**BỘ GIÁO DỤC & ĐÀO TẠO**
**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**
**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

----------Δ ----------

# BÁO CÁO MÔN TRÍ TUỆ - NHÂN TẠO

# ĐỀ TÀI: NHẬN DIỆN NGỦ GẬT

**SVTH: Đỗ Công Danh**
**MSSV: 19146163**

**TP HỒ CHÍ MINH NGÀY 20 THÁNG 06 NĂM 2022**

# MỤC LỤC

# 1. Tổng quan:

## 1.1. Giới thiệu về AI – Artifical Intelligent

Trí tuệ nhân tạo hay trí thông minh nhân tạo (Artificial intelligence – viết tắt là AI) là một ngành thuộc lĩnh vực khoa học máy tính (Computer science). Là trí tuệ do con người lập trình tạo nên với mục tiêu giúp máy tính có thể tự động hóa các hành vi thông minh như con người.

Trí tuệ nhân tạo khác với việc lập trình logic trong các ngôn ngữ lập trình là ở việc ứng dụng các hệ thống học máy (machine learning) để mô phỏng trí tuệ của con người trong các xử lý mà con người làm tốt hơn máy tính.

Cụ thể, trí tuệ nhân tạo giúp máy tính có được những trí tuệ của con người như: biết suy nghĩ và lập luận để giải quyết vấn đề, biết giao tiếp do hiểu ngôn ngữ, tiếng nói, biết học và tự thích nghi,…

Tuy rằng trí thông minh nhân tạo có nghĩa rộng như là trí thông minh trong các tác phẩm khoa học viễn tưởng, nó là một trong những ngành trọng yếu của tin học. Trí thông minh nhân tạo liên quan đến cách cư xử, sự học hỏi và khả năng thích ứng thông minh của máy móc.

## 1.2. Tổng quan đề tài:

Những năm gần đây, ở nước ta, cùng với quá trình phát triển nhanh của các phương tiện giao thông, con số tai nạn giao thông ngày càng tăng nhiều, đặt ra một mối nguy hiểm nghiêm trọng cho cuộc sống xã hội và người tham gia giao thông. Tai nạn giao thông (TNGT) đã và đang trở thành nỗi đau lớn của nhiều gia đình, trong những vụ tai nạn giao thông, người thì mang tật suốt đời, người tử vong để lại những khoảng trống không gì bù đắp nổi cho người thân. Và một trong những nguyên nhân chính của tai nạn giao thông là sự thiếu tập trung của người lái xe do mệt mỏi hay buồn ngủ.

Theo báo cáo của Tổng cục Thống kê cho biết, tính chung 7 tháng đầu năm 2015 trên địa bàn cả nước đã xảy ra 12.910 vụ tai nạn giao thông và bình quân mỗi ngày 61 vụ. Và theo phân tích của Cục Cảnh Sát Giao Thông, gần 70% số vụ TNGT xảy ra vào khoảng thời gian từ 12h đến 24h, đây là khoảng thời gian người điều khiển phương tiện bị tác động tâm lý của sự mệt mỏi, căng thẳng, sự chênh lệch về nhiệt độ, ánh sáng giữa ngày và đêm (đặc biệt đối với phương tiện vận tải hành khách, hàng hóa…)

Báo cáo về "Rối loạn giấc ngủ và tai nạn giao thông" tại hội nghị khoa học thường niên Hội Hô hấp Việt Nam và Chương trình đào tạo y khoa liên tục 2015,

giáo sư Telfilo Lee Chiong (Trung tâm Nationnal Jewish Health, Mỹ), cho biết buồn ngủ là một trong những nguyên nhân chính gây tai nạn giao thông trên thế giới. Ước tính khoảng 10-15% tai nạn xe có liên quan đến thiếu ngủ. Nghiên cứu về giấc ngủ ở các tài xế 19 quốc gia châu Âu cho thấy tỷ lệ buồn ngủ khi lái xe cao, trung bình 17%. Trong đó 10,8% người buồn ngủ khi lái xe ít nhất một lần trong tháng, 7% từng gây tai nạn giao thông do buồn ngủ, 18% suýt xảy ra tai nạn do buồn ngủ.

Những số liệu thống kê đáng báo động chỉ ra sự cần thiết để thực hiện các hệ thống có khả năng theo dõi và cảnh báo tình trạng mệt mỏi, buồn ngủ của người lái xe để có thể ngăn chặn những vụ TNGT đáng tiếc có thể xảy ra.

## 2. Phương pháp thực hiện

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.

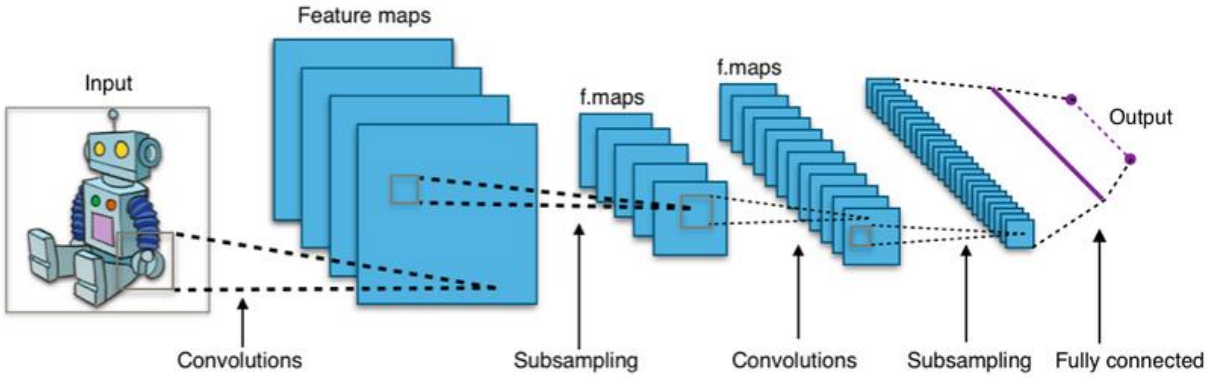
Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo. Trong mô hình mạng truyền ngược (feedforward neural network) thì mỗi neural đầu vào (input node) cho mỗi neural đầu ra trong các lớp tiếp theo.

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.




***Hình 1:*** *Cấu trúc mạng CNN*

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.

2.1. Đối tượng và phạm vi nghiên cứu:

Đối tượng nghiên cứu:

Ngôn ngữ lập trình Python.

Mạng CNN.

Các thuật toán cử chỉ gương mặt.

Phạm vi nghiên cứu:

Nghiên cứu về các phương pháp đã được đề xuất phát hiện và cảnh báo tình trạng buồn ngủ của người lái xe trên thế giới theo những bài báo và nghiên cứu khoa học.

Chương trình demo sử dụng ngôn ngữ lập Python, sử dụng mạng CNN trên nền tảng hệ điều hành Windows.

## 3. Nội dung thực hiện

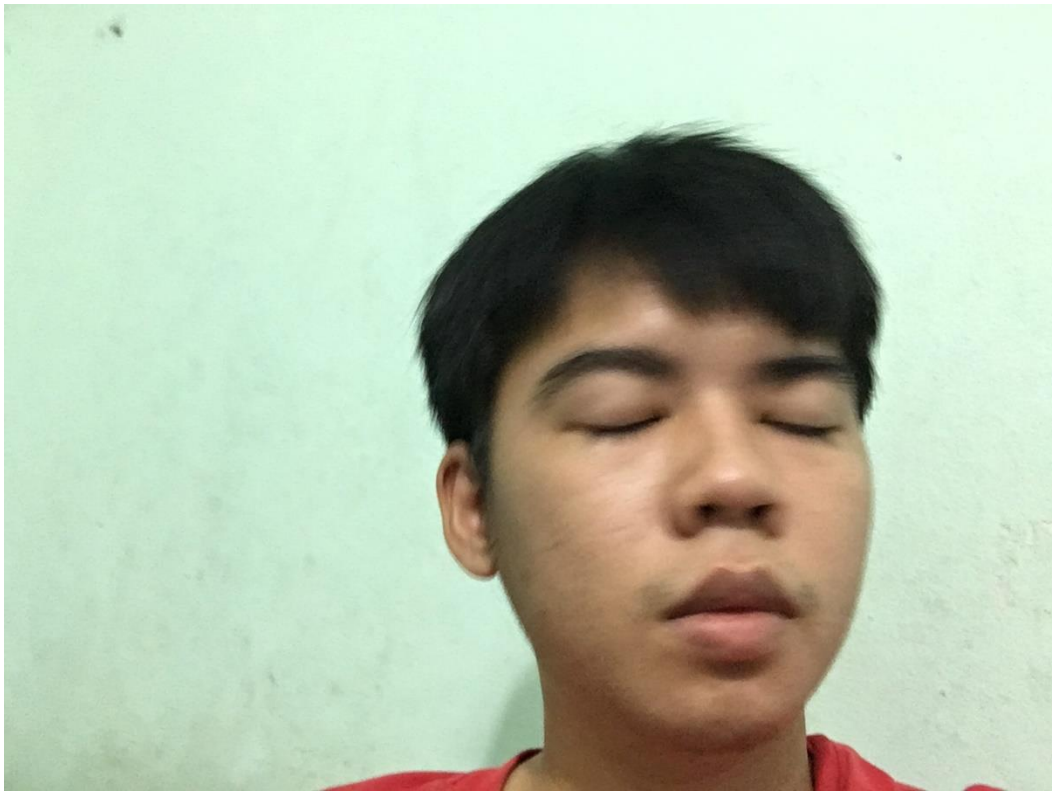
### 3.1. Thu thập dữ liệu

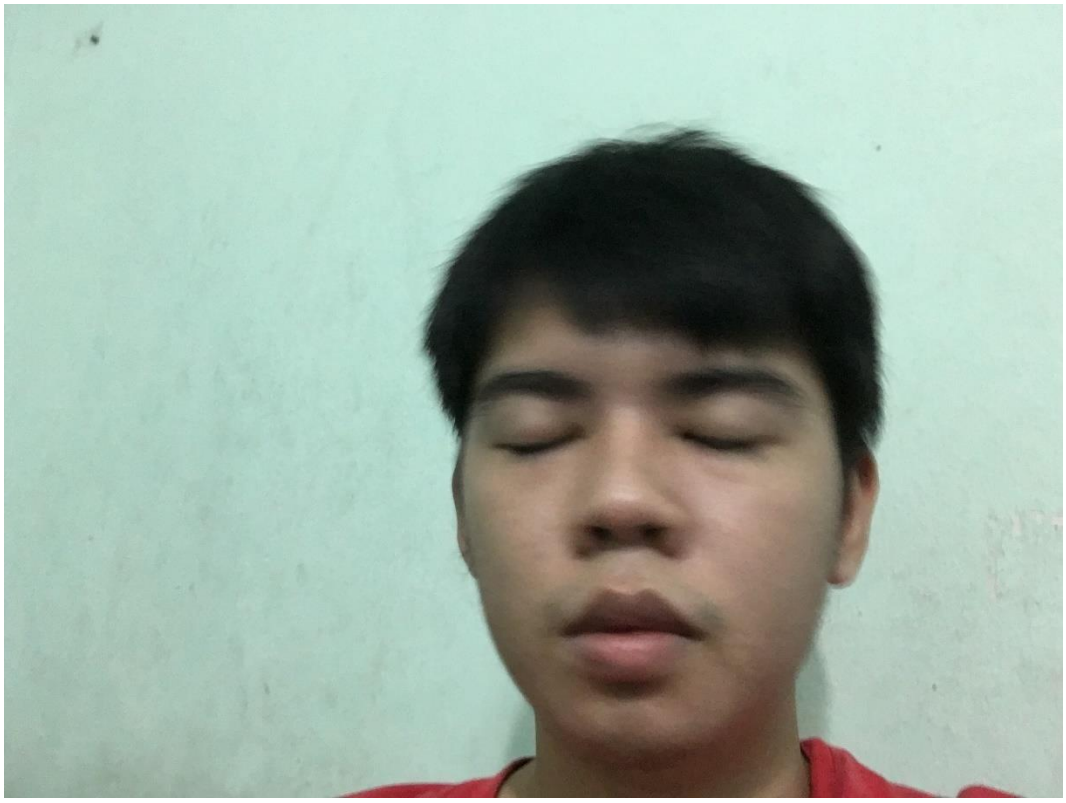
Với yêu cầu đề tài thì em đã lựa chọn 2 loại dữ liệu là nhắm và mở mắt và tiến hành chụp lại bằng điện thoại, xây dựng dataset gồm 2 classes mỗi classe gồm 150 tấm ảnh, tổng cộng 300 tấm.


***Hình 2:*** *Dataset về mắt nhắm, mắt mở và tập dữ liệu test*

Dưới đây là 1 số ảnh từ dataset từng loại gạo:

- Nhắm mắt:





-Mở mắt:

Với mỗi classes gồm 150 ảnh, chia thành 120 tấm train và 30 tấm test. Tổng cộng 300 ảnh train và 60 ảnh test.

## 3.2. Tạo model training

- Tải thư viện:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import Sequential,preprocessing

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,
Dropout
from keras.callbacks import EarlyStopping, ModelCheckpoint
```

- Kết nối với drive:

```
from google.colab import drive
drive.mount("/content/gdrive", force_remount = True)
```

- Tạo model training:

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D,Dense
model=Sequential()

model.add(Conv2D(32,(3,3),activation='relu',kernel_initializer='he_uniform',p
adding='same' ,input_shape=(150,150,3)))
model.add(Conv2D(32,(3,3),activation='relu',kernel_initializer='he_uniform',p
adding='same'))
model.add(MaxPooling2D((2,2)))

model.add(Conv2D(64,(3,3),activation='relu',kernel_initializer='he_uniform',p
adding='same'))
model.add(Conv2D(64,(3,3),activation='relu',kernel_initializer='he_uniform',p
adding='same'))
model.add(MaxPooling2D((2,2)))

model.add(Conv2D(128,(3,3),activation='relu',kernel_initializer='he_uniform',
padding='same'))
model.add(Conv2D(128,(3,3),activation='relu',kernel_initializer='he_uniform',
padding='same'))
model.add(MaxPooling2D((2,2)))

model.add(Flatten())
model.add(Dense(128,activation='relu',kernel_initializer='he_uniform'))
model.add(Dense(2,activation='softmax'))
model.summary()
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 150, 150, 32)      896
```

```
 conv2d_1 (Conv2D)           (None, 150, 150, 32)      9248

 max_pooling2d (MaxPooling2D  (None, 75, 75, 32)       0
 )

 conv2d_2 (Conv2D)           (None, 75, 75, 64)        18496

 conv2d_3 (Conv2D)           (None, 75, 75, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 37, 37, 64)       0
 2D)

 conv2d_4 (Conv2D)           (None, 37, 37, 128)       73856

 conv2d_5 (Conv2D)           (None, 37, 37, 128)       147584

 max_pooling2d_2 (MaxPooling  (None, 18, 18, 128)      0
 2D)

 flatten (Flatten)           (None, 41472)             0

 dense (Dense)               (None, 128)               5308544

 dense_1 (Dense)             (None, 2)                 258

=================================================================
Total params: 5,595,810
Trainable params: 5,595,810
Non-trainable params: 0
```

- Tải dataset từ drive lên và xử lý dữ liệu:

Với các ảnh được tải lên ta chuẩn hóa dữ liệu đầu vào bằng cách chia toàn bộ giá trị màu cho 255 để được giá trị min là 0.0 và max là 1.0

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255,shear_range =
0.2,zoom_range = 0.2,horizontal_flip = True)
training_set =
train_datagen.flow_from_directory('/content/gdrive/MyDrive/Colab
Notebooks/Final/dataset/train',target_size = (150,150),batch_size =
32,class_mode =  'categorical')
```

```
Found 3400 images belonging to 2 classes.
```

In [7]:

```
test_set = train_datagen.flow_from_directory('/content/gdrive/MyDrive/Colab
Notebooks/Final/dataset/test',target_size = (150,150),batch_size =
32,class_mode =  'categorical')
```

```
Found 600 images belonging to 2 classes.
```

In [8]:

```
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accu
racy'])
```

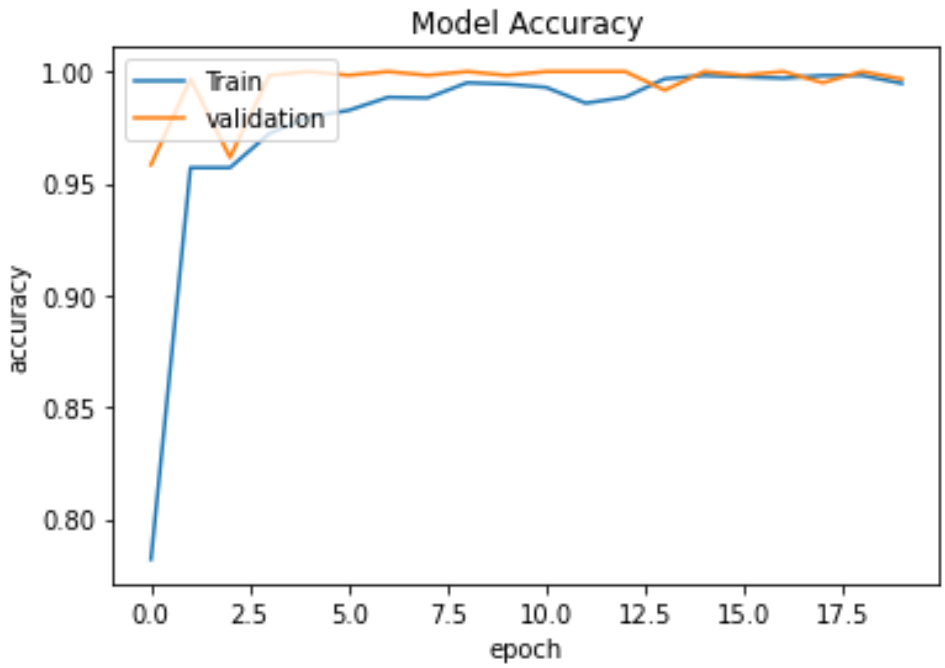
- Tiến hành train:

```
history=model.fit(training_set,epochs=20,batch_size=32,verbose=1, validati
on_data=test_set, callbacks=[EarlyStopping(monitor='val_loss', patience=20
)])
```

```
Epoch 1/20
107/107 [==============================] - 705s 6s/step - loss: 0.7471 - accuracy: 0.7821 - val_loss: 0.0885 - val_accuracy: 0.9583
Epoch 2/20
107/107 [==============================] - 507s 5s/step - loss: 0.1428 - accuracy: 0.9571 - val_loss: 0.0135 - val_accuracy: 0.9967
Epoch 3/20
107/107 [==============================] - 504s 5s/step - loss: 0.1360 - accuracy: 0.9571 - val_loss: 0.0692 - val_accuracy: 0.9617
Epoch 4/20
107/107 [==============================] - 505s 5s/step - loss: 0.0986 - accuracy: 0.9724 - val_loss: 0.0146 - val_accuracy: 0.9983
Epoch 5/20
107/107 [==============================] - 509s 5s/step - loss: 0.0734 - accuracy: 0.9797 - val_loss: 0.0089 - val_accuracy: 1.0000
Epoch 6/20
107/107 [==============================] - 508s 5s/step - loss: 0.0581 - accuracy: 0.9826 - val_loss: 0.0084 - val_accuracy: 0.9983
Epoch 7/20
107/107 [==============================] - 510s 5s/step - loss: 0.0347 - accuracy: 0.9885 - val_loss: 0.0016 - val_accuracy: 1.0000
Epoch 8/20
107/107 [==============================] - 511s 5s/step - loss: 0.0455 - accuracy: 0.9882 - val_loss: 0.0048 - val_accuracy: 0.9983
Epoch 9/20
107/107 [==============================] - 513s 5s/step - loss: 0.0188 - accuracy: 0.9950 - val_loss: 2.2721e-04 - val_accuracy: 1.0000
Epoch 10/20
107/107 [==============================] - 514s 5s/step - loss: 0.0192 - accuracy: 0.9944 - val_loss: 0.0027 - val_accuracy: 0.9983
Epoch 11/20
107/107 [==============================] - 503s 5s/step - loss: 0.0271 - accuracy: 0.9929 - val_loss: 0.0032 - val_accuracy: 1.0000
Epoch 12/20
107/107 [==============================] - 503s 5s/step - loss: 0.0549 - accuracy: 0.9859 - val_loss: 0.0145 - val_accuracy: 1.0000
Epoch 13/20
107/107 [==============================] - 503s 5s/step - loss: 0.0396 - accuracy: 0.9885 - val_loss: 5.8053e-04 - val_accuracy: 1.0000
Epoch 14/20
107/107 [==============================] - 510s 5s/step - loss: 0.0085 - accuracy: 0.9968 - val_loss: 0.0336 - val_accuracy: 0.9917
Epoch 15/20
107/107 [==============================] - 514s 5s/step - loss: 0.0061 - accuracy: 0.9982 - val_loss: 1.6595e-04 - val_accuracy: 1.0000
Epoch 16/20
107/107 [==============================] - 502s 5s/step - loss: 0.0062 - accuracy: 0.9976 - val_loss: 0.0090 - val_accuracy: 0.9983
Epoch 17/20
107/107 [==============================] - 523s 5s/step - loss: 0.0094 - accuracy: 0.9971 - val_loss: 0.0013 - val_accuracy: 1.0000
Epoch 18/20
107/107 [==============================] - 511s 5s/step - loss: 0.0050 - accuracy: 0.9982 - val_loss: 0.0364 - val_accuracy: 0.9950
Epoch 19/20
107/107 [==============================] - 505s 5s/step - loss: 0.0041 - accuracy: 0.9982 - val_loss: 9.0233e-04 - val_accuracy: 1.0000
Epoch 20/20
107/107 [==============================] - 504s 5s/step - loss: 0.0184 - accuracy: 0.9947 - val_loss: 0.0051 - val_accuracy: 0.9967
```

- Đánh giá model:

```
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['Train','validation'],loc='upper left')
plt.show()
```



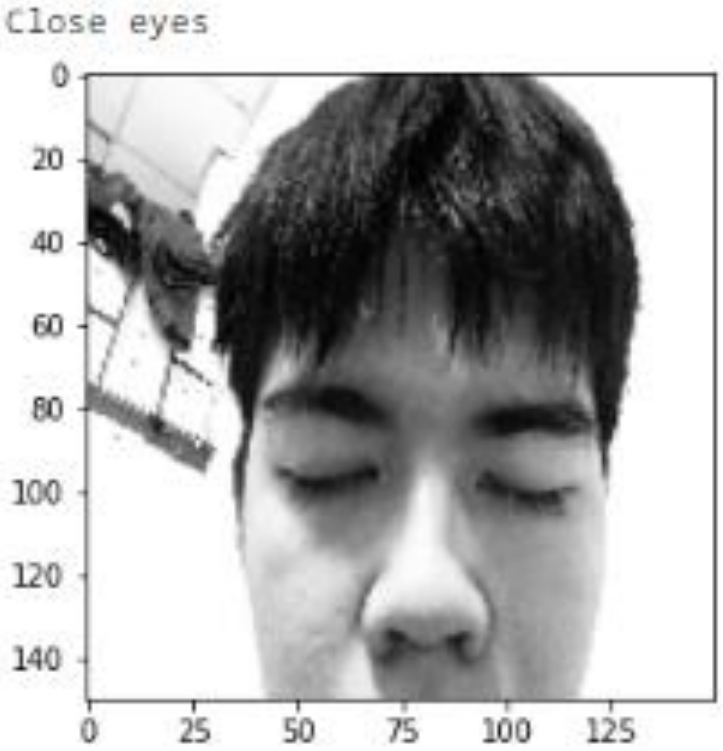

```
Score=model.evaluate(training_set,verbose=0)
print('Train Loss', Score[0])
print('Train Accuracy', Score[1])
Train Loss 0.01867879368364811
Train Accuracy 0.9941176176071167
```

## 3.3. Tiến hành nhận diện

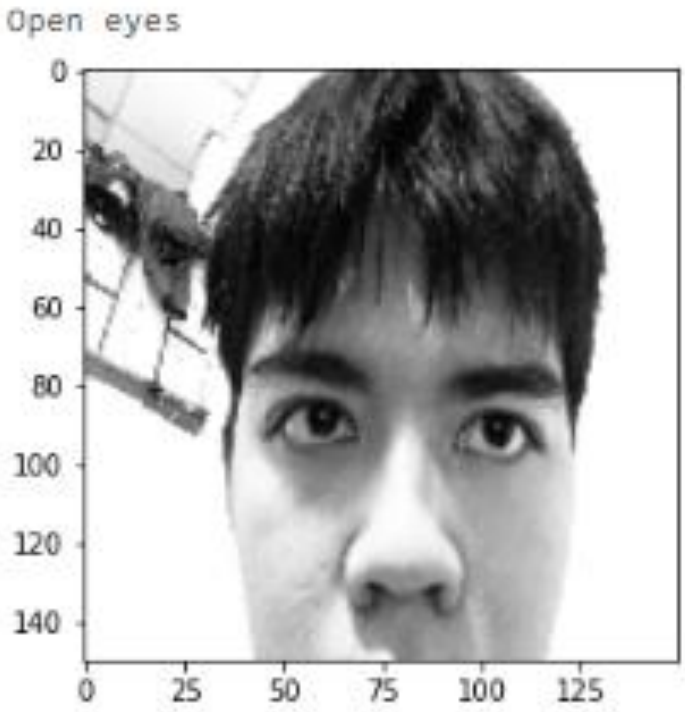
```
import os
import numpy as np
from tensorflow.keras.utils import load_img, img_to_array
import matplotlib.pyplot as plt
model_CNN = load_model('/content/gdrive/MyDrive/Colab
Notebooks/Final/dataset1/Cuoiki.h5')
test="/content/gdrive/MyDrive/Colab Notebooks/Final/dataset1/val"

for i in os.listdir(test):
  img=load_img(test+'/'+i,target_size=(150,150))
  plt.imshow(img)
  img=img_to_array(img)
  img=img.astype('float32')
  img=img/255
  img=np.expand_dims(img,axis=0)
  result=model_CNN.predict(img)
  if round(result[0][0])==1:
    prediction='Close eyes'
  if round(result[0][1])==1:
    prediction='Open eyes'


  print(prediction)
  plt.show()
```

Kết quả chạy với hình ảnh:

Nhắm mắt:




Mở mắt:



Kết quả khi chạy real-time:


## 4. Nhận xét, đánh giá

- Độ chính xác khi nhận diện của mô hình còn thấp, sai số cao
- Vì ảnh chụp có đặc điểm khá tương đồng, data được tạo từ ảnh chụp điện thoại, độ phân giải thấp, không được rõ nét.
- Độ phơi sáng mỗi lúc chụp khác nhau ảnh hưởng đến quá trình nhận diện.
- Model được tạo chưa đạt hiệu quả dẫn đến sai số cao.

## 5. Kết luận

Độ chính xác của mô hình nhận diện chưa chính xác cần điều chỉnh lại.

Tuy kết quả không đạt được như mong đợi, nhưng trong quá trình tìm hiểu để hoàn thành đề tài cuối kỳ và trong các buổi tham gia lớp học trực tiếp đã giúp cho em có thêm được những kiến thức, hiểu biết cần thiết hỗ trợ sau này.

Mã QR link github và video real-time:


***Hình 3:*** *Mã QR link Github*

## 6. Tài liệu tham khảo:

[1] Blog – Thuật toán CNN – Convolutional Neural Network - https://topdev.vn/

[2] Trí tuệ nhân tạo AI là gì? Ứng dụng như thế nào trong cuộc sống? - https://vnreview.vn/