

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



BÁO CÁO

Đồ án

XÁC ĐỊNH GIỚI TÍNH BẰNG HỌC SÂU

Thành viên :



Giảng viên hướng dẫn :

Nguyễn vinh tiệp

.

Tp. Hồ chí minh 1/2

Lời mở đầu

- Đạo gần đây, thế giới công nghệ đang dậy sóng bởi tính năng bảo mật mới được Apple giới thiệu trên chiếc điện thoại IphoneX. Mặc dù có rất nhiều nghi vấn liên quan đến tính bảo mật của chức năng này, tuy nhiên chúng ta không thể phủ nhận được thành quả công nghệ khá tuyệt vời mà họ đưa ra khi sử dụng một hệ thống hồng ngoại gọi là TrueDepth để tạo bản đồ lưới gồm 30.000 điểm ánh sáng trên gương mặt người dùng để tạo bản đồ 3D. Và có một điều dĩ nhiên rằng, chúng ta sẽ chỉ biết vậy thôi và không đời nào họ tiết lộ công nghệ của họ cho các bạn đâu =)) Nhưng về cơ bản thì công nghệ FaceID này đều được gọi chung là công nghệ nhận diện khuôn mặt và nó dĩ nhiên nó cao cấp hơn nhiều
- Một ví dụ khác, chắc hẳn mọi người đều đã từng một lần post một bức ảnh lên facebook, Bạn có để ý rằng, sau khi upload bức ảnh, facebook tự động tag một người bạn có mặt trong bức hình của bạn không? và rồi bạn thốt lên rằng, sao facebook giỏi vậy nhỉ, nó còn biết mình chụp ảnh với ai nữa này
- Thuật toán của Facebook có thể nhận ra khuôn mặt bạn bè của bạn sau khi họ đã được gắn thẻ chỉ một vài lần. Đó là công nghệ khá tuyệt vời - Facebook có thể nhận ra các khuôn mặt có độ chính xác 98% tương đương với con người!
- Trên cơ sở nhận diện khuôn mặt công nghệ hiện đại đòi hỏi chúng ta nhiều hơn nữa như nhận diện giới tính, cảm xúc con người thông qua khuôn mặt, đây như là bài toán mở rộng của nhận diện khuôn mặt.

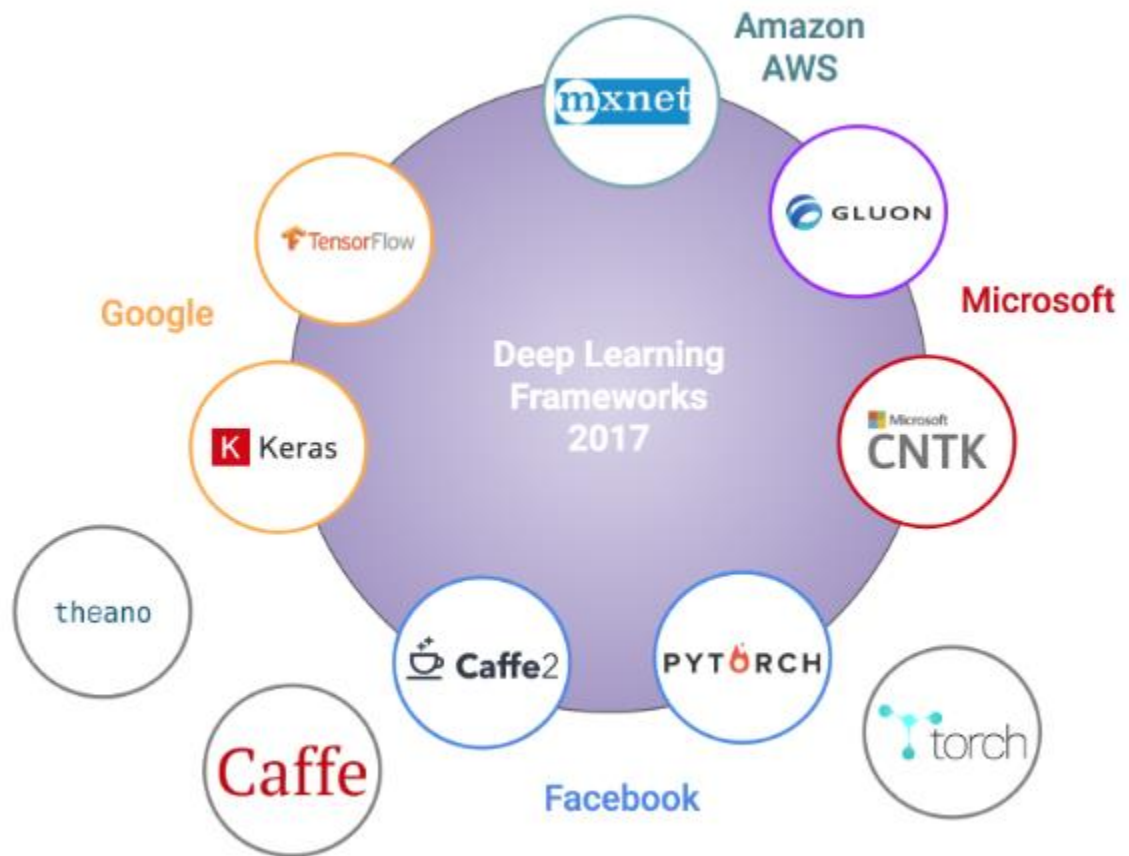
Mục lục

PHẦN . 1 .TÌM HIỂU KERAS.....	4
.1. Giới thiệu.....	4
.2. Keras core modules	7
.3. Keras metrics	10
.4. Keras models, losses, and optimizers.....	11
.5. Pre-trained CNN model in Keras	13
PHẦN . 2. PHÂN TÍCH VÀ THỰC HIỆN	16
.1. Tổng quan về bài toán phân lớp ảnh.....	16
.2. Xây dựng hệ thống	17
.2.1. Tiến hành xây dựng cơ sở dữ liệu ảnh khuôn mặt.....	18
PHẦN . 3.KẾT QUẢ , ĐÁNH GIÁ , SO SÁNH CÁC PHƯƠNG PHÁP	
KHÁC	29
.1. Kết quả	29
.2. Tài liệu tham khảo.....	36

PHẦN .1 .TÌM HIỂU KERAS

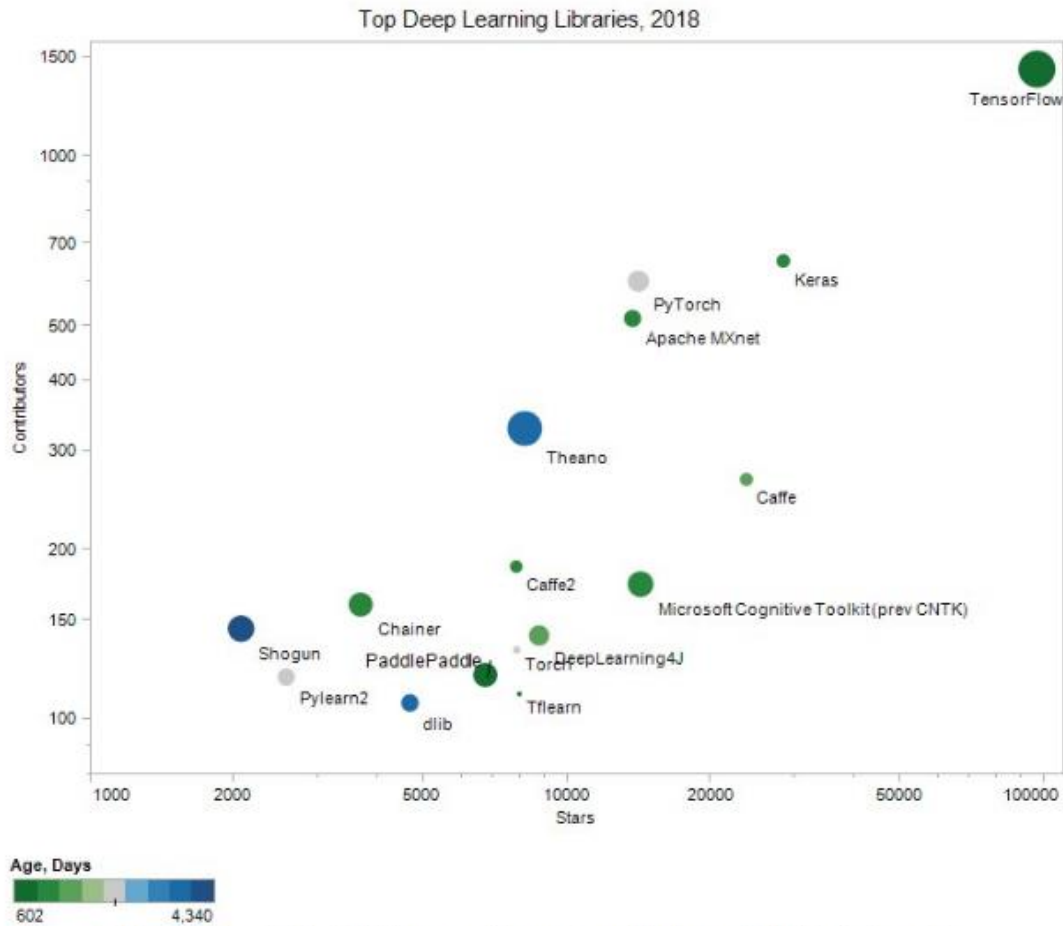
.1. Giới thiệu

- Machine Learning (ML), Deep Learning (DL) đang trở nên ngày càng phổ biến trong những năm gần đây. Chính vì sự phổ biến này cũng như tốc độ phát triển vô cùng nhanh chóng của ML cũng như DL trong hầu khắp các lĩnh vực, nhiều thư viện đã được xây dựng nhằm hỗ trợ cho việc xây dựng các mô hình ML, DL một cách nhanh chóng và dễ dàng hơn.
- Kể từ 2012 khi deep learning có bước đột phá lớn, hàng loạt các thư viện hỗ trợ deep learning ra đời. Cùng với đó, ngày càng nhiều kiến trúc deep learning ra đời, khiến cho số lượng ứng dụng và các bài báo liên quan tới deep learning tăng lên chóng mặt
- Các thư viện deep learning thường được ‘chống lưng’ bởi những hãng công nghệ lớn: Google (Keras, TensorFlow), Facebook (Caffe2, Pytorch), Microsoft (CNTK), Amazon (Mxnet), Microsoft và Amazon cũng đang bắt tay xây dựng Gluon (phiên bản tương tự như Keras). (Các hãng này đều có các dịch vụ cloud computing và muốn thu hút người dùng).



Các thư viện deep learning và các hãng công nghệ lớn.

- Thật khó có thể chỉ ra một thư viện đáp ứng tốt tất cả các yêu cầu .
- Xem xét sự phổ biến của keras



Số lượng 'stars' trên GitHub repo, số lượng 'contributors', và 'tuổi' của thư viện.

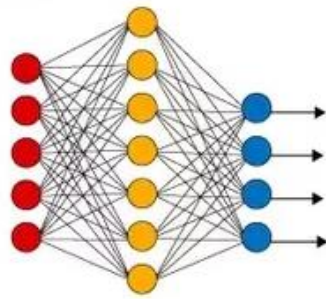
- So sánh trên đây chỉ ra rằng TensorFlow, Keras và Caffe là các thư viện được sử dụng nhiều nhất (gần đây có thêm PyTorch rất dễ sử dụng và đang thu hút thêm nhiều người dùng).
- Keras được coi là một thư viện 'high-level' với phần 'low-level' (còn được gọi là backend) có thể là TensorFlow, CNTK, hoặc Theano (Theano sẽ không được duy trì nâng cấp nữa). Keras có cú pháp đơn giản hơn TensorFlow rất nhiều.

- Một vài ưu điểm :
 - Keras ưu tiên trải nghiệm của người lập trình
 - Keras đã được sử dụng rộng rãi trong doanh nghiệp và cộng đồng nghiên cứu
 - Keras giúp dễ dàng biến các thiết kế thành sản phẩm
 - Keras hỗ trợ huấn luyện trên nhiều GPU phân tán
 - Keras hỗ trợ đa backend engines và không giới hạn bạn vào một hệ sinh thái
- Mô hình Keras có thể được huấn luyện trên một số nền tảng phần cứng khác nhau ngoài CPU:
 - NVIDIA GPU
 - Google TPUs, thông qua TensorFlow backend và Google Cloud
 - Các OpenCL GPU, chẳng hạn như các sản phẩm từ AMD, thông qua PlaidML Keras backend.

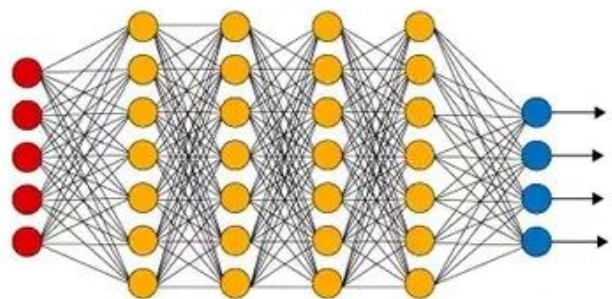
.2. Keras core modules

- Keras layers
 - DNN layer (Deep Neural Network) : DNN là kiến trúc mạng theo kiểu feed-forward network, các lớp neurons sẽ là các fully-connected layers được xếp liên tiếp nhau.

Simple Neural Network

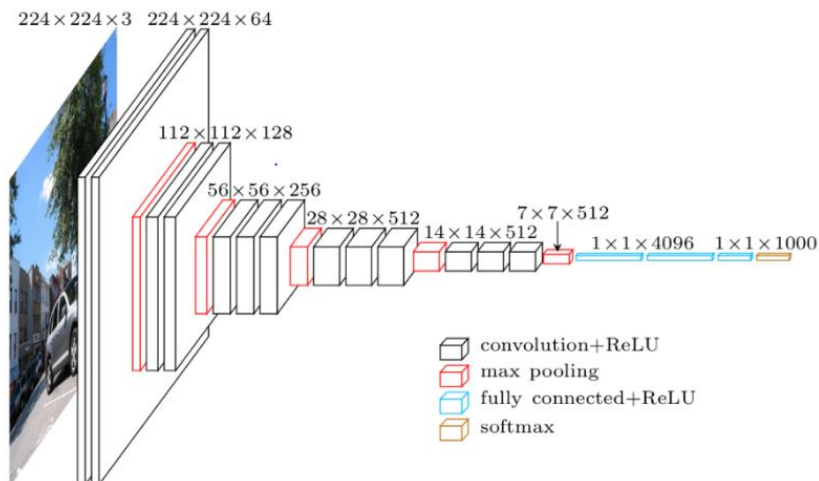


Deep Learning Neural Network



● Input Layer ● Hidden Layer ● Output Layer

- Đối với mạng DNN, Keras cung cấp 1 kiểu layers có tên là Dense (hay fully-connected-layers), cho phép chúng ta có thể xây dựng 1 Dense layer
- CNN layers



- CNN là một kiến trúc mạng neuron rất thích hợp cho các bài toán mà dữ liệu là ảnh hoặc video. Có 2 loại layer chính trong CNN: convolutional layer và pooling layer.

- Convolutional layer : có 2 loại convolutional layer hết sức cơ bản đó là convolutional layer 1D và 2D. Convolutional layer 1D hoạt động với dữ liệu đầu vào là một ma trận 3 chiều, trong khi convolutional layer 2D làm việc với dữ liệu đầu vào là ma trận 4 chiều
 - Keras hiện tại đang hỗ trợ nhiều loại pooling layers khác nhau như: max pooling, average pooling, global average pooling,...
 - RNN layer : Với kiến trúc mạng RNN, có 2 loại layer thường được dùng bao gồm: GRU, LSTM. Bên cạnh đó Keras còn cung cấp wrapper cho phép ta sử dụng các layers RNN 2 chiều (bi-directional).
- Keras advanced layers
 - Dropout layer :
 - Trong quá trình xây dựng các mô hình , hai vấn đề thường gặp nhất là overfitting và underfitting
 - Về cơ bản thì underfitting dễ dàng xử lý hơn overfitting rất nhiều
 - Để xử lý overfitting có hai cách được áp dụng rất phổ biến đó là: dropout và regularization.
 - Với dropout, keras cung cấp sẵn 1 layer, chúng ta chỉ cần gọi layer ra, truyền giá trị dropout rate (0-1) và sử dụng.
 - Về bản chất dropout ở đây sẽ ngắt ngẫu nhiên 1 số nút từ lớp hiện tại sang nút tiếp theo để kết quả không quá khớp với tập train.

- Flatten layer
 - Trong quá trình train ,ma trận ảnh ban đầu sẽ thay đổi kích thước rất nhiều , đến bước cuối cùng khi thực hiện hàm softmax ta cần phải biến ma trận thành vecto 1 chiều .
 - Có 2 cách để làm việc này: sử dụng Flatten layer hoặc Global Pooling layer
 - Hiểu 1 cách đơn giản thì nhiệm vụ của thằng Flatten layer là, đúng như cái tên của nó, "trải phẳng" ma trận ra thành 1 vector
- Global pooling layer
 - Flatten layer biến đổi ma trận thành vecto 1 chiều , nhưng output lại có quá nhiều chiều , global pooling là cách giải quyết
 - Keras cung cấp 2 loại Global Pooling layer khác nhau: GlobalMaxPooling và GlobalAveragePooling
 - GlobalMaxPooling sẽ tìm giá trị lớn nhất trong khi GlobalAveragePooling sẽ tính trung bình cộng
- Reshape layer : Reshape layer có nhiệm vụ thay đổi kích thước ma trận theo ý muốn

.3. Keras metrics

- Công cụ tuyệt vời giúp theo dõi độ chính xác của mô hình Deep Learning trong quá trình training.
- Binary accuracy : cho phép chúng ta biết độ chính xác hiện tại của mô hình trong bài toán phân loại và cụ thể là bài toán phân loại 2 lớp.

```
# compile model with metric
model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=[binary_accuracy,])
```

- Categorical accuracy và sparse categorical accuracy :
 - Với 2 metrics này, bạn có thể biết được độ chính xác của mô hình trong bài toán multiclass classification (phân lớp với số lượng class lớn hơn 2)
 - **categorical_crossentropy** được dùng trong bài toán multiclass-classification
 - **sparse categorical_crossentropy** cũng cho bài toán multiclass-classification nhưng với label chưa được đưa về dạng one-hot

.4. Keras models, losses, and optimizers

- Trong Keras, có 2 cách có thể xây dựng 1 mô hình DL: Model (functional API) và Sequential. Với cả 2 cách, các bạn đều có thể xây dựng được 1 mô hình 1 cách dễ dàng.
 - Model (functional API) :

```
from keras.models import Model
from keras.layers import Input
from keras.layers import Dense

# khai báo input_layer (lớp đầu vào của mạng neuron)
input_layer = Input(shape=(69,), name = 'input')
# Sử dụng 3 lớp Dense khác nhau
dense_1 = Dense(units = 100, activation='relu')(input_layer)
dense_2 = Dense(units = 50, activation='relu')(dense_1)
dense_3 = Dense(units = 2, activation='relu')(dense_2)
model = Model(inputs = input_layer, outputs = dense_3)
print (model.summary())
```

- Sequential

```
# 2. Build model
model = Sequential()
model.add(Dense(1, input_shape=(2,)))
model.add(Activation('linear'))
```

- Sau khi xây dựng được model, ta cần định nghĩa hàm loss và 1 optimizer để có thể bắt đầu quá trình training.
- Keras hỗ trợ rất nhiều built-in loss function.
- Một số hàm loss phổ biến
 - Mean squared error với bài toán regression,
 - Binary crossentropy cho bài toán classification,
 - Categorical crossentropy với bài toán multiclass-classification và sparse categorical crossentropy cũng cho bài toán multiclass-classification nhưng với label chưa được đưa về dạng one-hot.
- Đối với optimizers, tất cả các optimizer của Keras đều được xây dựng dựa trên thuật toán Gradient Descent .
 - SGD (Stochastic gradient descent optimizer)
 - Adagrad
 - Adadelta
 - Adam ...
- Trong số các optimizers mà Keras cung cấp thì Adam optimizer được sử dụng nhiều hơn cả do khả năng hội tụ nhanh của nó.

.5. Pre-trained CNN model in Keras

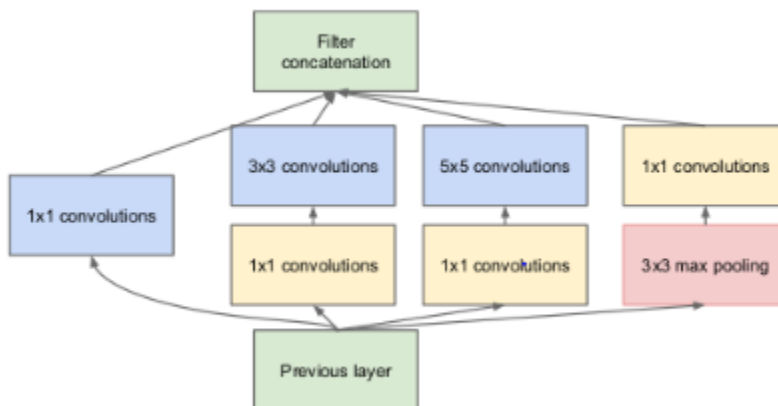
- Đây là các mô hình được xây dựng sẵn để train ảnh.
- VGG-16 : VGG ra đời năm 2015 và được giới thiệu tại hội thảo ICLR 2015. Kiến trúc của mô hình này có nhiều biến thể khác nhau: 11 layers, 13 layers, 16 layers, và 19 layers

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

- Nguyên tắc thiết kế của các mạng VGG nói chung rất đơn giản:

- 2 hoặc 3 layers Convolution (Conv) và tiếp nối sau đó là 1 layer Max Pooling 2D
- Ngay sau Conv cuối cùng là 1 Flatten layer để chuyển ma trận 4 chiều của Conv layer về ma trận 2 chiều
- Tiếp nối sau đó là các Fully-connected layers và 1 Softmax layer
- Trong Keras hiện tại có hỗ trợ 2 pre-trained model của VGG: VGG-16 và VGG 19

- InceptionNet



- Với các mạng CNN thông thường, khi thiết kế ta bắt buộc phải xác định trước các tham số của 1 Conv layer như: **kernel_size, padding, strides, etc.** Và thường thì rất khó để ta có thể nói trước được tham số nào là phù hợp
- Mạng Inception ra đời để giải quyết vấn đề đó
- yếu tố quan trọng nhất trong mạng Inception là **Inception module**, một mạng **Inception** hoàn chỉnh bao gồm nhiều module Inception nhỏ ghép lại với nhau.

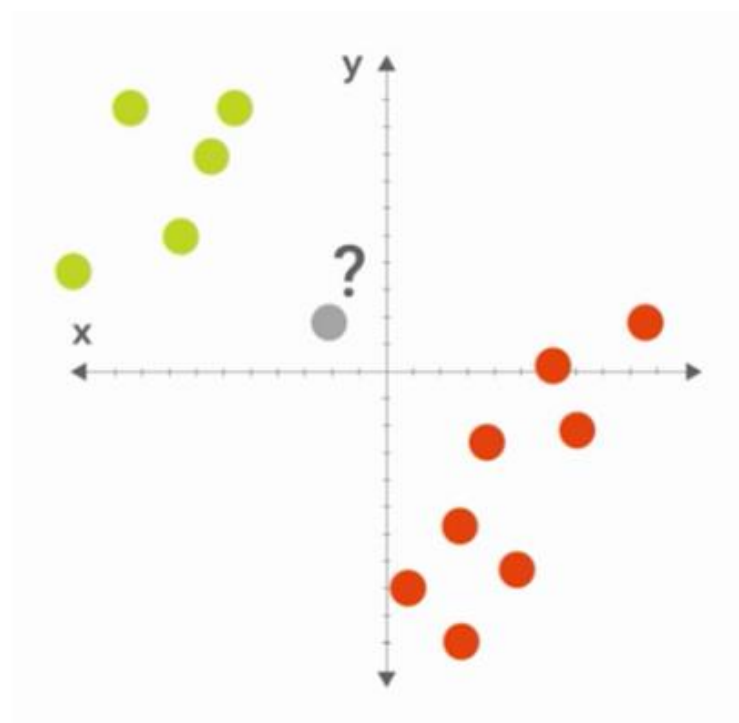
- ResNet

- Residual block có thể train các mô hình CNN có kích thước và độ phức tạp "khủng" hơn mà không lo bị exploding/vanishing gradient (lượng layer/parameter lớn)
- Mấu chốt của Residual block là cứ sau 2 layer, ta cộng input với output: $F(x) + x$
- Resnet là một mạng CNN bao gồm nhiều Residual block nhỏ tạo thành

PHẦN .2. PHÂN TÍCH VÀ THỰC HIỆN

.1. Tổng quan về bài toán phân lớp ảnh

- Bài toán phân lớp là quá trình phân lớp một đối tượng dữ liệu vào một hay nhiều lớp đã cho trước nhờ một mô hình phân lớp (model). Mô hình này được xây dựng dựa trên một tập dữ liệu được xây dựng trước đó có gán nhãn (hay còn gọi là tập huấn luyện). Quá trình phân lớp là **quá trình gán nhãn** cho đối tượng dữ liệu.



Liệu bi mới được thêm vào sẽ thuộc vào lớp bi xanh hay bi đỏ?

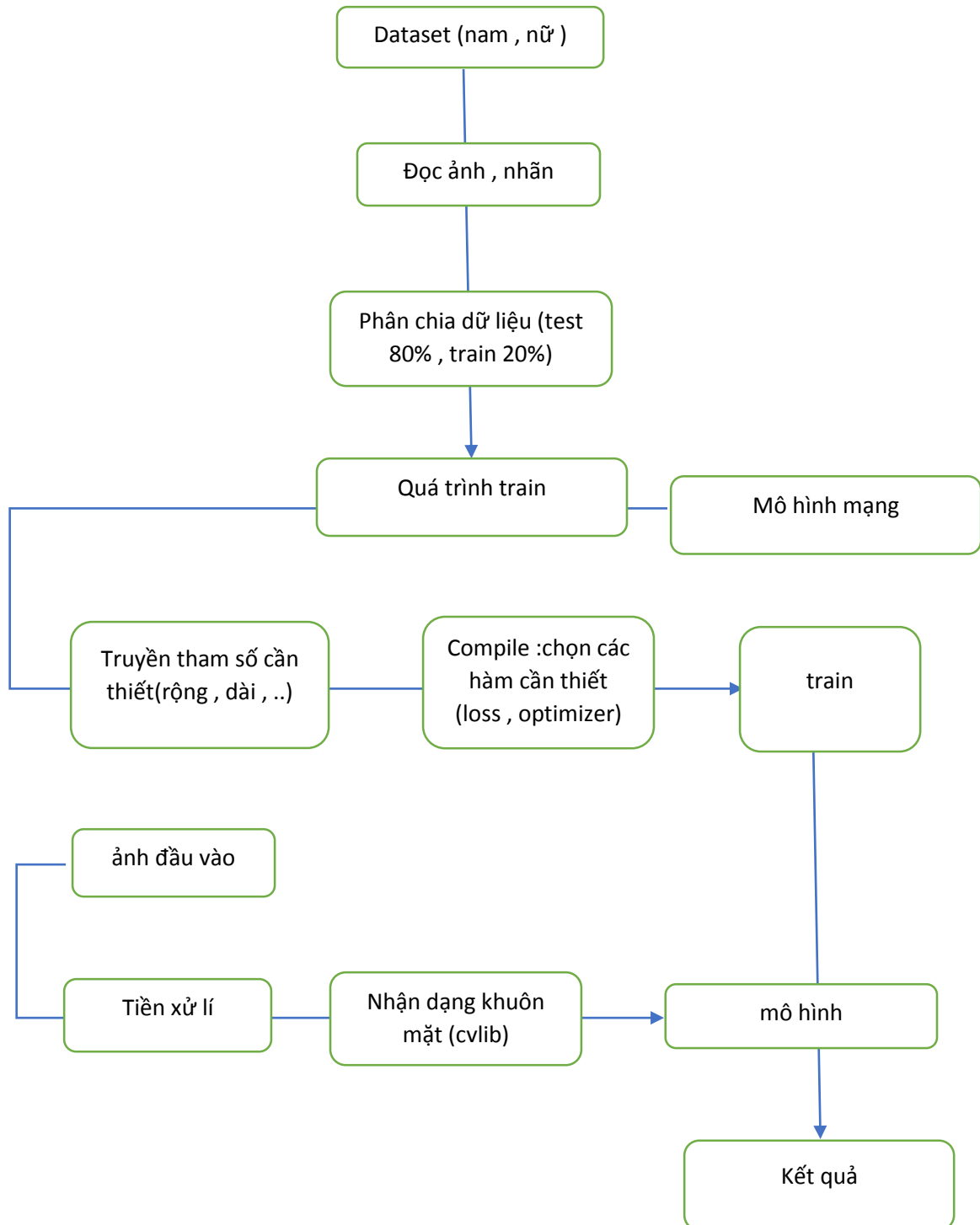
- Như vậy, nhiệm vụ của bài toán phân lớp là cần tìm một **mô hình phân lớp** để khi có dữ liệu mới thì có thể xác định được dữ liệu đó thuộc vào phân lớp nào.
- Có nhiều bài toán phân lớp dữ liệu như phân lớp nhị phân (binary), phân lớp đa lớp (multiclass), phân lớp đa trị.

- Trong phạm vi của đề án vì chỉ cần xác định 2 lớp đối tượng nên bài toán sẽ là phân lớp nhị phân

.2. Xây dựng hệ thống

- Ý tưởng xây dựng:
 - Input: ảnh khuôn mặt người.
 - Output: cho biết giới tính của người trong ảnh (Nam/Nữ)
 - Có 2 quá trình cần thực hiện :
 - Nhận diện khuôn mặt qua ảnh ban đầu
 - Phân lớp khuôn mặt đã nhận dạng được là nam hay nữ.
- Hoạt động :
 - Đây là mô hình phân lớp được xây dựng bằng Deep Learning , sử dụng keras và dùng Tensorflow là back end.
 - Tương tự như những thuật toán phân lớp khác , keras cần một bộ dữ liệu lớn để học và phân lớp , sau quá trình phân lớp keras sẽ tạo ra một model và sau đó dùng nó để xác định ảnh đầu vào.
 - Với bài toán là nhận dạng giới tính , ứng dụng thực hiện xác định khuôn mặt người trước rồi sau đó mới thực hiện quá trình phân lớp cho khuôn mặt dựa trên dữ liệu đã học.

.2.1. Tiến hành xây dựng cơ sở dữ liệu ảnh khuôn mặt



- Xây dựng mô hình :
 - Như đã trình bày ở trên , có 2 cách xây dựng 1 mô hình mạng là dùng API và model .
 - Trong phạm vi tìm hiểu này sẽ xây dựng 1 model để phù hợp , model được dựa trên các model VGG .
 - Để tạo ra 1 convNet chúng ta cần hiểu nó là một chuỗi các lớp và mỗi lớp của ConvNet sẽ biến đổi một khối lượng kích hoạt này sang một khối khác thông qua một chức năng khác biệt. thường sử dụng ba loại chính của lớp để xây dựng kiến trúc ConvNet: **Lớp Convolutional , Pooling , và Fully-Connected.**
 - Mạng được xây dựng dựa trên 1 ConvNet đơn giản cho CIFAR-10 với kiến trúc sau [INPUT - CONV - RELU - POOL - FC].
 - Model nhận đầu vào là các ảnh (48x48x3), đầu ra là lớp fc softmax với nodes = 2. Ở giữa là 1 kiến trúc CNN đơn giản với thứ tự các lớp như sau: CONV (relu) => POOL => (CONV (relu))*2 => POOL)=> FC(relu) => FC(softmax).

```

model = Sequential()
chanDim = -1
model.add(Conv2D(32, (3,3), padding="same", input_shape=(48,48,3)))
model.add(Activation("relu"))

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

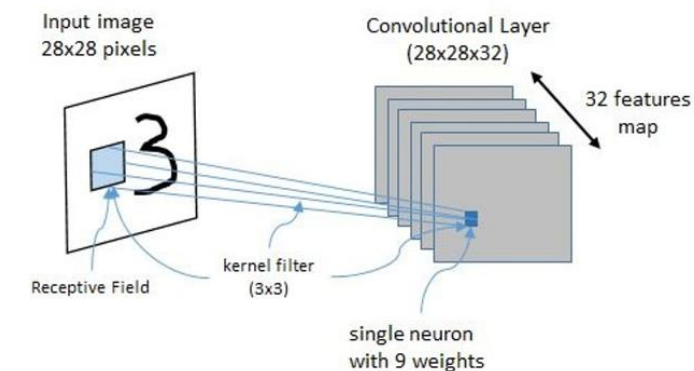
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(2))
model.add(Activation("sigmoid"))
model.summary()

```

- Đầu tiên thêm các lớp vào mô hình .
 - CONV => RELU => POOL
 - Dùng Conv2D để xử lí : là lớp cốt lõi của mạng thực hiện các công việc tính toán . dùng filter là 32 với kích thước là 3x3 . bộ lọc sẽ duyệt qua toàn bộ ảnh và sẽ tạo ra các features map .

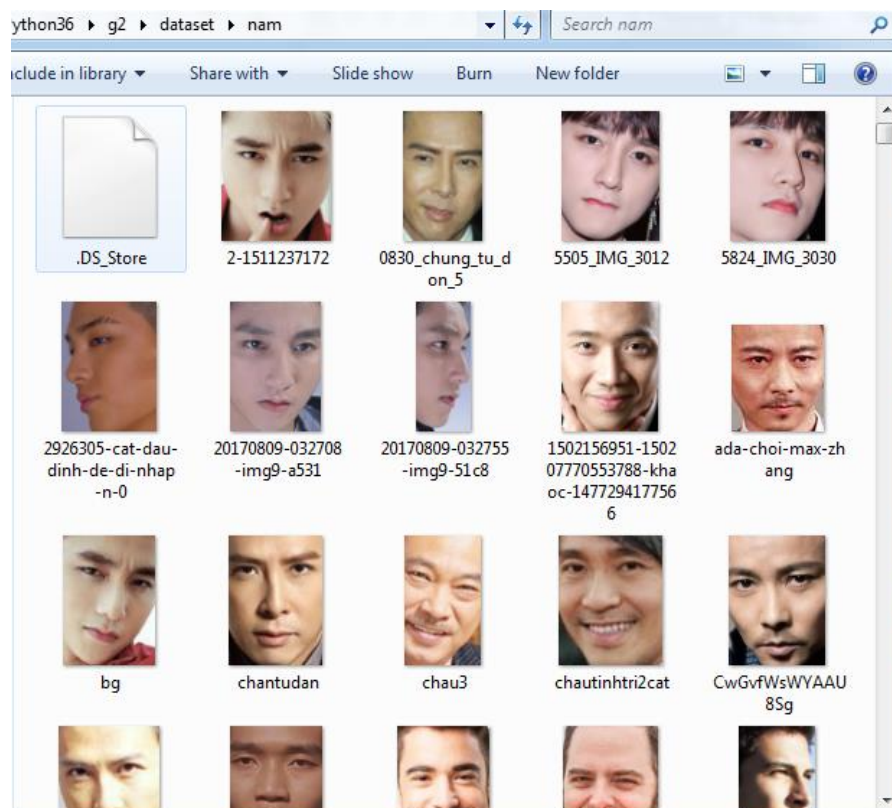


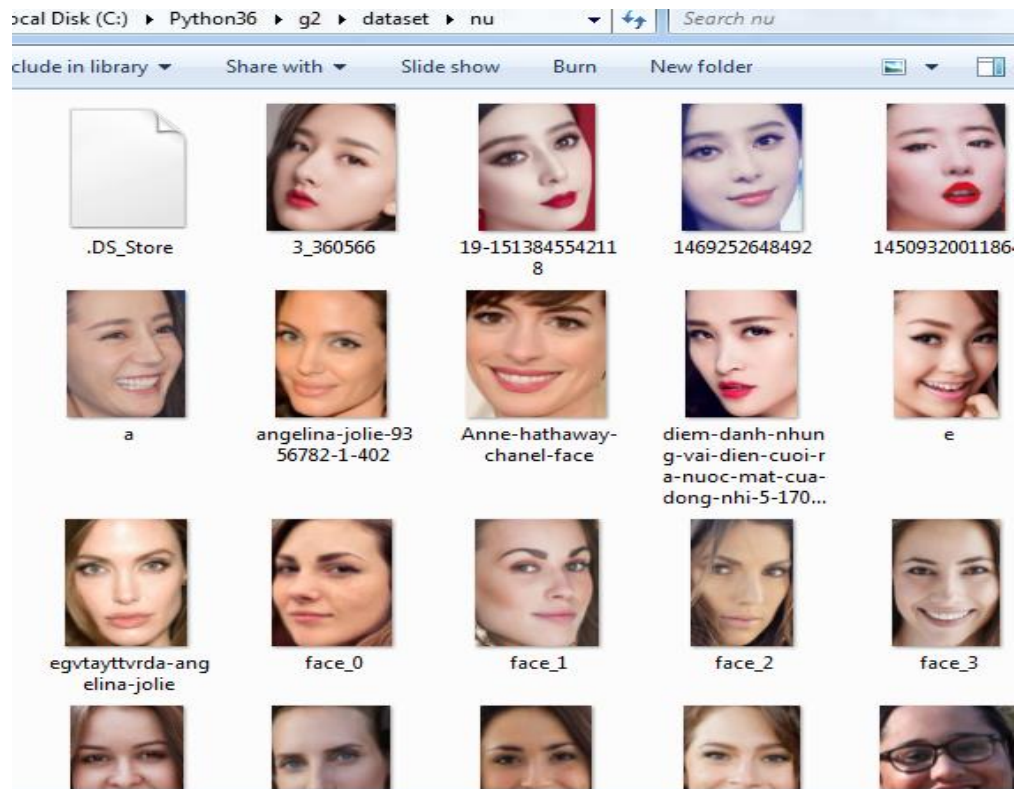
- Tiếp theo là dùng lớp Pooling :lớp này chỉ đơn giản là giảm kích thước ma trận input theo một cửa sổ kích thước (3x 3). Vì vậy lớp này ko có params (weights) nào cả. Và từ input = (48 x 48 x 32), output của lớp này sẽ là (16 x 16 x 32).
- Dùng dropout tại đây để bỏ học trong kiến trúc mạng , hoạt động bằng cách ngắt ngẫu nhiên nút từ lớp này sang lớp tiếp theo. => mục đích tránh overfit , không một nút nào trong lớp chịu trách nhiệm dự đoán một lớp, đối tượng, cạnh hoặc góc nhất định.
- Tiếp theo là chồng các lớp Conv2D (Relu) với nhau
 - Tăng kích thước bộ lọc lên cùng với kích thước của volume càng nhỏ giúp học được nhiều bộ lọc
- Tiếp theo giảm max pooling xuống 2x2 để kích thước không bị giảm quá nhanh . Tiếp theo là dùng dropout.
-
- Sau quá trình chồng các lớp output cuối sẽ là (8x8x64)
- Lớp tiếp theo là flatten và cuối cùng là lớp FC (**Fully-Connected**) .

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 48, 48, 32)	896
activation_1 (Activation)	(None, 48, 48, 32)	0
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 32)	0
dropout_1 (Dropout)	(None, 16, 16, 32)	0
conv2d_2 (Conv2D)	(None, 16, 16, 64)	18496
activation_2 (Activation)	(None, 16, 16, 64)	0
conv2d_3 (Conv2D)	(None, 16, 16, 64)	36928
activation_3 (Activation)	(None, 16, 16, 64)	0
max_pooling2d_2 (MaxPooling2D)	(None, 8, 8, 64)	0
dropout_2 (Dropout)	(None, 8, 8, 64)	0
flatten_1 (Flatten)	(None, 4096)	0
dense_1 (Dense)	(None, 512)	2097664
activation_4 (Activation)	(None, 512)	0
batch_normalization_1 (Batch Normalization)	(None, 512)	2048
dropout_3 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 2)	1026
activation_5 (Activation)	(None, 2)	0
Total params: 2,157,058		
Trainable params: 2,156,034		
Non-trainable params: 1,024		

- Bảng trên cho thấy các tham số được tạo ra , nó là khá lớn khoảng 2 triệu , với số lượng tham số này được xem là ít , để có độ chính xác cao chúng ta nên xây dựng nhiều lớp hơn .
Hoặc cũng có thể tăng số epochs lên để cải thiện Accuracy.

- Xây dựng cơ sở dữ liệu:
 - Quá trình tìm kiếm ảnh được thực hiện trên web , ảnh sau khi tải về sẽ được cắt phần khuôn mặt người để làm tập dataset.
 - Dữ liệu ảnh cũng được lấy từ những project về khuôn mặt người trên github.
 - Dữ liệu này đòi hỏi cần phải có sự bao quát cả về địa lí , vị trí khuôn mặt , xoay, che khuất ...
 - Các dữ liệu sau đó được chia vào 2 lớp được gán nhãn là nam và nữ để train , thuật toán sẽ xác định nhãn và dự liệu để phân lớp tốt nhất.





- Vì quá trình tìm kiếm ảnh tốn quá nhiều thời gian nên số lượng ảnh là khá ít so với đòi hỏi của kỹ thuật Deep Learning là dữ liệu càng nhiều càng tốt nên đôi khi không tránh được sai sót
- Huấn luyện ảnh :
 - ảnh trong dataset sẽ được train bằng mô hình của keras , mô hình này được xây dựng bằng tay , vì số lượng ảnh khá ít nên sử dụng những mô hình có sẵn có thể sẽ không tốt nên mạng cần có độ sâu phù hợp .
 - Dùng lệnh trong cmd để tiến hành train

```
C:\Python36\g2>python train.py -d dataset
```


- ở đây tệp train.py là tệp train ảnh , quá trình này sẽ dùng mô hình đã được khởi tạo và huấn luyện ảnh .
- dataset là nơi chứa dữ liệu ảnh.
- Quá trình train diễn ra :
 - Dữ liệu ảnh được đọc và ảnh được lấy , sau đó ảnh sẽ được resize thành (48x48) để phù hợp với mô hình train, sau đó ảnh sẽ được lưu vào
 - Trong dataset có 2 nhãn là nam và nữ , trong keras nhĩ sẽ được tách ra và phân chia thành số , ở đây máy sẽ gán nam=0 và nữ =1 .
 - Tiếp theo là quá trình chia dữ liệu để train , dữ liệu và nhãn được chia làm 2 phần là tập test và train , với tỉ lệ thông dụng là 80 và 20 .
 - Quá trình training cần rất nhiều dữ liệu vì vậy trong quá trình tiền xử lí ảnh chúng ta dùng hàm ImageDataGenerator trong thư viện keras để tạo ra thêm nhiều ảnh bằng cách scale , rotate...
 - Sau khi đã có số lượng ảnh đủ lớn ta gọi mô hình để train
 - Quá trình build : gọi đến mô hình và truyền các tham số cần thiết , các tham số này là
 - inputShape gồm có : chiều dài , rộng , sâu của ảnh.
 - Classes :số lớp cần chia.
 - Quá trình compile :
 - Optimizer : dùng chuẩn hóa Adam để chuẩn hóa mô hình.

```
keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
```

- Chúng ta cần quan tâm đến 2 tham số lr và decay
 - Lr : là tốc độ học của mạng , cần chọn tốc độ học phù hợp với mô hình vì nếu tốc độ quá chậm thì mô hình sẽ lâu hội tụ , còn quá nhanh thì sẽ khó hội tụ.
 - Decay : là sự phân rã , sau mỗi lần duyệt qua dữ liệu độ phân rã giảm xuống.
- Compile : cấu hình cho mô hình
 - Dùng optimizer là : Adam.
 - Hàm loss là : binary_crossentropy
 - Metrics : accuracy dùng để đánh giá mô hình trong khi đào tạo và test.
- Quá trình train:
 - Dùng chức năng fit_generator để train ảnh .
 - fit_generator(generator, steps_per_epoch=**None**, epochs=1, verbose=1, callbacks=**None**, validation_data=**None**, validation_steps=**None**, class_weight=**None**, max_queue_size=10, workers=1, use_multiprocessing=**False**, shuffle=**True**, initial_epoch=0)
 - Chức năng này cần truyền vào các dữ liệu đã tạo ở trước , quá trình train sẽ xảy ra bao lâu tùy thuộc vào số lần lặp qua tập dữ liệu ở đây là epochs .
 - Quá trình này sẽ trả về 1 đối tượng history , là 1 bản ghi các training loss và giá trị cho các epochs tiếp theo.

```
C:\Windows\system32\cmd.exe
Epoch 4/10
27/27 [=====] - 47s 2s/step - loss: 0.4508 - acc: 0.814
9 - val_loss: 0.3378 - val_acc: 0.8750
Epoch 5/10
27/27 [=====] - 48s 2s/step - loss: 0.3998 - acc: 0.838
4 - val_loss: 0.3322 - val_acc: 0.8557
Epoch 6/10
27/27 [=====] - 47s 2s/step - loss: 0.4206 - acc: 0.820
8 - val_loss: 0.2665 - val_acc: 0.8864
Epoch 7/10
27/27 [=====] - 49s 2s/step - loss: 0.3604 - acc: 0.848
8 - val_loss: 0.3324 - val_acc: 0.8761
Epoch 8/10
27/27 [=====] - 48s 2s/step - loss: 0.3182 - acc: 0.863
4 - val_loss: 0.4031 - val_acc: 0.8511
Epoch 9/10
27/27 [=====] - 49s 2s/step - loss: 0.3353 - acc: 0.861
9 - val_loss: 0.3245 - val_acc: 0.8773
Epoch 10/10
27/27 [=====] - 49s 2s/step - loss: 0.3269 - acc: 0.871
8 - val_loss: 0.2200 - val_acc: 0.9102
440/440 [=====] - 2s 5ms/step
acc: 91.02%
```

- Kết thúc chúng ta đã có được mô hình phân lớp tập dataset.

- Nhận dạng khuôn mặt dùng cvlib :
 - Một thư viện Computer Vision mã nguồn mở để sử dụng cho Python.
 - Nó được phát triển với trọng tâm là cho phép thử nghiệm dễ dàng và nhanh chóng.
 - Nguyên tắc hướng dẫn của cvlib được lấy cảm hứng rất nhiều từ Keras (thư viện học tập sâu).
 - sự đơn giản
 - Người dùng thân thiện
 - mô-đun
 - mở rộng
 - trong thư viện này còn có cả chức năng nhận dạng giới tính nhưng chúng ta không bàn đến nó.
 - Cvlib sử dụng mô-đun của OpenCV với một caffe model được đào tạo trước để phát hiện khuôn mặt.

PHẦN .3.KẾT QUẢ , ĐÁNH GIÁ , SO SÁNH CÁC PHƯƠNG PHÁP KHÁC

.1. Kết quả

- Cách sử dụng : vào cmd của tập và gõ lệnh

```
C:\Python36\g2>python anh.py -i anhtest/1.jpg
```

- Còn nếu dùng webcam gõ

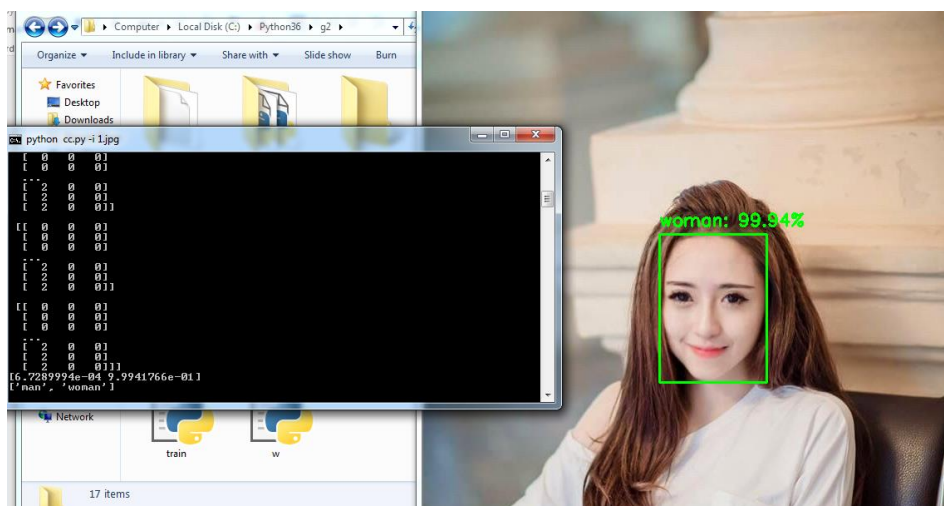
```
C:\Python36\g2>python video.py
```

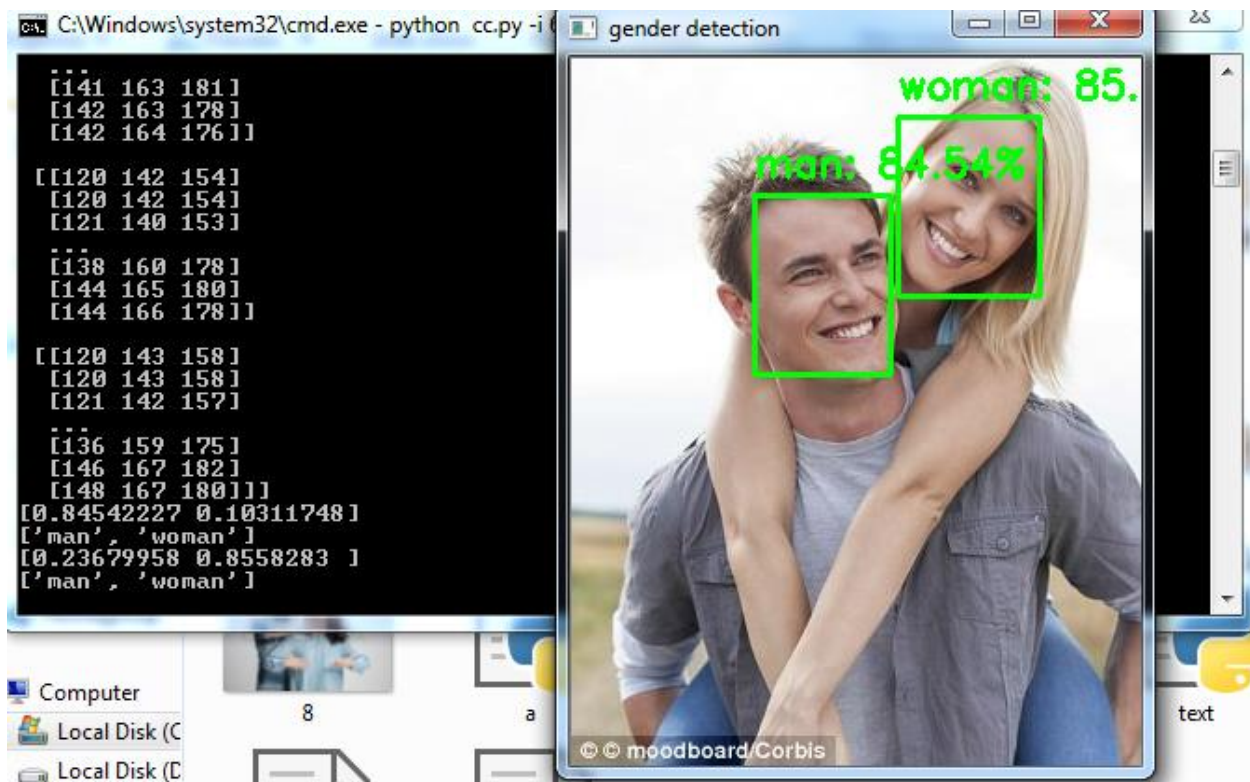
- Sau khi đã phân lớp xong chúng ta tiến hành xem kết quả.
- Input





- Output





- Thấy rằng độ chính xác khá cao.

- Đánh giá :



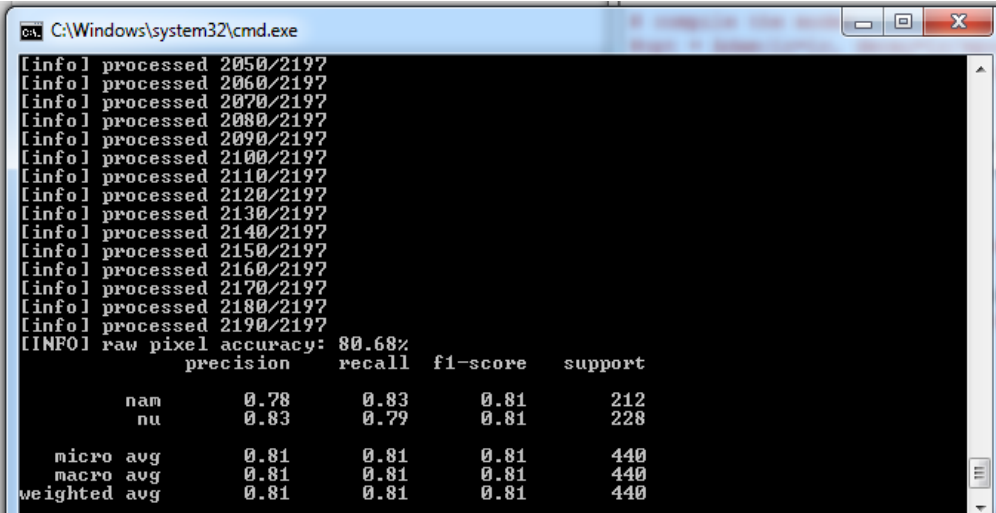
- Chúng ta nhận thấy rằng , trải qua mỗi epoch các giá trị loss đều có xu hướng giảm dần
- Để cải thiện loss chúng ta có thể có nhiều phương pháp
 - Tăng số lần duyệt qua bộ dữ liệu Epochs.
 - Tăng số lượng dữ liệu ban đầu , càng nhiều càng tốt.
 - Tăng độ sâu của mạng ban đầu.
 - Hàm loss có thể giảm sâu nhưng khó có thể xảy ra overfit bởi trong quá trình tạo mạng đã sử dụng phương pháp tránh overfit là dropout.

- So sánh với các bộ phân lớp khác.
 - K-NN :vì bài toán có 2 lớp nên ta có thể dùng K-NN
 - Cách dùng :

```
C:\Users\Tran Viet Huy\Desktop\thigiac\knn>python knn.py
```

- Trong phần so sánh này K-NN dùng với k=5 và phép đo là L2-norm.
- Trong quá trình xây dựng thử nghiệm đã chọn được các tham số trên.
- Dùng model của K-NN để train :

```
model = KNeighborsClassifier(n_neighbors=5,
                             n_jobs=2)
model.fit(trainX, trainY)
acc = model.score(testX, testY)
```



```
C:\Windows\system32\cmd.exe
[info] processed 2050/2197
[info] processed 2060/2197
[info] processed 2070/2197
[info] processed 2080/2197
[info] processed 2090/2197
[info] processed 2100/2197
[info] processed 2110/2197
[info] processed 2120/2197
[info] processed 2130/2197
[info] processed 2140/2197
[info] processed 2150/2197
[info] processed 2160/2197
[info] processed 2170/2197
[info] processed 2180/2197
[info] processed 2190/2197
[INFO] raw pixel accuracy: 80.68%
      precision    recall  f1-score   support

      nam         0.78         0.83         0.81         212
      nu          0.83         0.79         0.81         228

   micro avg         0.81         0.81         0.81         440
   macro avg         0.81         0.81         0.81         440
weighted avg         0.81         0.81         0.81         440
```

- Quá trình train ảnh diễn ra trên 2197 ảnh mẫu
- Tỷ lệ accuracy là 80%
- ở dưới là các thông số phân lớp trên tập test (cùng bộ dữ liệu với dùng keras.)
- Chúng ta thấy dù cố gắng tìm các tham số phù hợp nhất thì tỷ lệ của kỹ thuật vẫn không cao bằng so với keras , ở dưới là tỷ lệ accuracy của keras khá cao khoảng 91% , cho dù mạng khởi tạo vẫn còn đơn giản.

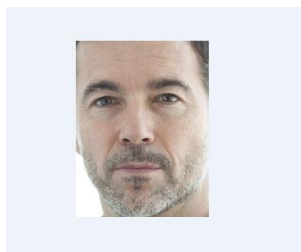
```

C:\Windows\system32\cmd.exe
Epoch 4/10
27/27 [=====] - 47s 2s/step - loss: 0.4508 - acc: 0.814
9 - val_loss: 0.3378 - val_acc: 0.8750
Epoch 5/10
27/27 [=====] - 48s 2s/step - loss: 0.3998 - acc: 0.838
4 - val_loss: 0.3322 - val_acc: 0.8557
Epoch 6/10
27/27 [=====] - 47s 2s/step - loss: 0.4206 - acc: 0.820
8 - val_loss: 0.2665 - val_acc: 0.8864
Epoch 7/10
27/27 [=====] - 49s 2s/step - loss: 0.3604 - acc: 0.848
8 - val_loss: 0.3324 - val_acc: 0.8761
Epoch 8/10
27/27 [=====] - 48s 2s/step - loss: 0.3182 - acc: 0.863
4 - val_loss: 0.4031 - val_acc: 0.8511
Epoch 9/10
27/27 [=====] - 49s 2s/step - loss: 0.3353 - acc: 0.861
9 - val_loss: 0.3245 - val_acc: 0.8773
Epoch 10/10
27/27 [=====] - 49s 2s/step - loss: 0.3269 - acc: 0.871
8 - val_loss: 0.2200 - val_acc: 0.9102
440/440 [=====] - 2s 5ms/step
acc: 91.02%

```

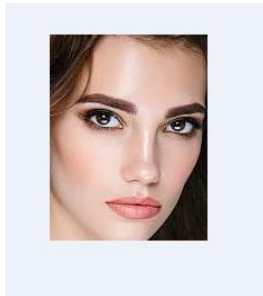
○ Test với K-NN

▪

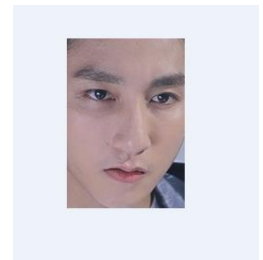


	precision	recall	f1-score	support
nam	0.78	0.83	0.81	212
nu	0.83	0.79	0.81	228
micro avg	0.81	0.81	0.81	440
macro avg	0.81	0.81	0.81	440
weighted avg	0.81	0.81	0.81	440

['nam']



	precision	recall	f1-score	support
nam	0.78	0.83	0.81	212
nu	0.83	0.79	0.81	228
micro avg	0.81	0.81	0.81	440
macro avg	0.81	0.81	0.81	440
weighted avg	0.81	0.81	0.81	440



	precision	recall	f1-score	support
nam	0.78	0.83	0.81	212
nu	0.83	0.79	0.81	228
micro avg	0.81	0.81	0.81	440
macro avg	0.81	0.81	0.81	440
weighted avg	0.81	0.81	0.81	440

○ SVM:

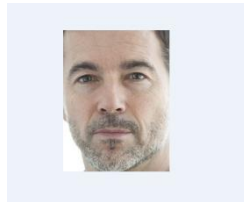
- Dùng model trong SVC để train
- Cách dùng

```
C:\Users\Tran Viet Huy\Desktop\thigiact\svm_1>python svm.py
```

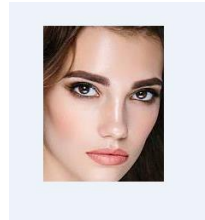
```
model = LinearSVC()
|
model.fit(trainX, trainY)
acc = model.score(testX, testY)
```

```
C:\Python36\svmm>python train.py -d dataset
[info] processed 600/2198
[info] processed 1200/2198
[info] processed 1800/2198
C:\Python36\lib\site-packages\sklearn\svm\base.py:931: ConvergenceWarning:
Near failed to converge, increase the number of iterations.
(the number of iterations.", ConvergenceWarning)
[INFO] raw pixel accuracy: 87.50%
precision    recall    f1-score   support
     nam      0.88      0.86      0.87      216
     nu      0.87      0.89      0.88      224
  micro avg      0.88      0.88      0.88      440
  macro avg      0.88      0.87      0.87      440
weighted avg      0.88      0.88      0.87      440
```

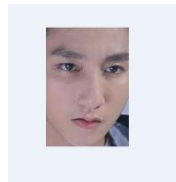
▪ Test :



	precision	recall	f1-score	support
nam	0.88	0.86	0.87	216
nu	0.87	0.89	0.88	224
micro avg	0.88	0.88	0.88	440
macro avg	0.88	0.87	0.87	440
weighted avg	0.88	0.88	0.87	440



	precision	recall	f1-score	support
nam	0.88	0.87	0.88	216
nu	0.88	0.89	0.88	224
micro avg	0.88	0.88	0.88	440
macro avg	0.88	0.88	0.88	440
weighted avg	0.88	0.88	0.88	440



	precision	recall	f1-score	support
nam	0.88	0.87	0.87	216
nu	0.87	0.89	0.88	224
micro avg	0.88	0.88	0.88	440
macro avg	0.88	0.88	0.88	440
weighted avg	0.88	0.88	0.88	440

.2. Tài liệu tham khảo

- Các thành phần của keras : <https://keras.io>
- Tìm hiểu DeepLearning : <https://viblo.asia/p/deep-learning-qua-kho-dung-lo-da-co-keras-LzD5dBqoZjY>
- Keras cơ bản : <https://machinelearningcoban.com/2018/07/06/deeplearning/>

