Gender Detection, Initial Approach

In [ ]:

```
import numpy as np
import os

import cv2
from google.colab.patches import cv2_imshow

from PIL import Image
import matplotlib.pyplot as plt

import torch
import torch.nn as nn
from torch.nn import Linear, ReLU, CrossEntropyLoss, Sequential, Conv2d, MaxPool2d, Module, Softmax, BatchNorm2
d, Dropout
from torch.optim import Adam, SGD
import torchvision
from torchvision import datasets, models, transforms


# loading dataset
!wget http://www.cs.toronto.edu/~fidler/teaching/2018/CSC420_assign/project1.zip
!unzip /content/project1.zip
```

Here we define 4 transformations, horizontal flip, perspective change, blur and color jitter. Each of these transformations will be applied to input data once to increase model robustness.

We also define a transformation for valid dataset which is just some resize and normalization.

In [2]:

```
transforms1_train = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])

transforms2_train = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomPerspective(),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])

transforms3_train = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.GaussianBlur(3),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])

transforms4_train = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ColorJitter(brightness=0.5, contrast=0.5, saturation=0.5, hue=0.5),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])

transforms_val = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])
```

Remove the unwanted `.mat` file

In [3]:

```
! rm /content/project1/train_data/female/*.mat
! rm /content/project1/train_data/male/*.mat
```

We load the training images 4 times, each time with a different type of augmentation to increase the model robustness.

In [ ]:

```
data_dir = '/content/project1'
train_sets = datasets.ImageFolder(os.path.join(data_dir, 'train_data'), transforms1_train)
train2_sets = datasets.ImageFolder(os.path.join(data_dir, 'train_data'), transforms2_train)
train3_sets = datasets.ImageFolder(os.path.join(data_dir, 'train_data'), transforms3_train)
train4_sets = datasets.ImageFolder(os.path.join(data_dir, 'train_data'), transforms4_train)

train_comb_sets = torch.utils.data.ConcatDataset([train_sets, train2_sets, train3_sets, train4_sets])

train_dataloader = torch.utils.data.DataLoader(train_comb_sets, batch_size=16, shuffle=True, num_workers=4)

class_names = train_sets.classes
```

We fine-tune our model based on a pretrained ResNet18.

We change the final classifier layers to a set of linear layers with batch norm and random dropout.

Then we also separate the classifier layers with the feature extraction convolutional layers.

In [6]:

```
model = models.resnet18(pretrained=True)

model.avgpool = nn.AdaptiveAvgPool2d(1)
model.fc = nn.Sequential(
    nn.BatchNorm1d(512),
    nn.Dropout(p=0.25),
    nn.Linear(in_features=512, out_features=512, bias=True),
    nn.ReLU(),
    nn.BatchNorm1d(512, eps=1e-05, momentum=0.1),
    nn.Dropout(p=0.5),
    nn.Linear(in_features=512, out_features=2, bias=True),
)
classifiers = []
features = []
for name,param in model.named_parameters():
    # print(name)
    if "fc" in name:
      classifiers.append(param)
    else:
      features.append(param)
print(model)
```

```
Downloading: "https://download.pytorch.org/models/resnet18-f37072fd.pth" to /root/.cache/torch/hub/c
heckpoints/resnet18-f37072fd.pth


ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (layer2): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
```

```
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (layer3): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (layer4): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=1)
  (fc): Sequential(
    (0): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (1): Dropout(p=0.25, inplace=False)
    (2): Linear(in_features=512, out_features=512, bias=True)
    (3): ReLU()
    (4): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=512, out_features=2, bias=True)
  )
)
```

Optimizer

We choose to use SGD, and adjust the learning rate for feature extraction layers and classifier layers separately.

In [7]:

```
# optimizer = Adam([
#                 {'params': classifiers},
#                 {'params': features, 'lr': 0.000005}
#             ], lr=0.000075)
optimizer = SGD([
                {'params': classifiers},
                {'params': features, 'lr': 0.0001}
            ], lr=0.001, momentum=0.9)
criterion = CrossEntropyLoss()
model = model.cuda()
```

In [8]:

```
from tqdm import tqdm
import statistics

def train_model(epochs, dataloader, criterion, optimizer):

  epochs = epochs

  for epoch in range(epochs):
    total_loss = []
    acc = []
    for i, (train_features, train_labels) in enumerate(tqdm(dataloader)):
        if train_features.shape[0] != 1:
            train_features = train_features.cuda()
            train_labels = train_labels.cuda()
            optimizer.zero_grad()
            out = model(train_features)
            _, pred = out.max(1)
            num_correct = torch.sum((pred == train_labels).int())
            loss = criterion(out, train_labels)
            loss.backward()
            optimizer.step()
            total_loss.append((loss / 16).detach().cpu().item())
            acc.append((num_correct / 16).detach().cpu().item())
    print("Epoch: {}, Avg loss: {}, Train accuracy: {}".format(epoch, statistics.mean(total_loss), statistics.mea
n(acc)))
```

In [9]:

```
train_model(epochs=6, dataloader=train_dataloader, criterion=criterion, optimizer=optimizer)
```

```
100%|██████████| 130/130 [00:05<00:00, 24.28it/s]

Epoch: 0, Avg loss: 0.026048324067288867, Train accuracy: 0.7971153846153847

100%|██████████| 130/130 [00:04<00:00, 26.45it/s]

Epoch: 1, Avg loss: 0.014246505396798826, Train accuracy: 0.9048076923076923

100%|██████████| 130/130 [00:04<00:00, 26.22it/s]

Epoch: 2, Avg loss: 0.011065836444210548, Train accuracy: 0.9293269230769231

100%|██████████| 130/130 [00:04<00:00, 26.48it/s]

Epoch: 3, Avg loss: 0.011179632119959794, Train accuracy: 0.93125

100%|██████████| 130/130 [00:04<00:00, 26.27it/s]

Epoch: 4, Avg loss: 0.0076411512223645475, Train accuracy: 0.9543269230769231

100%|██████████| 130/130 [00:04<00:00, 26.21it/s]

Epoch: 5, Avg loss: 0.007195076155655372, Train accuracy: 0.95625
```

In [10]:

```
image_path = "project1/clip_2"
```

In [11]:

```
transforms_simple = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]) # normalization
])
```

Here we clone and setup yolov5

In [12]:

```
!git clone https://github.com/ultralytics/yolov5.git
!pip install -r yolov5/requirements.txt
```

```
Cloning into 'yolov5'...
remote: Enumerating objects: 12870, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 12870 (delta 3), reused 5 (delta 1), pack-reused 12860
Receiving objects: 100% (12870/12870), 11.86 MiB | 43.07 MiB/s, done.
Resolving deltas: 100% (8938/8938), done.
Requirement already satisfied: matplotlib>=3.2.2 in /usr/local/lib/python3.7/dist-packages (from -r
yolov5/requirements.txt (line 4)) (3.2.2)
```

```
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.7/dist-packages (from -r yolo
v5/requirements.txt (line 5)) (1.21.6)
Requirement already satisfied: opencv-python>=4.1.2 in /usr/local/lib/python3.7/dist-packages (from
-r yolov5/requirements.txt (line 6)) (4.1.2.30)
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.7/dist-packages (from -r yolo
v5/requirements.txt (line 7)) (7.1.2)
Collecting PyYAML>=5.3.1
  Downloading PyYAML-6.0-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.man
ylinux2010_x86_64.whl (596 kB)
     |████████████████████████████████| 596 kB 5.0 MB/s
Requirement already satisfied: requests>=2.23.0 in /usr/local/lib/python3.7/dist-packages (from -r y
olov5/requirements.txt (line 9)) (2.23.0)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.7/dist-packages (from -r yolov
5/requirements.txt (line 10)) (1.4.1)
Requirement already satisfied: torch>=1.7.0 in /usr/local/lib/python3.7/dist-packages (from -r yolov
5/requirements.txt (line 11)) (1.10.0+cu111)
Requirement already satisfied: torchvision>=0.8.1 in /usr/local/lib/python3.7/dist-packages (from -r
yolov5/requirements.txt (line 12)) (0.11.1+cu111)
Requirement already satisfied: tqdm>=4.41.0 in /usr/local/lib/python3.7/dist-packages (from -r yolov
5/requirements.txt (line 13)) (4.64.0)
Requirement already satisfied: tensorboard>=2.4.1 in /usr/local/lib/python3.7/dist-packages (from -r
yolov5/requirements.txt (line 16)) (2.8.0)
Requirement already satisfied: pandas>=1.1.4 in /usr/local/lib/python3.7/dist-packages (from -r yolo
v5/requirements.txt (line 20)) (1.3.5)
Requirement already satisfied: seaborn>=0.11.0 in /usr/local/lib/python3.7/dist-packages (from -r yo
lov5/requirements.txt (line 21)) (0.11.2)
Collecting thop
  Downloading thop-0.0.31.post2005241907-py3-none-any.whl (8.7 kB)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from
matplotlib>=3.2.2->-r yolov5/requirements.txt (line 4)) (2.8.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/
dist-packages (from matplotlib>=3.2.2->-r yolov5/requirements.txt (line 4)) (3.0.8)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotl
ib>=3.2.2->-r yolov5/requirements.txt (line 4)) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from mat
plotlib>=3.2.2->-r yolov5/requirements.txt (line 4)) (1.4.2)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/d
ist-packages (from requests>=2.23.0->-r yolov5/requirements.txt (line 9)) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from re
quests>=2.23.0->-r yolov5/requirements.txt (line 9)) (2021.10.8)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests
>=2.23.0->-r yolov5/requirements.txt (line 9)) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from req
uests>=2.23.0->-r yolov5/requirements.txt (line 9)) (3.0.4)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from tor
ch>=1.7.0->-r yolov5/requirements.txt (line 11)) (4.1.1)
Requirement already satisfied: grpcio>=1.24.3 in /usr/local/lib/python3.7/dist-packages (from tensor
board>=2.4.1->-r yolov5/requirements.txt (line 16)) (1.44.0)
Requirement already satisfied: wheel>=0.26 in /usr/local/lib/python3.7/dist-packages (from tensorboa
rd>=2.4.1->-r yolov5/requirements.txt (line 16)) (0.37.1)
Requirement already satisfied: absl-py>=0.4 in /usr/local/lib/python3.7/dist-packages (from tensorbo
ard>=2.4.1->-r yolov5/requirements.txt (line 16)) (1.0.0)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-packages (from tenso
rboard>=2.4.1->-r yolov5/requirements.txt (line 16)) (3.3.6)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.7/dist-packag
es (from tensorboard>=2.4.1->-r yolov5/requirements.txt (line 16)) (1.8.1)
Requirement already satisfied: setuptools>=41.0.0 in /usr/local/lib/python3.7/dist-packages (from te
nsorboard>=2.4.1->-r yolov5/requirements.txt (line 16)) (57.4.0)
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.7/dist-packages (from ten
sorboard>=2.4.1->-r yolov5/requirements.txt (line 16)) (1.0.1)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.7/dis
t-packages (from tensorboard>=2.4.1->-r yolov5/requirements.txt (line 16)) (0.6.1)
Requirement already satisfied: protobuf>=3.6.0 in /usr/local/lib/python3.7/dist-packages (from tenso
rboard>=2.4.1->-r yolov5/requirements.txt (line 16)) (3.17.3)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.7/dist-pac
kages (from tensorboard>=2.4.1->-r yolov5/requirements.txt (line 16)) (0.4.6)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.7/dist-packages (from
tensorboard>=2.4.1->-r yolov5/requirements.txt (line 16)) (1.35.0)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=
1.1.4->-r yolov5/requirements.txt (line 20)) (2022.1)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from absl-py>=0.4->ten
sorboard>=2.4.1->-r yolov5/requirements.txt (line 16)) (1.15.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7/dist-packages (from
google-auth<3,>=1.6.3->tensorboard>=2.4.1->-r yolov5/requirements.txt (line 16)) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-packages (from google-
auth<3,>=1.6.3->tensorboard>=2.4.1->-r yolov5/requirements.txt (line 16)) (4.8)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.7/dist-packages (fro
m google-auth<3,>=1.6.3->tensorboard>=2.4.1->-r yolov5/requirements.txt (line 16)) (4.2.4)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.7/dist-packages (f
rom google-auth-oauthlib<0.5,>=0.4.1->tensorboard>=2.4.1->-r yolov5/requirements.txt (line 16)) (1.3
.1)
Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/python3.7/dist-packages (fr
om markdown>=2.6.8->tensorboard>=2.4.1->-r yolov5/requirements.txt (line 16)) (4.11.3)
```

```
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-m
etadata>=4.4->markdown>=2.6.8->tensorboard>=2.4.1->-r yolov5/requirements.txt (line 16)) (3.8.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist-packages (from
pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard>=2.4.1->-r yolov5/requirements.txt (line 1
6)) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-packages (from reque
sts-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard>=2.4.1->-r yolov5/requirements.tx
t (line 16)) (3.2.0)
Installing collected packages: thop, PyYAML
  Attempting uninstall: PyYAML
    Found existing installation: PyYAML 3.13
    Uninstalling PyYAML-3.13:
      Successfully uninstalled PyYAML-3.13
Successfully installed PyYAML-6.0 thop-0.0.31.post2005241907
```

Download the pretrained weight trained by me, using 70k images with 10 epochs.

In [13]:

```
## Download this for 50k images 8 epoches YOLOv5 medium model
# !wget https://www.dropbox.com/s/imoqdf12v59k9ov/yolo_face_50k_8epochs.zip?dl=1

## Download this for 70k images 10 epoches YOLOv5 medium model
!wget https://www.dropbox.com/s/8atygeezg7rvyf5/best.pt?dl=1
```

```
--2022-04-20 02:13:56--  https://www.dropbox.com/s/8atygeezg7rvyf5/best.pt?dl=1
Resolving www.dropbox.com (www.dropbox.com)... 162.125.5.18, 2620:100:601d:18::a27d:512
Connecting to www.dropbox.com (www.dropbox.com)|162.125.5.18|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/dl/8atygeezg7rvyf5/best.pt [following]
--2022-04-20 02:13:56--  https://www.dropbox.com/s/dl/8atygeezg7rvyf5/best.pt
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://ucc8b5bd104b2788260e66a5c631.dl.dropboxusercontent.com/cd/0/get/BjvRb1pyh5T3OGYI4K
eCpxzW0QBzQf4XxaTegPE_u5iZGaT6ejgxOjDN9MMVQqils0wxhPPwP9mSaIsi3fOUTUFO0K68zOwqKq5SeLFlv9N61-PQJOeq8G
RClUDlmDQXkw98beVaZAK9h-CiZieJJPUDU06o0z23I3vG5VkW4O2AGOslun9PWSPZ3xVJRuyFeLI/file?dl=1# [following]
--2022-04-20 02:13:57--  https://ucc8b5bd104b2788260e66a5c631.dl.dropboxusercontent.com/cd/0/get/Bjv
Rb1pyh5T3OGYI4KeCpxzW0QBzQf4XxaTegPE_u5iZGaT6ejgxOjDN9MMVQqils0wxhPPwP9mSaIsi3fOUTUFO0K68zOwqKq5SeLF
lv9N61-PQJOeq8GRClUDlmDQXkw98beVaZAK9h-CiZieJJPUDU06o0z23I3vG5VkW4O2AGOslun9PWSPZ3xVJRuyFeLI/file?dl
=1
Resolving ucc8b5bd104b2788260e66a5c631.dl.dropboxusercontent.com (ucc8b5bd104b2788260e66a5c631.dl.dr
opboxusercontent.com)... 162.125.5.15, 2620:100:601d:15::a27d:50f
Connecting to ucc8b5bd104b2788260e66a5c631.dl.dropboxusercontent.com (ucc8b5bd104b2788260e66a5c631.d
l.dropboxusercontent.com)|162.125.5.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 42155757 (40M) [application/binary]
Saving to: ‘best.pt?dl=1’

best.pt?dl=1        100%[===================>]  40.20M  49.1MB/s    in 0.8s

2022-04-20 02:13:58 (49.1 MB/s) - ‘best.pt?dl=1’ saved [42155757/42155757]
```

Here we rename the file to match the actual file name (remove the ?dl=1 from dropbox naming)

In [14]:

```
## Use this for 50k images 8 epoches YOLOv5 medium model
# !mv yolo_face_50k_8epochs.zip?dl=1 yolo_face_50k_8epochs.zip
# !unzip yolo_face_50k_8epochs.zip

## Use this for 70k images 10 epoches YOLOv5 medium model
!mv best.pt?dl=1 best.pt
```

Here, we use my pretrained yolo weight for face detection.

Remember to change the --source path to the clip path (i.e, /content/project1/clip_2)

Everytime you want to start the pipeline for a new clip, **start here** by changing --source path points to the clip

In [15]:

```
## Use this for 50k images 8 epoches YOLOv5 medium model's weight
# ! python yolov5/detect.py --save-txt --weights /content/content/yolov5/runs/train/exp2/weights/best.pt --sour
ce /content/project1/clip_1

## Use this for 70k images 10 epoches YOLOv5 medium model's weight
! python yolov5/detect.py --save-txt --weights /content/best.pt --source /content/project1/clip_2
```

```
Downloading https://ultralytics.com/assets/Arial.ttf to /root/.config/Ultralytics/Arial.ttf...
```

```
detect: weights=['/content/best.pt'], source=/content/project1/clip_2, data=yolov5/data/coco128.yaml
, imgsz=[640, 640], conf_thres=0.25, iou_thres=0.45, max_det=1000, device=, view_img=False, save_txt
=True, save_conf=False, save_crop=False, nosave=False, classes=None, agnostic_nms=False, augment=Fal
se, visualize=False, update=False, project=yolov5/runs/detect, name=exp, exist_ok=False, line_thickn
ess=3, hide_labels=False, hide_conf=False, half=False, dnn=False
YOLOv5  v6.1-139-gab5b917 torch 1.10.0+cu111 CUDA:0 (Tesla P100-PCIE-16GB, 16281MiB)

Fusing layers...
Model summary: 290 layers, 20852934 parameters, 0 gradients, 47.9 GFLOPs
image 1/135 /content/project1/clip_2/065.jpg: 384x640 4 faces, Done. (0.012s)
image 2/135 /content/project1/clip_2/066.jpg: 384x640 5 faces, Done. (0.010s)
image 3/135 /content/project1/clip_2/067.jpg: 384x640 5 faces, Done. (0.010s)
image 4/135 /content/project1/clip_2/068.jpg: 384x640 5 faces, Done. (0.010s)
image 5/135 /content/project1/clip_2/069.jpg: 384x640 5 faces, Done. (0.010s)
image 6/135 /content/project1/clip_2/070.jpg: 384x640 5 faces, Done. (0.010s)
image 7/135 /content/project1/clip_2/071.jpg: 384x640 5 faces, Done. (0.010s)
image 8/135 /content/project1/clip_2/072.jpg: 384x640 5 faces, Done. (0.010s)
image 9/135 /content/project1/clip_2/073.jpg: 384x640 5 faces, Done. (0.010s)
image 10/135 /content/project1/clip_2/074.jpg: 384x640 5 faces, Done. (0.010s)
image 11/135 /content/project1/clip_2/075.jpg: 384x640 5 faces, Done. (0.010s)
image 12/135 /content/project1/clip_2/076.jpg: 384x640 5 faces, Done. (0.010s)
image 13/135 /content/project1/clip_2/077.jpg: 384x640 5 faces, Done. (0.010s)
image 14/135 /content/project1/clip_2/078.jpg: 384x640 5 faces, Done. (0.010s)
image 15/135 /content/project1/clip_2/079.jpg: 384x640 5 faces, Done. (0.010s)
image 16/135 /content/project1/clip_2/080.jpg: 384x640 5 faces, Done. (0.010s)
image 17/135 /content/project1/clip_2/081.jpg: 384x640 5 faces, Done. (0.010s)
image 18/135 /content/project1/clip_2/082.jpg: 384x640 5 faces, Done. (0.010s)
image 19/135 /content/project1/clip_2/083.jpg: 384x640 5 faces, Done. (0.010s)
image 20/135 /content/project1/clip_2/084.jpg: 384x640 5 faces, Done. (0.010s)
image 21/135 /content/project1/clip_2/085.jpg: 384x640 5 faces, Done. (0.010s)
image 22/135 /content/project1/clip_2/086.jpg: 384x640 5 faces, Done. (0.010s)
image 23/135 /content/project1/clip_2/087.jpg: 384x640 5 faces, Done. (0.010s)
image 24/135 /content/project1/clip_2/088.jpg: 384x640 5 faces, Done. (0.010s)
image 25/135 /content/project1/clip_2/089.jpg: 384x640 5 faces, Done. (0.010s)
image 26/135 /content/project1/clip_2/090.jpg: 384x640 5 faces, Done. (0.010s)
image 27/135 /content/project1/clip_2/091.jpg: 384x640 5 faces, Done. (0.010s)
image 28/135 /content/project1/clip_2/092.jpg: 384x640 5 faces, Done. (0.010s)
image 29/135 /content/project1/clip_2/093.jpg: 384x640 5 faces, Done. (0.010s)
image 30/135 /content/project1/clip_2/094.jpg: 384x640 5 faces, Done. (0.010s)
image 31/135 /content/project1/clip_2/095.jpg: 384x640 5 faces, Done. (0.010s)
image 32/135 /content/project1/clip_2/096.jpg: 384x640 5 faces, Done. (0.010s)
image 33/135 /content/project1/clip_2/097.jpg: 384x640 5 faces, Done. (0.010s)
image 34/135 /content/project1/clip_2/098.jpg: 384x640 5 faces, Done. (0.010s)
image 35/135 /content/project1/clip_2/099.jpg: 384x640 5 faces, Done. (0.010s)
image 36/135 /content/project1/clip_2/100.jpg: 384x640 5 faces, Done. (0.010s)
image 37/135 /content/project1/clip_2/101.jpg: 384x640 5 faces, Done. (0.010s)
image 38/135 /content/project1/clip_2/102.jpg: 384x640 5 faces, Done. (0.010s)
image 39/135 /content/project1/clip_2/103.jpg: 384x640 5 faces, Done. (0.010s)
image 40/135 /content/project1/clip_2/104.jpg: 384x640 5 faces, Done. (0.010s)
image 41/135 /content/project1/clip_2/105.jpg: 384x640 5 faces, Done. (0.010s)
image 42/135 /content/project1/clip_2/106.jpg: 384x640 5 faces, Done. (0.010s)
image 43/135 /content/project1/clip_2/107.jpg: 384x640 5 faces, Done. (0.010s)
image 44/135 /content/project1/clip_2/108.jpg: 384x640 5 faces, Done. (0.010s)
image 45/135 /content/project1/clip_2/109.jpg: 384x640 5 faces, Done. (0.010s)
image 46/135 /content/project1/clip_2/110.jpg: 384x640 5 faces, Done. (0.009s)
image 47/135 /content/project1/clip_2/111.jpg: 384x640 5 faces, Done. (0.009s)
image 48/135 /content/project1/clip_2/112.jpg: 384x640 5 faces, Done. (0.009s)
image 49/135 /content/project1/clip_2/113.jpg: 384x640 5 faces, Done. (0.009s)
image 50/135 /content/project1/clip_2/114.jpg: 384x640 5 faces, Done. (0.009s)
image 51/135 /content/project1/clip_2/115.jpg: 384x640 5 faces, Done. (0.009s)
image 52/135 /content/project1/clip_2/116.jpg: 384x640 5 faces, Done. (0.009s)
image 53/135 /content/project1/clip_2/117.jpg: 384x640 5 faces, Done. (0.009s)
image 54/135 /content/project1/clip_2/118.jpg: 384x640 5 faces, Done. (0.009s)
image 55/135 /content/project1/clip_2/119.jpg: 384x640 5 faces, Done. (0.009s)
image 56/135 /content/project1/clip_2/120.jpg: 384x640 1 face, Done. (0.009s)
image 57/135 /content/project1/clip_2/121.jpg: 384x640 2 faces, Done. (0.009s)
image 58/135 /content/project1/clip_2/122.jpg: 384x640 4 faces, Done. (0.009s)
image 59/135 /content/project1/clip_2/123.jpg: 384x640 3 faces, Done. (0.009s)
image 60/135 /content/project1/clip_2/124.jpg: 384x640 2 faces, Done. (0.009s)
image 61/135 /content/project1/clip_2/125.jpg: 384x640 3 faces, Done. (0.009s)
image 62/135 /content/project1/clip_2/126.jpg: 384x640 4 faces, Done. (0.009s)
image 63/135 /content/project1/clip_2/127.jpg: 384x640 2 faces, Done. (0.009s)
image 64/135 /content/project1/clip_2/128.jpg: 384x640 2 faces, Done. (0.009s)
image 65/135 /content/project1/clip_2/129.jpg: 384x640 1 face, Done. (0.009s)
image 66/135 /content/project1/clip_2/130.jpg: 384x640 1 face, Done. (0.009s)
image 67/135 /content/project1/clip_2/131.jpg: 384x640 2 faces, Done. (0.009s)
image 68/135 /content/project1/clip_2/132.jpg: 384x640 1 face, Done. (0.009s)
image 69/135 /content/project1/clip_2/133.jpg: 384x640 3 faces, Done. (0.009s)
image 70/135 /content/project1/clip_2/134.jpg: 384x640 3 faces, Done. (0.009s)
image 71/135 /content/project1/clip_2/135.jpg: 384x640 2 faces, Done. (0.009s)
image 72/135 /content/project1/clip_2/136.jpg: 384x640 4 faces, Done. (0.009s)
image 73/135 /content/project1/clip_2/137.jpg: 384x640 2 faces, Done. (0.009s)
image 74/135 /content/project1/clip_2/138.jpg: 384x640 Done. (0.009s)
```

```
image 75/135 /content/project1/clip_2/139.jpg: 384x640 1 face, Done. (0.009s)
image 76/135 /content/project1/clip_2/140.jpg: 384x640 1 face, Done. (0.009s)
image 77/135 /content/project1/clip_2/141.jpg: 384x640 1 face, Done. (0.009s)
image 78/135 /content/project1/clip_2/142.jpg: 384x640 1 face, Done. (0.009s)
image 79/135 /content/project1/clip_2/143.jpg: 384x640 1 face, Done. (0.009s)
image 80/135 /content/project1/clip_2/144.jpg: 384x640 1 face, Done. (0.009s)
image 81/135 /content/project1/clip_2/145.jpg: 384x640 1 face, Done. (0.009s)
image 82/135 /content/project1/clip_2/146.jpg: 384x640 1 face, Done. (0.009s)
image 83/135 /content/project1/clip_2/147.jpg: 384x640 1 face, Done. (0.009s)
image 84/135 /content/project1/clip_2/148.jpg: 384x640 1 face, Done. (0.009s)
image 85/135 /content/project1/clip_2/149.jpg: 384x640 1 face, Done. (0.009s)
image 86/135 /content/project1/clip_2/150.jpg: 384x640 1 face, Done. (0.009s)
image 87/135 /content/project1/clip_2/151.jpg: 384x640 1 face, Done. (0.009s)
image 88/135 /content/project1/clip_2/152.jpg: 384x640 3 faces, Done. (0.009s)
image 89/135 /content/project1/clip_2/153.jpg: 384x640 3 faces, Done. (0.009s)
image 90/135 /content/project1/clip_2/154.jpg: 384x640 1 face, Done. (0.009s)
image 91/135 /content/project1/clip_2/155.jpg: 384x640 2 faces, Done. (0.009s)
image 92/135 /content/project1/clip_2/156.jpg: 384x640 2 faces, Done. (0.009s)
image 93/135 /content/project1/clip_2/157.jpg: 384x640 2 faces, Done. (0.009s)
image 94/135 /content/project1/clip_2/158.jpg: 384x640 2 faces, Done. (0.009s)
image 95/135 /content/project1/clip_2/159.jpg: 384x640 2 faces, Done. (0.009s)
image 96/135 /content/project1/clip_2/160.jpg: 384x640 2 faces, Done. (0.009s)
image 97/135 /content/project1/clip_2/161.jpg: 384x640 1 face, Done. (0.009s)
image 98/135 /content/project1/clip_2/162.jpg: 384x640 2 faces, Done. (0.009s)
image 99/135 /content/project1/clip_2/163.jpg: 384x640 1 face, Done. (0.009s)
image 100/135 /content/project1/clip_2/164.jpg: 384x640 1 face, Done. (0.009s)
image 101/135 /content/project1/clip_2/165.jpg: 384x640 1 face, Done. (0.009s)
image 102/135 /content/project1/clip_2/166.jpg: 384x640 1 face, Done. (0.009s)
image 103/135 /content/project1/clip_2/167.jpg: 384x640 1 face, Done. (0.009s)
image 104/135 /content/project1/clip_2/168.jpg: 384x640 1 face, Done. (0.009s)
image 105/135 /content/project1/clip_2/169.jpg: 384x640 1 face, Done. (0.009s)
image 106/135 /content/project1/clip_2/170.jpg: 384x640 1 face, Done. (0.009s)
image 107/135 /content/project1/clip_2/171.jpg: 384x640 1 face, Done. (0.009s)
image 108/135 /content/project1/clip_2/172.jpg: 384x640 1 face, Done. (0.009s)
image 109/135 /content/project1/clip_2/173.jpg: 384x640 1 face, Done. (0.009s)
image 110/135 /content/project1/clip_2/174.jpg: 384x640 1 face, Done. (0.009s)
image 111/135 /content/project1/clip_2/175.jpg: 384x640 1 face, Done. (0.009s)
image 112/135 /content/project1/clip_2/176.jpg: 384x640 Done. (0.009s)
image 113/135 /content/project1/clip_2/177.jpg: 384x640 1 face, Done. (0.009s)
image 114/135 /content/project1/clip_2/178.jpg: 384x640 1 face, Done. (0.009s)
image 115/135 /content/project1/clip_2/179.jpg: 384x640 1 face, Done. (0.009s)
image 116/135 /content/project1/clip_2/180.jpg: 384x640 1 face, Done. (0.009s)
image 117/135 /content/project1/clip_2/181.jpg: 384x640 1 face, Done. (0.009s)
image 118/135 /content/project1/clip_2/182.jpg: 384x640 1 face, Done. (0.009s)
image 119/135 /content/project1/clip_2/183.jpg: 384x640 1 face, Done. (0.009s)
image 120/135 /content/project1/clip_2/184.jpg: 384x640 1 face, Done. (0.009s)
image 121/135 /content/project1/clip_2/185.jpg: 384x640 1 face, Done. (0.009s)
image 122/135 /content/project1/clip_2/186.jpg: 384x640 1 face, Done. (0.009s)
image 123/135 /content/project1/clip_2/187.jpg: 384x640 3 faces, Done. (0.009s)
image 124/135 /content/project1/clip_2/188.jpg: 384x640 2 faces, Done. (0.009s)
image 125/135 /content/project1/clip_2/189.jpg: 384x640 2 faces, Done. (0.009s)
image 126/135 /content/project1/clip_2/190.jpg: 384x640 2 faces, Done. (0.009s)
image 127/135 /content/project1/clip_2/191.jpg: 384x640 2 faces, Done. (0.009s)
image 128/135 /content/project1/clip_2/192.jpg: 384x640 2 faces, Done. (0.009s)
image 129/135 /content/project1/clip_2/193.jpg: 384x640 2 faces, Done. (0.009s)
image 130/135 /content/project1/clip_2/194.jpg: 384x640 2 faces, Done. (0.009s)
image 131/135 /content/project1/clip_2/195.jpg: 384x640 2 faces, Done. (0.009s)
image 132/135 /content/project1/clip_2/196.jpg: 384x640 2 faces, Done. (0.009s)
image 133/135 /content/project1/clip_2/197.jpg: 384x640 2 faces, Done. (0.009s)
image 134/135 /content/project1/clip_2/198.jpg: 384x640 2 faces, Done. (0.009s)
image 135/135 /content/project1/clip_2/199.jpg: 384x640 2 faces, Done. (0.009s)
Speed: 0.4ms pre-process, 9.3ms inference, 0.9ms NMS per image at shape (1, 3, 640, 640)
Results saved to yolov5/runs/detect/exp
133 labels saved to yolov5/runs/detect/exp/labels
```

Save the path above that looks like "Results saved to **yolov5/runs/detect/expxxx**"

Remember the exp number, we will need to use it later.

Below are some auxillary codes for zipping and downloading intermediate results.

In [ ]:

```
!zip -r clip_2_res.zip /content/yolov5/runs/detect/exp
```

Here we use the detected faces label created by YOLO to populate a dictionary of bounding boxes used by later codes (face tracking / gender detection).

In [16]:

```
path = '/content/yolov5/runs/detect/exp/labels' # <-- '/content/yolov5/runs/detect/expxxx/labels' where xxx is th
e path where the detected labels are stored for yolo above.
                                                # For example, exp5 will be /content/yolov5/runs/detect/exp5/lab
els
image_path = '/content/project1/clip_2' # <-- This is where clip frames are stored. Change to clip_1/2/3

dir_list = os.listdir(path)
bbox = {}
for filename in dir_list:
  f = open(os.path.join(path,filename),'r')
  lines = f.read().split('\n')
  f.close()
  img_name = filename.split('.')[0]
  img = cv2.imread(os.path.join(image_path,img_name+'.jpg'))

  if img is not None:
    boxes = []
    for line in lines:
      f = filter(None, line.split(' '))
      part = list(f)
      if len(part) > 0:
        x = float(part[1])
        y = float(part[2])
        w = float(part[3])
        h = float(part[4])
        X, Y = img.shape[1], img.shape[0]
        W = w * X
        H = h * Y
        xcenter = x * X
        ycenter = y * Y
        xmin = int(xcenter - W/2)
        ymin = int(ycenter - H/2)
        xmax = int(xcenter + W/2)
        ymax = int(ycenter + H/2)
        boxes.append((xmin, ymin, xmax, ymax))
  bbox[os.path.join(image_path,img_name+'.jpg')] = boxes
```

In [ ]:

```
bbox
```

Face Tracking Across Frames

In [17]:

```
import cv2
import torch
import numpy as np
from collections import OrderedDict
from google.colab.patches import cv2_imshow
import dlib
import os
from PIL import Image
import matplotlib.pyplot as plt
from pathlib import Path
```

In [18]:

```
clip_img_paths = []
for file in Path(image_path).iterdir():
    if ".jpg" in file.name:
        clip_img_paths.append(file)
clip_img_paths = sorted(clip_img_paths, key=lambda i: int(os.path.splitext(os.path.basename(i))[0]))
```

In [19]:

```
def centroid_dist(curr, prev, curr_bbox, prev_bbox):
  """
  Compute the distance between two bounding box's centroid
  curr: current frame image
  prev: current frame image
  curr_bbox: one bounding box of the current frame
  prev_bbox: one bounding box of the current frame
  """
  curr_bbox_centroid = (curr_bbox[2]+curr_bbox[0])/2, (curr_bbox[3]+curr_bbox[1])/2
  prev_bbox_centroid = (prev_bbox[2]+prev_bbox[0])/2, (prev_bbox[3]+prev_bbox[1])/2
  # test
  #print(curr_bbox_centroid, "curr_bbox_centroid", prev_bbox_centroid, "prev_bbox_centroid")
  dist = np.linalg.norm(np.array(curr_bbox_centroid)-np.array(prev_bbox_centroid))
  return dist
```

In [20]:

```
def compute_distance(dist_function, curr, prev, bboxs_curr, bboxs_prev, boundary=50):
  # [(bbox1_index, [dist1, dist2...]), ()]
  bbox_dist = []
  bboxs_curr_assigned = []
  for bbox1_index in range(len(bboxs_curr)):
    dists = []
    for bbox2_index in range(len(bboxs_prev)):
      dist = dist_function(curr, prev, bboxs_curr[bbox1_index], bboxs_prev[bbox2_index])
      # test
      #print("test dist", dist,bboxs_curr[bbox1_index],bboxs_prev[bbox2_index], "test dist")
      dists.append(dist)
    if dists == []:
      dists = [float('inf')]
    bbox_dist.append((bbox1_index, dists))

  bbox_dist.sort(key=lambda s: min(s[1]))

  assigned_indexes = []
  for item in bbox_dist:
    bbox1_index = item[0]
    dists = item[1]
    # To prevent assigned a previously occurred value
    for i in assigned_indexes:
      dists[i] = float('inf')
    min_dist_bbox2_index = np.argmin(dists)
    assigned_indexes.append(min_dist_bbox2_index)
    # test
    #print("dists", dists)
    if dists[min_dist_bbox2_index] == float('inf'):
      min_dist_bbox2_index = None
    # If distance exceed boundary
    elif dists[min_dist_bbox2_index] >= boundary:
      min_dist_bbox2_index = None
    bboxs_curr_assigned.append((bbox1_index,min_dist_bbox2_index))
  return bboxs_curr_assigned
```

In [21]:

```
img_paths = clip_img_paths
face_detection_result = bbox
# Ordered dict to save a list of detected faces each time
# {id: (tracker, [(frame_id, bounding_box), ...])}
dist_function = centroid_dist
all_faces = {}
next_face_id = 0

# All the frames with face bounding boxes
frames_with_face_detection = set(face_detection_result.keys())

# Record the number of people that are currently tracking
# If become zero then we need to run face detection again
num_of_tracking = 0
prev_path = None
prev_img = None
bboxs_prev = []
for i in range(len(img_paths)):
  curr_path = str(img_paths[i])
  if curr_path in frames_with_face_detection:
    curr_img = cv2.imread(str(curr_path))
    boxes = face_detection_result[str(curr_path)]
    num_of_boxes = len(boxes) # If no face detected

    assigned_indexies = []
    if num_of_boxes > 0:

      #if bboxs_prev is not None and bboxs_prev != []:
      assignment = compute_distance(dist_function,curr_img, prev_img, boxes, bboxs_prev)


      all_faces[curr_path] = {}
      curr_bboxes_indexes = set(range(len(boxes)))
      for item in assignment:
        # Case 1 if curr has more boxes,
        # Case 2 if prev has more boxes
        # Case 3 if they are equal
        bbox1_index = item[0]
        assigned_indexies.append(bbox1_index)
        curr_box = boxes[bbox1_index]
        left_up_pt = curr_box[0], curr_box[1]
        righ_dow_pt = curr_box[2], curr_box[3]

        min_dist_bbox2_index = item[1]
        if min_dist_bbox2_index is not None:
          prev_box = bboxs_prev[min_dist_bbox2_index]
          prev_left_up_pt = prev_box[0], prev_box[1]
          prev_righ_dow_pt = prev_box[2], prev_box[3]
          prev_box = prev_left_up_pt, prev_righ_dow_pt
          prev_assigned_id = all_faces[prev_path][prev_box]
          box_pt = left_up_pt, righ_dow_pt
          all_faces[curr_path][box_pt] = prev_assigned_id
        else:
          box_pt = left_up_pt, righ_dow_pt
          all_faces[curr_path][box_pt] = next_face_id
          next_face_id += 1

      unassigned_indecies = curr_bboxes_indexes.difference(set(assigned_indexies))
      for unassigned_i in unassigned_indecies:
        curr_box = boxes[unassigned_i]
        left_up_pt = curr_box[0], curr_box[1]
        righ_dow_pt = curr_box[2], curr_box[3]
        box_pt = left_up_pt, righ_dow_pt
        all_faces[curr_path][box_pt] = next_face_id
        next_face_id += 1
      bboxs_prev = boxes
    else:
      all_faces[curr_path] = {}
      bboxs_prev = []

    prev_path = curr_path
    prev_img = curr_img
```

In [22]:

```
# Draw faces based on the tracks and save to some place
save_frames = {}
for key in all_faces.keys():
  frame_path = key
  curr_img = cv2.imread(str(frame_path))
  for box in all_faces[key].keys():
    left_up_pt = box[0]
    righ_dow_pt = box[1]
    image_drew_box = cv2.rectangle(curr_img,left_up_pt,
              righ_dow_pt, (36,255,12), 1)
    cv2.putText(image_drew_box, str(all_faces[key][box]), (int(left_up_pt[0]), int(left_up_pt[1])-10), cv2.FONT_H
ERSHEY_SIMPLEX, 0.8, (36,255,12), 2)
    save_frames[frame_path] = image_drew_box

# Since nota all frames has a bounding box
# here we save those frames without bounding box to save_frames
frames_with_bbox = set(save_frames.keys())
for path in img_paths:
  if str(path) not in frames_with_bbox:
    curr_img = cv2.imread(str(path))
    save_frames[str(path)] = curr_img
```

Show an example of frame #23

In [23]:

```
cv2_imshow(save_frames[list(save_frames.keys())[23]])
```

Do not run 2 cells below if you want to continue gender detection.

In [ ]:

```
outpath = '/content/result_clip_1'
if not os.path.isdir(outpath):
  os.makedirs(outpath)
for frame in save_frames:
    fname = os.path.split(frame)[-1]
    cv2.imwrite(os.path.join(outpath, fname),save_frames[frame])
```

In [ ]:

```
!ffmpeg -framerate 10 -pattern_type glob -i ./content/result_clip_1/'*.jpg' \
  -vf "pad=ceil(iw/2)*2:ceil(ih/2)*2"\
  -c:v libx264 -pix_fmt yuv420p out3.mp4
```

Gender Detection

In [24]:

```
for frame in list(save_frames.keys()):
    img_name = frame.split('.')[0]
    if frame in bbox:
        bounding_boxes = bbox[frame]
        img = cv2.imread(frame)
        for box in bounding_boxes:
            xmin = box[0]
            ymin = box[1]
            xmax = box[2]
            ymax = box[3]
            img22 = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            im_pil = Image.fromarray(img22)
            im_pil = im_pil.crop((xmin, ymin, xmax, ymax))

            # Use cropped face to detect gender
            im_pil = transforms_simple(im_pil).reshape(1, 3, 224, 224)
            model.eval()
            with torch.no_grad():
              output = model(im_pil.cuda())
              _, preds = torch.max(output, 1)

            # Draw gender
            cv2.putText(save_frames[frame], str(
                class_names[preds[0]]), (xmin, ymax), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2, cv2.LINE_AA)
```

Show an example of gender-detected frame #40

In [29]:

```
cv2_imshow(save_frames[list(save_frames.keys())[40]])
```

Below we save frames to file and zip it for downloading.

In [ ]:

```
if not os.path.exists('result_clip_1'):
      os.mkdir('result_clip_1')
```

In [ ]:

```
outpath = '/content/result_clip_1'
for frame in save_frames:
    fname = os.path.split(frame)[-1]
    cv2.imwrite(os.path.join(outpath, fname),save_frames[frame])
```

In [ ]:

```
!zip -r clip_3_results.zip /content/result_clip_1
```

You can also run below cell to generate video from frames

In [ ]:

```
# For clip 1 run this
# ! ffmpeg -framerate 10 -start_number 22 -i /content/result/%03d.jpg -vf "pad=ceil(iw/2)*2:ceil(ih/2)*2" output_
clip1.mp4
# For clip 2 run this
# ! ffmpeg -framerate 10 -start_number 65 -i /content/result2/%03d.jpg -vf "pad=ceil(iw/2)*2:ceil(ih/2)*2" output
_clip2.mp4
# For clip 3 run this
! ffmpeg -framerate 10 -start_number 16 -i /content/result_clip_3/%04d.jpg -vf "pad=ceil(iw/2)*2:ceil(ih/2)*2" ou
tput_clip3.mp4
```

```
ffmpeg version 3.4.8-0ubuntu0.2 Copyright (c) 2000-2020 the FFmpeg developers
  built with gcc 7 (Ubuntu 7.5.0-3ubuntu1~18.04)
  configuration: --prefix=/usr --extra-version=0ubuntu0.2 --toolchain=hardened --libdir=/usr/lib/x86
_64-linux-gnu --incdir=/usr/include/x86_64-linux-gnu --enable-gpl --disable-stripping --enable-avres
ample --enable-avisynth --enable-gnutls --enable-ladspa --enable-libass --enable-libbluray --enable-
libbs2b --enable-libcaca --enable-libcdio --enable-libflite --enable-libfontconfig --enable-libfreet
ype --enable-libfribidi --enable-libgme --enable-libgsm --enable-libmp3lame --enable-libmysofa --ena
ble-libopenjpeg --enable-libopenmpt --enable-libopus --enable-libpulse --enable-librubberband --enab
le-librsvg --enable-libshine --enable-libsnappy --enable-libsoxr --enable-libspeex --enable-libssh -
-enable-libtheora --enable-libtwolame --enable-libvorbis --enable-libvpx --enable-libwavpack --enabl
e-libwebp --enable-libx265 --enable-libxml2 --enable-libxvid --enable-libzmq --enable-libzvbi --enab
le-omx --enable-openal --enable-opengl --enable-sdl2 --enable-libdc1394 --enable-libdrm --enable-lib
iec61883 --enable-chromaprint --enable-frei0r --enable-libopencv --enable-libx264 --enable-shared
  libavutil      55. 78.100 / 55. 78.100
  libavcodec     57.107.100 / 57.107.100
  libavformat    57. 83.100 / 57. 83.100
  libavdevice    57. 10.100 / 57. 10.100
  libavfilter     6.107.100 /  6.107.100
  libavresample   3.  7.  0 /  3.  7.  0
  libswscale      4.  8.100 /  4.  8.100
  libswresample   2.  9.100 /  2.  9.100
  libpostproc    54.  7.100 / 54.  7.100
Input #0, image2, from '/content/result_clip_3/%04d.jpg':
  Duration: 00:00:27.50, start: 0.000000, bitrate: N/A
    Stream #0:0: Video: mjpeg, yuvj420p(pc, bt470bg/unknown/unknown), 854x480 [SAR 1:1 DAR 427:240],
10 fps, 10 tbr, 10 tbn, 10 tbc
Stream mapping:
  Stream #0:0 -> #0:0 (mjpeg (native) -> h264 (libx264))
Press [q] to stop, [?] for help
[libx264 @ 0x561bbfd57e00] using SAR=1/1
[libx264 @ 0x561bbfd57e00] using cpu capabilities: MMX2 SSE2Fast SSSE3 SSE4.2 AVX FMA3 BMI2 AVX2 AVX
512
[libx264 @ 0x561bbfd57e00] profile High, level 2.2
[libx264 @ 0x561bbfd57e00] 264 - core 152 r2854 e9a5903 - H.264/MPEG-4 AVC codec - Copyleft 2003-201
7 - http://www.videolan.org/x264.html - options: cabac=1 ref=3 deblock=1:0:0 analyse=0x3:0x113 me=he
x subme=7 psy=1 psy_rd=1.00:0.00 mixed_ref=1 me_range=16 chroma_me=1 trellis=1 8x8dct=1 cqm=0 deadzo
ne=21,11 fast_pskip=1 chroma_qp_offset=-2 threads=12 lookahead_threads=2 sliced_threads=0 nr=0 decim
ate=1 interlaced=0 bluray_compat=0 constrained_intra=0 bframes=3 b_pyramid=2 b_adapt=1 b_bias=0 dire
ct=1 weightb=1 open_gop=0 weightp=2 keyint=250 keyint_min=10 scenecut=40 intra_refresh=0 rc_lookahea
d=40 rc=crf mbtree=1 crf=23.0 qcomp=0.60 qpmin=0 qpmax=69 qpstep=4 ip_ratio=1.40 aq=1:1.00
Output #0, mp4, to 'output_clip3.mp4':
  Metadata:
    encoder         : Lavf57.83.100
    Stream #0:0: Video: h264 (libx264) (avc1 / 0x31637661), yuvj420p(pc), 854x480 [SAR 1:1 DAR 427:2
40], q=-1--1, 10 fps, 10240 tbn, 10 tbc
    Metadata:
      encoder         : Lavc57.107.100 libx264
    Side data:
      cpb: bitrate max/min/avg: 0/0/0 buffer size: 0 vbv_delay: -1
frame=  275 fps=125 q=-1.0 Lsize=    2754kB time=00:00:27.20 bitrate= 829.5kbits/s speed=12.3x
video:2751kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: 0.136694%
[libx264 @ 0x561bbfd57e00] frame I:8     Avg QP:18.61  size: 29036
[libx264 @ 0x561bbfd57e00] frame P:112   Avg QP:20.62  size: 13572
[libx264 @ 0x561bbfd57e00] frame B:155   Avg QP:24.22  size:  6861
[libx264 @ 0x561bbfd57e00] consecutive B-frames: 21.1% 10.2%  3.3% 65.5%
[libx264 @ 0x561bbfd57e00] mb I  I16..4: 31.2% 64.7%  4.2%
[libx264 @ 0x561bbfd57e00] mb P  I16..4: 16.2% 29.2%  1.8%  P16..4: 13.3%  7.2%  6.7%  0.0%  0.0%
skip:25.6%
[libx264 @ 0x561bbfd57e00] mb B  I16..4:  1.6%  5.5%  0.6%  B16..8: 26.8%  8.4%  3.9%  direct: 6.4%
skip:46.8%  L0:48.4% L1:42.9% BI: 8.7%
[libx264 @ 0x561bbfd57e00] 8x8 transform intra:63.7% inter:87.0%
[libx264 @ 0x561bbfd57e00] coded y,uvDC,uvAC intra: 49.9% 72.0% 27.1% inter: 18.3% 19.2% 6.3%
[libx264 @ 0x561bbfd57e00] i16 v,h,dc,p: 51% 30% 15%  4%
[libx264 @ 0x561bbfd57e00] i8 v,h,dc,ddl,ddr,vr,hd,vl,hu: 27% 19% 37%  3%  2%  2%  2%  4%  5%
[libx264 @ 0x561bbfd57e00] i4 v,h,dc,ddl,ddr,vr,hd,vl,hu: 41% 23% 13%  4%  4%  4%  4%  4%  3%
[libx264 @ 0x561bbfd57e00] i8c dc,h,v,p: 35% 24% 33%  8%
[libx264 @ 0x561bbfd57e00] Weighted P-Frames: Y:7.1% UV:4.5%
[libx264 @ 0x561bbfd57e00] ref P L0: 57.4% 14.1% 15.5% 12.9%  0.1%
[libx264 @ 0x561bbfd57e00] ref B L0: 75.5% 18.7%  5.7%
[libx264 @ 0x561bbfd57e00] ref B L1: 90.5%  9.5%
[libx264 @ 0x561bbfd57e00] kb/s:819.15
```

Gender classification result looks bad

Mainly due to the lack of data (only ~250 faces each gender)

Huge dataset to the rescue!

We got our gender classification dataset (~20k images) from kaggle:

https://www.kaggle.com/datasets/cashutosh/gender-classification-dataset (https://www.kaggle.com/datasets/cashutosh/gender-classification-dataset)

In [30]:

```
!pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.7/dist-packages (1.5.12)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from kaggle) (2021
.10.8)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from kaggle) (2.2
3.0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.7/dist-packages (from kaggle
) (6.1.1)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.7/dist-packages (from kaggle) (1.
15.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.7/dist-packages (from kaggl
e) (2.8.2)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from kaggle) (4.64.0)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (from kaggle) (1.24
.3)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.7/dist-packages (from p
ython-slugify->kaggle) (1.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from req
uests->kaggle) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests
->kaggle) (2.10)
```

Below please upload your `kaggle.json` from your PC.

If you don't have one, you can use mine https://drive.google.com/file/d/1PtOXY_jvQYBT3TFIOi0ovjHFdQc_71i-/view?usp=sharing (https://drive.google.com/file/d/1PtOXY_jvQYBT3TFIOi0ovjHFdQc_71i-/view?usp=sharing)

Please no distribution

A "Choose Files" prompt will show up allowing you to upload your .json file.

In [31]:

```
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))

# Then move kaggle.json into the folder where the API expects to find it.
!mkdir -p ~/.kaggle/ && mv kaggle.json ~/.kaggle/ && chmod 600 ~/.kaggle/kaggle.json
```

Choose Files No file selected

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving kaggle.json to kaggle.json
User uploaded file "kaggle.json" with length 64 bytes
```

Download the dataset.

In [32]:

```
!kaggle datasets download "cashutosh/gender-classification-dataset"
```

```
Downloading gender-classification-dataset.zip to /content
 95% 257M/269M [00:01<00:00, 245MB/s]
100% 269M/269M [00:01<00:00, 210MB/s]
```

In [ ]:

```
!unzip gender-classification-dataset.zip
```

Run below cell to train the model on new dataset.

In [34]:

```
data_dir = '/content'
train_sets = datasets.ImageFolder(os.path.join(data_dir, 'Training'), transforms1_train)
train2_sets = datasets.ImageFolder(os.path.join(data_dir, 'Training'), transforms2_train)
train3_sets = datasets.ImageFolder(os.path.join(data_dir, 'Training'), transforms3_train)
train4_sets = datasets.ImageFolder(os.path.join(data_dir, 'Training'), transforms4_train)

train_comb_sets = torch.utils.data.ConcatDataset([train_sets, train2_sets, train3_sets, train4_sets])

train_dataloader_big = torch.utils.data.DataLoader(train_comb_sets, batch_size=16, shuffle=True, num_workers=4)
criterion = nn.CrossEntropyLoss()

from tqdm import tqdm
import statistics

epochs = 5

for epoch in range(epochs):
  total_loss = []
  acc = []
  for i, (train_features, train_labels) in enumerate(tqdm(train_dataloader_big)):
      if train_features.shape[0] != 1:
          train_features = train_features.cuda()
          train_labels = train_labels.cuda()
          optimizer.zero_grad()
          out = model(train_features)
          _, pred = out.max(1)
          num_correct = torch.sum((pred == train_labels).int())
          loss = criterion(out, train_labels)
          loss.backward()
          optimizer.step()
          total_loss.append((loss / 16).detach().cpu().item())
          acc.append((num_correct / 16).detach().cpu().item())
  print("Epoch: {}, Avg loss: {}, Train accuracy: {}".format(epoch, statistics.mean(total_loss), statistics.mean(
acc)))
```

```
100%|██████████| 11753/11753 [06:58<00:00, 28.11it/s]

Epoch: 0, Avg loss: 0.007003380372533586, Train accuracy: 0.9594997022036926

100%|██████████| 11753/11753 [06:58<00:00, 28.11it/s]

Epoch: 1, Avg loss: 0.004747259096441121, Train accuracy: 0.9731664255934654

100%|██████████| 11753/11753 [06:58<00:00, 28.10it/s]

Epoch: 2, Avg loss: 0.00391523335136265, Train accuracy: 0.9777663149834085

100%|██████████| 11753/11753 [06:58<00:00, 28.10it/s]

Epoch: 3, Avg loss: 0.003200997192229729, Train accuracy: 0.981579171275419

100%|██████████| 11753/11753 [06:58<00:00, 28.11it/s]

Epoch: 4, Avg loss: 0.0026570982401105127, Train accuracy: 0.9848336594911937
```

Run below cell to validate the model.

In [ ]:

```
valid_dir = '/content'
valid_sets = datasets.ImageFolder(os.path.join(data_dir, 'Validation'), transforms1_train)
valid_dataloader_big = torch.utils.data.DataLoader(valid_sets, batch_size=16, shuffle=True, num_workers=4)
criterion = nn.CrossEntropyLoss()

from tqdm import tqdm
import statistics

epochs = 1

for epoch in range(epochs):
  model.eval()
  total_loss = []
  acc = []
  for i, (valid_features, valid_labels) in enumerate(tqdm(valid_dataloader_big)):
      if valid_features.shape[0] != 1:
        with torch.no_grad():
          valid_features = valid_features.cuda()
          valid_labels = valid_labels.cuda()
          out = model(valid_features)
          _, pred = out.max(1)
          num_correct = torch.sum((pred == valid_labels).int())
          loss = criterion(out, valid_labels)
          total_loss.append((loss / 16).detach().cpu().item())
          acc.append((num_correct / 16).detach().cpu().item())
  print("Epoch: {}, Avg loss: {}, Valid accuracy: {}".format(epoch, statistics.mean(total_loss), statistics.mean(
acc)))
```

```
100%|██████████| 729/729 [00:14<00:00, 51.05it/s]

Epoch: 0, Avg loss: 0.004863296787996836, Valid accuracy: 0.9730425824175825
```

Below cells re-inference the model again and detect gender.

The codes below are copy pasted from above, starting from "Face Tracking Across Frames" title to the end of "Gender Detection" title.

If you want you can go up and run those cells instead. Or you can run all cells below to produce the final output.

In [41]:

```
img_paths = clip_img_paths
face_detection_result = bbox
# Ordered dict to save a list of detected faces each time
# {id: (tracker, [(frame_id, bounding_box), ...])}
dist_function = centroid_dist
all_faces = {}
next_face_id = 0

# All the frames with face bounding boxes
frames_with_face_detection = set(face_detection_result.keys())

# Record the number of people that are currently tracking
# If become zero then we need to run face detection again
num_of_tracking = 0
prev_path = None
prev_img = None
bboxs_prev = []
for i in range(len(img_paths)):
  curr_path = str(img_paths[i])
  if curr_path in frames_with_face_detection:
    curr_img = cv2.imread(str(curr_path))
    boxes = face_detection_result[str(curr_path)]
    num_of_boxes = len(boxes) # If no face detected

    assigned_indexies = []
    if num_of_boxes > 0:

      #if bboxs_prev is not None and bboxs_prev != []:
      assignment = compute_distance(dist_function,curr_img, prev_img, boxes, bboxs_prev)


      all_faces[curr_path] = {}
      curr_bboxes_indexes = set(range(len(boxes)))
      for item in assignment:
        # Case 1 if curr has more boxes,
        # Case 2 if prev has more boxes
        # Case 3 if they are equal
        bbox1_index = item[0]
        assigned_indexies.append(bbox1_index)
        curr_box = boxes[bbox1_index]
        left_up_pt = curr_box[0], curr_box[1]
        righ_dow_pt = curr_box[2], curr_box[3]

        min_dist_bbox2_index = item[1]
        if min_dist_bbox2_index is not None:
          prev_box = bboxs_prev[min_dist_bbox2_index]
          prev_left_up_pt = prev_box[0], prev_box[1]
          prev_righ_dow_pt = prev_box[2], prev_box[3]
          prev_box = prev_left_up_pt, prev_righ_dow_pt
          prev_assigned_id = all_faces[prev_path][prev_box]
          box_pt = left_up_pt, righ_dow_pt
          all_faces[curr_path][box_pt] = prev_assigned_id
        else:
          box_pt = left_up_pt, righ_dow_pt
          all_faces[curr_path][box_pt] = next_face_id
          next_face_id += 1

      unassigned_indecies = curr_bboxes_indexes.difference(set(assigned_indexies))
      for unassigned_i in unassigned_indecies:
        curr_box = boxes[unassigned_i]
        left_up_pt = curr_box[0], curr_box[1]
        righ_dow_pt = curr_box[2], curr_box[3]
        box_pt = left_up_pt, righ_dow_pt
        all_faces[curr_path][box_pt] = next_face_id
        next_face_id += 1
      bboxs_prev = boxes
    else:
      all_faces[curr_path] = {}
      bboxs_prev = []

    prev_path = curr_path
    prev_img = curr_img
```

In [42]:

```
# Draw faces based on the tracks and save to some place
save_frames = {}
for key in all_faces.keys():
  frame_path = key
  curr_img = cv2.imread(str(frame_path))
  for box in all_faces[key].keys():
    left_up_pt = box[0]
    righ_dow_pt = box[1]
    image_drew_box = cv2.rectangle(curr_img,left_up_pt,
              righ_dow_pt, (36,255,12), 1)
    cv2.putText(image_drew_box, str(all_faces[key][box]), (int(left_up_pt[0]), int(left_up_pt[1])-10), cv2.FONT_H
ERSHEY_SIMPLEX, 0.8, (36,255,12), 2)
    save_frames[frame_path] = image_drew_box

# Since nota all frames has a bounding box
# here we save those frames without bounding box to save_frames
frames_with_bbox = set(save_frames.keys())
for path in img_paths:
  if str(path) not in frames_with_bbox:
    curr_img = cv2.imread(str(path))
    save_frames[str(path)] = curr_img
```

In [43]:

```
for frame in list(save_frames.keys()):
    img_name = frame.split('.')[0]
    if frame in bbox:
        bounding_boxes = bbox[frame]
        img = cv2.imread(frame)
        for box in bounding_boxes:
            xmin = box[0]
            ymin = box[1]
            xmax = box[2]
            ymax = box[3]
            img22 = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            im_pil = Image.fromarray(img22)
            im_pil = im_pil.crop((xmin, ymin, xmax, ymax))

            # Use cropped face to detect gender
            im_pil = transforms_simple(im_pil).reshape(1, 3, 224, 224)
            model.eval()
            with torch.no_grad():
              output = model(im_pil.cuda())
              _, preds = torch.max(output, 1)

            # Draw gender
            cv2.putText(save_frames[frame], str(
                class_names[preds[0]]), (xmin, ymax), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2, cv2.LINE_AA)
```

In [44]:

```
if not os.path.exists('result_clip_2'):
      os.mkdir('result_clip_2')
```

In [45]:

```
outpath = '/content/result_clip_2'
for frame in save_frames:
    fname = os.path.split(frame)[-1]
    cv2.imwrite(os.path.join(outpath, fname),save_frames[frame])
```

Download results

Lastly we zip our results, allowing it to be downloaded.

In [ ]:

```
!zip -r clip_2_results.zip /content/result_clip_2
```

Below code is to convert ipynb file to html for submission, please do not run.

In [ ]:

```
%%shell
jupyter nbconvert --to html /content/detection.ipynb
```

```
[NbConvertApp] Converting notebook /content/detection.ipynb to html
[NbConvertApp] Writing 2024890 bytes to /content/detection.html
```

Out[ ]: