CS 5334

Dr. Shirley Moore

Oscar Galindo

Lab 5 Report

Lab 5 Objective:

Choose a large dataset and a machine learning method. Implement for both CPU and GPU on Bridges-2. Measure speedup of GPU over CPU.

Dataset

For this lab assignment I decided to implement a neural network to solve the problem of distinguishing if an X-ray reading from the chest of people shows that a person has Covid-19 or not. The dataset was found in Kaggle and the dataset can be accessed through this link: <https://www.kaggle.com/andyczhao/covidx-cxr2>

The dataset is composed of ~16,000 images for training purposes and 400 images from testing purposes. Of course, when compared to other datasets this is not “big” by any means, nevertheless one key aspect must be considered. Obtaining images from x-ray machines is a hard, full of regulations, and slow task. So, for the domain of the problem a dataset containing around ~16,000 images can be considered a “big” dataset.

Platforms used

The platforms used for this lab were anaconda to set up an environment for experimentation (a comprehensive list of what the environment can be found if the command “module load AI” is run, and then the packages installed are listed by doing pip3 list). In particular, for this experimentation it was utilized tensorflow-gpu 2.0.0, keras-gpu 2.3.1, cudnn 7.6.5, and CUDA 11.2. The versions used on different software packages were taken from the environment for experimentation loaded when the command “module load AI” is ran.

A comprehensive list of modules used:

Package Version

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

absl-py 0.11.0

argon2-cffi 20.1.0

astor 0.8.1

astroid 2.4.2

astropy 4.2

async-generator 1.10

attrs 20.3.0

backcall 0.2.0

bitarray 1.6.1

bleach 3.2.1

blessings 1.7

bokeh 2.2.3

brotlipy 0.7.0

certifi 2020.12.5

cffi 1.14.4

cftime 1.4.1

chardet 4.0.0

click 7.1.2

cloudpickle 1.6.0

conda-pack 0.5.0

cryptography 3.3.1

cycler 0.10.0

Cython 0.29.21

cytoolz 0.11.0

dask 2020.12.0

decorator 4.4.2

defusedxml 0.6.0

entrypoints 0.3

gast 0.2.2

google-pasta 0.2.0

gpustat 0.6.0

grpcio 1.31.0

h5py 2.8.0

idna 2.10

imagecodecs 2020.5.30

imageio 2.9.0

imagesize 1.2.0

importlib-metadata 2.0.0

iniconfig 1.1.1

ipykernel 5.3.4

ipython 7.19.0

ipython-genutils 0.2.0

ipywidgets 7.6.1

jedi 0.17.2

Jinja2 2.11.2

joblib 1.0.0

json5 0.9.5

jsonschema 3.2.0

jupyter 1.0.0

jupyter-client 6.1.7

jupyter-console 6.2.0

jupyter-core 4.7.0

jupyterlab 2.2.6

jupyterlab-pygments 0.1.2

jupyterlab-server 1.2.0

jupyterlab-widgets 1.0.0

Keras 2.3.1

Keras-Applications 1.0.8

Keras-Preprocessing 1.1.0

kiwisolver 1.3.0

lazy-object-proxy 1.4.3

leveldb 0.201

llvmlite 0.34.0

lxml 4.6.2

Mako 1.1.3

Markdown 3.3.3

MarkupSafe 1.1.1

matplotlib 3.3.2

mistune 0.8.4

mkl-fft 1.0.14

mkl-random 1.0.4

mkl-service 2.3.0

more-itertools 8.6.0

mpmath 1.1.0

nbclient 0.5.1

nbconvert 6.0.7

nbformat 5.0.8

nest-asyncio 1.4.3

netCDF4 1.4.2

networkx 2.5

nltk 3.5

notebook 6.1.6

numba 0.51.2

numexpr 2.7.2

numpy 1.17.0

nvidia-ml-py3 7.352.0

olefile 0.46

opt-einsum 3.1.0

packaging 20.8

pandas 1.2.0

pandocfilters 1.4.3

parso 0.7.0

pexpect 4.8.0

pickleshare 0.7.5

Pillow 8.1.0

pip 20.3.3

pluggy 0.13.1

prometheus-client 0.9.0

prompt-toolkit 3.0.8

protobuf 3.13.0

psutil 5.8.0

ptyprocess 0.7.0

py 1.10.0

pycparser 2.20

pyerfa 1.7.1.1

Pygments 2.7.3

pygpu 0.7.6

pyOpenSSL 20.0.1

pyparsing 2.4.7

pyrsistent 0.17.3

PySocks 1.7.1

pytest 0.0.0

python-dateutil 2.8.1

python-gflags 3.1.2

pytz 2020.5

PyWavelets 1.1.1

PyYAML 5.3.1

pyzmq 20.0.0

qtconsole 4.7.7

QtPy 1.9.0

regex 2020.11.13

requests 2.25.1

scikit-image 0.17.2

scikit-learn 0.23.2

scipy 1.5.2

seaborn 0.11.1

Send2Trash 1.5.0

setuptools 51.0.0.post20201207

six 1.15.0

tensorboard 2.0.0

tensorflow 2.0.0

tensorflow-estimator 2.0.0

termcolor 1.1.0

terminado 0.9.2

testpath 0.4.4

Theano 1.0.4

threadpoolctl 2.1.0

tifffile 2020.12.8

toml 0.10.1

toolz 0.11.1

torch 1.5.0

tornado 6.1

tqdm 4.55.1

traitlets 5.0.5

typed-ast 1.4.1

typing-extensions 3.7.4.3

urllib3 1.26.2

wcwidth 0.2.5

webencodings 0.5.1

Werkzeug 0.16.1

wheel 0.36.2

widgetsnbextension 3.5.1

wrapt 1.11.2

zipp 3.4.0

Model Architecture

The model used to solve this problem was based on other architectures that had been tested previously for other experiments, it just so happens to be that the architecture presented in this document was “adequate” for achieving a good score when predicting on the dataset for COVID-19 images. It is important to remember that if this architecture had too many parameters it would quickly converge to >90% accuracy on the training set and the validation data/test data would not necessarily have a score as high, which would mean overfitting occurred. In essence, if the problem does not have as many complex relationships and we applied a construction that is too complex nothing meaningful is learned and instead the model would learn to look at the samples and almost predict based on the specific traits of every sample. On the other side of the spectrum if the model had too few parameters the model would never learn to identify general aspects of the data being feed and instead it would learn to guess, or rather, it would learn to forget some traits of the data as newer aspects are learned.

Table

Description automatically generated

Comment: With the previously presented architecture the accuracy reached on the test set was 80%. Without any further tunning for better accuracy.

Speed up

To calculate the speed up I divided the time it took to run the training of the COVID-19 dataset while using only a single node with 128 cores in Bridges 2 and then divide the time when executed with a CPU by the time taken when the training is executed using a single V100 GPU on bridges.

Time taken on 128 cores: 899 seconds



Time taken on a single GPU: 229 seconds



* Speedup = 899/229 = 3.925 times

Conclusions

According to results it definitely makes a difference to run a neural network model on a GPU vs a CPU. Hence, using a GPU makes more sense than a CPU since, for example, taking a day for training on a GPU would require more theoretically 4 nodes and 512 cores to theoretically approach the same time taken that it takes to train the model on a GPU. In addition, we know that having many cores does not guarantee a speedup, actually it could reduce the velocity of the computations.