TRƯỜNG ĐẠI HỌC BÁCH KHOA TP HCM
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

- - - - - - - - - - - -

TÍNH TOÁN SONG SONG (CO3067)

BÁO CÁO BÀI TẬP LỚN

Báo cáo tổng kết

Giảng viên hướng dẫn: PGS.TS Thoại Nam
ThS Nguyễn Mạnh Thìn

SV thực hiện: Lê Đức Mạnh 1611985
Văn Tiến Cường 1610386
Nguyễn Khắc Quang Huy 1611288

Tp. Hồ Chí Minh, Tháng 4/2019

Mục lục

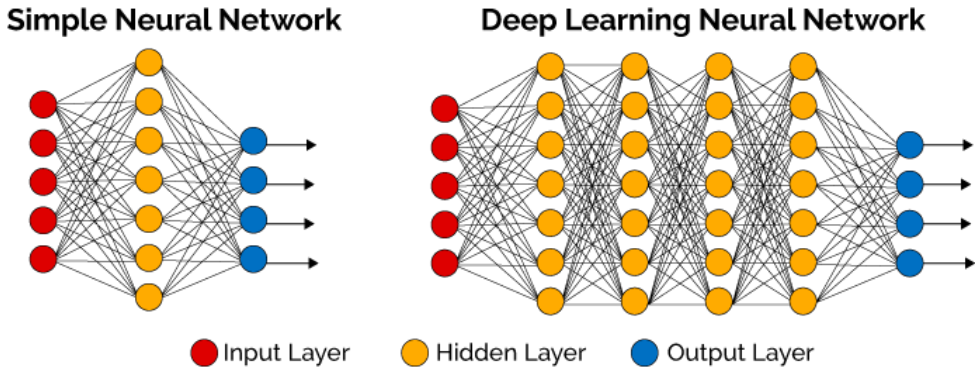
Danh sách hình vẽ

1 Giới thiệu về Deep Learning

1.1 Định nghĩa

Deep learning là một nhánh trong họ các phương pháp liên quan đến neural network [3].

Deep learning là một neural network gồm một hệ thống phân cấp nhiều lớp, trong đó mỗi tầng biến đổi dữ liệu đi vào thành một dạng thể hiện trừu tượng hơn. Những lớp này thể hiện các đặc tính (feature) của dữ liệu đầu vào được tạo ra bởi neural network. Lớp đầu ra cuối cùng kết hợp tất cả các đặc tính và đưa ra dự đoán. Deep learning càng nhiều lớp thì các có nhiều đặc tính trích xuất, do đó cần một lượng dữ liệu rất lớn.



Hình 1: *Deep learning và neural network, nguồn: [2]*

1.2 Ứng dụng

Deep learning đang được ứng dụng rộng rãi vào rất nhiều lĩnh vực khác nhau như thị giác máy tính, nhận dạng giọng nói, xử lý ngôn ngữ tự nhiên, nhận dạng âm thanh, dịch máy, đề xuất dịch vụ, ...

2 Giới thiệu về VGG16 và ResNet152

2.1 VGG16

Ra đời năm 2005, VGG có nhiều biến thể khác nhau: 11, 13, 16 và 19 layers. Ý tưởng của VGG như sau: Cứ sau 2 hoặc 3 convolution layer là sẽ có 1 max pooling layer. Ở đầu ra của mạng ta có những fully connected layer theo sau bởi 1 softmax layer. VGG cho kết quả tốt hơn so với một model khác là AlexNet vào ILSVRC 2014 (mạng AlexNet giành chiến thắng cuộc thi đó vào năm 2012).

VGG16 là VGG với 16 layer.

2.2 ResNet152

Là một model ra đời nhằm giải quyết vấn đề vanishing gradient trong deep CNN. Khi số lượng layer của một Deep CNN càng nhiều thì độ chính xác càng tăng. Tuy nhiên số lượng layer này quá nhiều (>50) thì độ chính xác lại giảm đi. Residual blocks ra đời để giải quyết vấn đề trên: cứ sau mỗi 2 layer, ta cộng input với output: F(x) + x. ResNet là một mạng CNN bao gồm nhiều residual block nhỏ tạo thành. ResNet giành chiến thắng cuộc thi ILSVRC 2015 với error rate 3.57%.

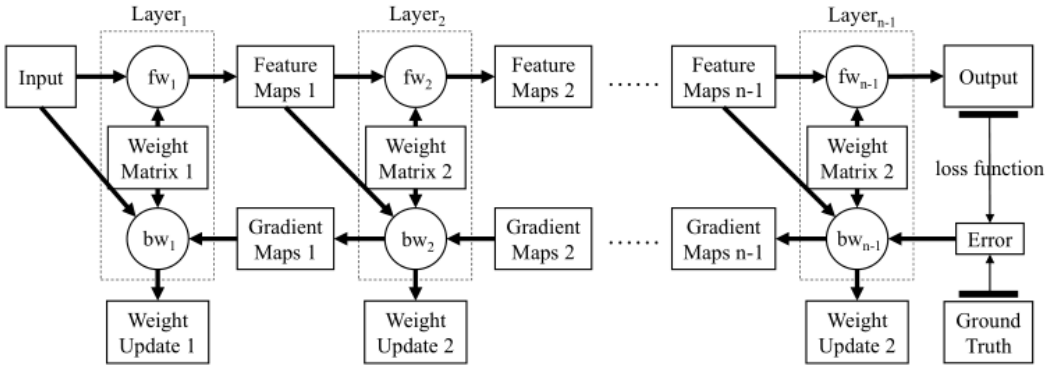
ResNet152 là ResNet với 152 layer.

3 Cơ sở lý thuyết

3.1 Forward pass và backward pass

Giải thuật train cho deep learning là một giải thuật lặp, gồm nhiều iteration, mỗi iteration gồm một forward pass và một backward pass. Ở forward pass, input là dữ liệu train/test sẽ được tính toán qua từng lớp của mạng neural. Kết quả thu được ở forward pass sẽ được so sánh với giá trị thực để tính ra giá trị sai

số (lost/cost), cũng là input ở backward pass. Ở backward pass, việc tính toán cũng được thực hiện qua từng lớp của mạng neural như ở forward pass, nhưng theo chiều ngược lại.



Hình 2: *Forward pass và backward pass, nguồn: [7]*

3.2 Inference và training

Inference nói đến một forward pass, trong khi training nói đến một forward pass và một backward pass. Khi deep learning được triển khai trên một hệ thống nào đó (sau khi đã hoàn tất việc train), chỉ có thời gian inference được quan tâm, vì nó ảnh hưởng đến chất lượng dịch vụ. Thời gian này càng nhỏ thì độ trễ cho việc xử lý dữ liệu cho ra kết quả càng thấp, phản hồi tới người dùng càng nhanh.

3.3 Stochastic gradient descent

Gradient descent [5] là một phương pháp để giảm sai số ở forward pass và tìm một điểm tốt ưu cục bộ. Stochastic gradient descent (SGD) [6] là một dạng của gradient descent, tuy nhiên thay vì tính toán trên toàn bộ tập dữ liệu (có thể rất lớn), chỉ một phần của tập dữ liệu (gọi là một batch) sẽ được tính toán đưa qua forward pass. Việc này có một số lợi ích như:

- Giảm thời gian tính toán cho mỗi pass.
- Khiến model hội tụ nhanh hơn với batch size hợp lý.
- Batch size thích hợp sẽ nằm vừa trong bộ nhớ của GPU, giảm thiểu thời gian chuyển dữ liệu từ bộ nhớ chính lên GPU, tận dụng tối đa hiệu năng của GPU.

Thí nghiệm được thực hiện trên nhiều batch size khác nhau để tìm ra batch size cho hiệu năng tốt nhất đối với mỗi cặp GPU/model.

4 Độ đo hiệu năng của các framework trên GPU

Thí nghiệm này đo và so sánh các framework dựa trên hiệu năng train/test các model, chứ không quan tâm đến độ chính xác của quá trình train/test.

Có nhiều độ đo về hiệu năng train/test của các framework đã được dùng, điển hình như:

4.1 Thời gian train một model

Là thời gian cần thiết để train một model một cấu hình cho trước. Mỗi framework thường có một bộ thông số mặc định về số lần chạy giải thuật (iteration, epoch, batch size) cho một model nào đó. Các thông số này đã được nhiều người thử nghiệm để đảm bảo model đạt được độ chính xác nhất định. Độ đo có thể đánh giá độ hiệu quả của framework/model tốt, giúp người dùng chọn lựa framework/model hợp lý để triển khai sao cho số tiền bỏ ra là tốt nhất. Tuy nhiên, việc train một model từ đầu đến hoàn thành là rất tốn kém và phức tạp.

4.2 Thời gian xử lý một batch dữ liệu

Là thời gian inference hoặc training một batch dữ liệu. Do thời gian này luôn tăng khi batch size tăng nên không có ý nghĩa nhiều khi so sánh thời gian trên nhiều batch size. Do đó, batch size thường được chọn cố định sao cho model hội tụ tốt nhất. Nhiều nghiên cứu đã cho thấy batch size quá nhỏ hoặc quá lớn có thể khiến model kém hội tụ [1]. Batch size cần đủ lớn để sử dụng tối đa hiệu năng tính toán của GPU nhưng không được quá lớn vì gây tràn bộ nhớ của GPU.

4.3 Thời gian train một model đến một độ chính xác nhất định

Là thời gian train một model cho đến khi nó đạt được độ chính xác nào đó trên tập dữ liệu validation. Sử dụng độ đo này giúp tìm ra batch size hợp lý khiến model hội tụ đến một mức nào đó trong thời gian nhanh nhất. Độ do này được sử dụng trong công cụ benchmark DAWNBench [1].

4.4 Số ảnh xử lý trong một đơn vị thời gian

Độ đo tính theo số lượng ảnh được xử lý trong một đơn vị thời gian. Thực chất nó là kết quả của việc chia số lượng ảnh của một batch ảnh cho thời gian xử lý batch ảnh đó. Số ảnh trên giây khi test càng lớn đồng nghĩa với thời gian xử lý ảnh và độ trễ của các ứng dụng tích hợp model càng thấp. Nếu không xét đến việc batch size ảnh hưởng đến tốc độ hội tụ của model, số ảnh trên giây khi train càng lớn sẽ khiến việc train càng nhanh. Độ đo này đã từng được Google sử dụng để benchmark Tensorflow [4].

Độ đo được dùng trong thí nghiệm này là số ảnh xử lý trong một đơn vị thời gian.

5 Mô hình thí nghiệm

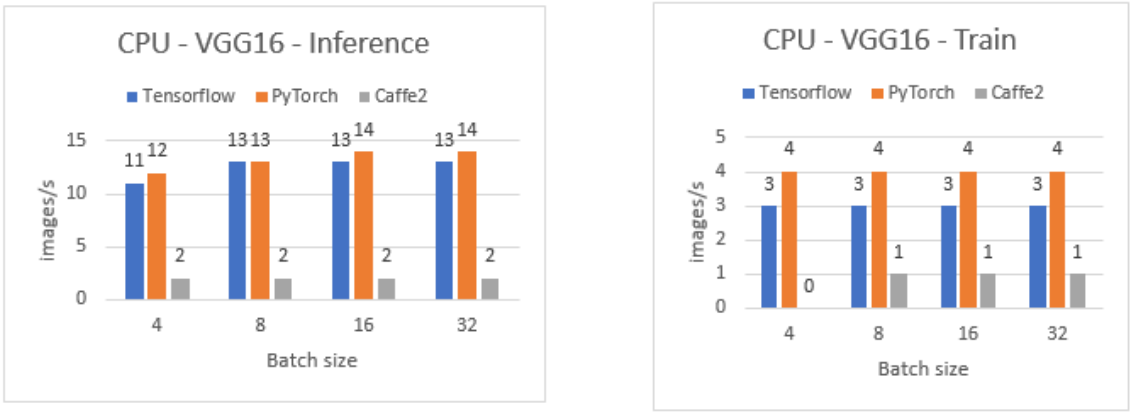
- Thực hiện phân loại ảnh hai model là VGG và ResNet152.
- Độ đo là số ảnh trên giây (ảnh/giây), được tính bằng batch size chia thời gian xử lý một batch.
- Dữ liệu được sinh ra ngẫu nhiên. Theo kết quả từ [4], với một GPU, thời gian xử lý dữ liệu giả (sinh ngẫu nhiên) so với dữ liệu thật (ảnh thật) chênh lệch rất ít, nên có thể đơn giản hóa bằng cách sinh dữ liệu ngẫu nhiên.
- Không tiền xử lý dữ liệu. Việc này làm giảm sai số do ảnh hưởng của tiền xử lý có thể gây ra.
- Thời gian backward pass không tính thời gian cập nhật lại hệ số (weight, bias).
- Model sẽ được chạy warm up 10 iteration và sau đó là 20 iteration. Thời gian xử lý được tính bằng trung bình gian chạy của 20 iteration.
- CPU sử dụng gồm 40 logical cpu, tạo bởi 2 Intel(R) Xeon(R) Silver 4114 CPU @ 2.20-3.0GHz 10/20 13.75 MB L3
- Phiên bản phần mềm cho GPU bao gồm Nvidia Driver 410.48, CUDA V10.0.130, cuDNN 7.5.0
- Phiên bản framework sử dụng cùng Python 3.6 để benchmark:
  - Tensorflow 1.13.1
  - Pytorch 1.0.1.post2
  - Caffe2 0.8.1
- 2 GPU sử dụng cho quá trình benchmark và thông số kỹ thuật được trình bày trong bảng sau:

GPU	Tesla P100	Tesla T4
Architecture	Pascal	Turing
CUDA Cores	3584	2560
Tensor Cores		320
Memory	16GB HBM2	16GB GDDR6
FP64 TFLOPS	4.7	4.7
FP32 TFLOPS	9.3	8.1
FP16 TFLOPS	18.7	
Tensor Performance (Mixed Precision FP16/FP32)		65

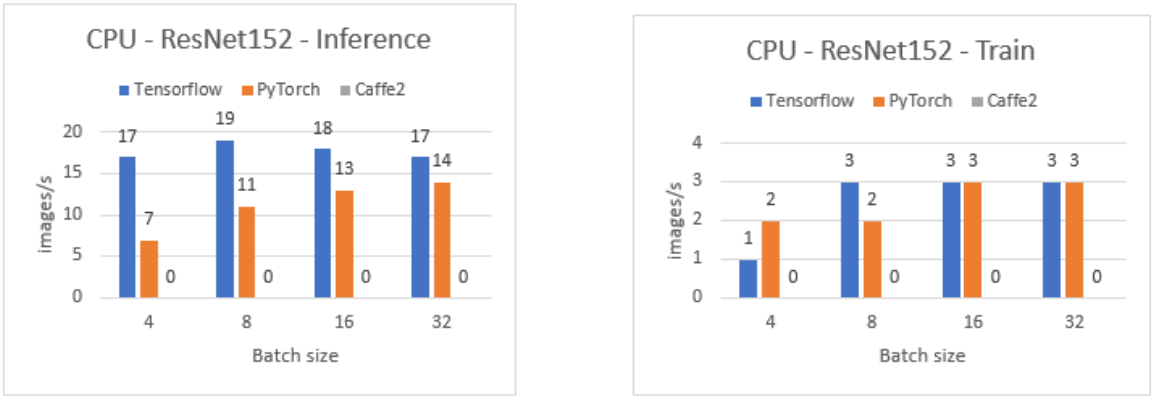
6 Kết quả

6.1 Kết quả chia theo các loại CPU/GPU

6.1.1 CPU

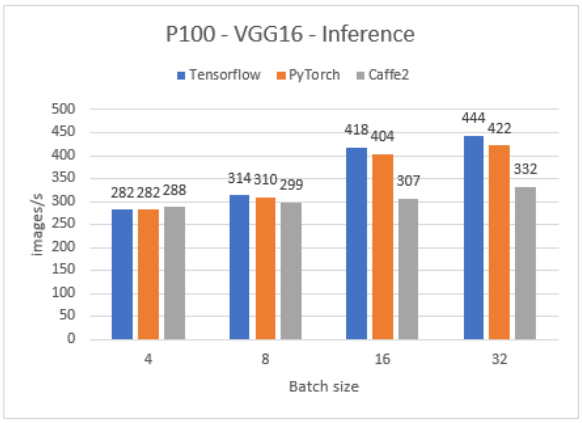


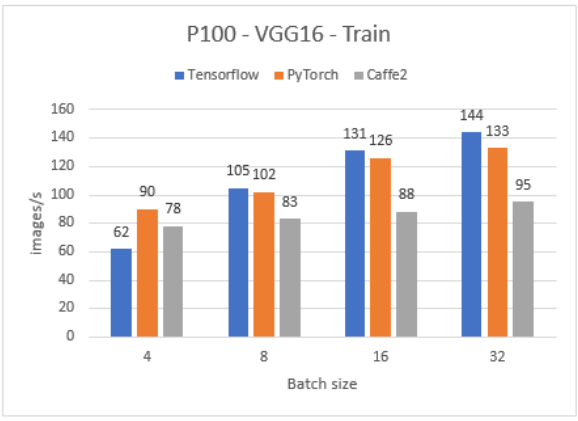
Hình 3: *Hiệu năng các framework trên CPU, model VGG16*



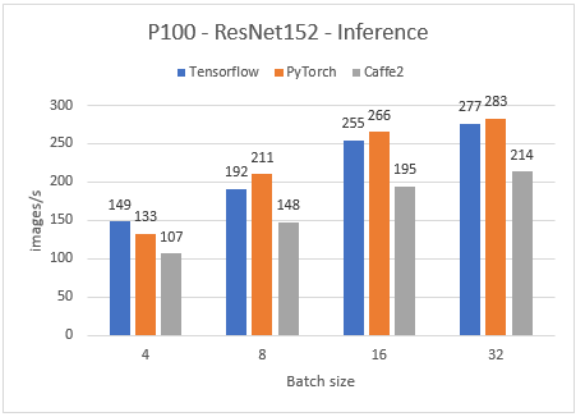
Hình 4: *Hiệu năng các framework trên CPU, model ResNet152*

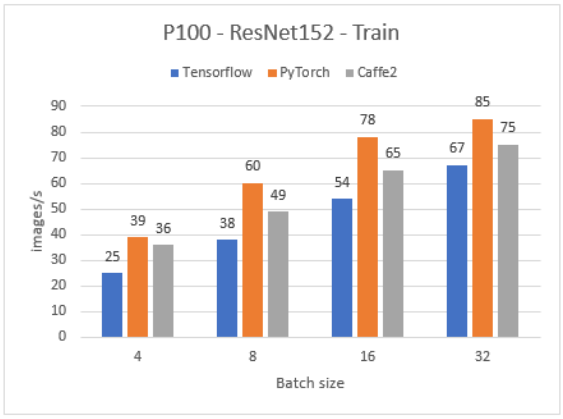
6.1.2 GPU Tesla P100





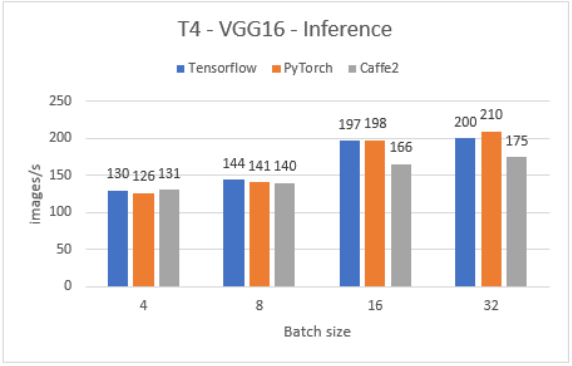
Hình 5: *Hiệu năng các framework trên GPU Tesla P100, model VGG16*

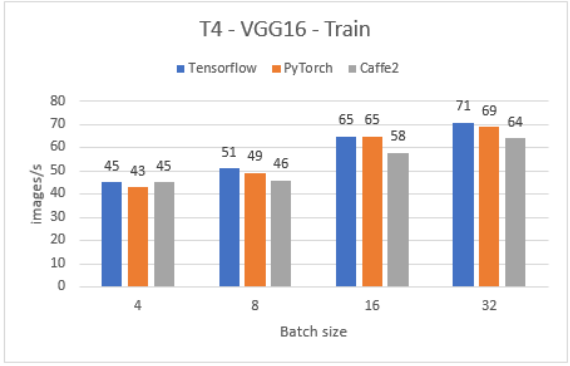




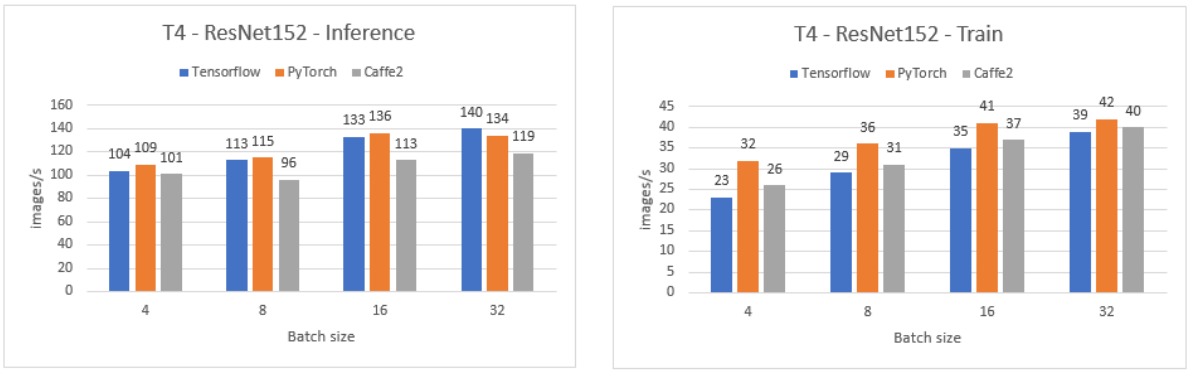
Hình 6: *Hiệu năng các framework trên GPU Tesla P100, model ResNet152*

6.1.3 GPU Tesla T4





Hình 7: *Hiệu năng các framework trên GPU Tesla T4, model VGG16*

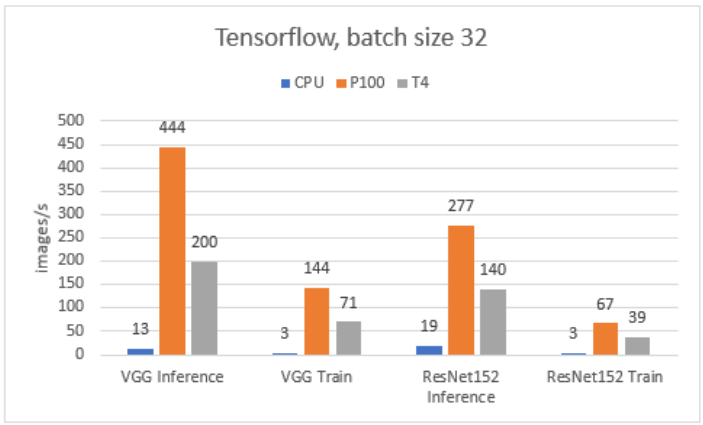


Hình 8: *Hiệu năng các framework trên GPU Tesla T4, model ResNet152*

6.2 Kết quả chia theo từng framework

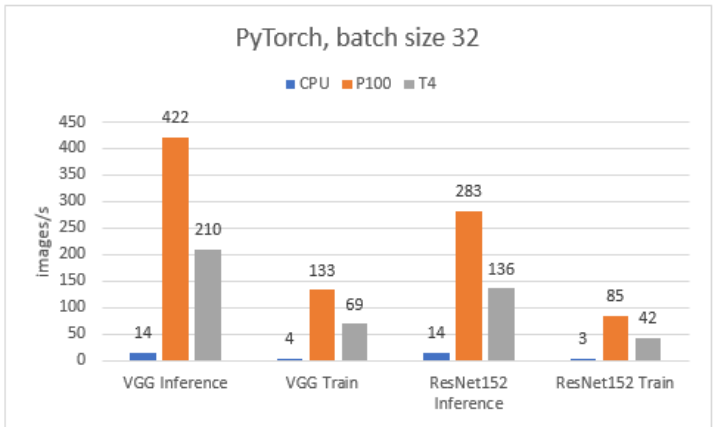
Kết quả tham chiếu từ kết quả ở phần trên, nhưng chỉ lấy kết quả tốt nhất, chính là với batch size 32.

6.2.1 Tensorflow



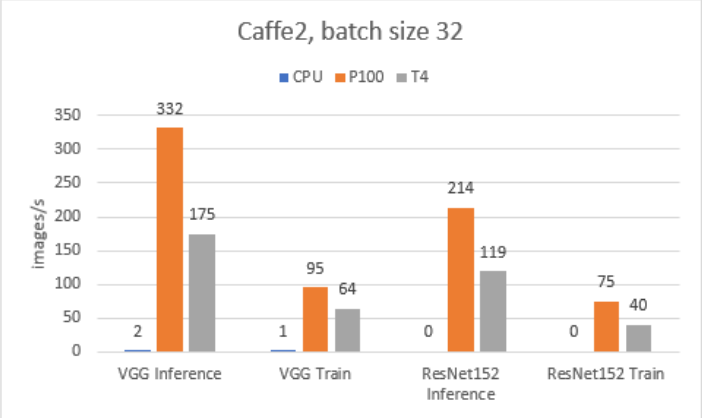
Hình 9: *Hiệu năng các CPU/GPU trên Tensorflow*

6.2.2 PyTorch



Hình 10: *Hiệu năng các CPU/GPU trên PyTorch*

6.2.3 Caffe2



Hình 11: *Hiệu năng các CPU/GPU trên Caffe2*

7 Đánh giá kết quả

7.1 Về phần cứng

Qua bài benchmark, ta thấy được sự ưu việt của GPU so với CPU trong các tác vụ deep learning mà quá trình tính toán với ma trận và song song cực kì quan trọng. Điều đó được thể hiện qua các kết quả benchmark khi GPU luôn có tốc độ cao gấp hàng chục lần so với CPU. Trong đó, mặc dù với kiến trúc cũ hơn, nhưng với lượng nhân tính toán lớn hơn so với Tesla T4, Tesla P100 là phần cứng mạnh nhất trong bài benchmark này, cho kết quả gần gấp đôi Tesla T4.

Vì bài benchmark chưa làm việc với kiểu dữ liệu Mixed Precision FP16/FP32 mà chỉ cố định ở FP32 do đó Tesla T4 với 320 core Tensor core chưa thể hiện được hết khả năng của mình.

Từ các biểu đồ có nhận thấy một số điểm sau:

- Số ảnh trên giây tăng khi batch size tăng trên GPU. Điều này do batch size lớn nhất được thí nghiệm (32) vẫn chưa gây tràn bộ nhớ GPU.
- GPU Tesla T4 cho kết quả khá đồng đều khi chạy các framework khác nhau, cả trên inference time và train time.
- Thời gian inference của Tensorflow và PyTorch là khá đồng đều và nhìn chung là cao hơn so với Caffe2.

7.2 Về framework

Pytorch dù là framework mới ra đời (Torch trên Lua đã có từ trước), nhưng với khả năng tối ưu rất tốt của mình, đã thể hiện là framework cho tốc độ tốt nhất trong hầu hết các trường hợp.

Tensorflow, cùng với sự phổ biến và thời gian phát triển từ lâu, cũng là một framework chỉ thua kém Pytorch một phần nhỏ trong các kết quả benchmark. Vì phiên bản Tensorflow sử dụng trong bài benchmark này là bản stable mới nhất tính tới thời điểm làm benchmark, trong khi Tensorflow đã phát hành bản alpha của phiên bản 2.0 với nhiều thay đổi vượt trội để bắt kịp Pytorch, do đó Tensorflow hoàn toàn có thể trở thành framework với khả năng ngang bằng với Pytorch, thậm chí là hơn.

Caffe2, một framework với sự hỗ trợ cũng như cộng đồng kém hơn hẳn so với Pytorch và Tensorflow, cho kết quả benchmark thấp hơn 2 framework còn lại, đặc biệt là khả năng tối ưu khi thực hiện trên CPU.

8 SourceCode và cách sử dụng

SourceCode có sẵn tại https://github.com/ititandev/deep-learning-benchmark

Lệnh để chạy benchmark toàn bộ cho cả 3 framework bằng cả CPU lẫn GPU `python3 benchmark.py`

Để chạy benchmark với framework cụ thể và phần cứng cụ thể: `python3 benchmark.py -f option`
Các option gồm: `tensorflow` | `tensorflowcpu` | `pytorch` | `pytorchcpu` | `caffe2` | `caffe2cpu`

Tài liệu

[1] Cody Coleman et al. "Dawnbench: An end-to-end deep learning benchmark and competition". In: *NIPS ML Systems Workshop*. 2017.

[2] Deep Learning. *Deep Learning made easy with Deep Cognition — becomminghuman.ai*. URL: https://becominghuman.ai/deep-learning-made-easy-with-deep-cognition-403fbe445351.

[3] Deep Learning. *Deep learning — Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Deep_learning.

[4] Google. "TensorFlow Performance Benchmarks". In: (2017). URL: https://www.tensorflow.org/guide/performance/benchmarks.

[5] Gradient descent. *Gradient descent — Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Gradient_descent.

[6] Stochastic gradient descent. *Stochastic gradient descent — Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Stochastic_gradient_descent.

[7] Hongyu Zhu et al. "TBD: Benchmarking and Analyzing Deep Neural Network Training". In: *CoRR* abs/1803.06905 (2018). arXiv: 1803.06905. URL: http://arxiv.org/abs/1803.06905.