National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Saylee Vijay More
Student ID: X18180167

School of Computing
National College of Ireland

Supervisor: Christian Horn

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Saylee Vijay More ……. …… ……
Student ID:	X18180167 …… ……..……
Programme:	Msc. in Data Analytics …… … **Year:** 2019-2020 ……..
Module:	Msc. Research Project …… ……..……
Lecturer:	Christian Horn …… ……..……
Submission Due Date:	17/08/2020 …… ……..……
Project Title:	Virtual Garment Imposition using ACGPN …… …..……
Word Count:	…… **Page Count:** ……..…….……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.
I agree to an electronic copy of my thesis being made publicly available on NORMA the National College of Ireland's Institutional Repository for consultation.

Signature: …… ……

Date: …… ……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Saylee Vijay More
Student ID: X18180167

1 Hardware/Software Requirement

Below is the hardware requirement for running the experiment smoothly.

1.1 Hardware Requirement

Operating system: - Windows 8.1

RAM: - 16.0 GB

Processor: - Intel (R) Core (TM) i5-5200U CPU @ 2.20GHz 2.19 GHz

Hard Disk Space: - Minimum 100GB (100 GB drive space)

1.2 Software Requirements

Programming Language Tools: - Google Colaboratory (Cloud-based Jupyter notebook environment), Python version 3,

Web Browser: - Google Chrome

Email: - Access to a Gmail account

2 Environmental Setup for Google Colaboratory

This section will explain how to set up Google colaboratory environment.
Screenshots are added below for better understanding. First, the login into the Gmail account (make sure that Gmail account has enough space) or else make a new account and sign-up in the Google colab.

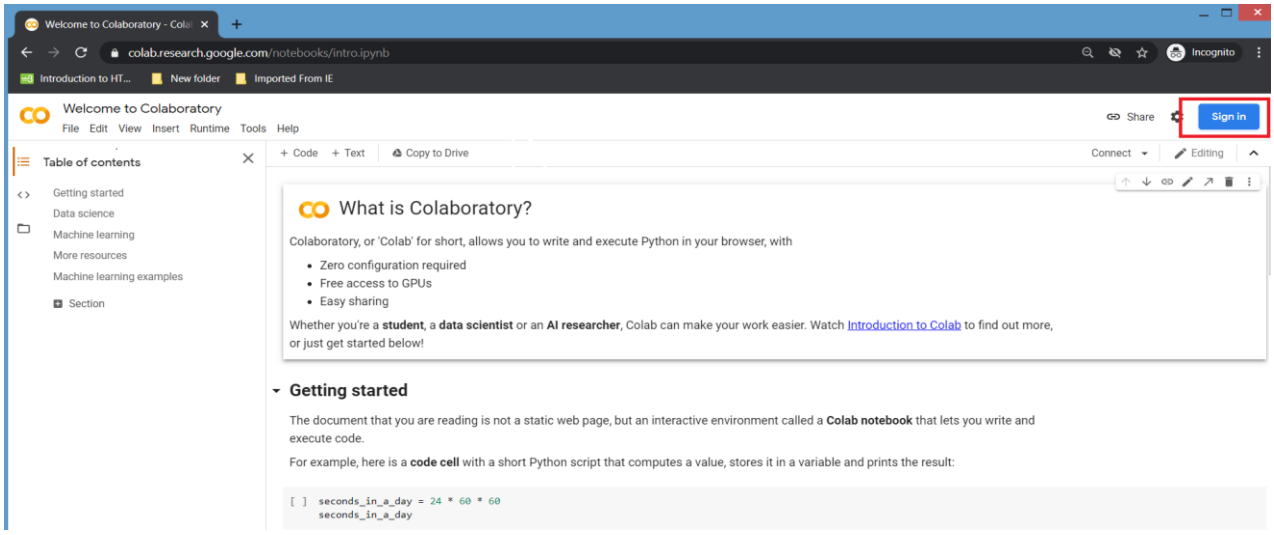
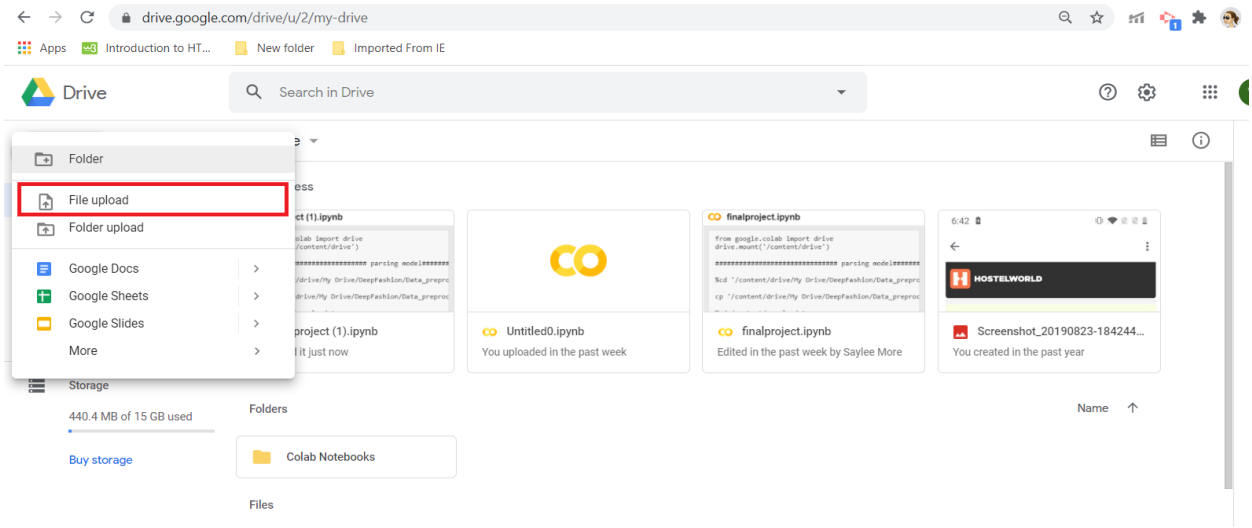


Fig 1: - Sign in to Gmail account.

2. Adding all the files to Google drive (Open Google Drive)



2.1 Adding Deep Fashion File in Google Drive

I. **Link for Deep Fashion file: - C:\Users\saylee more\Downloads\DeepFashion-20200817T062902Z-001.zip**

II. Link for dataset: - C:\Users\saylee more\Downloads\test-20200817T061227Z-001.zip

III. Add new file → select DeepFashion file and upload it.

IV. Add Dataset → Open DeepFashion → in that open Data_preprocessing →From dataset file upload only test_colormask and test_mask into the Data_preprocessing.

V. Garment: - Add a folder named test_color and test_edge. Add one image from test_color folder and respectively add one image from test_edge

from provided Data_preprocessing. This will be the garment which will be imposed on the test_img,
Note: - File name for test_color file and test_edge file should be same and should be added with care. As test_edge is the outer lining of the garment present in test_color.

2.2 Adding Self-Correction Human Parsing in Google Drive.

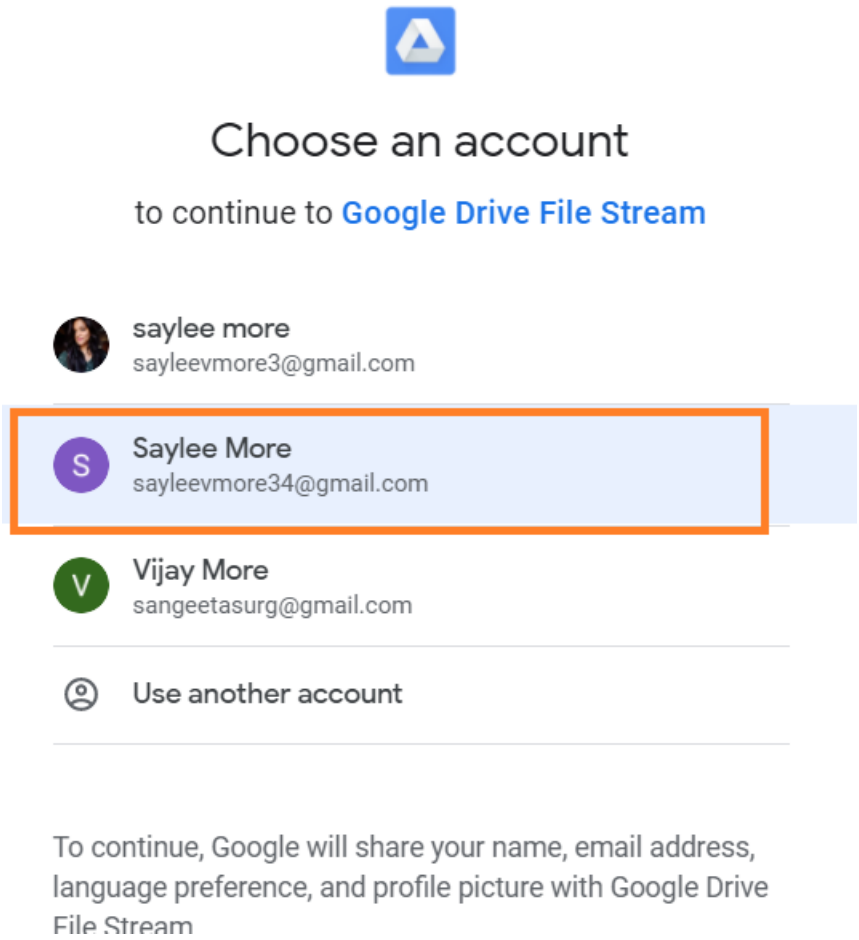
I. Link for Self-Correction Human Parsing: - C:\Users\saylee more\Downloads\Self-Correction-Human-Parsing-master-20200817T061244Z-001.zip
II. Link for dataset: - https://drive.google.com/drive/folders/1ldMxi8LOHTtXK_XnMOL0tWD3ZvBaiFM4?usp=sharing
III. Link for checkpoints: - https://drive.google.com/drive/folders/14xMgW0Ho0oaCGWBTE47qcUADcR0DkYe1?usp=sharing
IV. Similarly, add new file → Into ‘” content/drive/My drive” add Self-Correction-Human-Parsing-master.
V. Add dataset into “content/drive/My drive/ Self-Correction-Human-Parsing-master’’.
VI. Add check points into: - “content/drive/My drive/ Self-Correction-Human-Parsing-master/checkpoint’’.

3 Mounting the Google colaboratory Notebook to Google Drive

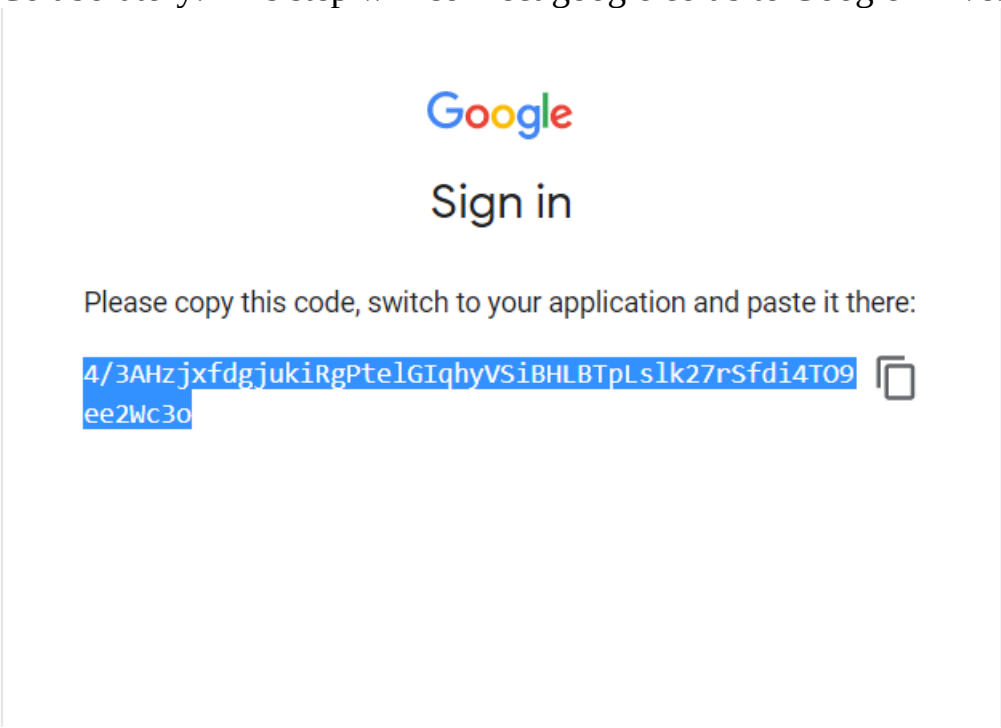
. Connect google Colaboratory to google drive.

1. Run the below-given code and click on the link

2. Select Google Drive Account where files have been uploaded.



3. Allow Access and copy the Passcode and Paste in the google Colaboratory. This step will connect google colab to Google Drive.



2. Create a test_img folder in "**/content/drive/My Drive/DeepFashion/Data_preprocessing".**
3. Add the images into test_img folder which you want to impose garments on.

4. Implementation

1. Install Pillow to resize the test_img to 192x256 resolution.

```
pip install Pillow
```

```
Requirement already satisfied: Pillow in /usr/local/lib/python3.6/dist-packages (7.0.0)
```

```
import PIL
import os
import os.path
from PIL import Image

f = r'/content/drive/My Drive/DeepFashion/Data_preprocessing/test_img'
for file in os.listdir(f):
    f_img = f+"/"+file
    img = Image.open(f_img)
    img = img.resize((192,256))
    img.save(f_img)
```

This will convert the standard image to the image of resolution 192x256.

2. We are using Parsing Model to get test_label.

2.1 Change the directory.

```
    img = img.resize((192,256))
    img.save(f_img)
```

```
[ ] %cd '/content/drive/My Drive/Self-Correction-Human-Parsing-master'
    !ls
```

```
/content/drive/My Drive/Self-Correction-Human-Parsing-master
checkpoint            evaluate.py     mhp_extension  'simple_extractor(1.py'
cmake-3.17.2          Input_images    modules         simple_extractor.py
cmake-3.17.2.tar.gz   Input_Images1   networks        test_labels
datasets              LICENSE         Output_Images   train.py
demo                  LIP             README.md       utils
```

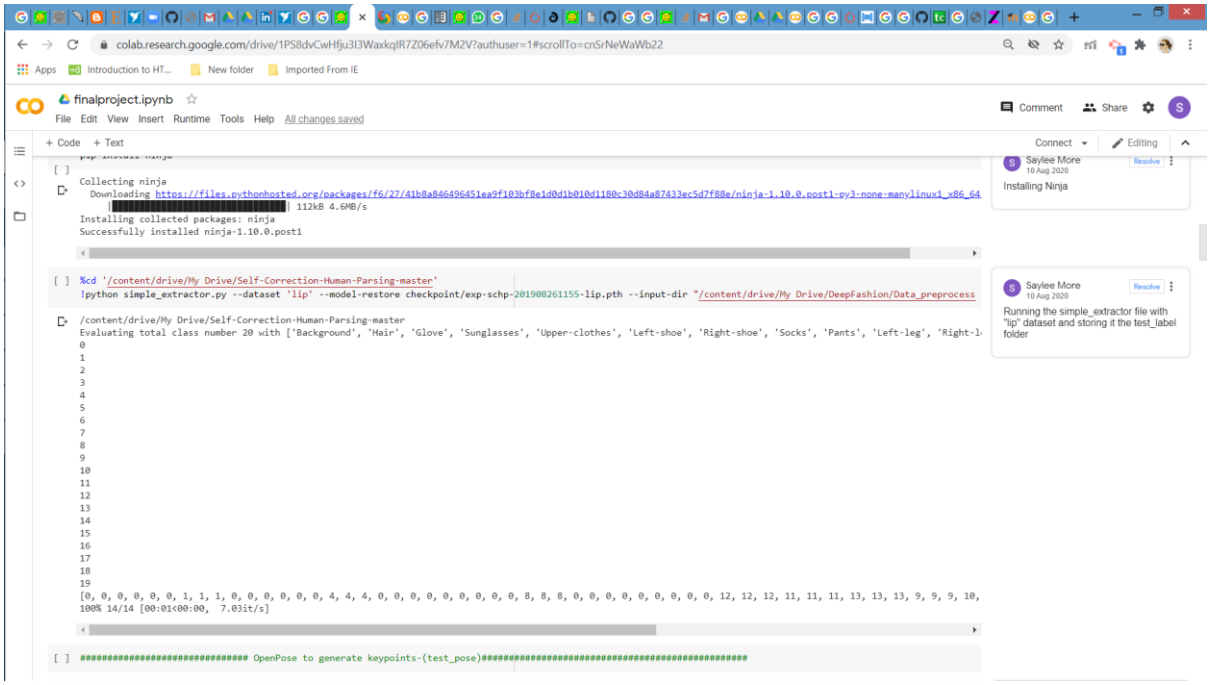
2.2 Install Ninja for speeding the process.

```
[ ] pip install ninja

Collecting ninja
  Downloading https://files.pythonhosted.org/packages/f6/27/41b8a846496451ea9f103bf8e1d0d1b010d1180c30d84a87433ec5d7f88e/ninja-1.10.0.post1-py3-none-manylinux1_x86_64
     |████████████████████████████████| 112kB 4.6MB/s
Installing collected packages: ninja
Successfully installed ninja-1.10.0.post1
```

2.3 Run this code

Here the input-dir path is given as "/content/drive/My Drive/DeepFashion/Data_preprocessing/test_img" and the output-dir as "/content/drive/My Drive/DeepFashion/Data_preprocessing/test_label" because we want to take input from test_img file and store it in test_label. Both this file is in Data_preprocessing folder in DeepFashion.



The output of this will like this.

3. Generate key points using an open pose.
   As Open pose requires Nvidia, Nvidia-smi has been installed.

```
[ ] ############################## OpenPose to generate keypoints-(test_pose)#############################################

!nvcc --version
!nvidia-smi

nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2019 NVIDIA Corporation
Built on Sun_Jul_28_19:07:16_PDT_2019
Cuda compilation tools, release 10.1, V10.1.243
Wed Aug 12 00:51:57 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 450.57       Driver Version: 418.67       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
```

4. Install cuDNN and check the version of CMake.

```
[ ] ver_openpose = "v1.6.0"

[ ] ! cmake --version

cmake version 3.12.0

CMake suite maintained and supported by Kitware (kitware.com/cmake).
```

5. Install new Cmake version.

```
[ ] ! wget -c "https://github.com/Kitware/CMake/releases/download/v3.17.2/cmake-3.17.2.tar.gz"
    ! tar xf cmake-3.17.2.tar.gz
    ! cd cmake-3.17.2 && ./configure && make && sudo make install

Streaming output truncated to the last 5000 lines.
-- Performing Test curl_cv_func_recv_test
-- Performing Test curl_cv_func_recv_test - Failed
-- Tested: int recv(SOCKET, char *, unsigned int, int)
-- Performing Test curl_cv_func_recv_test
-- Performing Test curl_cv_func_recv_test - Failed
-- Tested: int recv(SOCKET, char *, unsigned int, unsigned int)
```

6. Install the necessary libraries and dependencies Open pose

   As we are using python, we will require OpenCV, Opencl, python3 and other libraries to work on Open pose.

```
[ ]  # Install library

     # Basic
     ! sudo apt-get --assume-yes update
     ! sudo apt-get --assume-yes install build-essential
     # OpenCV
     ! sudo apt-get --assume-yes install libopencv-dev
     # General dependencies
     ! sudo apt-get --assume-yes install libatlas-base-dev libprotobuf-dev libleveldb-dev libsnappy-dev libhdf5-serial-dev protobuf-compiler
     ! sudo apt-get --assume-yes install --no-install-recommends libboost-all-dev
     # Remaining dependencies, 14.04
     ! sudo apt-get --assume-yes install libgflags-dev libgoogle-glog-dev liblmdb-dev
     # Python3 libs
     ! sudo apt-get --assume-yes install python3-setuptools python3-dev build-essential
     ! sudo apt-get --assume-yes install python3-pip
     ! sudo -H pip3 install --upgrade numpy protobuf opencv-python
     # OpenCL Generic
     ! sudo apt-get --assume-yes install opencl-headers ocl-icd-opencl-dev
     ! sudo apt-get --assume-yes install libviennacl-dev
```

7. Clone Open pose with the help of cuDNN

```
[ ]  # Openpose clone
     # Clone Openpose
     ! git clone  --depth 1 -b "$ver_openpose" https://github.com/CMU-Perceptual-Computing-Lab/openpose.git
```

8. After cloning Open pose build a directory.

```
[ ]  # build
     # Create build directory
     ! cd openpose && mkdir build && cd build
```

9. Add COCO dataset into Open pose/build.

```
# https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/installation.md#cmake-command-line-configuration-ubuntu-only
# Scenario 1 - Caffe not installed and OpenCV installed using apt-get
! cd openpose/build && cmake ..

# COCO
# Add cocomodel in Cmake
! cd openpose/build && cmake .. -D DOWNLOAD_BODY_COCO_MODEL=ON
```

10. Create an output directory in Open pose.

```
[ ]  # Openpose
     # Openpose Building
     ! cd openpose/build && make -j`nproc`
     # output
     ! cd openpose && mkdir output
```

11. Create root path and file path so that the input is taken from ““‘/content/drive/My Drive/DeepFashion/Data_preprocessing/test_img” and the output is directly stored into the ““‘/content/drive/My Drive/DeepFashion/Data_preprocessing/test_pose”

```
[ ]  root_path = "/content/drive/My Drive/DeepFashion/Data_preprocessing"
     !ls
     file_path = "test_img"

     ! cd openpose && ./build/examples/openpose/openpose.bin --image_dir "$root_path/$file_path" --display 0  --write_json "$root_path/test_pose" --render_pose 0
```

Here we are saving the file in JSON as we want key points and not the images of key points.

12. Key points are generated into the test_pose folder. The pose_keypoints were generated in 2d, so we replace the string stating to pose_keypoints.

    Note: - the file name should be put accordingly as we have taken 14 input images all required to be changed. This depends on how many pictures you input in test_img.

```
[ ]  %cd "/content/drive/My Drive/DeepFashion/Data_preprocessing/test_pose"

     !sed -i "s/pose_keypoints_2d/pose_keypoints/g" 0_keypoints.json
     !sed -i "s/pose_keypoints_2d/pose_keypoints/g" 1_keypoints.json
     !sed -i "s/pose_keypoints_2d/pose_keypoints/g" 2_keypoints.json
     !sed -i "s/pose_keypoints_2d/pose_keypoints/g" 3_keypoints.json
     !sed -i "s/pose_keypoints_2d/pose_keypoints/g" 4_keypoints.json
     !sed -i "s/pose_keypoints_2d/pose_keypoints/g" 5_keypoints.json
     !sed -i "s/pose_keypoints_2d/pose_keypoints/g" 6_keypoints.json
     !sed -i "s/pose_keypoints_2d/pose_keypoints/g" 7_keypoints.json
     !sed -i "s/pose_keypoints_2d/pose_keypoints/g" 8_keypoints.json
     !sed -i "s/pose_keypoints_2d/pose_keypoints/g" 9_keypoints.json
     !sed -i "s/pose_keypoints_2d/pose_keypoints/g" 10_keypoints.json
     !sed -i "s/pose_keypoints_2d/pose_keypoints/g" 11_keypoints.json
     !sed -i "s/pose_keypoints_2d/pose_keypoints/g" 12_keypoints.json
     !sed -i "s/pose_keypoints_2d/pose_keypoints/g" 13_keypoints.json
```

13. Import checkpoints, Data_preprocessing and ACGPN inference to Google Colaboratory Drive.

```
[ ] %cd '/content/drive/My Drive/DeepFashion/ACGPN_inference/checkpoints'
```

```
/content/drive/My Drive/DeepFashion/ACGPN_inference/checkpoints
```

```
ls '/content/drive/My Drive/DeepFashion/ACGPN_inference/checkpoints'
```

```
label2city/  readme.txt
```

```
[ ] cp -r label2city/ /content/sample_data/
```

```
[ ] %cd '/content/drive/My Drive/DeepFashion/'
```

```
/content/drive/My Drive/DeepFashion
```

```
[ ] ls '/content/drive/My Drive/DeepFashion/'
```

```
[ ] cp -r Data_preprocessing/ /content/sample_data/
```

ADD ACGPN_inference to the goggle colab drive

```
[ ] %cd '/content/drive/My Drive/DeepFashion/'
```

```
/content/drive/My Drive/DeepFashion
```

```
[ ] ls '/content/drive/My Drive/DeepFashion/'
```

```
' '/                cmake-3.17.2/          images/
 ACGPN_inference/   Data_preprocessing/    pretrain/
 ACGPN_train/       Data_preprocessing2/   README.md
```

```
[ ] cp -r ACGPN_inference/ /content/sample_data/
```

14. Install ipdp and Tensorflow

```
pip install ipdb
```

```
pip install tensorboardX
```

15.Change the directory and run the test.py file.

```
[ ] cd /content/sample_data/

/content/sample_data

!python test.py --dataroot /content/sample_data/Data_preprocessing/test// --load_pretrain /content/sample_data/label2city/
```

16.Final output will be in the sample file in ACGPN_inference.