TRƯỜNG ĐẠI HỌC HÀNG HẢI VIỆT NAM

VIỆN ĐÀO TẠO CHẤT LƯỢNG CAO

KHOA CÔNG NGHỆ THÔNG TIN

*****

BÀI TẬP LỚN

KĨ THUẬT HỌC SÂU VÀ ỨNG DỤNG

Đề tài:

Thuật toán CNN

Ứng dụng trong bài toán nhận diện giới tính bằng mô hình học sâu

Sinh viên: Vũ Tiến Chung

MSV: 82418

TRƯỜNG ĐẠI HỌC HÀNG HẢI VIỆT NAM
VIỆN ĐÀO TẠO CHẤT LƯỢNG CAO
KHOA CÔNG NGHỆ THÔNG TIN

-----*-----**

BÀI TẬP LỚN

KĨ THUẬT HỌC SÂU VÀ ỨNG DỤNG

1. Nội dung đề tài

Xây dựng chương trình cài đặt nhận diện giới tính bằng mô hình học sâu thuật toán CNN cho phép nhận diện giới tính qua webcam hoặc video.

2. Mục đích

Dựa vào kiến thức đã học của môn "Kĩ thuật học sâu và ứng dụng" để xây dựng chương trình nhận diện giới tính.

3. Công việc cần thực hiện

- Khảo sát, xác định các vấn đề mà hệ thống yêu cầu.
- Phân tích, tìm hiểu các thuật toán.
- Thiết kế và lập trình cho phù hợp với yêu cầu.
- Demo hoạt động của trương trình
- Kết luận về ưu, nhược điểm.

4. Yêu cầu

- Kết quả bài tập lớn: có sản phẩm thực tế và báo cáo bài tập lớn hoàn thiện.

Hải Phòng - 12/12/2022

MỤC LỤC

CHƯƠNG I: TỔNG QUAN ĐỀ TÀI

1.1. Lí do lựa chọn đề tài

Trong xu thế phát triển hiện nay, trên thế giới khoa học và công nghệ luôn có những thay đổi mạnh mẽ. Sự phát triển như vũ bão của ngành CNTT đã tác động to lớn đến mọi mặt đời sống kinh tế xã hội. Ngày nay, CNTT đã trở thành một trong những động lực quan trọng nhất của sự phát triển. Với khả năng số hóa mọi thông tin (số, đồ thị, văn bản, hình ảnh,tiếng nói,..), máy tính trở thành phương tiện xử lí thông tin thống nhất và đa năng, thực hiện được nhiều chức năng khác nhau trên mọi dạng thông tin thuộc mọi lĩnh vực: nghiên cứu, quản lí, kinh doanh…

Với vốn kiến thức đã học tại trường và nhu cầu cấp thiết của xã hội cộng thêm gợi ý của giảng viên bộ môn - thầy Nguyễn Hữu Tuân, tôi đã chọn đề tài " Xây dựng chương trình cài đặt nhận diện giới tính bằng mô hình học sâu thuật toán CNN cho phép nhận diện giới tính qua webcam hoặc video." với mong muốn giúp cho phát hiện và nhận diện giới tính của khuôn mặt.

1.2. Mục tiêu nghiên cứu

Bài toán áp dụng nhưngc kiến thức về môn "Kĩ thuật học sâu và ứng dụng" để xây dựng chương trình nhận diện giới tính.

Trong bài toán này mình sẽ xây dựng model nhận diện những khuôn mặt nam giới hay nữ giới được thu lại bởi webcam của máy tính hoặc video đưa vào chương trình, sau đó dùng model đó để nhận diện.

Bài toán xây dựng model sử dụng mạng Neuron, sử dụng thư viện Opencv với Anaconda, Tensorflow, karas, sklearn để nhận dạng giới tính của khuôn mặt được thu từ webcam của máy tính hoặc video.

1.3. Phạm vi bài toán

Chương trình hoạt động được và nhận diện được giới tính của gương mặt được đưa vào từ file video và webcam.

CHƯƠNG II: TỔNG QUAN VỀ MẠNG CNN

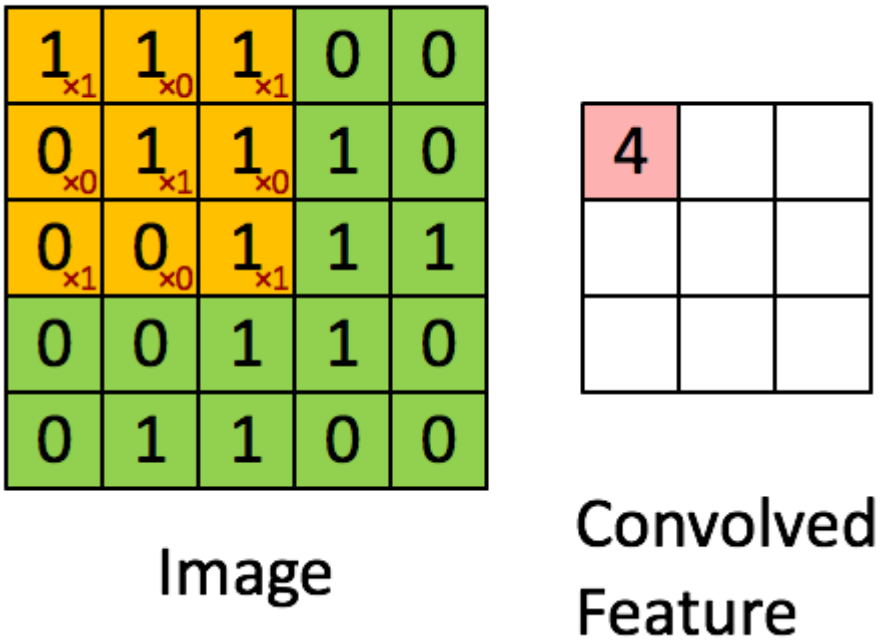
2.1. Mạng CNN (Convolutional Neural Network) là gì?

CNN là gì? Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến. Nó giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay.

CNN được sử dụng nhiều trong các bài toán nhận dạng các object trong ảnh. Để tìm hiểu tại sao thuật toán này được sử dụng rộng rãi cho việc nhận dạng (detection), chúng ta hãy cùng tìm hiểu về thuật toán này.

2.2. Tìm hiểu Convolutional là gì?

Là một cửa sổ trượt (Sliding Windows) trên một ma trận như mô tả hình dưới:

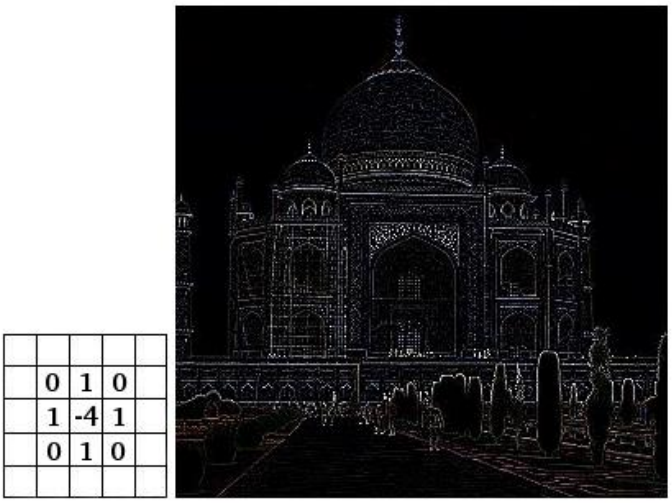


Các convolutional layer có các parameter(kernel) đã được học để tự điều chỉnh lấy ra những thông tin chính xác nhất mà không cần chọn các feature.

Trong hình ảnh ví dụ trên, ma trận bên trái là một hình ảnh trắng đen được số hóa. Ma trận có kích thước **5×5** và mỗi điểm ảnh có giá trị 1 hoặc 0 là giao điểm của dòng và cột.

Convolution hay tích chập là nhân từng phần tử trong ma trận 3. Sliding Window hay còn gọi là kernel, filter hoặc feature detect là một ma trận có kích thước nhỏ như trong ví dụ trên là 3×3.

Convolution hay tích chập là nhân từng phần tử bên trong ma trận 3×3 với ma trận bên trái. Kết quả được một ma trận gọi là Convoled feature được sinh ra từ việc nhận ma trận Filter với ma trận ảnh 5×5 bên trái.



	0	1	0	
	1	-4	1	
	0	1	0	

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

2.3. Cấu trúc mạng CNN

Mạng CNN là một tập hợp các lớp Convolution chồng lên nhau và sử dụng các hàm nonlinear activation như ReLU và tanh để kích hoạt các trọng số trong các node. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo.

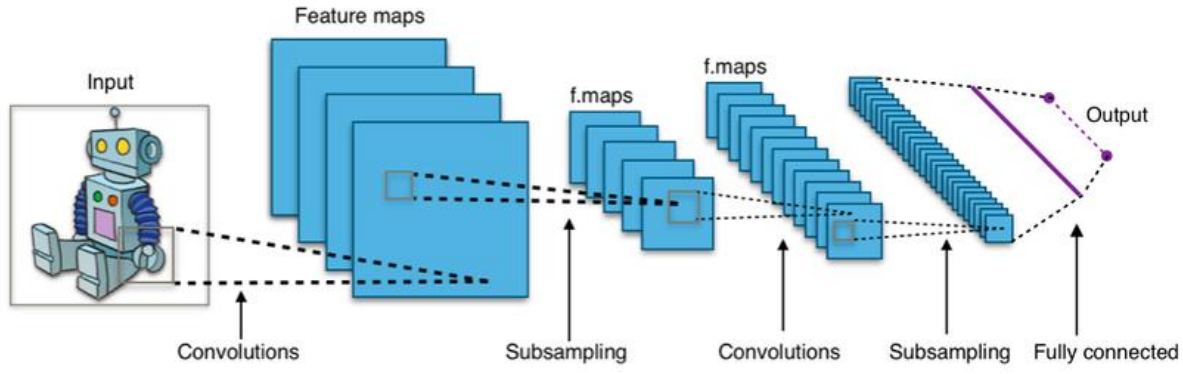
Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo. Trong mô hình mạng truyền ngược (feedforward neural network) thì mỗi neural đầu vào (input node) cho mỗi neural đầu ra trong các lớp tiếp theo.

Mô hình này gọi là mạng kết nối đầy đủ (fully connected layer) hay mạng toàn vẹn (affine layer). Còn trong mô hình CNNs thì ngược lại. Các layer liên kết được với nhau thông qua cơ chế convolution.

Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Như vậy mỗi neuron ở lớp kế tiếp sinh ra từ kết quả của filter áp đặt lên một vùng ảnh cục bộ của neuron trước đó.

Mỗi một lớp được sử dụng các filter khác nhau thông thường có hàng trăm hàng nghìn filter như vậy và kết hợp kết quả của chúng lại. Ngoài ra có một số layer khác như pooling/subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu).

Trong quá trình huấn luyện mạng (traning) CNN tự động học các giá trị qua các lớp filter dựa vào cách thức mà bạn thực hiện. Ví dụ trong tác vụ phân lớp ảnh, CNNs sẽ cố gắng tìm ra thông số tối ưu cho các filter tương ứng theo thứ tự raw pixel > edges > shapes > facial > high-level features. Layer cuối cùng được dùng để phân lớp ảnh.



Trong mô hình CNN có 2 khía cạnh cần quan tâm là **tính bất biến** (Location Invariance) và **tính kết hợp** (Compositionality). Với cùng một đối tượng, nếu đối tượng này được chiếu theo các gốc độ khác nhau (translation, rotation, scaling) thì độ chính xác của thuật toán sẽ bị ảnh hưởng đáng kể.

Pooling layer sẽ cho bạn tính bất biến đối với phép dịch chuyển (translation), phép quay (rotation) và phép co giãn (scaling). Tính kết hợp cục bộ cho ta các cấp độ biểu diễn thông tin từ mức độ thấp đến mức độ cao và trừu tượng hơn thông qua convolution từ các filter.

Đó là lý do tại sao CNNs cho ra mô hình với độ chính xác rất cao. Cũng giống như cách con người nhận biết các vật thể trong tự nhiên.

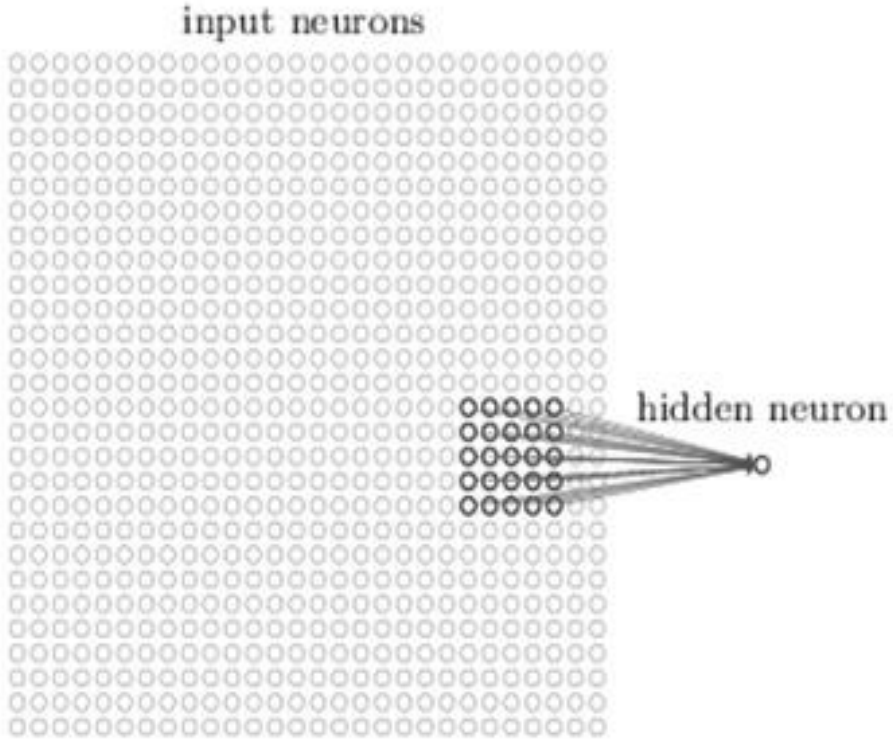
Mạng CNN sử dụng 3 ý tưởng cơ bản:

- **các trường tiếp nhận cục bộ** (local receptive field)
- **trọng số chia sẻ** (shared weights)
- **tổng hợp** (pooling).

Trường tiếp nhận cục bộ (local receptive field)

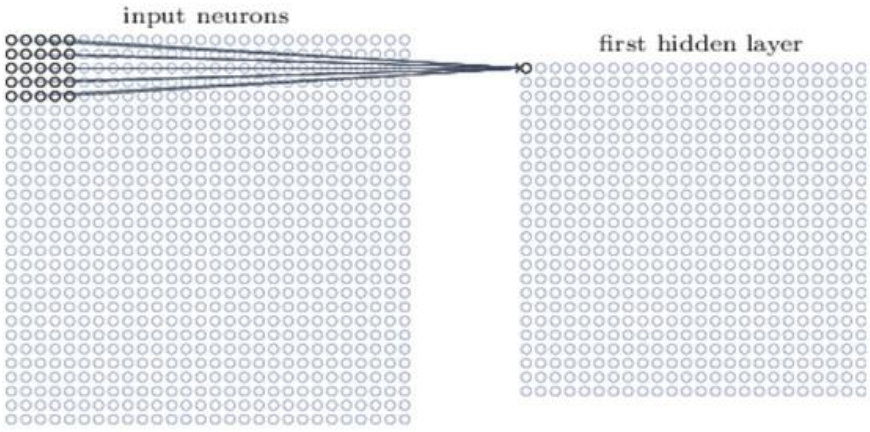
Đầu vào của mạng CNN là một ảnh. Ví dụ như ảnh có kích thước 28×28 thì tương ứng đầu vào là một ma trận có 28×28 và giá trị mỗi điểm ảnh là một ô trong ma trận. Trong mô hình mạng ANN truyền thống thì chúng ta sẽ kết nối các neuron đầu vào vào tầng ảnh.

Tuy nhiên trong CNN chúng ta không làm như vậy mà chúng ta chỉ kết nối trong một vùng nhỏ của các neuron đầu vào như một filter có kích thước 5×5 tương ứng (28- 5 + 1) 24 điểm ảnh đầu vào. Mỗi một kết nối sẽ học một trọng số và mỗi neuron ẩn sẽ học một bias. Mỗi một vùng 5×5 đấy gọi là một trường tiếp nhận cục bộ.

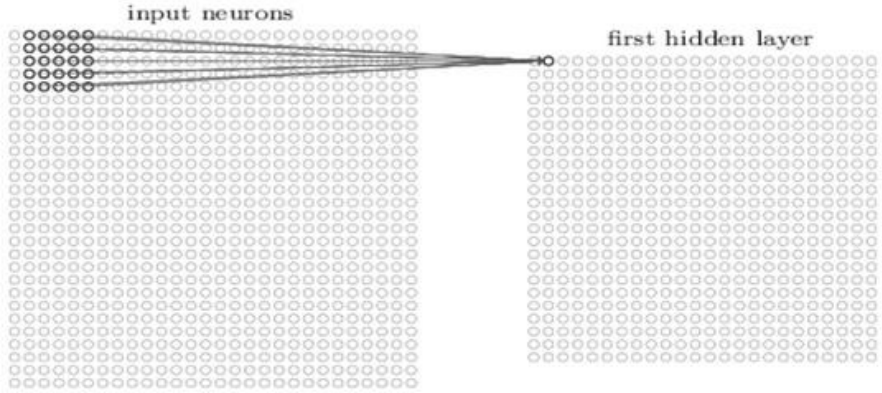


Một cách tổng quan, ta có thể tóm tắt các bước tạo ra 1 hidden layer bằng các cách sau:

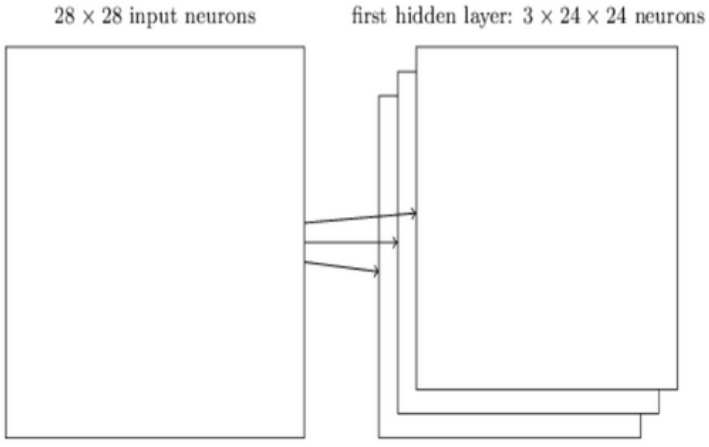
Tạo ra neuron ẩn đầu tiên trong lớp ẩn 1



Dịch filter qua bên phải một cột sẽ tạo được neuron ẩn thứ 2.



với bài toán nhận dạng ảnh người ta thường gọi ma trận lớp đầu vào là feature map, trọng số xác định các đặc trương là shared weight và độ lệch xác định một feature map là shared bias. Như vậy đơn giản nhất là qua các bước trên chúng ta chỉ có 1 feature map. Tuy nhiên trong nhận dạng ảnh chúng ta cần nhiều hơn một feature map.



Như vậy, local receptive field thích hợp cho việc phân tách dữ liệu ảnh, giúp chọn ra những vùng ảnh có giá trị nhất cho việc đánh giá phân lớp.

Trọng số chia sẻ (shared weight and bias)

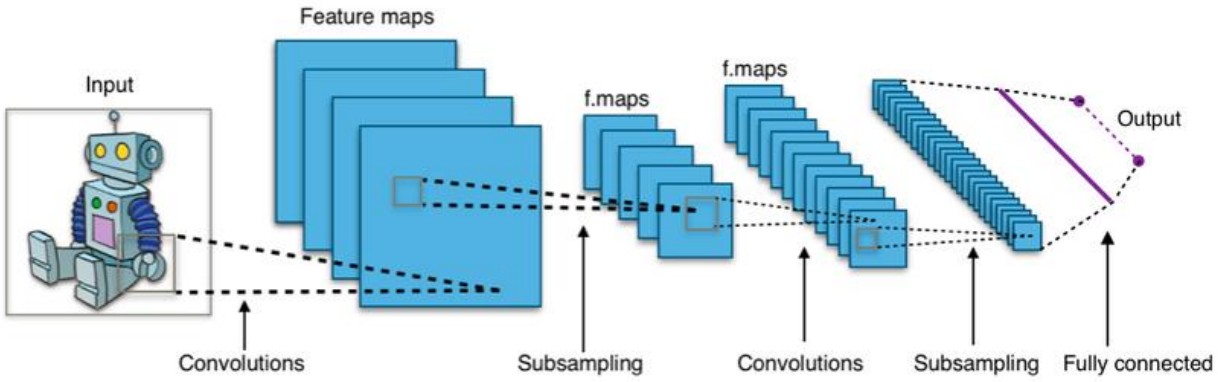
Đầu tiên, các trọng số cho mỗi filter (kernel) phải giống nhau. Tất cả các nơ-ron trong lớp ẩn đầu sẽ phát hiện chính xác feature tương tự chỉ ở các vị trí khác nhau trong hình ảnh đầu vào. Chúng ta gọi việc map từ input layer sang hidden layer là một feature map. Vậy mối quan hệ giữa số lượng Feature map với số lượng tham số là gì?

Chúng ta thấy mỗi fearture map cần 25 = 5×5 shared weight và 1 shared bias. Như vậy mỗi feature map cần 5×5+1 = 26 tham số. Như vậy nếu có 10 feature map thì có 10×26 = 260 tham số. Chúng ta xét lại nếu layer đầu tiên có kết nối đầy đủ nghĩa là chúng ta có 28×28=784 neuron đầu vào như vậy ta chỉ có 30 neuron ẩn. Như vậy ta cần 28x28x30 shared weight và 30 shared bias. Tổng số tham số là 28x28x30+30 tham số lớn hơn nhiều so với CNN. Ví dụ vừa rồi chỉ mô tả để thấy được sự ước lượng số lượng tham số chứ chúng ta không so sánh được trực tiếp vì 2 mô hình khác nhau. Nhưng điều chắc chắn là nếu mô hình có số lượng tham số ít hơn thì nó sẽ chạy nhanh hơn.

Tóm lại, một convolutional layer bao gồm các feature map khác nhau. Mỗi một feature map giúp detect một vài feature trong bức ảnh. Lợi ích lớn nhất của trọng số chia sẻ là giảm tối đa số lượng tham số trong mạng CNN.

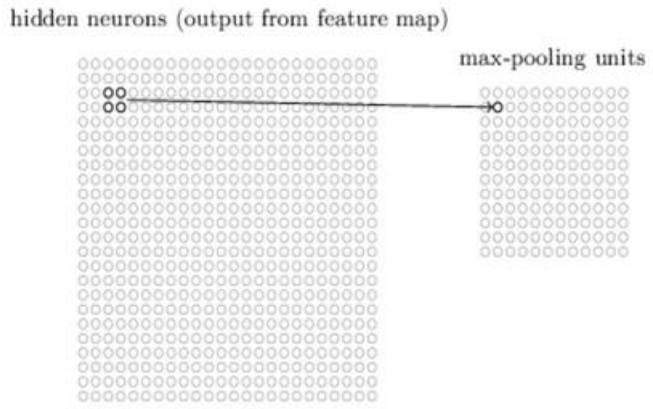
Lớp tổng hợp (pooling layer)

Lớp pooling thường được sử dụng ngay sau lớp convulational để đơn giản hóa thông tin đầu ra để giảm bớt số lượng neuron.



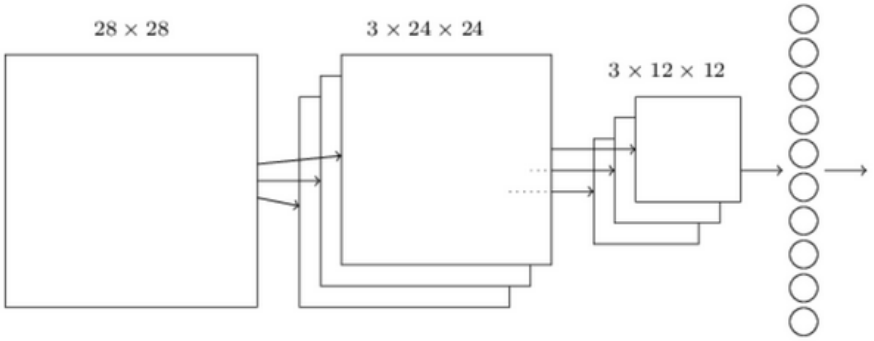
Thủ tục pooling phổ biến là max-pooling, thủ tục này chọn giá trị lớn nhất trong vùng đầu

vào 2×2.



Như vậy qua lớp Max Pooling thì số lượng neuron giảm đi phân nửa. Trong một mạng CNN có nhiều Feature Map nên mỗi Feature Map chúng ta sẽ cho mỗi Max Pooling khác nhau. Chúng ta có thể thấy rằng Max Pooling là cách hỏi xem trong các đặc trưng này thì đặc trưng nào là đặc trưng nhất. Ngoài Max Pooling còn có L2 Pooling.

Cuối cùng ta đặt tất cả các lớp lại với nhau thành một CNN với đầu ra gồm các neuron với số lượng tùy bài toán.



2 lớp cuối cùng của các kết nối trong mạng là một lớp đầy đủ kết nối (fully connected layer). Lớp này nối mọi nơron từ lớp max pooled tới mọi nơron của tầng ra.

Cách chọn tham số cho CNN

1. Số các convolution layer: càng nhiều các convolution layer thì performance càng được cải thiện. Sau khoảng 3 hoặc 4 layer, các tác động được giảm một cách đáng kể
2. Filter size: thường filter theo size 5×5 hoặc 3×3
3. Pooling size: thường là 2×2 hoặc 4×4 cho ảnh đầu vào lớn
4. Cách cuối cùng là thực hiện nhiều lần việc train test để chọn ra được param tốt nhất.

CHƯƠNG III: TỔNG QUAN VỀ CHƯƠNG TRÌNH

3.1. Các thành phần chính

3.1.1. Các thư viện sử dụng

TensorFlow cho phép xây dựng được nhiều neural network. Nó còn cho phép bạn tính toán song song trên nhiều máy tính khác nhau, thậm chí trên nhiều CPU, GPU trong cùng 1 máy hay tạo ra các dataflow graph – đồ thị luồng dữ liệu để dựng nên các model. Nếu bạn muốn chọn con đường sự nghiệp trong lĩnh vực A.I. này, nắm rõ những điều cơ bản của TensorFlow thực sự rất quan trọng.

Scikit-learn là một thư viện Python mã nguồn mở dành cho học máy. Thư viện này hỗ trợ các thuật toán hiện đại như KNN, XGBoost, random forest, SVM và một số thuật toán khác. Scikit-learn được sử dụng rộng rãi trong các cuộc thi kaggle cũng như trong các công ty công nghệ nổi tiếng. Nó giúp tiền xử lý, giảm chiều dữ liệu (lựa chọn tham số), phân loại, hồi quy, phân cụm và model selection.

OpenCV là một thư viện các chức năng lập trình chủ yếu nhắm vào thị giác máy tính thời gian thực.

Matplotlib Để thực hiện các suy luận thống kê cần thiết, cần phải trực quan hóa dữ liệu của bạn và Matplotlib là một trong những giải pháp như vậy cho người dùng Python. Nó là một thư viện vẽ đồ thị rất mạnh mẽ hữu ích cho những người làm việc với Python và NumPy. Module được sử dụng nhiều nhất của Matplotib là Pyplot cung cấp giao diện như MATLAB nhưng thay vào đó, nó sử dụng Python và nó là nguồn mở.

Pillow là một fork từ thư viện PIL của Python được sử dụng để xử lý hình ảnh. So với PIL thì Pillow được cập nhật thường xuyên và đánh giá cao hơn. (PIL đã không được cập nhật từ năm 2009).

Imageio là một thư viện Python cung cấp giao diện dễ đọc và ghi nhiều loại dữ liệu hình ảnh, bao gồm hình ảnh động, dữ liệu thể tích và định dạng khoa học.

3.1.2. Chương trình sử dụng

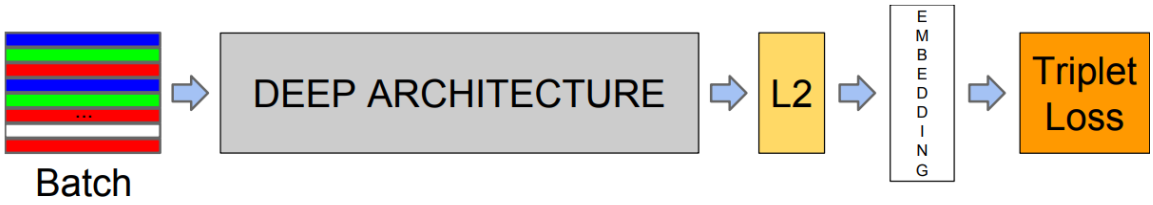
Anaconda là bản phân phối các ngôn ngữ lập trình Python và R cho tính toán khoa học, nhằm mục đích đơn giản hóa việc quản lý và triển khai gói. Bản phân phối bao gồm các

gói khoa học dữ liệu phù hợp với Windows, Linux và macOS. Anaconda Distribution hay Anaconda Individual Edition là những sản phẩm miễn phí thuộc Anaconda, Inc., trong khi các sản phẩm khác của công ty là Anaconda Team Edition và Anaconda Enterprise Edition đều không miễn phí.

Pycharm là một nền tảng kết kết hợp được JetBrains phát triển như một IDE (Môi trường phát triển tích hợp) để phát triển các ứng dụng cho lập trình trong Python. Một số ứng dụng lớn như Tweeter, Facebook, Amazon và Pinterest sử dụng Pycharm để làm IDE Python của họ.

Quá trình Training:

- *Sử dụng một tập Dataset với rất nhiều các cá thể người khác nhau, mỗi cá thể có một số lượng ảnh nhất định.*
- *Xây dựng một mạng CNN dùng để làm Feature Extractor cho Dataset trên, kết quả là 1 embedding 128-Dimensions. Trong paper có 2 đại diện mạng là Zeiler&Fergus và InceptionV1.*
- *Huấn luyện mạng CNN để kết quả embedding có khả năng nhận diện tốt, bao gồm 2 việc là sử dụng l2l2 normalization (Khoảng cách Euclide) cho các embeddings đầu ra và tối ưu lại các parameters trong mạng màng bằng Triplet Loss.*
- *Hàm Triplet Loss sẽ sử dụng phương pháp Triplet Selection, lựa chọn các embeddings sao cho việc học diễn ra tốt nhất.*



Quá trình Inference:

- *Truyền ảnh mặt cần classify vào trong mạng Feature Extractor, thu được 1 embedding.*
- *Tiến hành sử dụng hàm l2l2 và so sánh với các embedding khác trong tập embeddings đã có. Việc classify sẽ giống như thuật toán k-NN với k = 1.*

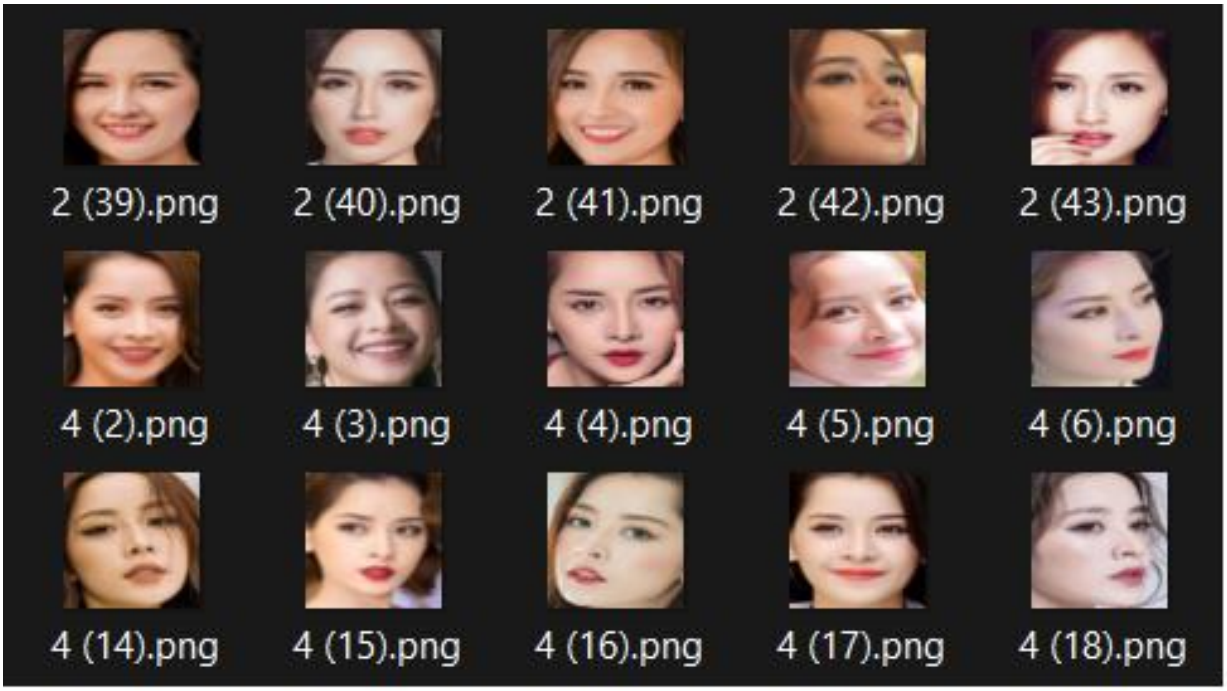
3.2. Code chương trình

3.2.1. Chuẩn bị

Dữ liệu được chuẩn bị là các hình ảnh gương mặt của những nghệ sĩ ở trong và ngoài nước bao gồm 1552 ảnh mặt nam và 1254 ảnh mặt nữ.

Mặt nam:

Mặt nữ:



Sau khi đã thu thập đủ dữ liệu về hình ảnh chuẩn bị thì bây giờ là build Model để sử dụng cho việc training.

Model CNN sử dụng cho bài toán nhận diện giới tính:

Một số những Layer Types tham chiếu sử dụng:

- Convolutional (CONV)
- Activation (ACT or RELU)
- Pooling (POOL)
- Fully-connected (FC)
- Batch Normalization (BN)
- Dropout (DO)

Stacking một tập các layers này theo một cách thức xác định chúng ta sẽ có được một mô hình CNN, một mô hình CNN có thể là từ đơn giản đến phức tạp tuỳ thuộc vào bao nhiêu tham số mà mô hình sử dụng. Sau đây sẽ là diagram cho một mô hình CNN đơn giản: INPUT => CONV => RELU => FC => SOFTMAX

Một điều quan trọng chúng ta cần phải nhớ đó là trong tập các layers này thì CONV, FC và BN là những layers sẽ chứa tham số, những tham số này chúng sẽ được học trong quá trình training, nhờ đó chúng ta có thể biết được các vị trí nào trong mô hình dẫn đến việc quá nhiều tham số để điều chỉnh lại. Activation và dropout không thực sự được coi là các layers, nhưng chúng vẫn được thêm vào mô hình để chúng ta thấy được kiến trúc mô hình rõ ràng và mạch lạc hơn.

Model tham chiếu:

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

Tham số sử dụng trong model CNN của bài toán:

epochs = 100

lr = 1e-3

batch_size = 64

img_dims = (96,96,3)

data = []

labels = []

Model thực tế của chương trình:

```
model.add(Conv2D(32, (3,3), padding="same", input_shape=inputShape))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(3,3)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3,3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))

model.add(Conv2D(64, (3,3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, (3,3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))

model.add(Conv2D(128, (3,3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
```

3.2.2. Model training

```
import tensorflow as ts
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from keras.utils import img_to_array
from keras.utils import to_categorical, plot_model
from keras.models import Sequential
from keras.layers import BatchNormalization, Conv2D, MaxPooling2D, Activation, Flatten, Dropout, Dense
from keras import backend as K
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
import random
import cv2
import os
import glob

# initial parameters
epochs = 100
lr = 1e-3
batch_size = 64
img_dims = (96,96,3)

data = []
labels = []

# load image files from the dataset
image_files = [f for f in glob.glob(r'gd-data' + "/**/*", recursive=True) if not os.path.isdir(f)]
```

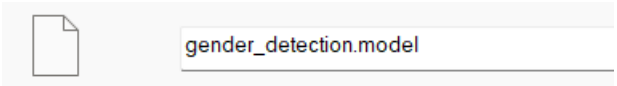
```
random.shuffle(image_files)
# converting images to arrays and labelling the categories
for img in image_files:

    image = cv2.imread(img)
    image = cv2.resize(image, (img_dims[0],img_dims[1]))
    image = img_to_array(image)
    data.append(image)

    label = img.split(os.path.sep)[-2] # gender_dataset_face\woman\face_1162.jpg
    if label == "woman":
        label = 1
    else:
        label = 0

    labels.append([label]) # [[1], [0], [0], ...]

# pre-processing
data = np.array(data, dtype="float") / 255.0
labels = np.array(labels)

# split dataset for training and validation
(trainX, testX, trainY, testY) = train_test_split(data, labels, test_size=0.2,
                                  random_state=42)
trainY = to_categorical(trainY, num_classes=2) # [[1, 0], [0, 1], [0, 1], ...]
testY = to_categorical(testY, num_classes=2)

# augmenting datset
aug = ImageDataGenerator(rotation_range=25, width_shift_range=0.1,
                         height_shift_range=0.1, shear_range=0.2, zoom_range=0.2,
```

```
                    horizontal_flip=True, fill_mode="nearest")


# define model
def build(width, height, depth, classes):
    model = Sequential()
    inputShape = (height, width, depth)
    chanDim = -1


    if K.image_data_format() == "channels_first": #Returns a string, either 'channels_first' or 'channels_last'
        inputShape = (depth, height, width)
        chanDim = 1


    # The axis that should be normalized, after a Conv2D layer with data_format="channels_first",
    # set axis=1 in BatchNormalization.


    model.add(Conv2D(32, (3,3), padding="same", input_shape=inputShape))
    model.add(Activation("relu"))
    model.add(BatchNormalization(axis=chanDim))
    model.add(MaxPooling2D(pool_size=(3,3)))
    model.add(Dropout(0.25))


    model.add(Conv2D(64, (3,3), padding="same"))
    model.add(Activation("relu"))
    model.add(BatchNormalization(axis=chanDim))


    model.add(Conv2D(64, (3,3), padding="same"))
    model.add(Activation("relu"))
    model.add(BatchNormalization(axis=chanDim))
```

```
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.25))

    model.add(Conv2D(128, (3,3), padding="same"))
    model.add(Activation("relu"))
    model.add(BatchNormalization(axis=chanDim))

    model.add(Conv2D(128, (3,3), padding="same"))
    model.add(Activation("relu"))
    model.add(BatchNormalization(axis=chanDim))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.25))

    model.add(Flatten())
    model.add(Dense(1024))
    model.add(Activation("relu"))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))

    model.add(Dense(classes))
    model.add(Activation("sigmoid"))

    return model

# build model
model = build(width=img_dims[0], height=img_dims[1], depth=img_dims[2],
                        classes=2)

# compile the model
opt = Adam(lr=lr, decay=lr/epochs)
```

```
model.compile(loss="binary_crossentropy", optimizer=opt, metrics=["accuracy"])

# train the model
H = model.fit_generator(aug.flow(trainX, trainY, batch_size=batch_size),
                        validation_data=(testX,testY),
                        steps_per_epoch=len(trainX) // batch_size,
                        epochs=epochs, verbose=1)

# save the model to disk
model.save('gender_detection.model')

# plot training/validation loss/accuracy
plt.style.use("ggplot")
plt.figure()
N = epochs
plt.plot(np.arange(0,N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0,N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0,N), H.history["acc"], label="train_acc")
plt.plot(np.arange(0,N), H.history["val_acc"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="upper right")

# save plot to disk
plt.savefig('plot.png')
```

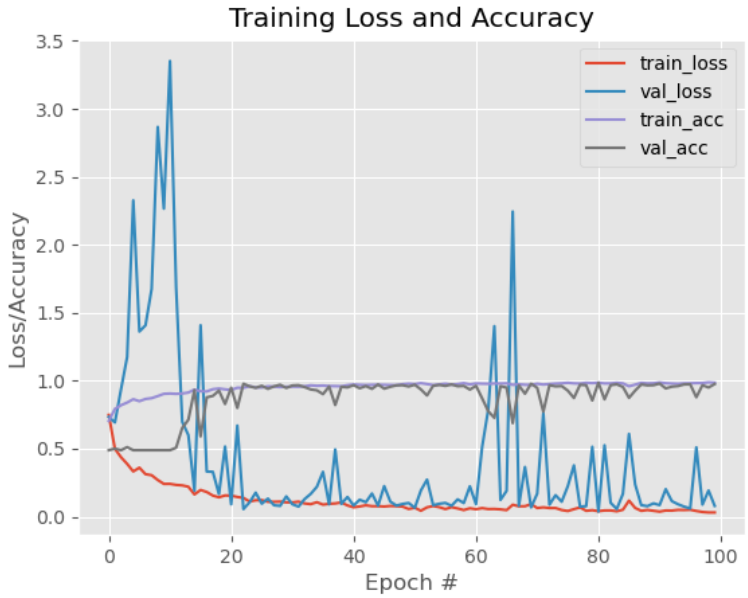
Kết quả training thu được là một Model được lưu vào như hình bên dưới:



```
Epoch 92/100
28/28 [==============================] - 34s 1s/step - loss: 0.0578 - accuracy: 0.9815 - val_loss: 0.1275 - val_accuracy: 0.9589
Epoch 93/100
28/28 [==============================] - 29s 1s/step - loss: 0.0469 - accuracy: 0.9820 - val_loss: 0.0952 - val_accuracy: 0.9805
Epoch 94/100
28/28 [==============================] - 25s 908ms/step - loss: 0.0365 - accuracy: 0.9888 - val_loss: 0.2241 - val_accuracy: 0.9502
Epoch 95/100
28/28 [==============================] - 24s 846ms/step - loss: 0.0406 - accuracy: 0.9837 - val_loss: 0.1200 - val_accuracy: 0.9610
Epoch 96/100
28/28 [==============================] - 21s 761ms/step - loss: 0.0324 - accuracy: 0.9876 - val_loss: 0.1829 - val_accuracy: 0.9567
Epoch 97/100
28/28 [==============================] - 23s 804ms/step - loss: 0.0467 - accuracy: 0.9860 - val_loss: 0.1535 - val_accuracy: 0.9589
Epoch 98/100
28/28 [==============================] - 30s 1s/step - loss: 0.0344 - accuracy: 0.9871 - val_loss: 0.0646 - val_accuracy: 0.9762
Epoch 99/100
28/28 [==============================] - 32s 1s/step - loss: 0.0411 - accuracy: 0.9832 - val_loss: 0.2044 - val_accuracy: 0.9459
Epoch 100/100
28/28 [==============================] - 21s 763ms/step - loss: 0.0306 - accuracy: 0.9893 - val_loss: 0.0891 - val_accuracy: 0.9762
```

Kết quả đánh giá độ chính xác của model là 98%

Đồ thì đánh giá độ chính xác của Model



Trong khi training model, em có sử dụng một đoạn code là EarlyStopping. Chương trình sẽ dừng quá trình training lại khi hàm đánh giá không giảm đi sau 3 epochs. Mục đích là để tiết kiệm thời gian và tài nguyên sử dụng cho việc training.

```
callback = ts.keras.callbacks.EarlyStopping(monitor='loss', patience=3)
model = ts.keras.models.Sequential([ts.keras.layers.Dense(10)])
model.compile(ts.keras.optimizers.SGD(), loss='mse')
history = model.fit(np.arange(100).reshape(5, 20), np.zeros(5),
```

```
                epochs=10, batch_size=1, callbacks=[callback],
                verbose=0)
len(history.history['loss'])  # Only 4 epochs are run.

model.save('gender_detection.model')
```

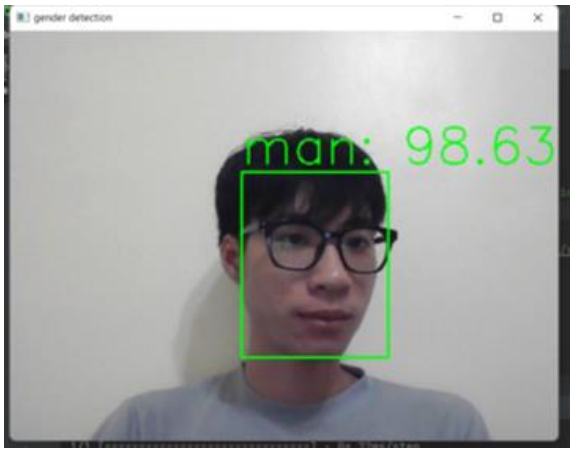
3.2.3. Chương trình chạy với web cam

```
import tensorflow as ts
from keras.utils import img_to_array
from keras.models import load_model
import numpy as np
import cv2
import os
import cvlib as cv

# load model
model = load_model('gender_detection.model')

# open webcam
webcam = cv2.VideoCapture(0)

classes = ['man','woman'
# loop through frames
while webcam.isOpened():

    # read frame from webcam
    status, frame = webcam.read()
```

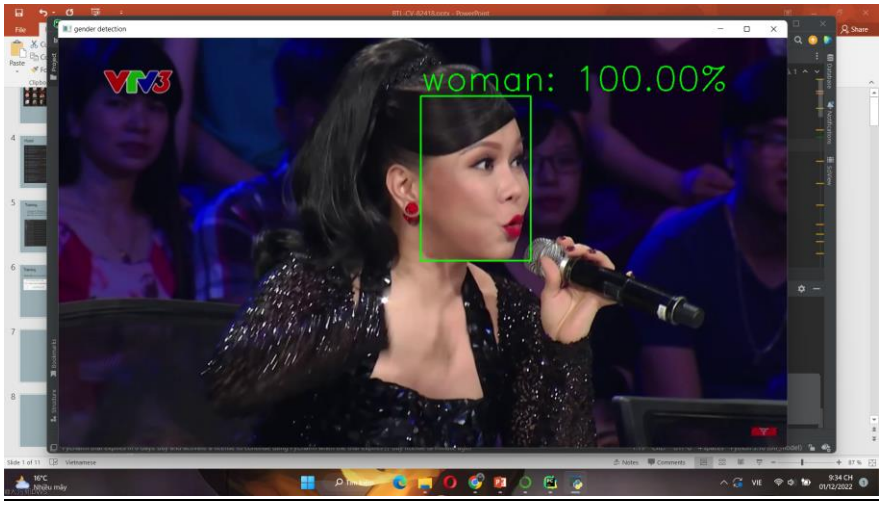
```
    # apply face detection
    face, confidence = cv.detect_face(frame)
    # loop through detected faces
    for idx, f in enumerate(face):

        # get corner points of face rectangle
        (startX, startY) = f[0], f[1]
        (endX, endY) = f[2], f[3]

        # draw rectangle over face
        cv2.rectangle(frame, (startX,startY), (endX,endY), (0,255,0), 2)

        # crop the detected face region
        face_crop = np.copy(frame[startY:endY,startX:endX])

        if (face_crop.shape[0]) < 10 or (face_crop.shape[1]) < 10:
            continue

        # preprocessing for gender detection model
        face_crop = cv2.resize(face_crop, (96,96))
        face_crop = face_crop.astype("float") / 255.0
        face_crop = img_to_array(face_crop)
        face_crop = np.expand_dims(face_crop, axis=0)

# apply gender detection on face
        conf = model.predict(face_crop)[0] # model.predict return a 2D matrix, ex:
[[9.9993384e-01 7.4850512e-05]]

        # get label with max accuracy
        idx = np.argmax(conf)
```

```
        label = classes[idx]

        label = "{}: {:.2f}%".format(label, conf[idx] * 100)

        Y = startY - 10 if startY - 10 > 10 else startY + 10

        # write label and confidence above face rectangle
        cv2.putText(frame, label, (startX, Y),  cv2.FONT_HERSHEY_SIMPLEX,
                    0.7, (0, 255, 0), 2)

    # display output
    cv2.imshow("gender detection", frame)

    # press "Q" to stop
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# release resources
webcam.release()
cv2.destroyAllWindows()
```

Ảnh Demo kết quả test với webcam:



Ảnh Demo kết quả test với video:



Chỉ có nữ trong khung hình.

Chỉ có nam trong khung hình.

Có cả nam nữ trong khung hình.

3.2.4. Ưu nhược điểm của chương trình

Ưu điểm:

- Xác định giới tính khá đúng.
- Độ chính xác tương đối cao, cụ thể là 98%.
- Cực nhạy trong việc thay đổi ánh sáng.
- Thuật toán hoạt động ổn định.
- đáp ứng được nhu cầu cơ bản.
- đưa ra được kết quả hiển thị.

Nhược điểm:

- Chạy chậm.

3.3. Kết luận

Đã chạy được chương trình.

Chương trình nhận diện tương đối chính xác cụ thể là 98%.

Đáp ứng được nhu cầu cơ bản.

3.4. Tài liệu tham khảo

https://www.youtube.com/watch?v=snmYELRQvME

https://vi.wikipedia.org/

https://github.com/

https://stackoverflow.com/

GVHD: thầy Nguyễn Hữu Tuân