Categorization of Fashion Clothes from WildImages using Object Detection andSegmentation based Models

MSc Research Project
Data Analytics

Rohan Indrajeet Jadhav
Student ID: x20169043

School of Computing
National College of Ireland

Supervisor: Dr. Paul Stynes, Dr. Pramod Pathak

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Rohan Indrajeet Jadhav
Student ID:	x20169043
Programme:	Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Dr. Paul Stynes, Dr. Pramod Pathak
Submission Due Date:	31/01/2022
Project Title:	Categorization of Fashion Clothes from WildImages using Object Detection andSegmentation based Models
Word Count:	NA
Page Count:	12

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.

Signature:	
Date:	31st January 2022

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):	

Categorization of Fashion Clothes from WildImages using Object Detection andSegmentation based Models

Rohan Indrajeet Jadhav
x20169043

1 Introduction

The configuration manual shows all the requirements that are needed to replicate the work in other environment similar fashion. Below sections mainly includes, the azure cloud deployment, dataset configuration over cloud, and run the code.

2 System Specification

2.1 Hardware Requirement

Minimum hardware requirement for running the azure cloud environment.

- CPU with operating frequency of 1 GHz minimum
- 64-bit operating system
- RAM: 4 GB minimum
- Disk space: 10 GB minimum

Figure 1 shows the device configuration from which the azure cloud setup is accessed.

IdeaPad S540-15IML D

Device name	LAPTOP-LN0GUSPQ
Processor	Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz
Installed RAM	20.0 GB (19.8 GB usable)
Device ID	0B32945A-3B68-428A-A794-8C59A61A0B33
Product ID	00327-35894-71662-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Figure 1: Device configuration used to access the Azure cloud

2.2 Software Requirement

For software it requires only internet browser from one can access the cloud environment.

- Microsoft Edge (recommended: Chromium-based Edge)
- Google Chrome
- Apple Safari
- Internet Explorer 11

Figure 2 and 3 shows the versions and details of browser which are used for accessing the azure cloud setup.

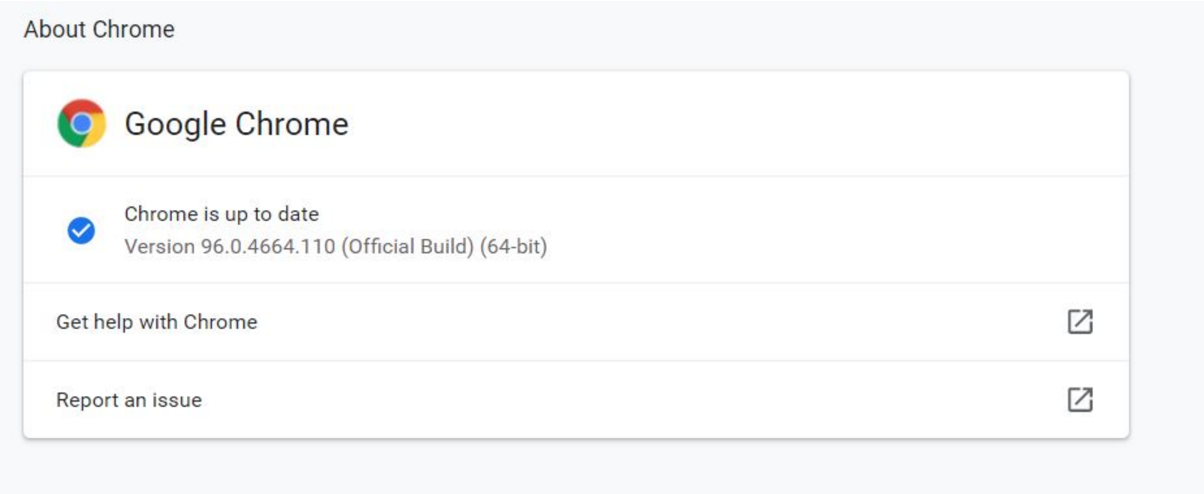

Figure 2: Google Chrome information

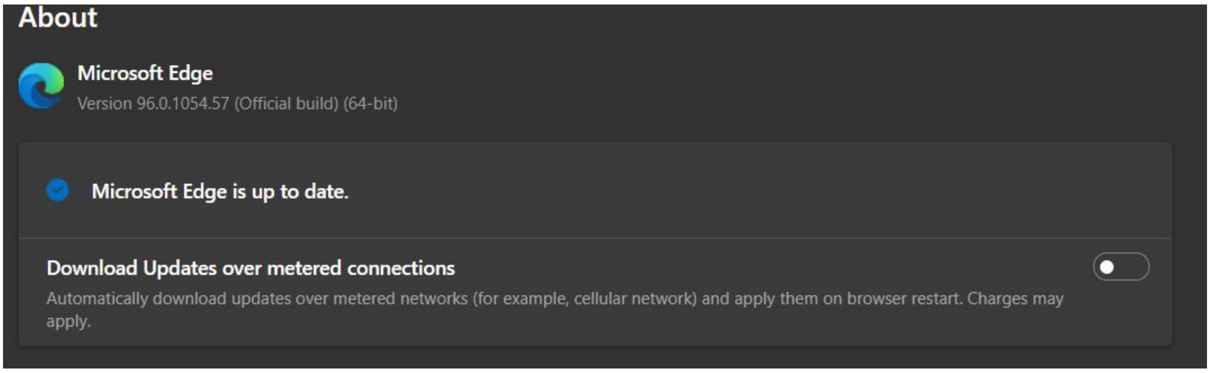

Figure 3: Microsoft Edge information

3 Azure Cloud Environment Setup

Once we satisfied with all the system configuration to access the cloud setup need to create the below services to train and run the models further.

3.1 Setting up the Resource group

For working on cloud user need to have an resource group under which all the other services which are required can be created and access withing given network of resource group.Here workspace created is RJ-workspace shown in figuree 4

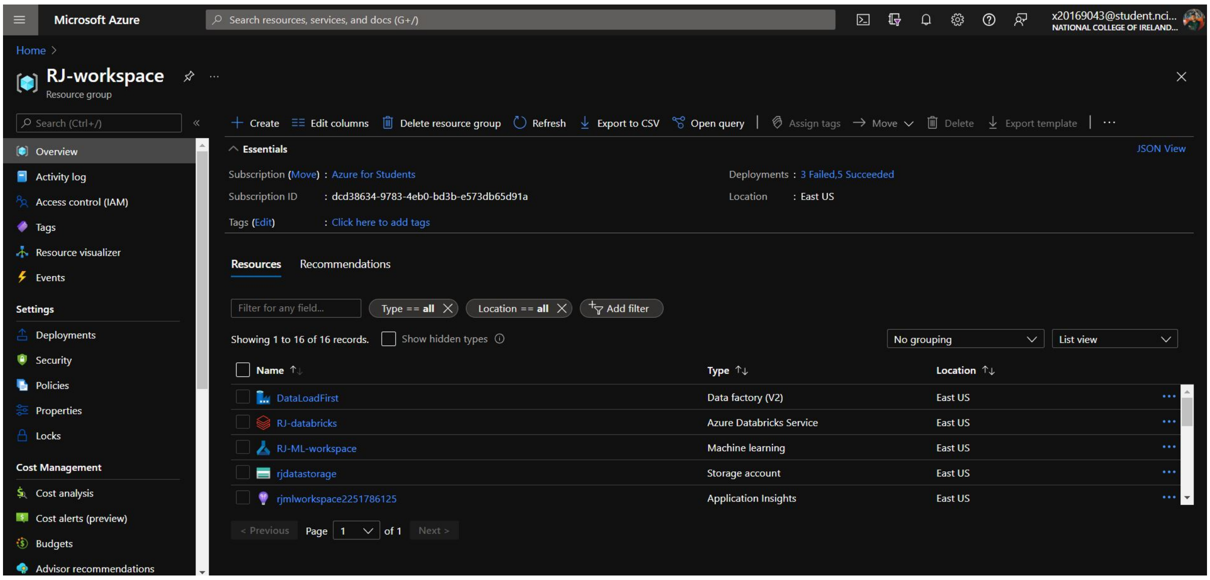

Figure 4: Creating Resource group

3.2 Setting up the Blob storage

Blob storage setup is also required, there are two ways manually can create one service under on resource group RJ-workspace or while uploading the dataset it will automatically creates a file system on blob storage as shown in figure 5

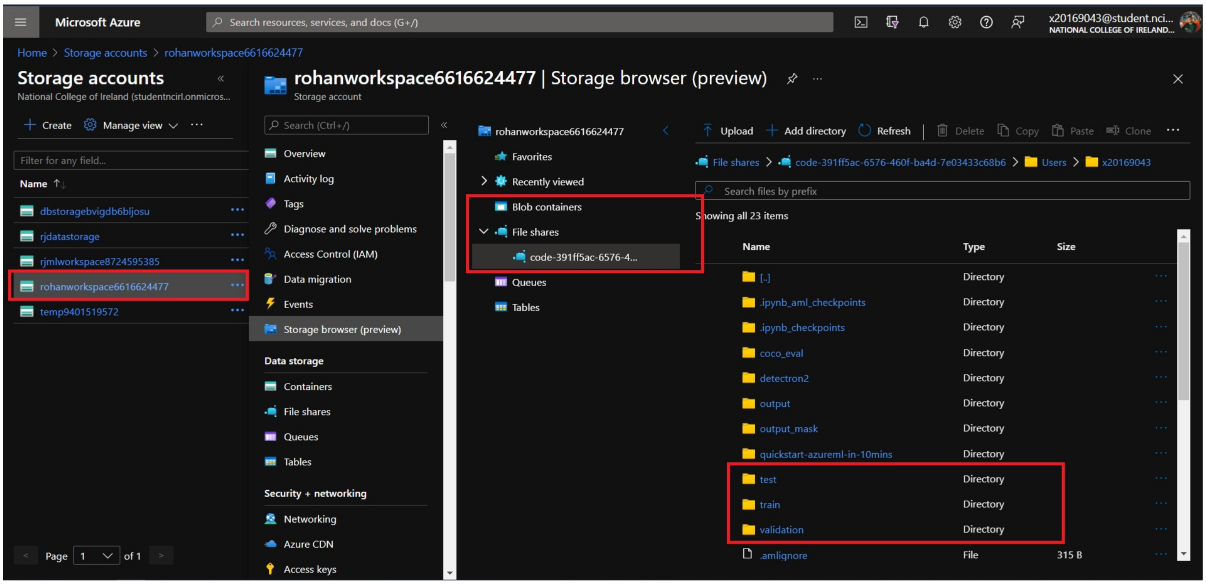

Figure 5: File system on blob storage

3.3 Setting up Azure ML studio

The key part in all implementation is azure ML studio where actual studio in which development is done. This includes code, dataset, required cluster spin which is given below.

Figure 6 shows how to create the ML studio.

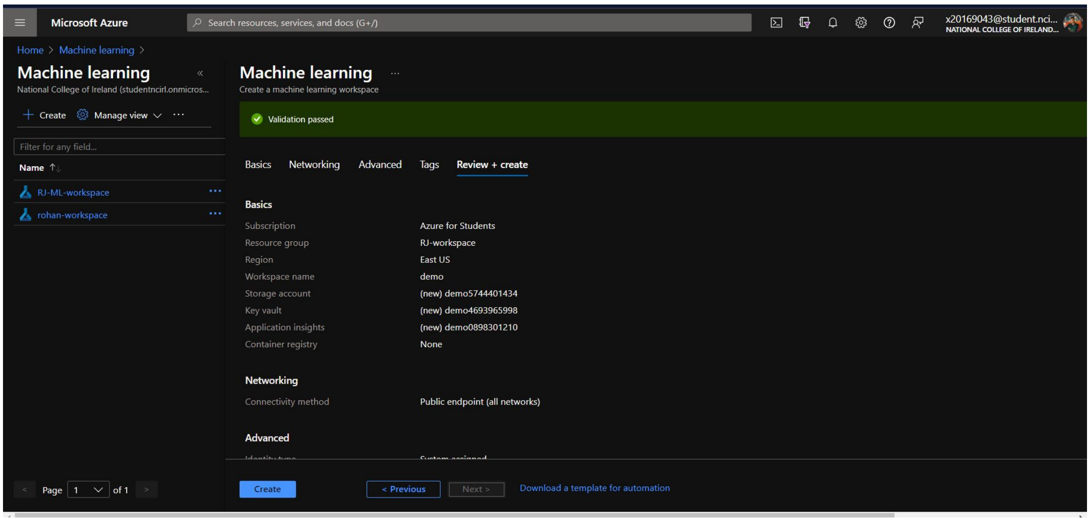

Figure 6: Creating ML studio

Once the ML studio is created the figure 7 shows how to launch the studio.

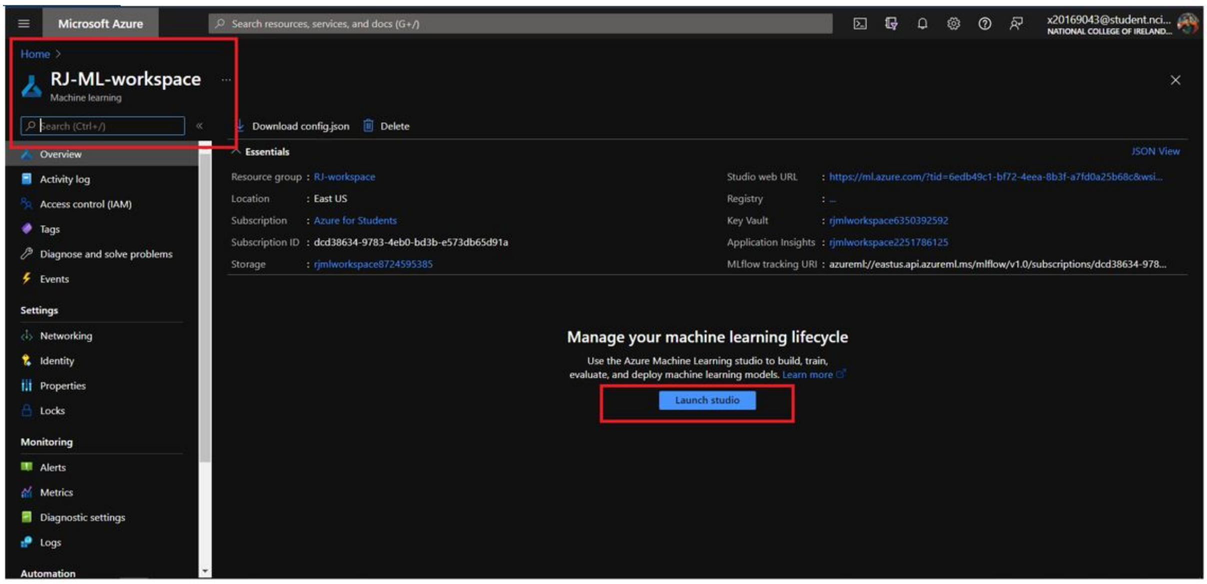

Figure 7: Launching ML studio

Figure 8 shows the view of the ML studio how the notebook is listed along with dataset and other required functionalities such on left panel shows the deepfashion2 dataset which I have uploaded, the ipnyb file of Faster RCNN model which is already trained. Till now this guide helps in creating the studio where developer can develop the code but it requires the high configured GPU which is explained in next section that is 3.4.

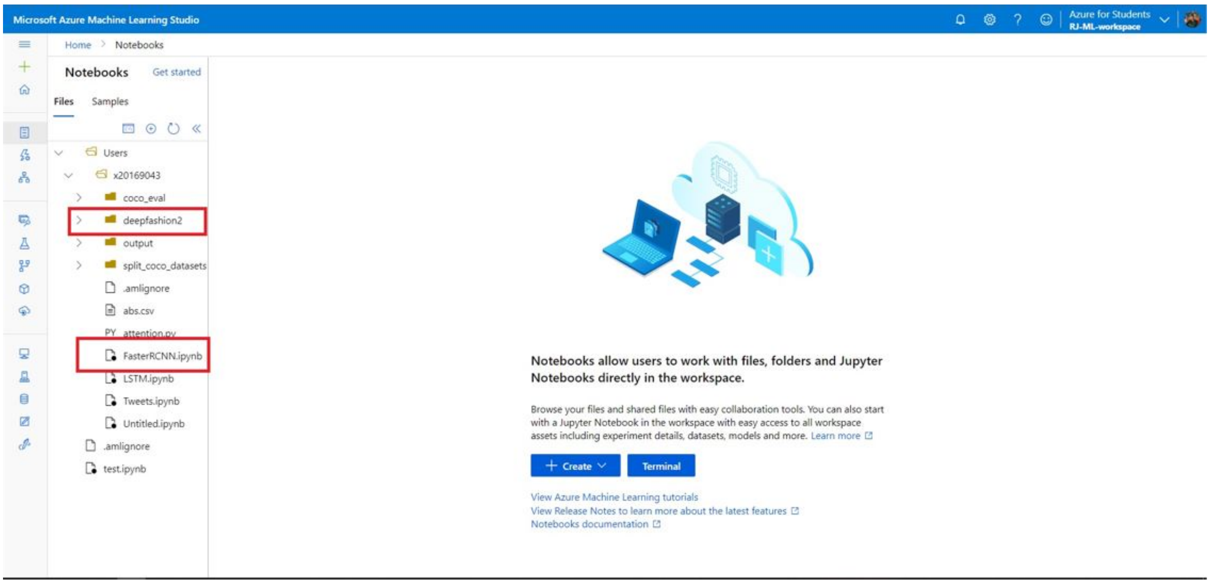

Figure 8: View of ML studio

3.4 Spin GPU based Compute node

As all the environment is created now its time to create the compute node which is the most crucial part to run such high visualization tasks on GPU based cluster.

Figure 9 show how to create the GPU based cluster, basic requirement which is required to complete these tasks are:

- 2 core GPU
- 20 GB RAM
- 200 GB of disk space

In this case I have got the least is 2 core GPU, 20 RAM, 256 GB of disk space. This varies user to user.

4 Importing the deepfashion2 dataset

Deepfashion2 dataset is availed from authors of the dataset creator which they have given the access to google drive and password to unzip the dataset. As thi model building part is going to carry out on cloud need to upload the dataset to cloud environment. Figure 10 shows how to upload the folder of dataset to Azure ML studio. There is another to do this is uploading the zip file to cloud and then unzip it there using unix command, this will be a much easier way to upload.

The actual dataset deepfashion2 will get at location: `https://github.com/switchablenorms/DeepFashion2`

5 Locating and parsing the dataset location

Data directories are crucial element when it deals with multiple file access while building the model. In this case the key datasets are, annotation file and fashion images for each dataset that is train, test and validation.

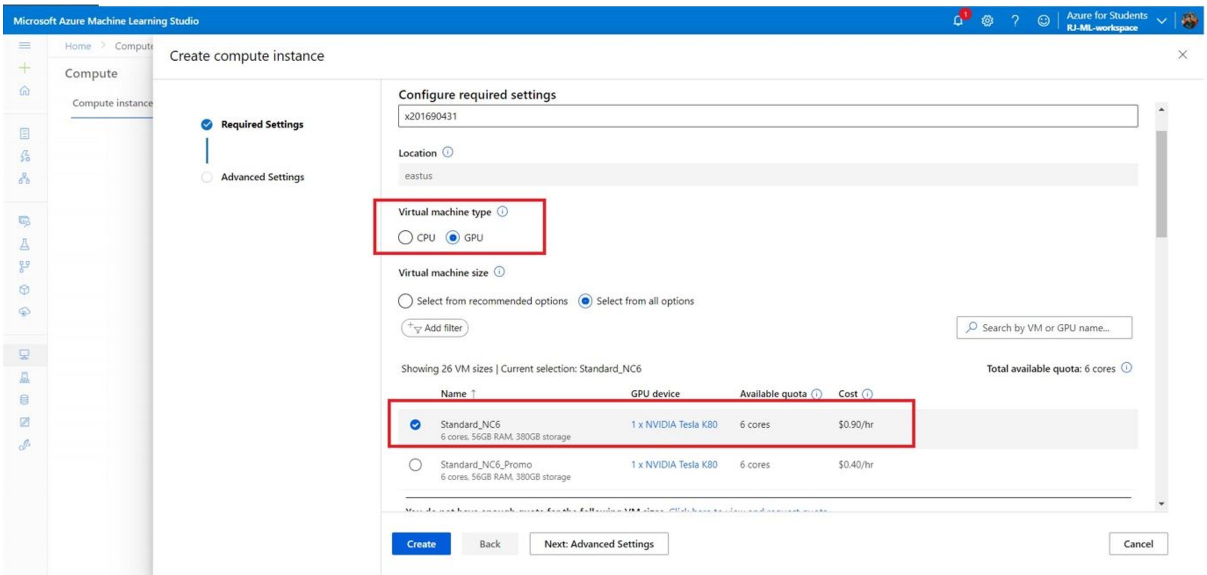

Figure 9: Creating GPU based compute node

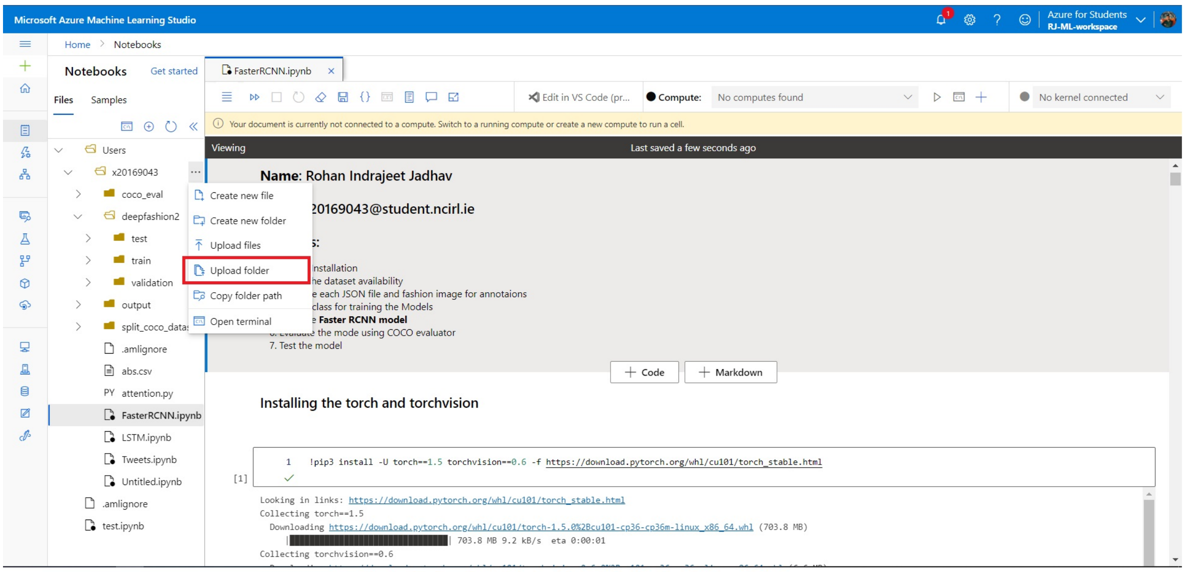

Figure 10: Uploading the deepfashion2

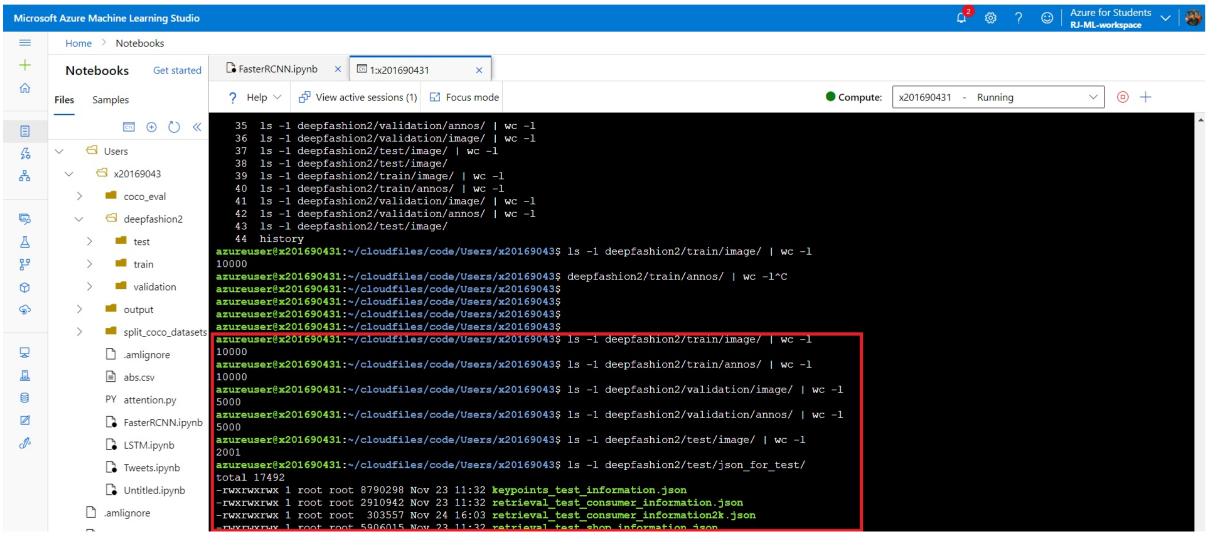

Figure 11: Directory structure on ML studio for dataset

The directory structure which I have followed is /root/deepfashion2 under will get the /train, /test and /validation dataset. For validation and train will get the /annos folder for annotation files in JSON format and /images for actual fashion images. Figure 11 shows the all paths which is captured from compute node terminal.

6 Run the code

Lastly one can run the code by uploading the ipnyb file to ML studio.Here I have ran in two different azure environment both of the models which Faster RCNN and Mask RCNN. All of the process will be same but only paths of dataset has to be changed.

Below paths represents the annotation and image file paths for train dataset

```
json_name = '/mnt/batch/tasks/shared/LS_root/mounts/clusters/
    x201690431/code/Users/x20169043/deepfashion2/train/annos/'
image_name = '/mnt/batch/tasks/shared/LS_root/mounts/clusters
    /x201690431/code/Users/x20169043/deepfashion2/train/image'
consolidated_json = '/mnt/batch/tasks/shared/LS_root/mounts/
    clusters/x201690431/code/Users/x2016904/deepfashion2/train
    /train.json'
```

In the same way the validation dataset would have file and data location such as:

```
json_name = '/mnt/batch/tasks/shared/LS_root/mounts/clusters/
    x201690431/code/Users/x20169043/deepfashion2/validation/
    annos/'
image_name = '/mnt/batch/tasks/shared/LS_root/mounts/clusters
    /x201690431/code/Users/x20169043/deepfashion2/validation/
    image/
consolidated_json_name = '/mnt/batch/tasks/shared/LS_root/
    mounts/clusters/x201690431/code/Users/x20169043/
```

```
deepfashion2/validation/validation.json'
```

The locations for test would be a bit different as it will not require annotation for all the fashion images of test data. This will be used for testing and evaluating the results. Below dataset registration needed to change when one trying to replicate the work.

```
import os
import numpy as np
import json

from detectron2.data.datasets import register_coco_instances
register_coco_instances("deep_fashion_train", {},
↪  "/mnt/batch/tasks/shared/LS_root/mounts/clusters/x201690431/cod
↪  e/Users/x20169043/deepfashion2/train/train.json",
"/mnt/batch/tasks/shared/LS_root/mounts/clusters/x201690431/cod
↪  e/Users/x20169043/deepfashion2/train/image/")
register_coco_instances("deep_fashion_valid", {},
↪  "/mnt/batch/tasks/shared/LS_root/mounts/clusters/x201690431/cod
↪  e/Users/x20169043/deepfashion2/validation/validation.json",
"/mnt/batch/tasks/shared/LS_root/mounts/clusters/x201690431/cod
↪  e/Users/x20169043/deepfashion2/validation/image/")
register_coco_instances("deep_fashion_test", {},
↪  "/mnt/batch/tasks/shared/LS_root/mounts/clusters/x201690431/cod
↪  e/Users/x20169043/d
↪  eepfashion2/test/json_for_test/retrieval_test_consumer_information.json",
↪  "/mnt/batch/tasks/shared/LS_root/mounts/clusters/x201690431/cod
↪  e/Users/x20169043/deepfashion2/test/image/")
```

Below code can help to install the required libraries such as torch, deetctron2, cython and so on.

```
# Installing the torch and torch vision
!pip3 install -U torch==1.5 torchvision==0.6 -f
↪  https://download.pytorch.org/whl/cu101/torch_stable.html
# Installling cython
!pip3 install cython pyyaml==5.1
#Installing coco related libraries
!pip3 install -U 'git+https://github.com/cocodataset/cocoapi.git#subd
↪  irectory=PythonAPI'
import torch, torchvision
print(torch.__version__, torch.cuda.is_available())
!gcc --version
# Installing Google colab library
!pip install google.colab
#Installing the detevtron2 API
!pip3 install detectron2==0.1.3 -f
↪  https://dl.fbaipublicfiles.com/detectron2/wheels/cu101/index.html
```

7 Actual Code

This section shows the actual python code of both models how the development is carried out so far.

7.1 Faster RCNN

Below code shows the consolidated JSON file creation for train dataset same way the validation dataset is also used just paths are changed as explained in section 6.

```
## The same code is running for validation dataaset as well as:
from PIL import Image
import numpy as np
import json

dataset = {
    "info": {},
    "licenses": [],
    "images": [],
    "annotations": [],
    "categories": []
}

lst_name = ['short_sleeved_shirt', 'long_sleeved_shirt',
↪  'short_sleeved_outwear', 'long_sleeved_outwear',
            'vest', 'sling', 'shorts', 'trousers', 'skirt',
             ↪  'short_sleeved_dress',
            'long_sleeved_dress', 'vest_dress', 'sling_dress']

for idx, e  in enumerate(lst_name):
    dataset['categories'].append({
        'id': idx + 1,
        'name': e,
        'supercategory': "clothes",
        'keypoints': ['%i' % (i) for i in range(1, 295)],
        'skeleton': []
    })

num_images = 5000  #32153
sub_index = 0  # the index of ground truth instance
for num in range(1, num_images + 1):
    json_name =
     ↪  '/mnt/batch/tasks/shared/LS_root/mounts/clusters/x201690431/cod⌋
     ↪  e/Users/x20169043/deepfashion2/validation/annos/' +
     ↪  str(num).zfill(6) + '.json'
    image_name =
     ↪  '/mnt/batch/tasks/shared/LS_root/mounts/clusters/x201690431/cod⌋
     ↪  e/Users/x20169043/deepfashion2/validation/image/' +
     ↪  str(num).zfill(6) + '.jpg'
```

```python
if (num >= 0):
    print(image_name)
    imag = Image.open(image_name)
    width, height = imag.size
    with open(json_name, 'r') as f:
        temp = json.loads(f.read())
        pair_id = temp['pair_id']

        dataset['images'].append({
            'coco_url': '',
            'date_captured': '',
            'file_name': str(num).zfill(6) + '.jpg',
            'flickr_url': '',
            'id': num,
            'license': 0,
            'width': width,
            'height': height
        })
        for i in temp:
            if i == 'source' or i == 'pair_id':
                continue
            else:
                points = np.zeros(294 * 3)
                sub_index = sub_index + 1
                box = temp[i]['bounding_box']
                w = box[2] - box[0]
                h = box[3] - box[1]
                x_1 = box[0]
                y_1 = box[1]
                bbox = [x_1, y_1, w, h]
                cat = temp[i]['category_id']
                style = temp[i]['style']
                seg = temp[i]['segmentation']
                landmarks = temp[i]['landmarks']

                points_x = landmarks[0::3]
                points_y = landmarks[1::3]
                points_v = landmarks[2::3]
                points_x = np.array(points_x)
                points_y = np.array(points_y)
                points_v = np.array(points_v)
                case = [0, 25, 58, 89, 128, 143, 158, 168, 182, 190,
                ↪ 219, 256, 275, 294]
                idx_i, idx_j = case[cat - 1], case[cat]

                for n in range(idx_i, idx_j):
                    points[3 * n] = points_x[n - idx_i]
```

```
                    points[3 * n + 1] = points_y[n - idx_i]
                    points[3 * n + 2] = points_v[n - idx_i]

                num_points = len(np.where(points_v > 0)[0])

                dataset['annotations'].append({
                    'area': w * h,
                    'bbox': bbox,
                    'category_id': cat,
                    'id': sub_index,
                    'pair_id': pair_id,
                    'image_id': num,
                    'iscrowd': 0,
                    'style': style,
                    'num_keypoints': num_points,
                    'keypoints': points.tolist(),
                    'segmentation': seg,
                })
# Writing the consolidated JSON file
json_name =
↪  '/mnt/batch/tasks/shared/LS_root/mounts/clusters/x201690431/cod
↪  e/Users/x20169043/deepfashion2/validation/validation.json'
with open(json_name, 'w') as f:
    json.dump(dataset, f)
```

Below code in snippet 12 shows the configurations which is used for Faster RCNN to train the 10k fashion images using the consolidated JSON file:

```
# select from modelzoo here: https://github.com/facebookresearch/detectron2/blob/master/MODEL_ZOO.md#coco-object-detection-baselines

from detectron2.config import get_cfg
#from detectron2.evaluation.coco_evaluation import COCOEvaluator
import os

cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/faster_rcnn_R_50_FPN_1x.yaml"))
cfg.DATASETS.TRAIN = ("deep_fashion_train",)
cfg.DATASETS.TEST = ("deep_fashion_valid",)

cfg.DATALOADER.NUM_WORKERS = 4
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/faster_rcnn_R_50_FPN_1x.yaml")  # Let training initialize from model zoo
cfg.SOLVER.IMS_PER_BATCH = 4
cfg.SOLVER.BASE_LR = 0.001


cfg.SOLVER.WARMUP_ITERS = 1000
cfg.SOLVER.MAX_ITER = 15000 #adjust up if val mAP is still rising, adjust down if overfit
cfg.SOLVER.STEPS = (1000, 1500)
cfg.SOLVER.GAMMA = 0.05

cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 64
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 14 #your number of classes + 1

#cfg.TEST.EVAL_PERIOD = 2500


os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
trainer = CocoTrainer(cfg)
trainer.resume_or_load(resume=True)
trainer.train()
```

Figure 12: Faster RCNN model configuration

7.2 Mask RCNN

For consolidated JSON creation is the same code which is explained in above section but the configuration which is used is bit different which is shown in below code snippet 13

```
#from .detectron2.tools.train_net import Trainer
#from detectron2.engine import DefaultTrainer
# select from modelzoo here: https://github.com/facebookresearch/detectron2/blob/master/MODEL_ZOO.md#coco-object-detection-baselines

from detectron2.config import get_cfg
#from detectron2.evaluation.coco_evaluation import COCOEvaluator
import os

cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rcnn_R_101_FPN_3x.yaml"))
cfg.DATASETS.TRAIN = ("deep_fashion_train",)
cfg.DATASETS.TEST = ("deep_fashion_valid",)

cfg.DATALOADER.NUM_WORKERS = 4
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask_rcnn_R_101_FPN_3x.yaml")  # Let training initialize from model zoo
cfg.SOLVER.IMS_PER_BATCH = 4
cfg.SOLVER.BASE_LR = 0.001


cfg.SOLVER.WARMUP_ITERS = 1000
cfg.SOLVER.MAX_ITER = 15000 #adjust up if val mAP is still rising, adjust down if overfit
cfg.SOLVER.STEPS = (1000, 1500)
cfg.SOLVER.GAMMA = 0.05




cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 64
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 14 #your number of classes + 1

#cfg.TEST.EVAL_PERIOD = 2500


os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
trainer = CocoTrainer(cfg)
trainer.resume_or_load(resume=True)
trainer.train()
```

Figure 13: Mask RCNN model configuration

Below code shows how the interferences are created on validation dataset. In the same way it has implemented in Faster RCNN.

```
from detectron2.data import DatasetCatalog, MetadataCatalog,
↪ build_detection_test_loader
from detectron2.evaluation import COCOEvaluator, inference_on_dataset

cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.75
predictor = DefaultPredictor(cfg)
evaluator = COCOEvaluator("deep_fashion_valid", cfg, False,
↪ output_dir="./output/")
val_loader = build_detection_test_loader(cfg, "deep_fashion_valid")
inference_on_dataset(trainer.model, val_loader, evaluator)
```

As part of evaluation used the COCO evaluator below code snippet shows how the summary has been collected.

```
# running evaluation
cocoEval = COCOeval(cocoGt,cocoDt,annType)
cocoEval.params.imgIds  = imgIds
cocoEval.evaluate()
cocoEval.accumulate()
cocoEval.summarize()
```