**BỘ THÔNG TIN VÀ TRUYỀN THÔNG**



**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**CƠ SỞ TẠI THÀNH PHỐ HỒ CHÍ MINH**

****

**XỬ LÝ ẢNH**

**ĐỀ TÀI: NHẬN DIỆN TIỀN VIỆT NAM**

**Giảng viên hướng dẫn : ThS. Huỳnh Trung Trụ**

**Lớp : D20CQCNPM01-N**

**Sinh viên thực hiện : Đoàn Long Âu – N20DCCN004**

**Phạm Ngọc Bảo – N20DCCN006  
 Lê Văn Phúc – N20DCCN055  
 Nguyễn Thái Trưởng – N20DCCN083**

**Hồ Chí Minh, ngày 21 tháng 12 năm 2023**

**MỤC LỤC**

[**I.** **GIỚI THIỆU ĐỀ TÀI** 1](#_Toc153872743)

[**1.** **Đặt vấn đề** 1](#_Toc153872744)

[**2.** **Mục tiêu nghiên cứu** 1](#_Toc153872745)

[**3.** **Ứng dụng** 1](#_Toc153872746)

[**II.** **CƠ SỞ LÝ THUYẾT** 2](#_Toc153872747)

[**1.** **Giới thiệu về Python** 2](#_Toc153872748)

[**2.** **Các thư viện của Python được dùng trong chương trình** 2](#_Toc153872749)

[**III.** **PHÂN TÍCH BÀI TOÁN** 3](#_Toc153872750)

[**1.** **Tạo dữ liệu cho bài toán** 3](#_Toc153872751)

[**2.** **Xử lý dữ liệu ảnh** 4](#_Toc153872752)

[**3.** **Thiết kế mạng CNN classify dùng để train** 6](#_Toc153872753)

[**4.** **Sử dụng augmentation cho dữ liệu** 10](#_Toc153872754)

[**5.** **Run model** 11](#_Toc153872755)

[**IV.** **CHƯƠNG TRÌNH** 14](#_Toc153872756)

# **GIỚI THIỆU ĐỀ TÀI**

### **Đặt vấn đề**

Ngày nay, với sự phát triển mạnh mẽ của công nghệ thông tin và kỹ thuật máy tính. Con người đã áp dụng chúng vào các lĩnh vực khác nhau và thu lại được những thành tựu to lớn. Cùng với sự tiến bộ vượt bậc về tốc độ tính toán và sức mạnh xử lý, giá cả và chi phí đã giảm xuống mức mà máy tính và các thiết bị liên quan đến xử lý khác không còn là thiết bị chuyên dụng nữa. Kể từ đó, khái niệm hình ảnh kỹ thuật số đã trở nên phổ biến đối với hầu hết mọi người trong xã hội, và việc thu nhận hình ảnh kỹ thuật số và xử lý hình ảnh thông qua thiết bị cá nhân hoặc thiết bị đặc biệt trở nên dễ dàng và đơn giản.

Cùng với xu hướng phát triển của công nghệ, xử lý ảnh cũng trở thành lĩnh vực đang được quan tâm và trở thành môn học chuyên ngành để đào tạo các sinh viên ngành công nghệ và kỹ thuật trên khắp thế giới. Xử lý ảnh mang lại nhiều kiến thức về hình ảnh cũng như xử lý thông tin hình ảnh, qua đó mang lại nhiều lợi ích cho công việc của con người trong nghiên cứu cũng như sản xuất, nâng cao được hiệu quả làm việc, giảm thiểu các sai sót, đưa ra được kết quả chính xác sau quá trình xử lý hình ảnh.

Tổng kết cả quá trình học tập và nghiên cứu, chúng em xin giới thiệu đề tài của nhóm: “Nhận diện tiền Việt Nam”. Việc tìm hiểu, nghiên cứu đề tài này sẽ cung cấp kiến thức, kinh nghiệm giúp chúng em tự tin giải quyết các vấn đề về nhận dạng đặc trưng ảnh, ứng dụng vào thực tiễn đời sống.

1. **Mục tiêu nghiên cứu**

Tạo chương trình nhận dạng tiền Việt Nam với 7 mệnh giá thông qua camera trên python.

### **Ứng dụng**

Nhận dạng các vật thể bằng mạng neural tích chập CNN hay deep learning là một xu thế hiện đại trong nhiều ngành công nghệ nói chung và xử lý ảnh nói riêng

Mạng CNN hỗ trợ trích lọc các đặc trưng của vật thể bằng việc nhân tích chập ma trận nên dễ dàng nhận diện được nhiều đối tượng khác nhau.

Tiền tệ hiện là phương thức giao dịch phổ biến tại các TTTM, máy bán hàng tự động… Nên việc nhận dạng tiền sẽ hỗ trợ rất nhiều trong cuộc sống, tự động hóa giao dịch…

1. **CƠ SỞ LÝ THUYẾT**

### **Giới thiệu về Python**

Python là một ngôn ngữ lập trình thông dịch (interpreted), hướng đối tượng (object-oriented), và là một ngôn ngữ bậc cao (high-level) ngữ nghĩa động (dynamic semantics). Python hỗ trợ các module và gói (package), khuyến khích chương trình module hóa và tái sử dụng mã. Trình thông dịch Python và thư viện chuẩn mở rộng có sẵn dưới dạng mã nguồn hoặc nhị phân miễn phí cho tất cả các nền tảng chính và có thể được phân phối tự do.

Sau đây là đặc điểm của Python

* Ngữ pháp đơn giản, dễ đọc
* Vừa hướng thủ tục (procedural-oriented), vừa hướng đối tượng (object-oriented)
* Hỗ trợ module và hỗ trợ gói (package)
* Xử lý lỗi bằng ngoại lệ (Exception)
* Kiểu dữ liệu động ở mức cao.
* Có các bộ thư viện chuẩn và các module ngoài, đáp ứng tất cả các nhu cầu lập trình.
* Có khả năng tương tác với các module khác viết trên C/C++ (Hoặc Java cho Jython, hoặc .Net cho IronPython).
* Có thể nhúng vào ứng dụng như một giao tiếp kịch bản (scripting interface).

1. **Các thư viện của Python được dùng trong chương trình**

OpenCv, PIL: là một thư viện mã nguồn mở hàng đầu cho thị giác máy tính (computer vision), xử lý ảnh và máy học, và tính năng tăng tốc GPU trong hoạt động thời gian thực. Trong khi PIL được sử dụng để xử lý ảnh.

Numpy: là một thư viện toán học phổ biến và mạnh mẽ của Python. Cho phép làm việc hiệu quả với ma trận và mảng, đặc biệt là dữ liệu ma trận và mảng lớn với tốc độ xử lý nhanh hơn nhiều lần khi chỉ sử dụng “core Python” đơn thuần.

Tkinter: là thư viện GUI tiêu chuẩn dùng cho Python. Tkinter trong Python cung cấp một cách nhanh chóng và dễ dàng để tạo các ứng dụng GUI. Tkinter cung cấp giao diện hướng đối tượng cho bộ công cụ Tk GUI.

Keras, tensorflow, sklearn… và các thư viện hỗ trợ khác.

1. **PHÂN TÍCH BÀI TOÁN**

**Dự kiến thực hiện sẽ gồm các bước sau:**

* Tạo dữ liệu bằng cách đọc ảnh các tờ tiền từ camera
* Thiết kế mạng NN với đầu vào là ảnh (128, 128, 3), đưa vào mạng VGG16 và đầu ra của VGG16 sẽ dùng để đưa vào 1 mạng NN nhỏ kết thúc bằng 1 lớp Dense và hàm softmax.
* Đầu ra sẽ là 1 vector softmax chứa các probality p(i) ứng với mỗi class i, in ra giá trị max trong vector đó và chọn đó làm class dự đoán.

### **Tạo dữ liệu cho bài toán**

Cách tạo dữ liệu đơn giản là viết một đoạn python đọc liên tục từ camera và save lại vào các thư mục tương ứng ảnh các tờ tiền.

**import cv2**

**import os**

***# Label: 000000 là ko cầm tiền, còn lại là các mệnh giá***

**label = "500000"**

**cap = cv2.VideoCapture(0)**

***# Biến đếm, để chỉ lưu dữ liệu sau khoảng 60 frame, tránh lúc đầu chưa kịp cầm tiền lên***

**i = 0**

**while True:**

***# Capture frame-by-frame***

***#***

**i += 1**

**ret, frame = cap.read()**

**if not ret:**

**continue**

**frame = cv2.resize(frame, dsize=None, fx=0.3, fy=0.3)**

***# Hiển thị***

**cv2.imshow('frame', frame)**

***# Lưu dữ liệu***

**if i >= 60:**

**print("Số ảnh capture = ", i-60)**

***# Tạo thư mục nếu chưa có***

**if not os.path.exists('data/' + str(label)):**

**os.mkdir('data/' + str(label))**

**cv2.imwrite('data/' + str(label) + "/" + str(i) + ".png", frame)**

**if cv2.waitKey(1) & 0xFF == ord('q'):**

**break**

***# When everything done, release the capture***

**cap.release()**

**cv2.destroyAllWindows()**

Cách tạo dữ liệu đơn giản là viết một đoạn python đọc liên tục từ camera và save lại vào các thư mục tương ứng ảnh các tờ tiền

Sau khi chạy thành công các class tương ứng với các nhãn của các tờ tiền, chúng ta sẽ có thư mục data với các subfolders như sau:



Ví dụ 1 số ảnh data với mỗi class:





1. **Xử lý dữ liệu ảnh**

Trong quá trình chúng ta train model, thử nghiệm model chúng ta sẽ đọc hết file một lần và thực hiện:

* Convert nhãn (00000, 10000, 20000…) thành one-hot
* Resize ảnh về 128x128

**def** save\_data(raw\_folder=raw\_folder):

dest\_size = (128, 128)

print(**"Bắt đầu xử lý ảnh..."**)

pixels = []

labels = []

*# Lặp qua các folder con trong thư mục raw*

**for** folder **in** listdir(raw\_folder):

**if** folder!=**'.DS\_Store'**:

print(**"Folder="**,folder)

*# Lặp qua các file trong từng thư mục chứa các em*

**for** file **in** listdir(raw\_folder + folder):

**if** file!=**'.DS\_Store'**:

print(**"File="**, file)

pixels.append( cv2.resize(cv2.imread(raw\_folder + folder +**"/"** + file),dsize=(128,128)))

labels.append( folder)

pixels = np.array(pixels)

labels = np.array(labels)*#.reshape(-1,1)*

**from** sklearn.preprocessing **import** LabelBinarizer

encoder = LabelBinarizer()

labels = encoder.fit\_transform(labels)

print(labels)

file = open(**'pix.data'**, **'wb'**)

*# dump information to that file*

pickle.dump((pixels,labels), file)

*# close the file*

file.close()

**return**

* Sau khi lưu xong chúng ta sẽ có file pix.data, file này chứa cả ảnh và labels. Khi chúng ta train dữ liệu thì chỉ cần load file pix.data lên
* Load file lên như sau:

**def** load\_data():

file = open(**'pix.data'**, **'rb'**)

*# dump information to that file*

(pixels, labels) = pickle.load(file)

*# close the file*

file.close()

print(pixels.shape)

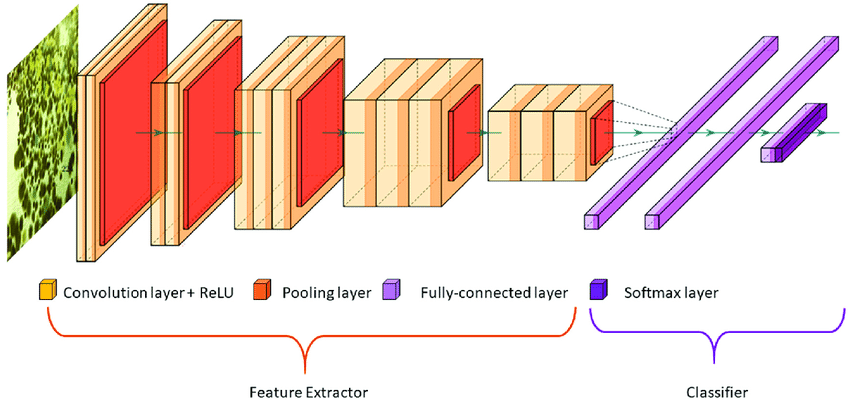
print(labels.shape)

**return** pixels, labels

X,y = load\_data()

### **Thiết kế mạng CNN classify dùng để train**

Sau khi xử lý, chuẩn bị dữ liệu xong, đây là phần quan trọng nhất - thiết kế mạng để train

Chúng ta đã biết cấu trúc mạng CNN, cụ thể là VGG16 sẽ có dạng như sau:

Nhìn vào cấu trúc VGG16 sẽ gồm 2 phần, phần màu cam là trích đặc trưng của ảnh còn phần màu tím sẽ là các lớp FC để classify. Tuy nhiên chúng ta sẽ không cần sử dụng classify nên ta sẽ khai báo include\_top=False khi implement mạng này:

model\_vgg16\_conv = VGG16(weights=**'imagenet'**,include\_top=**False**)

Bây giờ chúng ta thực hiện ghép nối FC vào bằng đoạn lệnh:

*# Dong bang cac layer*

**for** layer **in** model\_vgg16\_conv.layers:

layer.trainable = **False**

*# Tao model*

input\_ = Input(shape=(128, 128, 3), name=**'image\_input'**)

output\_vgg16\_conv = model\_vgg16\_conv(input\_)

*# Them cac layer FC va Dropout*

x = Flatten(name=**'flatten'**)(output\_vgg16\_conv)

x = Dense(4096, activation=**'relu'**, name=**'fc1'**)(x)

x = Dropout(0.5)(x)

x = Dense(4096, activation=**'relu'**, name=**'fc2'**)(x)

x = Dropout(0.5)(x)

x = Dense(8, activation=**'softmax'**, name=**'predictions'**)(x)

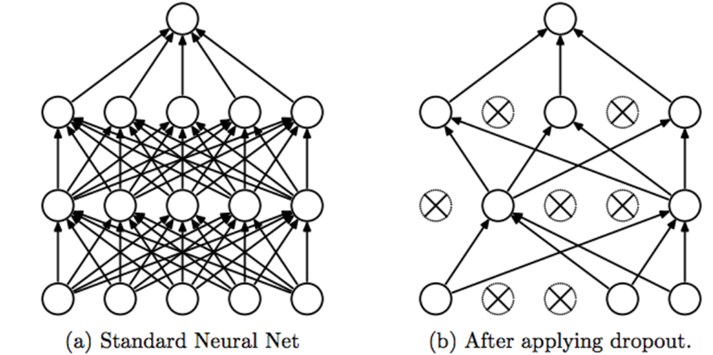
Qua 2 lớp neural với hàm kích hoạt reLU - lọc giá trị âm

Dropout - Nếu một lớp fully connected có quá nhiều tham số và chiếm hầu hết tham số, các nút mạng trong lớp đó quá phụ thuộc lẫn nhau trong quá trình huấn luyện thì sẽ hạn chế sức mạnh của mỗi nút, dẫn đến việc kết hợp quá mức.

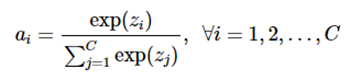
Dropout sẽ được học thêm các tính năng mạnh mẽ hữu ích

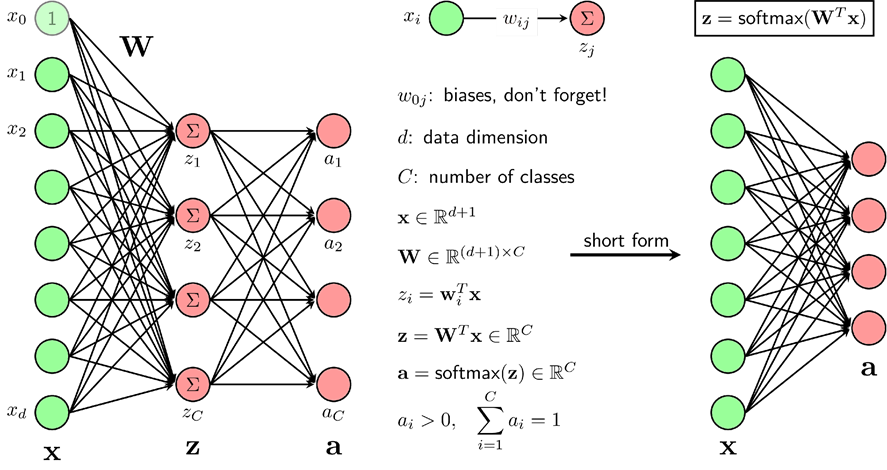
Nó gần như tăng gấp đôi số epochs cần thiết để hội tụ. Tuy nhiên, thời gian cho mỗi epoch là ít hơn.

Ta có H đơn vị ẩn, với xác suất bỏ học cho một đơn vị là (1 - p) thì ta có thể có 2H mô hình có thể có. Nhưng trong giai đoạn test, tất cả các nút mạng phải được xét đến, và mỗi activation sẽ giảm đi 1 hệ số p.

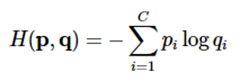


Layer cuối có 8 classes (0, 10k, 20k…) với activation softmax với output sẽ là 1 vector chứa xác suất từng class

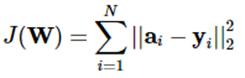


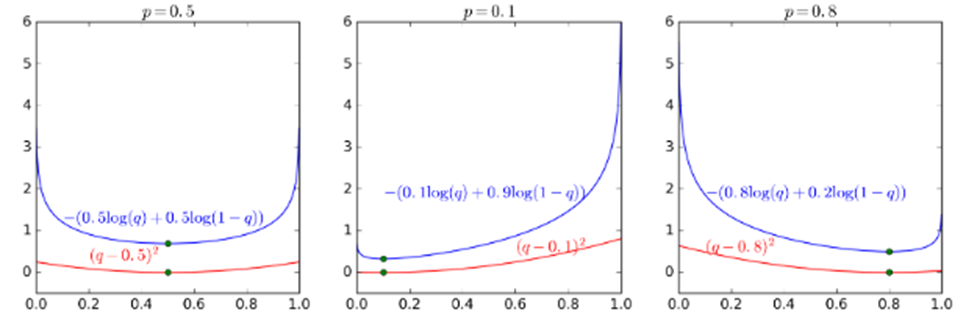


Hàm mất mát (loss) - cross\_entropy với p và q (hay y và a) là rời rạc (như ngõ ra thực sự và ngõ ra dự đoán của class thứ i) ta có công thức



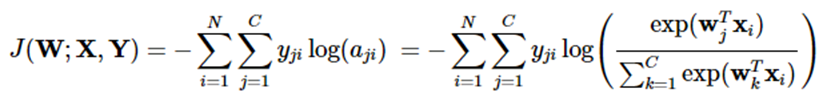
Hàm loss - bình phương khoảng cách



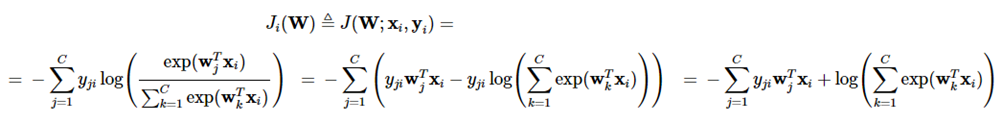


Hàm cross\_entropy sẽ cho nghiện gần với p hơn hàm bình phương khoảng cách vì các nghiệm ở xa sẽ bị ‘trừng phạt’ rất nặng

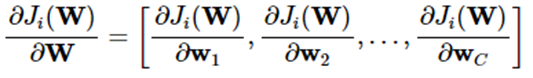
Kết hợp tất cả các cặp dữ liệu xi, yi, i=1, 2, …N chúng ta sẽ có hàm mất mát cho Softmax Regression như sau:



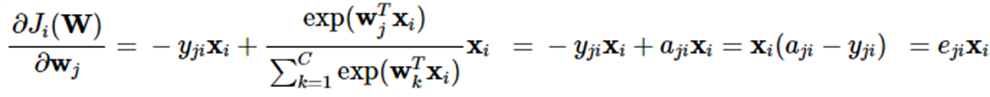
Với chỉ một cặp dữ liệu (xi,yi) ta có hàm loss:



Gradient của hàm loss:

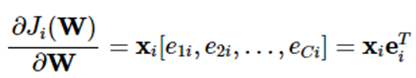


Trong đó, Gradient theo từng cột có thể tính từ hàm Ji (W):

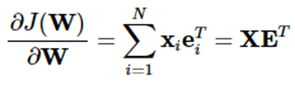


Giá trị eji=aji−yji có thể được coi là sai số dự đoán.

⇨ Gradient trên 1 cặp data thứ j



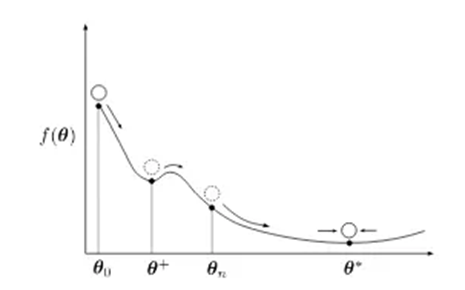
⇨ Gradient trên toàn bộ data

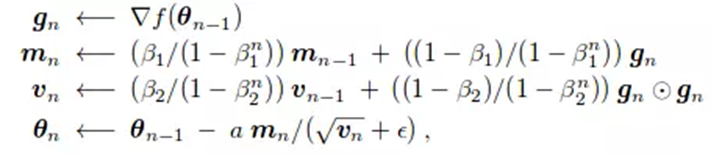
với **E = A – Y**

Giả sử rằng chúng ta sử dụng SGD, công thức cập nhật cho ma trận trọng số W sẽ là:



Tối ưu hàm loss sử dụng Adam



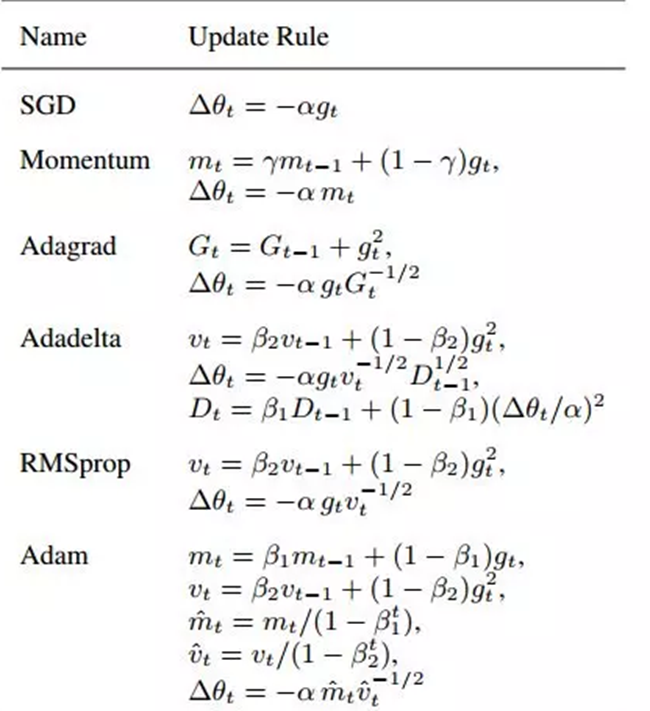


Giống với Adadelta và RMSprop, nó duy trì trung bình bình phương độ dốc (slope) quá khứ vt và cũng đồng thời duy trì trung bình độ dốc quá khứ mt, giống momentum.

Trong khi momentum giống như một quả cầu lao xuống dốc, thì Adam lại giống như một quả cầu rất nặng và có ma sát (friction), nhờ vậy nó dễ dàng vượt qua local minimum và đạt tới điểm tối ưu nhất (flat minimum)

Nó đạt được hiệu ứng Heavy Ball with Friction (HBF) nhờ vào hệ số (mt/ sqrt(vt))

Các công thức update khác



Đánh giá chất lượng model - [Accuracy](https://www.atoha.com/blogs/kien-thuc/accuracy-vs-precision-trong-pmp). Độ chính xác [Accuracy](https://www.atoha.com/blogs/kien-thuc/accuracy-vs-precision-trong-pmp) được định nghĩa là cách các giá trị đo gần với giá trị đích.

### **Sử dụng augmentation cho dữ liệu**

aug = ImageDataGenerator(rotation\_range=20, zoom\_range=0.1, rescale=1./255, width\_shift\_range=0.1, height\_shift\_range=0.1, horizontal\_flip=**True**, brightness\_range=[0.2,1.5], fill\_mode=**"nearest"**)

aug\_val = ImageDataGenerator(rescale=1./255)

vgghist=vggmodel.fit\_generator(aug.flow(X\_train, y\_train, batch\_size=64),

epochs=50, *# steps\_per\_epoch=len(X\_train)//64,*

validation\_data=aug.flow(X\_test,y\_test,

batch\_size=64),

callbacks=callbacks\_list)

vggmodel.save(**"vggmodel.h5"**)

Bây giờ nếu chúng ta sử dụng ngay ảnh nói trên để train cho model CNN Classify thì sẽ bị hiện tượng Overfit vì dữ liệu nhiều nhưng đa phần giống nhau. Dẫn đến train có chất lượng tốt nhưng khi test sẽ thấy không nhận chuẩn lắm.

Chúng ta sẽ thực hiện augment dữ liệu để làm phong phú hơn dữ liệu, tăng data variance, tăng tính tổng quát cho model bằng ImageDataGenrator của Keras

### **Run model**

**from** os **import** listdir

**import** cv2

**import** numpy **as** np

**import** pickle

**from** sklearn.preprocessing **import** OneHotEncoder

**from** sklearn.model\_selection **import** train\_test\_split

**from** keras.applications.vgg16 **import** VGG16

**from** keras.layers **import** Input, Flatten, Dense, Dropout

**from** keras.models **import** Model

**from** keras.callbacks **import** ModelCheckpoint

**import** matplotlib.pyplot **as** plt

**import** random

**from** keras.models **import** load\_model

**import** sys

cap = cv2.VideoCapture(0)

*# Dinh nghia class*

class\_name = [**'00000'**, **'1000'**, **'2000'**, **'5000', '10000', '20000', '200000', '500000'**]

**def** get\_model():

model\_vgg16\_conv = VGG16(weights=**'imagenet'**, include\_top=**False**)

*# Dong bang cac layer*

**for** layer **in** model\_vgg16\_conv.layers:

layer.trainable = **False**

*# Tao model*

input = Input(shape=(128, 128, 3), name=**'image\_input'**)

output\_vgg16\_conv = model\_vgg16\_conv(input)

*# Them cac layer FC va Dropout*

x = Flatten(name=**'flatten'**)(output\_vgg16\_conv)

x = Dense(4096, activation=**'relu'**, name=**'fc1'**)(x)

x = Dropout(0.5)(x)

x = Dense(4096, activation=**'relu'**, name=**'fc2'**)(x)

x = Dropout(0.5)(x)

x = Dense(8, activation=**'softmax'**, name=**'predictions'**)(x)

*# Compile*

my\_model = Model(inputs=input, outputs=x)

my\_model.compile(loss=**'categorical\_crossentropy'**, optimizer=**'adam'**, metrics=[**'accuracy'**])

**return** my\_model

*# Load weights model da train*

my\_model = get\_model()

my\_model.load\_weights(**"weights-04-0.96.hdf5"**)

**while True**:

*# Capture frame-by-frame*

*#*

ret, image\_org = cap.read()

**if not** ret:

**continue**

image\_org = cv2.resize(image\_org, dsize=**None**,fx=0.5,fy=0.5)

*# Resize*

image = image\_org.copy()

image = cv2.resize(image, dsize=(128, 128))

image = image.astype(**'float'**)\*1./255

*# Convert to tensor*

image = np.expand\_dims(image, axis=0)

*# Predict*

predict = my\_model.predict(image)

print(**"This picture is: "**, class\_name[np.argmax(predict[0])], (predict[0]))

print(np.max(predict[0],axis=0))

**if** (np.max(predict)>=0.8) **and** (np.argmax(predict[0])!=0):

*# Show image*

font = cv2.FONT\_HERSHEY\_SIMPLEX

org = (50, 50)

fontScale = 1.5

color = (0, 255, 0)

thickness = 2

cv2.putText(image\_org, class\_name[np.argmax(predict)], org, font,

fontScale, color, thickness, cv2.LINE\_AA)

cv2.imshow(**"Picture"**, image\_org)

**if** cv2.waitKey(1) & 0xFF == ord(**'q'**):

**break**

*# When everything done, release the capture*

cap.release()

cv2.destroyAllWindows()

1. **CHƯƠNG TRÌNH**

* Đọc ảnh từ webcam, resize về 128x128
* Chuyển ảnh về tensor bằng lệnh mở rộng mảng thêm 1 chiều
* Dự đoán ảnh thuộc classes nào:
* In ra output hàm softmax là một vector chứa tất cả các xác suất của từng classes
* In ra class có xác suất cao nhất: argmax của f(x) là giá trị x khi f(x) lớn nhất, ở đây chính là tên của class có xác suất lớn nhất
* Nếu 1 classes có xác suất > 80% và không phải class 000000 (không có tiền) thì in ra màn hình tên của class đó
* Nhấn ‘q’ để kết thúc chương trình

