FINAL PROJECT REPORT - [by Ndifreke OKORIE & Mohammad AQEEL]

RESEARCH ANALYTIC QUESTION

How fast can a face recognition algorithm work on about 2GB of data so has to be efficiently used in production to train and capture faces for the purpose of pandemic control or crime detection in real time?

STATEMENT OF PROBLEM

In the twenty first century speed is key in technology deployments, that should be considered as working or better put as working to safe lives; whether in shopping malls environs, public gatherings, railway stations, etc.

So, if we could know and perhaps find ways to efficiently tune our system to be a step ahead of situations that poses as challenges to human survival, we would have helped the evolution of life and tranquility.

As people go about their daily activities, cameras could collect images of people entering and exiting critical environments and live training of models/ predictions could be going on at the same time, to identify those found to engage in suspicious activities or not following laid down rules of cubbing the spread of diseases e.g. the proper wearing of face masks in some locations and not just wearing it over the chin or over the mouth only.

So, we propose building a face recognition algorithm using machine learning to see how fast it could train using big data and to also see what happens under the hood with the view to making it faster.

DATA

We used already labels images of celebrities from pinterest
The image size was originally 229MB
But we employed the use of data augmentation techniques to ramp it up to 1.69GB

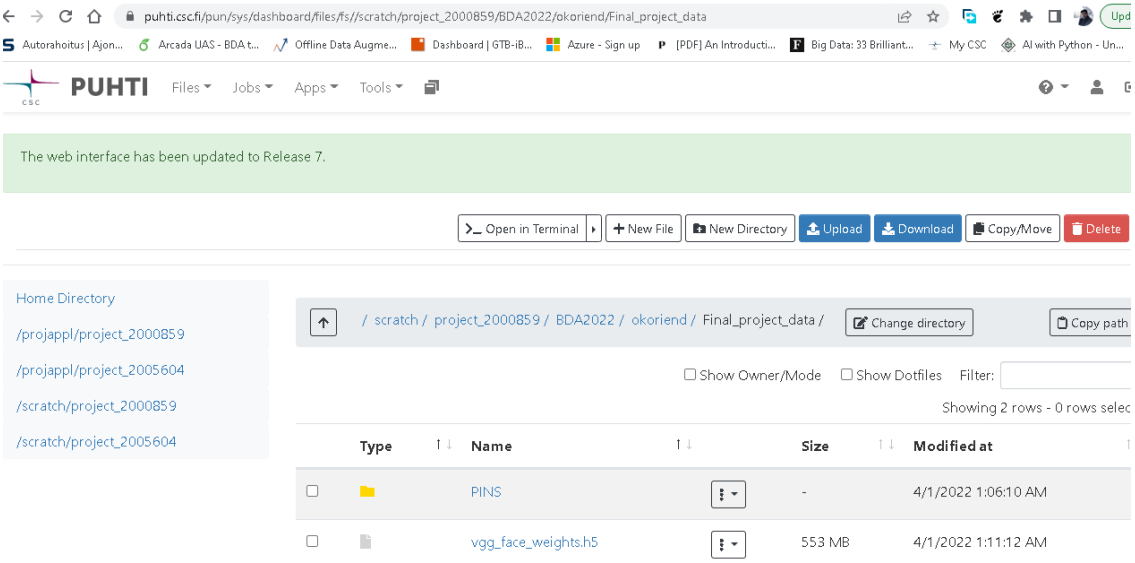
- Image Augmentation technique that were used in an image data generator were:
  i - rotation_range = 40,
  ii - shear_range = 0.2,
  iii - zoom_range = 0.2,
  iv - horizontal_flip = True,
  v - brightness_range = (0.5, 1.5)

50 images for every original image was generated and added to the original ones

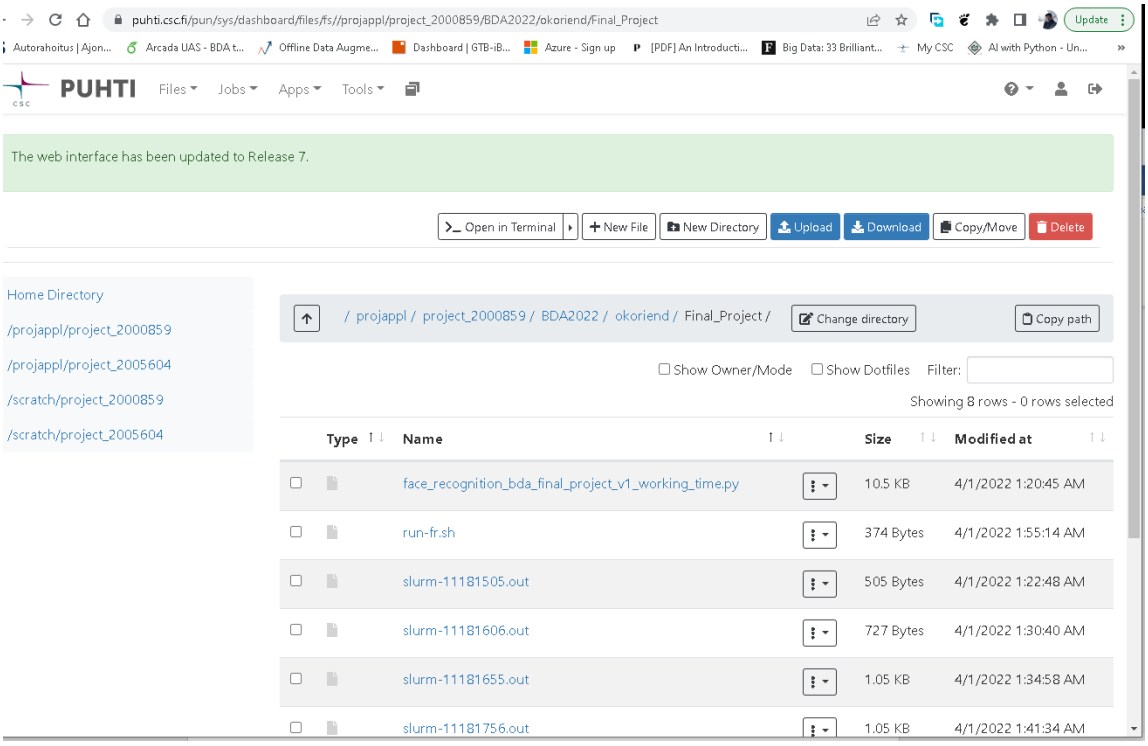
BIG DATA TOOL FOR SOLVING PROBLEM

We attempted to use puhti the supercomputer; see the screenshots:

Data in Scratch:



Code in Application path:



OUR FINDINGS

Due to modules issues and lack of experience using puhti, we could not run our ‘.py’ notebook successfully, we had issues as shown by one of the slum files (we will see how to get it to work in the next course module):

```
===============

Lmod is automatically replacing "python-data/3.9-2" with "tensorflow/2.8".

NOTE: This module uses Singularity. Some commands execute inside the container
(e.g. python3, pip3).

Currently Loaded Modules:
  1) gcc/7.4.0   2) tensorflow/2.8

Collecting opencv-python
  Downloading
https://files.pythonhosted.org/packages/67/50/665a503167396ad347957bea0bd8d5c08c865030b2d1565ff06
eba613780/opencv_python-4.5.5.64-cp36-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(60.5MB)
ERROR: Could not install packages due to an EnvironmentError: [Errno 28] No space left on device

srun python3 $*
+ srun python3 face_recognition_bda_final_project_v1_working_time.py
Traceback (most recent call last):
  File "face_recognition_bda_final_project_v1_working_time.py", line 91, in <module>
    import cv2
ModuleNotFoundError: No module named 'cv2'
srun: error: r01g03: task 0: Exited with exit code 1
srun: launch/slurm: _step_signal: Terminating StepId=11181939.0
======================
```

Our batch file

```
#!/bin/bash
#SBATCH --job-name=face_recognition
#SBATCH --account=project_2000859
#SBATCH --partition=gpu
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=10
#SBATCH --mem=64G
#SBATCH --time=2:00:10
#SBATCH --gres=gpu:v100:1

module purge
module load python-data/3.9-2
module load tensorflow
module list
pip install --user opencv-python

set -xv
srun python3 $*
```

But we were able to run our python notebook in google colab, though with gpu, it took about 2 hours to do this, majorly because of creating embeddings for each image took way too long on colab (maybe there was a malfunction and gpu was not allocating, so we resorted to Kaggle gpu although we had to use the original 229MB data, since we couldn’t use puhti for now). But our model accuracy was good. See it below:

Modeling

Build a Classifier

- Use SVM Classifier to predict the person in the given image
- Fit the classifier and print the score

```
[ ]  1  from sklearn.svm import SVC
     2
     3  clf = SVC(kernel='rbf', class_weight=None , C=10000000, gamma='auto')
     4  clf.fit(X_train, y_train)
     5  clf.score(X_test, y_test)
```

```
0.9615705931495405
```

And was able to easily identify the fed in test images with a labelled name as shown:

Test results

- Take 10th or any image number from test set and plot the image
- Report to which person(folder name in dataset) the image belongs to

```
import warnings
# Suppress LabelEncoder warning
warnings.filterwarnings('ignore')

example_idx = 100

example_image = load_image(metadata[test_idx][example_idx].image_path())
example_prediction = clf.predict([X_test[example_idx]])
example_identity = encoder.inverse_transform(example_prediction)[0]

plt.imshow(example_image)
plt.title(f'Identified as {example_identity}');
```

It took 955.10 seconds which is 15.92mins to train this model on Kaggle gpu.

If we were to use the big data (1.69GB, which is about 7.56 times the original; it would take about 2hours in training with Kaggle).
We would love to find ways to make it much faster in production, I hope it will be when we finally get it to run in puhti.