3803ICT
Big Data Analysis

Lab 08 – Data Analytics for Images

Table of Contents

1. Setup (Not Graded)

1.1. Run your code with Google Colab

You are recommended to use Google Colab when:

- Your computer is not powerful enough to run Deep Learning
- You want to avoid the hassle of install Deep Learning libraries

Follow this tutorial on running your code with Google Colab:
https://colab.research.google.com/github/googlecolab/colabtools/blob/master/notebooks/colab-github-demo.ipynb

1.2. Install Tensorflow/Keras (Optional if using Colab)

Keras is now implemented in Tensorflow. You can install Keras by:
https://www.ibm.com/docs/en/wmlce/1.6.2?topic=frameworks-getting-started-keras-team-keras

and install Tensorflow by:

Install Tensorflow on Windows (GPU-possible) or Linux (GPU-possible) or MacOS (CPU-only):
https://docs.anaconda.com/anaconda/user-guide/tasks/tensorflow/

Install Tensorflow on MacOS m1 with GPU (tested):
https://makeoptim.com/en/deep-learning/tensorflow-metal#install-tensorflow
+ General idea: install Miniforge, it has another conda command inside. Uninstall Anaconda/Anaconda Navigator and other related previously installed version of conda-based installations. Anaconda (which has another conda command inside) and Miniforge cannot co-exist together.
+ Uninstall Anaconda using anaconda-clean: https://docs.anaconda.com/anaconda/install/uninstall/
+ Install Miniforge: brew install miniforge
+ Create and activate a conda virtual environment with python 3.9.5 (as required for TensorFlow).

```
conda create -n tensorflow python=3.9.5
conda activate tensorflow
conda install -c apple tensorflow-deps
python -m pip install tensorflow-macos
python -m pip install tensorflow-metal
brew install libjpeg
conda install -y matplotlib jupyterlab
```

1.3. Install OpenCV (Optional if using Colab)

For Anaconda or Miniforge: conda install opencv
For Pip: pip install opencv-python

Install OpenCV on Mac M1 (Tested):
https://blog.roboflow.com/m1-opencv/
+ General idea: use Miniforge like Tensorflow
+ brew install miniforge
+ Install a new or use an existing conda environment
+ conda install opencv

1.4. Install Detectron2 (Optional if using Colab)

Install Detectron2 in Windows, Linux with Anaconda and GPU-enabled:

- https://medium.com/@yogeshkumarpilli/how-to-install-detectron2-on-windows-10-or-11-2021-aug-with-the-latest-build-v0-5-c7333909676f
- https://medium.com/@sujoydebnath.92/installing-detectron2-with-anaconda-and-cuda-on-linux-7b710663326c
- https://anaconda.org/conda-forge/detectron2

Install Detectron2 in Google Colab:

- Check torch and cuda version then install the correponding pre-built detectron2: https://detectron2.readthedocs.io/en/latest/tutorials/install.html

- Run this cell:

```
import torch, torchvision
print(torch.__version__, torch.cuda.is_available())
!gcc --version
```

- Run this cell

```
!python -m pip install detectron2 -f
https://dl.fbaipublicfiles.com/detectron2/wheels/cu111/torch1.10/index.html
# exit(0) # After installation, you need to "restart runtime" in Colab. This line can also restart runtime
```

Installing Detectron2 natively for Mac M1 Pro / Apple silicon (use Miniforge) (tested, still CPU because of Pytorch):
https://medium.com/@hakon.hukkelas/installing-detectron2-natively-for-mac-m1-pro-apple-silicon-a89517f1c913

1.5 Troubleshooting

When running keras/tensorflow on CPU, you might have this error:

- Error #15: Initializing libiomp5.dylib, but found libiomp5.dylib already initialized https://stackoverflow.com/questions/53014306/error-15-initializing-libiomp5-dylib-but-found-libiomp5-dylib-already-initial
- Solution: run these lines before compile/run the Keras model

```
import os

os.environ['KMP_DUPLICATE_LIB_OK']='True'
```

2. Image classification with Multilayer Perceptron (MLP)

In this exercise, we will try to use a neural network on a simple classification task: classifying images of clothes into 10 classes. We will use FASHION_MNIST data

Open Fashion.ipynb and follow the instructions. Your model will look like this:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 10)                7850
_________________________________________________________________
dense_1 (Dense)              (None, 10)                110
_________________________________________________________________
dense_2 (Dense)              (None, 10)                110
=================================================================
Total params: 8,070
Trainable params: 8,070
Non-trainable params: 0
_________________________________________________________________
```

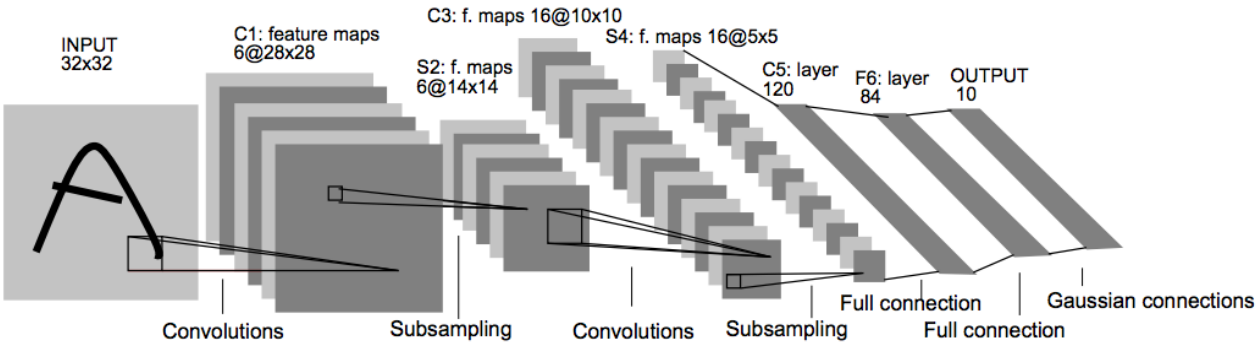
Note:

- You can use full dataset if using Colab and might use a subset of the data if your hardware is slow.
- You might want to normalize the input into [0,1] to avoid the overflow problem:
  X_train_norm = X_train/255.
  X_test_norm = X_test/255.
- Try to reduce the number of epochs if your hardware is slow
- Try to increase batch_size if GPU is enabled; otherwise you might want to decrease the batch_size if your hardware is slow
- You can specify the batch_size in the forward propagation (predict, evaluate) of a neural network model

3. Image classification with Convolutional Neural Networks (CNN) (OPTIONAL)

In this exercise, we will apply the LeNet5 architecture to the Fashion MNIST dataset and improve your performances.

Open LeNet5.ipynb and follow the instructions. Your model will look like this:




```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
C1 (Conv2D)                  (None, 26, 26, 6)         60
_________________________________________________________________
S2 (MaxPooling2D)            (None, 13, 13, 6)         0
_________________________________________________________________
C3 (Conv2D)                  (None, 11, 11, 16)        880
_________________________________________________________________
S4 (MaxPooling2D)            (None, 5, 5, 16)          0
_________________________________________________________________
flatten (Flatten)            (None, 400)               0
_________________________________________________________________
C5 (Dense)                   (None, 120)               48120
_________________________________________________________________
F6 (Dense)                   (None, 84)                10164
_________________________________________________________________
dense (Dense)                (None, 10)                850
=================================================================
Total params: 60,074
Trainable params: 60,074
Non-trainable params: 0
_________________________________________________________________
```

Note:

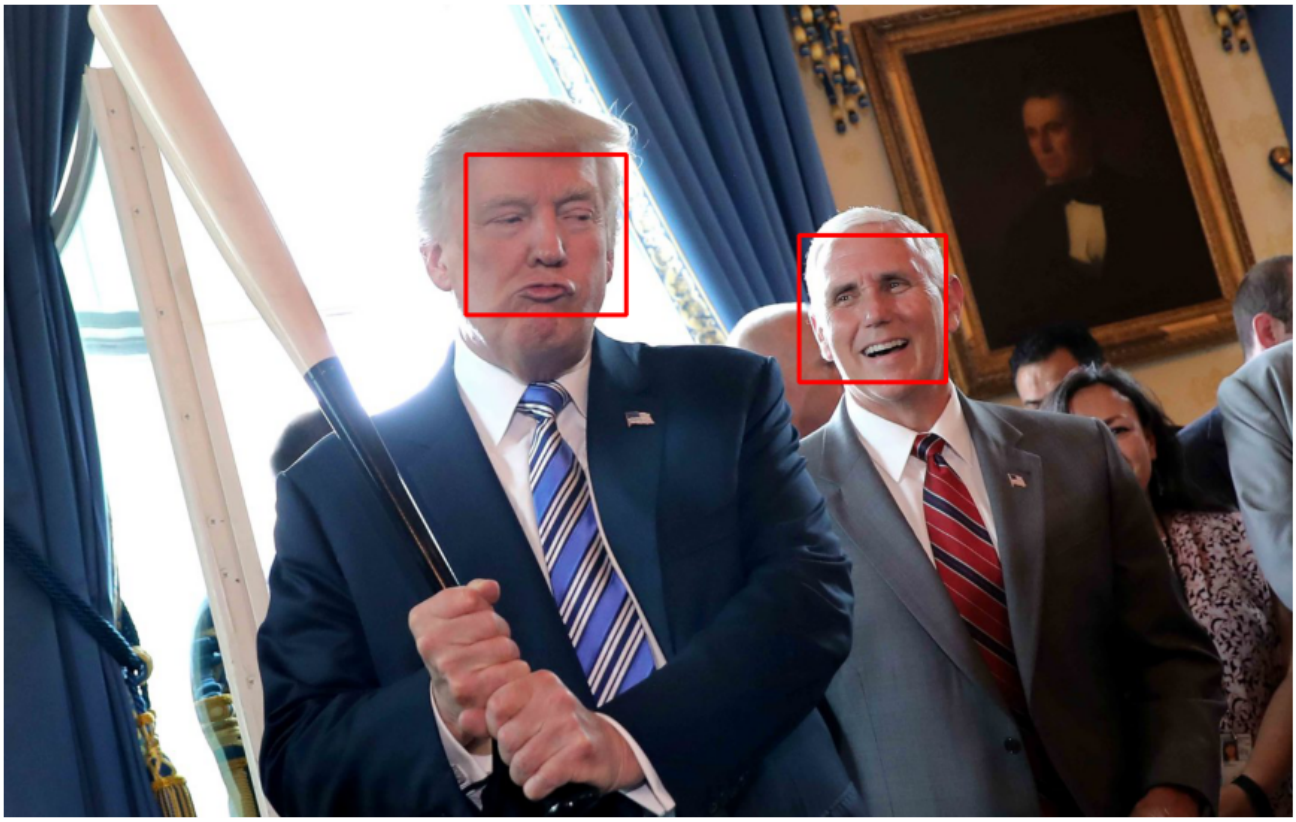
- You can use full dataset if using Colab and might use a subset of the data if your hardware is slow.
- You might want to normalize the input into [0,1] to avoid the overflow problem:
  X_train_norm = X_train/255.

X_test_norm = X_test/255.

- Try to reduce the number of epochs if your hardware is slow
- Try to increase batch_size if GPU is enabled; otherwise you might want to decrease the batch_size if your hardware is slow
- You can specify the batch_size in the forward propagation (predict, evaluate) of a neural network model

4. Face Detection with Classical ML (OpenCV)

In this first exercise, you will apply a basic face detection algorithm called Haar Cascade Classifier. Your result will look like this:



You can also use this classifier to detect faces in real-time with your webcam!

5. Object Detection in Large-Scale (Detectron2) (OPTIONAL)

Detectron2 is Facebook AI Research's next generation library that provides state-of-the-art detection and segmentation algorithms. It is the successor of Detectron and maskrcnn-benchmark. It supports a number of computer vision research projects and production applications in Facebook.

Open Detectron2.ipynb and follow the instructions.

You don’t need to train a model but using a pre-trained model for the object detection task. So it should be quite fast to run the inference.

Your result will look like this:


0
100
200
300
400
500
600
0
200
400
600
800
bus 100%
bus 99%
bus 100%
traffic light 95%
traffic light 93%
person 72%
person 79%
person 65%
person 69%
person 76%
person 97%
person 94%
person 100%
person 87%
person 71%
person 96%
person 91%
backpack 94%
backpack 74%
handbag 67%