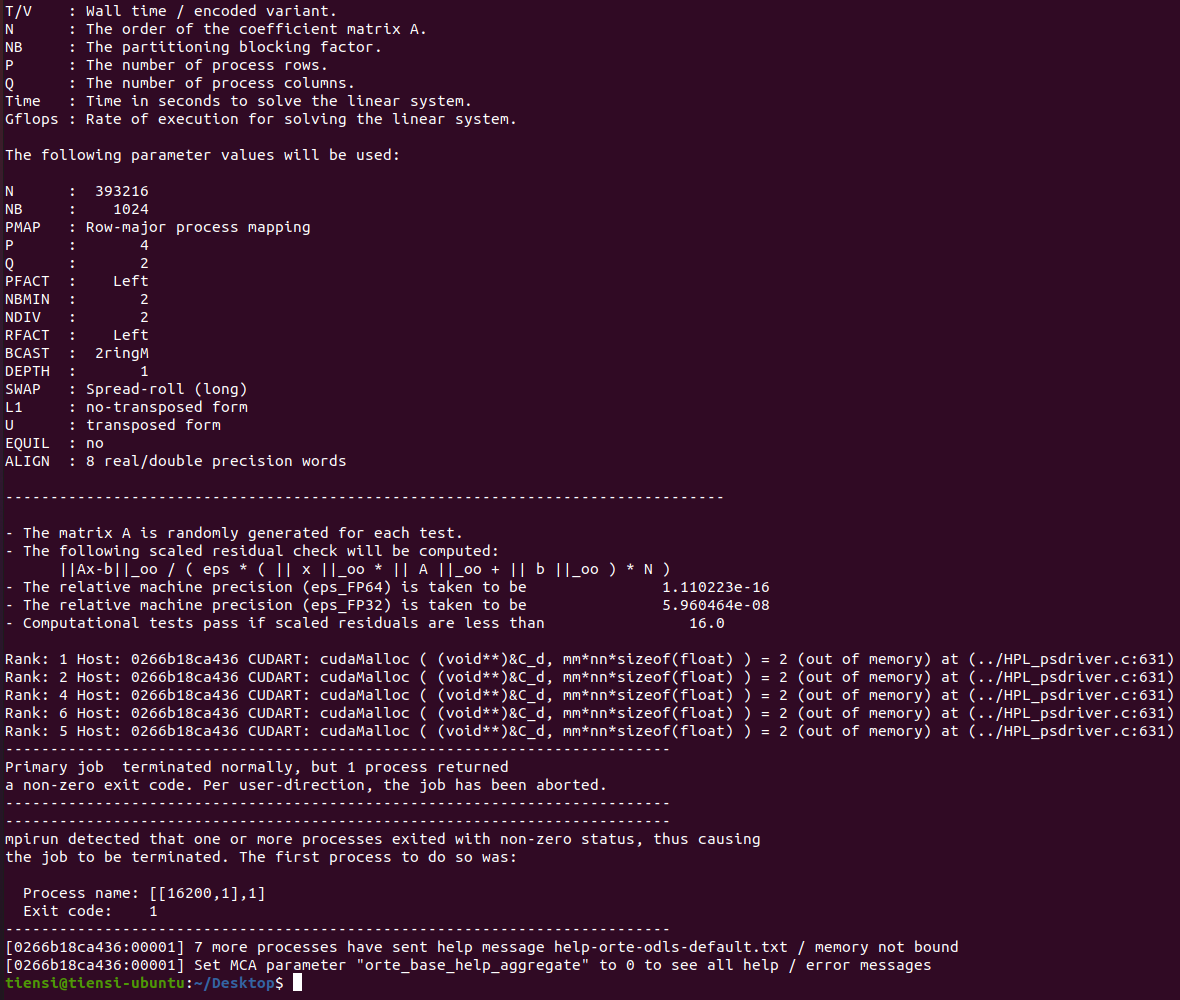
<https://www.researchgate.net/publication/335419992_Demystifying_the_MLPerf_Benchmark_Suite>

Benchmarks

1. Hashcat
   1. hashcat -b
2. AI-Benchmark

[https://medium.com/analytics-vidhya/installing-tensorflow-with-cuda-cudnn-gpu-support-on-ubuntu-20-04-f6f67745750a](https://medium.com/analytics-vidhya/installing-tensorflow-with-cuda-cudnn-gpu-support-on-ubuntu-20-04-f6f67745750aBenchmarks)

* 1. pip install tensorflow-gpu
  2. pip install ai-benchmark
  3. ai-benchmark

1. PyTorch
2. MLPerf
   1. <https://infohub.delltechnologies.com/p/running-the-mlperf-tm-inference-v1-0-benchmark-on-dell-emc-systems/>
   2. Not enough disk space :< need 3TB
      1. <https://www.digitalocean.com/community/tutorials/how-to-create-raid-arrays-with-mdadm-on-ubuntu-18-04>
   3. Example guide (1) <https://www.codenong.com/s1190000022834920/>
   4. metric <https://arxiv.org/pdf/1910.01500.pdf>
   5. **GMNT**
      1. Dataset link <https://drive.google.com/file/d/0B_bZck-ksdkpM25jRUN2X2UxMm8/view?resourcekey=0-KdGKlcAjpJ8q_j2b5ImItQ>
      2. $ pip install --extra-index-url https://developer.download.nvidia.com/compute/redist/cuda/10.0 nvidia-dali
   6. **Resnet**
      1. Dataset link <https://image-net.org/challenges/LSVRC/2012/2012-downloads.php#images> (need all 3)
3. XHPL
   1. <https://developer.nvidia.com/rdp/assets/cuda-accelerated-linpack-linux64>
   2. <https://www.nvidia.com/content/PDF/sc_2010/theater/Phillips_SC10.pdf>
   3. <https://github.com/scamicha/HybridHPL/blob/master/literature/p46-fatica.pdf>
   4. <https://www.netlib.org/benchmark/hpl/tuning.html>
   5. <https://arxiv.org/ftp/arxiv/papers/1108/1108.3268.pdf>
4. HPC LinPack - **OLD** 
   1. <https://www.pugetsystems.com/labs/hpc/Outstanding-Performance-of-NVIDIA-A100-PCIe-on-HPL-HPL-AI-HPCG-Benchmarks-2149/>
   2. CONT='nvcr.io/nvidia/hpc-benchmarks:21.4-hpl'
   3. docker run --gpus all ${CONT} mpirun --bind-to none -np 8 hpl.sh --xhpl-ai --cpu-affinity 0:2:4:6:8:10:12:14 --cpu-cores-per-rank 1 --gpu-affinity 0:0:0:0:0:0:0:0 --dat /workspace/hpl-ai-linux-x86\_64/sample-dat/HPL-dgx-a100-1N.dat
   4. 
   5. :(

Link dump

<https://medium.com/syncedreview/tensorflow-pytorch-or-mxnet-a-comprehensive-evaluation-on-nlp-cv-tasks-with-titan-rtx-cdf816fc3935>

<https://github.com/HewlettPackard/dlcookbook-dlbs>

<https://lambdalabs.com/blog/2080-ti-deep-learning-benchmarks/>

<https://ai-benchmark.com/alpha.html>

<https://timdettmers.com/2020/09/07/which-gpu-for-deep-learning/>

<https://www.forbes.com/sites/karlfreund/2021/04/21/nvidia-dominates-a-near-empty-field-in-ai-benchmarks-again/>

<https://lambdalabs.com/blog/choosing-a-gpu-for-deep-learning/#imagemodels>

<https://biodatamining.biomedcentral.com/articles/10.1186/s13040-017-0154-4>

<https://ngc.nvidia.com/catalog/containers/nvidia:hpc-benchmarks>

<https://www.servethehome.com/dual-nvidia-geforce-rtx-3090-nvlink-performance-review-asus-zotac/5/>

<https://infohub.delltechnologies.com/p/running-the-mlperf-tm-inference-v1-0-benchmark-on-dell-emc-systems/>

<https://www.hpcwire.com/2019/04/09/digging-into-mlperf-benchmark-suite-to-inform-ai-infrastructure-decisions/>

<https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/install-guide.html>

<https://www.pugetsystems.com/labs/hpc/TensorFlow-Performance-with-1-4-GPUs----RTX-Titan-2080Ti-2080-2070-GTX-1660Ti-1070-1080Ti-and-Titan-V-1386/>

<https://www.netlib.org/benchmark/hpl/tuning.html>

Problem Statement

Graphics processing units (GPUs) have been increasingly used in various IT aspects, such as Artificial Intelligence (AI), Machine Learning (ML) as well as Information Security (IS), with an example being hash cracking. With increased usage of GPU in these fields, it is important to find a suitable approach to efficiently complete these tasks.

To correctly identify the most efficient method to conduct these tasks, a proper standardization of benchmarks should be identified and carried out to various approaches such as the usage of a multi-GPU system or a traditional approach of GPUs being plugged into the computer directly. Analysis as well as documentation of different results would help the team in understanding more and will give us an edge to optimize the usage of GPUs in these related fields.

Justification for using which benchmark tools

**Artificial Intelligence**

Kinda clashes with ML, maybe can read up more

**Machine Learning**

[https://www.run.ai/guides/gpu-deep-learning/#:~:text=GPUs%20can%20perform%20multiple%2C%20simultaneous,without%20sacrificing%20efficiency%20or%20power.](https://www.run.ai/guides/gpu-deep-learning/%23:~:text=GPUs%20can%20perform%20multiple%2C%20simultaneous,without%20sacrificing%20efficiency%20or%20power.)

<https://towardsdatascience.com/what-is-a-gpu-and-do-you-need-one-in-deep-learning-718b9597aa0d>

<https://www.quora.com/Why-is-GPU-useful-for-machine-learning-and-deep-learning>

- Memory bandwidth

- Dataset size

- Optimization

**Hash Cracking**

<https://radove.medium.com/hackers-use-gpu-technology-to-crack-passwords-ec6bf1ba4fc7>

<https://www.linuxjournal.com/content/hack-and-password-cracking-gpus-part-iii-tune-your-attack>

<https://security.stackexchange.com/questions/32816/why-are-gpus-so-good-at-cracking-passwords>

GPU has a lot of cores inside it -> each core can compute one 32bit arithmetic operation per clock cycle as a pipeline (data processing elements connected in series, where the output of one element is the input of the next one. Basically executed in parallel), which makes it more efficient as the GPU shares instruction decoding as well

Tools used

**Hashcat**

[https://nikitushka.github.io/passwhttps://www.run.ai/guides/gpu-deep-learning/#:~:text=GPUs%20can%20perform%20multiple%2C%20simultaneous,without%20sacrificing%20efficiency%20or%20power.ords.html](https://nikitushka.github.io/passwords.html)

<https://security.stackexchange.com/questions/211999/estimating-password-cracking-speed-based-on-gpu>

<https://tutorials.technology/blog/08-Hashcat-GPU-benchmarking-table-Nvidia-and-amd.html>

<https://security.stackexchange.com/questions/152339/password-cracking-speeds-according-to-hashcat>

shows the cracking speed of gpu, different gpus have different speeds

**AI-Benchmark**

<https://ai-benchmark.com/ranking_deeplearning.html>

<https://ai-benchmark.com/tests.html>

Detailed benchmark sections – check links for information

In the end, it will generate an AI score, based on different neural networks

**Hashcat**

https://github.com/siseci/hashcat-benchmark-comparison/blob/master/4x%20Nvidia%20GTX%201080%20TI%20Hashcat%20Benchmark

**LinPack**

<https://en.wikipedia.org/wiki/LINPACK_benchmarks>

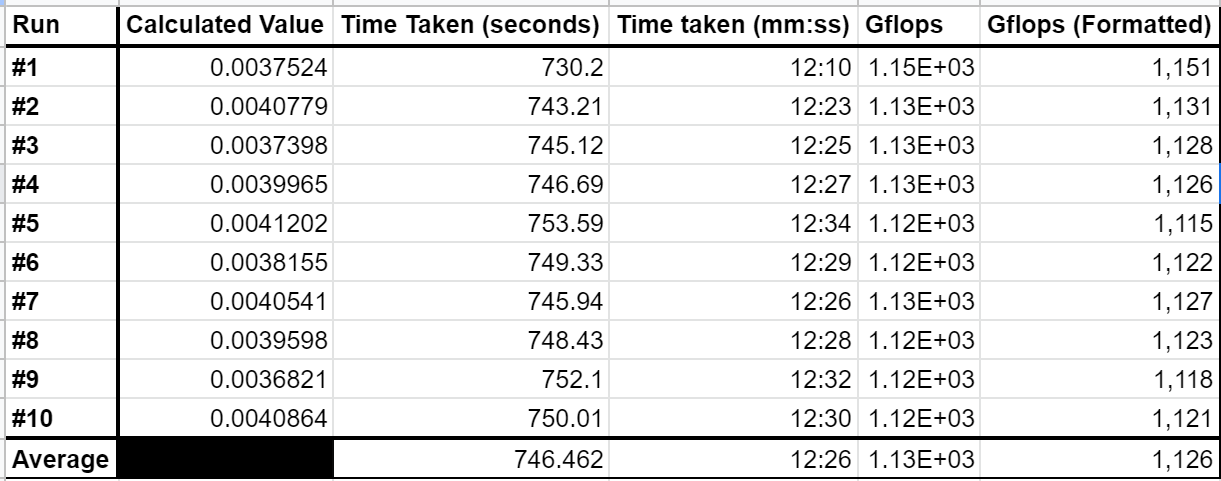
<https://www.researchgate.net/publication/227060498_Optimizing_Linpack_Benchmark_on_GPU-Accelerated_Petascale_Supercomputer>

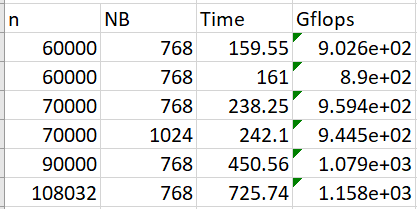
<http://www.netlib.org/utk/people/JackDongarra/faq-linpack.html#:~:text=The%20Linpack%20Benchmark%20is%20a,dense%20system%20of%20linear%20equations.&text=The%20benchmark%20stated%20as%20an,Guide%20was%20published%20in%201979>.

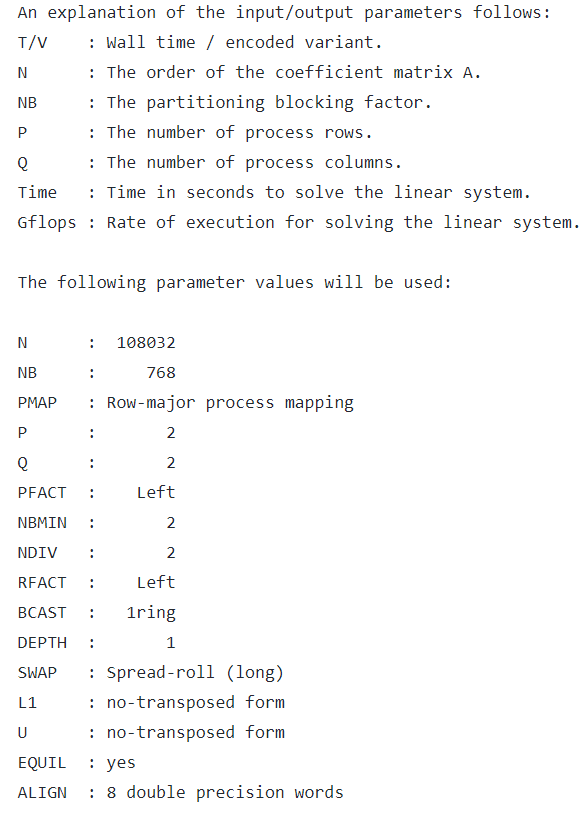
<http://www.netlib.org/utk/people/JackDongarra/PAPERS/The-LINPACK-Benchmark-An-Explanation.pdf>

**xHPL**

[Results]







The HPL.dat file has variables that can be modified so as to perform better (produces higher GFLOPs). The variables which can be modified are as follows:

* N: Indicates the number of problem sizes to execute
* NB: Indicates the block size to run
* P: Number of process rows. (P x Q) should equal to the number of processes
* Q: Number of process columns. (P x Q) should equal to the number of processes

The purpose of our experiment is to compare the results of the performance from different environment setups. Hence, we will not be looking at changing the variables to achieve highest possible performance.

[Comparison?]

Dawnbench

<https://www.pugetsystems.com/labs/hpc/How-To-Install-TensorFlow-1-15-for-NVIDIA-RTX30-GPUs-without-docker-or-CUDA-install-2005/>

**Baidu Deepbench**

Released in 2016, the microbenchmark suite DeepBench, analyzes basic, low-level operations like matrix multiplication and dealing with recurrent layers. It handles both inference and training. DeepBench’s main focus is on assessing hardware performance at the kernel level. Its goal is to benchmark the deep learning model’s underlying operations.

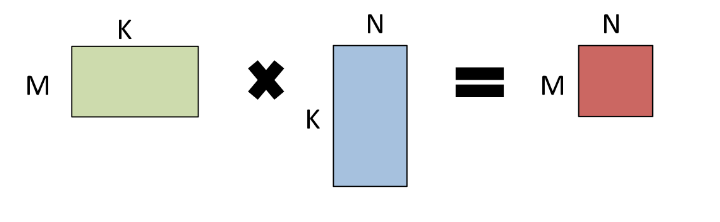
Its operational features cover a wide range of applications in machine learning. Some of which that are mentioned includes DeepSpeech, Language Modeling, Machine Translation, Speaker Identification, etc.

Operational Features

* **Dense Matrix Multiply**

Most neural networks today employ the use of Dense Matrix Multiplication. Its purpose is to establish fully linked layers and vanilla Recurrent Neural Networks (RNN). They also serve as the building blocks for the other recurrent layers.

Optionally, one or both of A and B can be transposed when executing the (General Matrix Multiply) GEMM operation A \* B = C. A common example widely used for better understanding can be represented by a matrix problem like the triple (M, N, K) shown below. Note the sizes of the matrices included in the operation as well as the “op” which informs us of any transposed matrices. The image also shows how the triple (M, N, K) correlates to the sizes of the multiplied matrices.



* **Convolutions**

Convolutions account for the great majority of flops in networks that work on images and video. They are essential aspects of networks like speech and natural language modeling, making them the single most significant layer in terms of performance. Having 4 or 5 dimensional inputs or outputs means that there are a lot of different ways to order these dimensions. This benchmark is interested in the performance in NCHW format, thus producing data that is presented in pictures, feature maps, rows, and columns.

* **Recurrent Layers**

Recurrent Neural Networks (RNN) are a type of [Neural Network](https://www.geeksforgeeks.org/tag/neural-network/) consisting of recurrent layers where the inputs from the current layer are derived from outputs of the previous layer. Inputs and outputs from a traditional neural network are typically independent of one another. However, there can be circumstances such as when predicting the next word of a phrase, the preceding words are necessary to be stored.

DeepBench includes support for three types of recurrent layers; vanilla RNNs, Long Short-Term Memory layers (LSTM) and Gated Recurrent Unit (GRU).

* **All-Reduce**

In this age, neural networks are frequently trained on many GPUs or numerous systems (with multiple GPUs). This can be done either through asynchronous or synchronous procedures. Synchronous methods ensure that all instances of the model maintain the same copy of the gradients before performing an optimization step. Hence, this method relies on keeping the parameters on all instances of the model to be synchronised. All-Reduce is the Message Passing Interface (MPI) primitive which is commonly used to execute this function. Depending on the number of ranks, the size of the data, and the network topology, there are a variety of approaches to implement All-Reduce.

<https://github.com/baidu-research/DeepBench#types-of-operations>

DeepBench from Baidu is an open-source benchmark covering both training and inference. DeepBench focuses on measuring the performance of basic operations in neural network libraries. It aims at determining the most suitable hardware for specific operations, and communicating requirements to hardware manufacturers. DeepBench tries to find the most suitable hardware that can provide the best performance on the basic operations used in deep neural networks. Baidu DeepBench can test Deep Learning operations on different hardware and architectures.

<https://link.springer.com/chapter/10.1007/978-3-030-32813-9_5>

Who uses it?

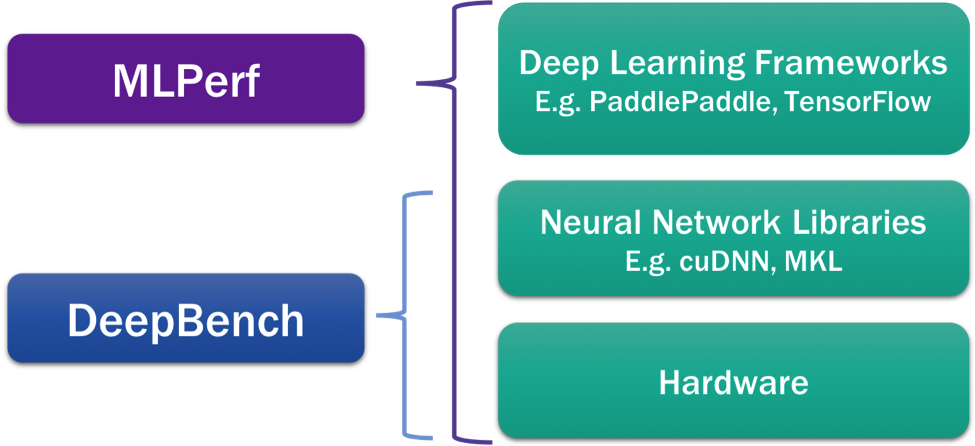
Intel, Nvidia, Graphcore and AMD are among the organizations that have implemented DeepBench.

<http://research.baidu.com/Blog/index-view?id=100>

Why to use or not to use?

Although Baidu DeepBench can operate on a variety of hardware platforms, it is worth noting that it is a difficult benchmark to utilise CPUs with a non-x86 architecture. This is because only some ARM CPUs can work with DeepBench. DeepBench’s code as well as the supporting libraries on ARM CPUs have to be modified due to the many bugs present when attempting to run the benchmark. Furthermore, the datasets used on the many different hardware were not consistent with one another. As a result, this makes for an inaccurate result when comparing the performance.

Given that DeepBench is an old benchmark, Baidu has worked hand-in-hand with other companies to release the newer benchmark known as MLPerf. MLPerf measures the time required to achieve a given accuracy of a deep learning model trained on a fixed dataset. MLPerf assesses the time it takes for a deep learning model that is trained using a fixed dataset to reach an acceptable level of accuracy. This method allows benchmark model performance across different hardware systems.



<https://cs.stanford.edu/~matei/papers/2017/nips_sysml_dawnbench.pdf>

<https://rajpurkar.github.io/SQuAD-explorer/>

<https://opendatascience.com/what-is-mlperf/>

<https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270>

<https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/>

<https://en.wikipedia.org/wiki/Google_Neural_Machine_Translation>

<https://owdt.com/the-big-breakthrough-in-digital-language-translation/>

<https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html>

<https://arxiv.org/pdf/1910.01500.pdf>

<https://alittlepain833.medium.com/simple-understanding-of-mask-rcnn-134b5b330e95#:~:text=Mask%20RCNN%20is%20a%20deep,bounding%20boxes%2C%20classes%20and%20masks>.

<https://developers.arcgis.com/python/guide/how-maskrcnn-works/>

<https://paperswithcode.com/method/mask-r-cnn>

<https://developers.arcgis.com/python/guide/how-maskrcnn-works/>

<https://alittlepain833.medium.com/simple-understanding-of-mask-rcnn-134b5b330e95#:~:text=Mask%20RCNN%20is%20a%20deep,bounding%20boxes%2C%20classes%20and%20masks>.

<https://paperswithcode.com/method/mask-r-cnn>

<https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173>

Dawnbench

* What is dawnbench (overview)

DAWNBench is a benchmark suite that is used to benchmark GPUs based on end-to-end deep learning training as well as inference. The reference used for these benchmarks to quantify how each system fares are the training time, training cost and inference cost across different models such as Canadian Institute For Advanced Research (CIFAR10) as well as Stanford Question Answering Database(SQuAD) which are models focused on Image Classification as well as Question Answering.

DAWNBench started out as a project targeted to

DAWNBench was also one major part of another benchmark’s success - MLPerf. DAWNBench has then ceased taking submissions from the community in early 2020 to consolidate efforts in the benchmarking community to allow MLPerf to be the industry standard for measuring machine learning system performances.

* Features
  + Image Classification

Image Classification is the ability for the computer to analyze an image and assign it a classification based on what the computer perceives it falls under. Some simple classifications would include: ‘animal’, ‘human’, ‘vehicles’ and so on.

In deep learning, the computer makes use of an artificial neural network to process information, which in the case of image classification - the images. The neural network will first go through training, where lots of information will be accessed by the neural network. For instance, we can train the network to identify ‘vehicles’ by loading a massive collection of pictures of ‘vehicles’ to the network. The network will then take these images and attempt to find patterns from these images. The network will then identify and assign weights based on the images that the network received to deem it more or less important when compared to the rest of the samples. Once a network is trained well enough, it should be able to identify images of ‘vehicles’ when a new image is being presented to the network.

* + Question Answering
* Who uses it

Many companies make use of DAWNBench to access their systems. Companies such as Alibaba, Apple as well as Huawei have used and contributed to DAWNBench with results of their benchmarks which can be found on the github page of stanford-futuredata.

* Why to use and why not to use

MLPerf

* What is MLPerf

MLPerf is a benchmark suite that can be used to measure how fast systems can train models to achieve a targeted quality metric. It is a necessary tool to allow companies all over the world to create and test out their product to check how their product fares against other existing products.

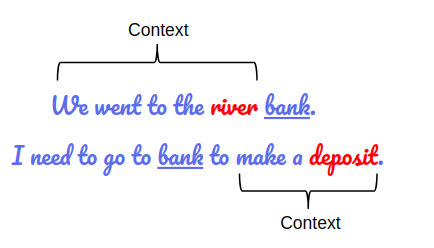
MLPerf’s main purpose is to have a broad approach to machine learning that can be supported by both industry and research academia. MLPerf has over 40 organisations backing and deciding together on a consistent set of benchmarks, which can be seen as an attempt to keep benchmarking consistent in the field of machine learning.

* Features
  + **Bidirectional Encoder Representations from Transformers (BERT)**

One of the benchmark tests in MLPerf involves BERT, with the main goal being to generate a language model. This model is designed to pre-train deep bidirectional representations from texts that it is being provided with by jointly conditioning on both the left and right context.

How BERT mainly works is that it makes use of a Transformer, which learns contextual relations between words that are by its side. The transformer has two main roles, which is to read the text input via an encoder. BERT’s encoder reads the whole sequence of text to allow BERT to learn the context of a word based on the entirety of the text.

One example of why taking the entirety of the text is important in this model would be that in English for example, there are multiple meanings for a single word, the image below would be an example of when the left and right context makes a difference in training the model.



If BERT takes in the first context the nature of the word ‘bank’, the word would not make sense for the model in the next sentence. BERT avoids this problem by taking in the whole sentence and processing it before making a prediction and helping to train the model.

To allow ease of training the model, BERT is pre-trained with reliable sources - Wikipedia as well as Book Corpus, with a grand total of 3,300 million words in total. This gives the model a general and brief context of words to help it learn better with new texts.

To allow BERT to understand and learn, the model takes in each word as an input embedding. There are three types of embeddings: Position embeddings, segment embeddings as well as token embeddings. The position embedding takes note of the position of words in a sentence to learn and use the words in that particular order. Segment embeddings take in sentence pairs as inputs to learn the different ways the same words can be used such that it takes note of a unique embedding and learns from it. Token embeddings are mainly vocabulary words that are learned to further describe items.

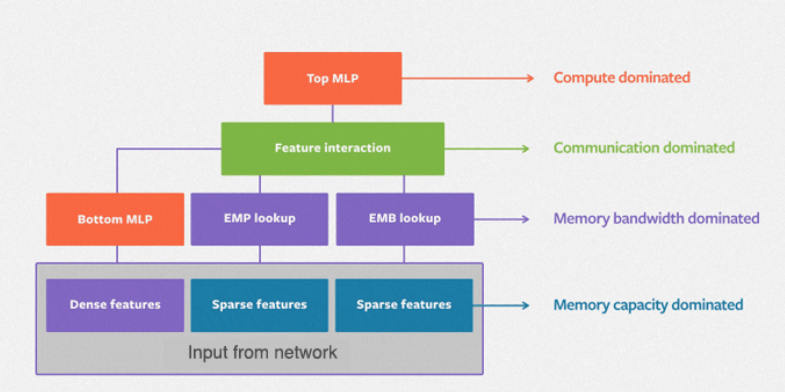
How BERT is being used in MLPerf is that it takes the time for the model to learn based on a specific dataset but on the same seed to ensure consistency. Training data will be created with the benchmark and the evaluation samples will be taken from the training data that was created. Evaluation will be taken if it reaches above a specific amount of accuracy and the score is generated by accuracy as well as time taken to train the model.

* + **Deep Learning Recommendation Model (DLRM)**

DLRM is a neural network based recommendation model that is used to build recommendation systems in different environments. One example of this would be Facebook, where the neural network learns of what the users are looking for and recommend items based on it. The model makes use of collaborative filtering as well as predictive analytics based approach to work efficiently in providing recommendations.

One example of DLRM usage in real life would be social media websites, where the algorithm detects websites that the user uses and understands what the users wants by displaying products or advertisements related to it.

The model makes use of a layered approach, where the model goes from the bottom up. The bottom layer processes categorical features using embeddings as well as continuous features, which are processed with a bottom multilayer perceptron (MLP). The middle layer takes care of feature interactions, which is mainly about communication between the bottom and the top. The top MLP will process the result and pass the result through a function to give a probability of recommendation.



Benchmarking with this DLRM model would be using PyTorch, where it keeps track of the speed at which the model performs as well as the accuracy. The benchmark makes use of custom generation of indices according to the categorical features found in the lower tier of the model. The speed of which the model performs on is heavily dependent on architectural heterogeneity, which is how the layout of the hardware that is available. Parallelism also affects the speed of computation as the model used to address memory capacity requirements of embeddings is critical for most MLP models.

* + **Google Neural Machine Translation system (GNMT)**

GNMT is a neural machine translation system developed by Google that makes use of an artificial neural network to increase the accuracy in Google Translate. The benchmark used in MLPerf with GNMT is the language translation from a source language to a targeted language.

GNMT makes use of an example-based machine translation(EBMT), which learns by taking in as many examples as possible. With this method, the model can be seen as getting better as time progresses with more data, assuming that the examples that the model receives are true. The model also takes in sentences instead of words or single phrases as words and phrases have different meanings.

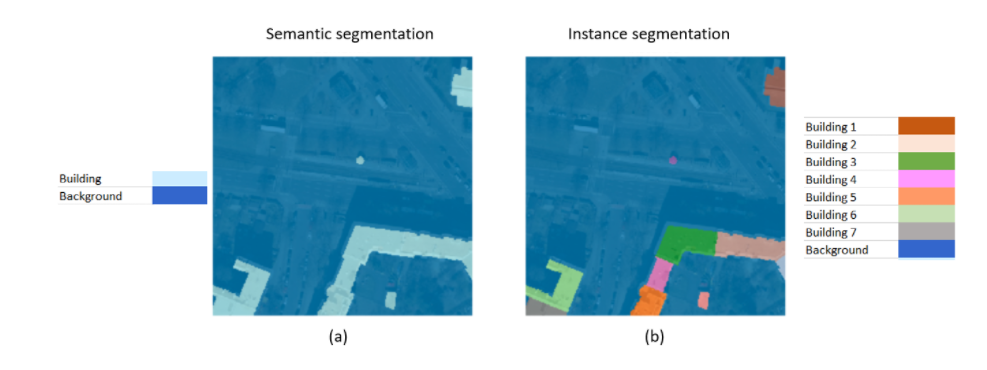
A clear example of GNMT being put in use would be on Google Translate, as it is a system developed by Google for this exact purpose. The translation program is able to pick up subtleties of languages according to people from different regions of the world, which is a result of gaining example sample data as mentioned above.

The benchmark makes use of two specific languages. In MLPerf, it takes a source language as English and a target language as German. For consistency, these parameters are kept constant. The datasets will be the same for any benchmarking done, training will be done, where the model takes in the two dataset and searches for similarities and attempts to translate the texts based on the dataset it receives. The model will keep taking in the datasets and learn, and then be evaluated after a certain amount of steps.

The benchmark makes use of the Bilingual Evaluation Understudy (BLEU) as a metric for this test. BLEU is a method for evaluating a generated sentence to a reference sentence, this metric is developed to evaluate predictions made by automatic machine translation systems and is a trusted metric source by machine learning advocates.

* + **Mask Region-Based Convolutional Neural Network (Mask R-CNN)**

Mast R-CNN is a type of object instance segmentation which integrates object detection alongside semantic segmentation. This allows the model to detect objects in an image and separate them into defined instances based on pixels. An example would be shown below.



From image a, semantic segmentation separates the backgrounds with the buildings due to the nature of semantic segmentation. However in image b, each building is able to be differentiated from one another as a separate entity. This is the main reason why Mask R-CNN is different from the previous image detection algorithms.

How Mask R-CNN works is that it makes use of two separate networks - one of them being the backbone which is in charge of creating shortcuts between layers to skip connections, one example would be ResNet. The other network would be a region proposal network, which gives the model a set of region proposals, where the model attempts to separate instances based on the image. Mask R-CNN then predicts bounding boxes and object class for each of the regions that has been detected by the model. In addition to this, Mask R-CNN also predicts segmentation masks on each region, which creates layers and generates a mask for each region.

The benchmark makes mean average precision (mAP) for both mask and box as metrics for the benchmark. <may need to elaborate more here>

* + **Minimalist Go Engine (minigo)**

Minigo is a benchmark model that is focused on reinforcement learning. Go is a 19 by 19 board game played by two people. In this benchmark, a 9 by 9 model is used instead.

The model will first play the game against itself to produce the first few steps of the game, as well as to learn board positionings for this training. The model will then self play data with previous models. In order to be the best model, the current best model will play against the most recently trained model. The model must win 55% of the games for it to become the current best model.

The benchmark model takes reference from records of human games to start off with professional players in real life Go. The benchmark checks for position prediction of where the model thinks its the best move. The target that the model is trying to achieve is 40% move prediction rate.

* + **Residual Neural Network (ResNet)**

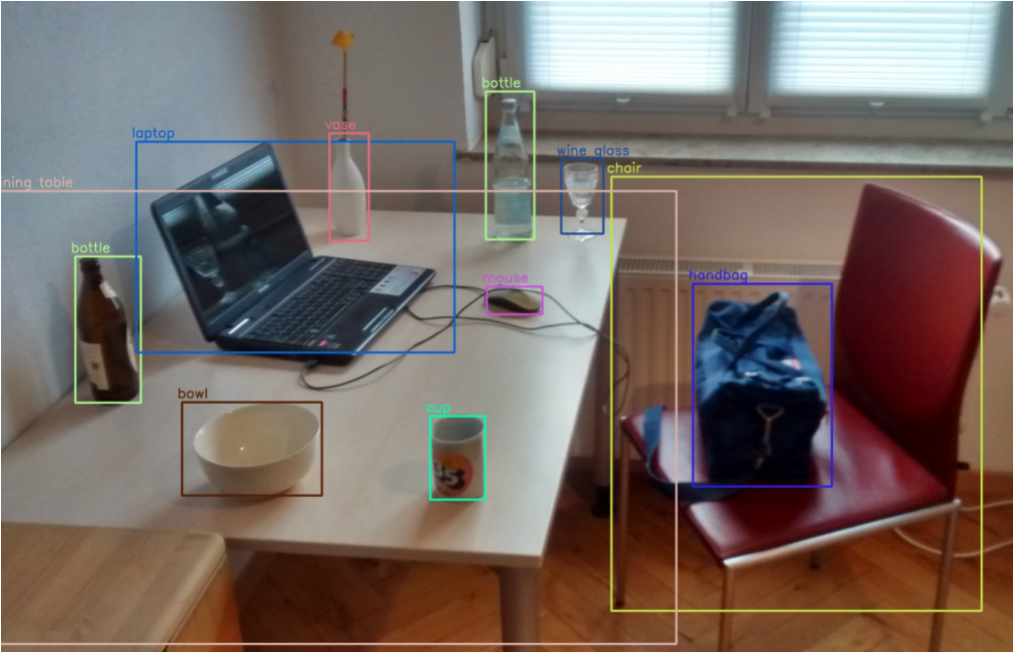
MLPerf makes use of ResNet for image classification benchmark tests. The model will do some data preprocessing, where images go through steps to make sure that the images are uniform, such as making the images the same size as well as doing some morphological transformations on the image itself.

ResNet makes use of layers as well as weights to perform deep residual learning. The model will take the dataset - in MLPerf’s case, from ImageNet, and load the image as well as the classification for the model to learn. This continues on for the rest of the ImageNet database to allow the model to learn. The more images the model is able to take in and learn, the more accurate the classification will be.

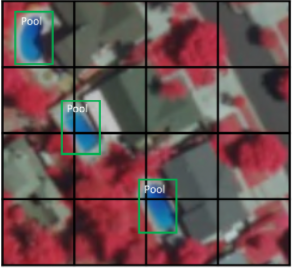
The benchmark metric is based on the amount of correct classifications on the ImageNet test dataset after training. The target for this benchmark is 0.759.

* + **Single Shot Detector (SSD)**

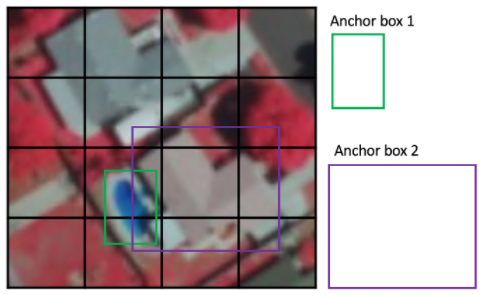
Single Shot Detector is an object detection network, one good example would be that the model will be able to identify the objects inside a single photo and output the new image with boxes surrounding the item. One example would be as shown below:



SSD makes use of the grid cell method, where the model divides the image using a grid and each grid cell will find objects that are within it. The model will then go through the image cell by cell and mark it as object or background. Background means that there are no objects within it and the model can safely ignore that particular cell. The model will then go back to the cells with objects and output the position and shape of the object the cell contains.



There may be times where there are multiple objects in a cell, or where the full object takes up more than a single cell, what the model does is that it would make use of an anchor box, where each anchor box are predefined and each one is responsible for a specific size and shape within a cell.



The benchmark is gauged by using COCO box mAP, which is a metric in measuring the accuracy of object detectors like SSD.

* Who uses it

As of today, many prominent companies in the field of machine learning have used MLPerf as their benchmark for their products. Companies such as Nvidia, Fujitsu, Google as well as Intel have their results posted on training results repository of the mlcommons github page, which can be found on: <https://github.com/mlcommons/training_results_v0.7>

* Why to use and why not to use
  + Recognized benchmark

MLPerf is an industrially recognized benchmark with companies such as Nvidia, Qualcomm and Dell using it to publish their results from their products. One good example would be on Nvidia, where their results are published on their website, with references to MLPerf on their product site. There are countless amounts of research papers making use of MLPerf to justify the improvements made on the AL/ML fields.

By making use of a recognized benchmark that the industry approves, our research done in this project will prove to be more trustworthy.

* + Comprehensive

MLPerf contains 8 different benchmark tests, ranging from image classification to reinforced learning. Each benchmark serves their own purpose, covering different areas and aspects of AI and ML.

* + Unsupported hardware

As MLPerf is an industrially recognized benchmark, most hardware tests that have been done and posted on the MLPerf repository are high-end GPUs that are made specifically for AI/ML tasks. Commercially available GPUs such as multiple GTX 1080s in a specific setup do not have results based on that on the github repository. This could be a potential challenge as we are unable to reference and infer the results and compare them with existing ones to attempt to make sense out of it.

AI-Benchmark

* What is AI-Benchmark (overview)

AI-Benchmark is an open source python library whose purpose is to evaluate AI performance over different hardware setups(CPU,GPU,TPU). The goal of AI-Benchmark is to create a professional, accurate and lightweight tool for the field of Artificial Intelligence as well as Machine Learning. AI-Benchmark also hopes to do their part in setting a professional standard in standardizing the ever growing field of AI and ML in terms of benchmarking to fairly evaluate hardware based on fair tests.

* Features
  + **MobileNet-V2**

AI-Benchmark is capable of running on mobile phones. It introduces MobileNet which is a convolutional neural network (CNN) meant to perform on mobile devices. MobileNet’s goal is to conduct image classification. With MobileNet-V1, it uses lightweight “Depthwise Separable Convolution” which significantly lowers the model size of the network and the complexity cost to cater to the low computational power of mobile devices. Yet, this is further enhanced with MobileNet-V2 which is built on an inverted residual structure instead. Another improvement was the removal of non-linearities in the narrow layers. MobileNet-V2 is also able to perform object detection and segmentation.

<https://arxiv.org/pdf/1704.04861.pdf>

<https://arxiv.org/pdf/1801.04381.pdf>

* + **Inception-V3**

The Inception model convolutional neural network (CNN) was designed for image classification. The model was created to address concerns such as computational cost and overfitting, to name a few. How Inception works is that it performs a convolution on three different filter sizes (1x1, 3x3, 5x5) followed by max pooling. Concatenated outputs are then transmitted to the next tier.

<https://arxiv.org/pdf/1512.00567.pdf>

* + **Inception-V4**

There are several versions of its architecture like the ‘Inception-V3’, ‘Inception-V4’ and ‘Inception-ResNet-V2’ mentioned in this paper. Each version of the model till Inception-V4 has been an upgrade in order to solve issues with its previous versions such as the vanishing gradient problem.

<https://arxiv.org/pdf/1602.07261.pdf>

* + **Inception-ResNet-V2**

Just like its name suggests, this architecture of Inception was the result of Inception being upgraded to include residual connections. This is because doing so substantially speeds up the training of Inception networks. Inception-ResNet-V2 was found to have better image recognition performance than its predecessor. Despite that, its performance is almost identical to Inception-V4.

<https://arxiv.org/pdf/1602.07261.pdf>

* + **ResNet-V2-50**

Residual networks (ResNet) focuses on doing image classification, detection, localization and segmentation with the help of residual units. It utilizes identity skip connections also known as shortcuts which help to reduce time complexity. A high time complexity is the adverse effect of having a very deep network. Enhancements to the ResNet training model (ResNet-V2) have made training easier, improved generalization and produced a further decrease to the rate of error.

<https://arxiv.org/pdf/1603.05027.pdf>

* + **ResNet-V2-152**

As with VGG, the numbers ‘50’ and ‘152’ represent the difference in the number of layers that the residual network adopts. Although ResNet-V2-50 is the more popular option, ResNet-V2-152 has been shown to perform better and yielded an even lower error rate.

<https://arxiv.org/pdf/1603.05027.pdf>

* + **VGG-16**

Similar to VGG-19, VGG-16 is another very deep convolutional neural network albeit with 16 layers instead of 19. However, AI-Benchmark uses it for image classification, categorising images into a huge number of object categories. The model is being trained on images taken from the ImageNet database.

<https://arxiv.org/pdf/1409.1556.pdf>

* + **SRCNN 9-5-5**

SRCNN 9-5-5 performs image super-resolution via a fully convolutional neural network. The SRCNN model is generally made up of 3 components which are the feature extractor, non-linear mapping and the reconstruction process. The model is trained to reduce the pixel-wise cumulative squared error between the reconstructed and actual images. 9-5-5 is one of the many model architectures which aims to provide an optimal balance between performance and speed.

<https://arxiv.org/pdf/1501.00092.pdf>

<https://medium.com/analytics-vidhya/srcnn-paper-summary-implementation-ad5cea22a90e>

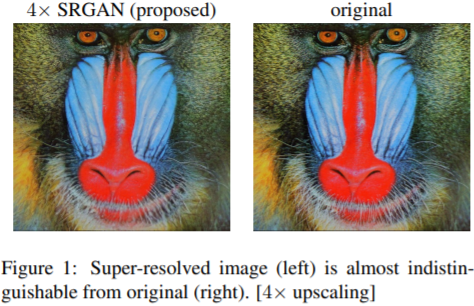
* + **VGG-19**

VGG-19 is another single-image super-resolution approach that uses a very deep convolutional neural network to solve the challenge of producing a high-resolution picture from a low-resolution image. The ‘19’ in Visual Geometry Group-19 (VGG-19) represents the number of weight layers that the model uses in training. However, AI-Benchmark uses an improved version of the model with 20 layers.

<https://arxiv.org/pdf/1511.04587.pdf>

* + **ResNet-SRGAN**

For this, ResNet is implemented with SRGAN to produce high-resolution natural-looking photorealistic pictures even when upscaled. SRGAN is described as conducting Super-Resolution (SR) using a Generative Adversarial Network (GAN). In order to further understand this, it is best to split SRGAN into two separate terms. Essentially, ResNet-SRGAN has the goal of generating a high-resolution image from its low-resolution equivalent and this process is known as Super-Resolution. Meanwhile, GAN puts two neural networks (namely the generator and discriminator) against one another with the sole purpose of generating fresh, synthetic instances of the image that is as identical to the real high-resolution image.



<https://arxiv.org/pdf/1609.04802.pdf>

<https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV1011/Super_Resolution_CVonline.pdf>

<https://wiki.pathmind.com/generative-adversarial-network-gan>

* + **ResNet-DPED**

ResNet-DPED seeks to solve the photo enhancement problem. Ordinary photographs are converted to DSLR-quality images for the image enhancement. It utilises the DPED dataset which is a large dataset of real photos captured from a high-end reflex camera along with three different phones. Using the photos captured with a phone as input, it trains and outputs a comparable image in high-quality as if it was captured from a DSLR. The output is compared to the actual DSLR-quality photo with an aim to minimise the loss function. The loss function is determined by colour, texture and content.

<https://arxiv.org/pdf/1704.02470.pdf>

* + **U-Net**

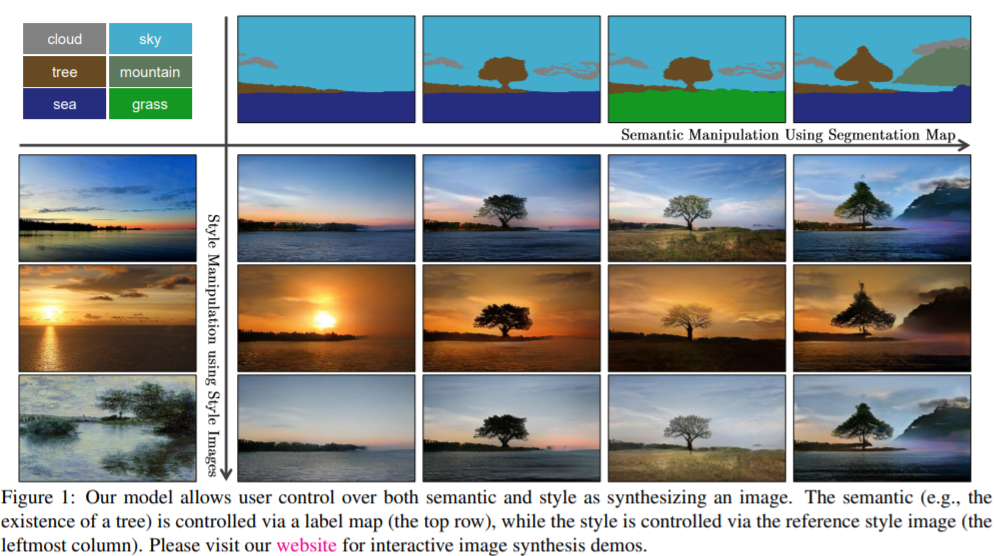
Worth noting that U-Net is a convolutional neural network (CNN). CNNs are typically used for image classification. As such, U-Net was created with the sole purpose of processing biomedical images. It aimed to resolve issues relating to the determining presence of diseases as well as the identifying the location of abnormalities.

<https://arxiv.org/pdf/1505.04597.pdf>

<https://towardsdatascience.com/unet-line-by-line-explanation-9b191c76baf5>

* + **Nvidia-SPADE**

The goal of Nvidia-SPADE is to perform semantic image synthesis in order to create photorealistic pictures (fig below). This allows for the generating of content as well as image editing. Nvidia-SPADE employs an optimal method of a spatially adaptable, learnt transformation that uses the input layout to modulate the activations in normalizing layers.



<https://arxiv.org/pdf/1903.07291.pdf>

* + **ICNet**

Also known as Image Cascade Network (ICNet), it focuses on performing semantic segmentation in real-time. In order to achieve the best possible combination of speed and accuracy, ICNet employs the cascade feature fusion unit to swiftly accomplish high-quality segmentation on both high-resolution and low-resolution images.

<https://arxiv.org/pdf/1704.08545.pdf>

* + **PSPNet**

Known as Pyramid Scene Parsing Network (PSPNet), its objective is to assign a category name to each pixel in the picture.This process is aided by the pyramid pooling module which serves as the main driving force of this model. What this module does is to capture the overall context in the picture. By providing a context of the scene, this enables it to categorize the pixels based on the picture's overall information. This results in an improvement on the ability of the model to accurately predict the type of label or location.

<https://arxiv.org/pdf/1612.01105.pdf>

* + **DeepLab**

DeepLab focuses on performing semantic image segmentation. This is the process of applying a semantic label to each and every pixel in a picture. Some examples of the labels can include objects like car, flower, etc. Thus, this works to identify and classify all the objects in the pictures. In particular, AI-Benchmark employs the DeepLab-CRF method. This considers the use of a Deep Convolutional Neural Network (DCNN) in conjunction with a fully-connected Conditional Random Field (CRF). Via the CRF method, it is possible to capture the finer details in pictures, resulting in high resolution segmentation while simultaneously accounting for long-range dependencies.

<https://arxiv.org/pdf/1502.02734.pdf>

<https://arxiv.org/pdf/1606.00915.pdf>

* + **Pixel-RNN**

Pixel-RNN is a neural network that incorporates many LSTM layers and sequentially predicts pixel values in two-dimensional pictures. What this means is that, given a set of natural images, it learns from it, predicts and generates images from the predictions. AI-Benchmark makes use of Pixel-RNN in image inpainting; filling in the missing data in pictures. This thus serves to repair damaged or incomplete images.

<https://medium.com/a-paper-a-day-will-have-you-screaming-hurray/day-4-pixel-recurrent-neural-networks-1b3201d8932d>

<https://arxiv.org/pdf/1601.06759.pdf>

* + **LSTM**

Also mentioned as one of the features of DeepBench, it is created as a solution to short-term memory, is good at processing long sequences of information and present in most recurrent neural networks now. The decision to save or discard data is made by different gates and math operations within a LSTM cell. It’s cell state enables considerably longer storage of important information, thus proving to be an advantage over short-term memory. Data that is kept is presumed to be useful and is utilized to generate predictions. For AI-Benchmark, LSTM is used for sentiment analysis in sentences.

<https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>

<https://www.bioinf.jku.at/publications/older/2604.pdf>

* + **GNMT**

This was explained earlier as one of the features provided by MLPerf. Similarly here, AI-Benchmark uses GNMT to perform text translation on a set of input text; thus producing another set of text in another language that has the same or similar meaning to that of the text run by the model. Being the same feature used by MLPerf, this benchmark uses the same two languages, English and German. This is not forgetting the datasets and metrics which are not dissimilar from its MLPerf counterpart.

<https://arxiv.org/pdf/1609.08144.pdf>

* Who uses it

Based on crunchbase, a website that keeps track of business information about companies, over 10,000 organisations have made use of AI-Benchmark and listed them on their profile. AI-Benchmark is also open source, and submissions for test results from users of the benchmark from all over the world. There is an official forum for the benchmark and submission items can be approved after proper research analysis done stating the benefits for the tests recommended. There are many users around the world using this benchmark and data on the webpage is updated bi-weekly - ensuring the dataset is kept updated.

* Why to use and why not to use
  + Supported hardware with results
  + Comprehensive benchmark tests

AI Benchmarks contains 16 different benchmark

# 

# Hardware writeup

Motherboard: Asus X99-E WS/USB 3.1

4x PCIe 3.0 x16 slots

BIOS: Ver. 4001

CPU: Intel Xeon(R) E5-1650 v4 @3.60GHz

6 cores 12 threads

Memory: 131072 MB DDR4 2400 MHz

8x Crucial 16GB 2400 MHz

Storage: 2x Crucial\_CT480M500SSD1 480GB

Average sequential read (MB/s) 407

Average sequential write (MB/s) 366

GPU: 4x Asus TURBO-GTX1080TI-11G

PCIe 3.0

3584 CUDA cores

GDDR5X 11GB

11010 MHz

PSU: 1600W

PCIe Switch: PLX technology PEX 8614

<https://docs.broadcom.com/doc/12351827>

Port configuration for this card: 

Reported maximum latency of 140ns x4 to x1 configuration

Non-blocking internal switch architecture

Max payload size of 2048 bytes

Guaranteed error-free packets

Each port has auto-negotiation, lane reversal and polarity reversal

## Baseline setup

Directly attach the 4 GTX 1080 Tis to the motherboard.