HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 2

--

Báo cáo môn học
XỬ LÍ ẢNH

Đề tài:

NHẬN DIỆN KHỐI U NÃO Ở NGƯỜI

Giảng viên: Huỳnh Trung Trụ
Sinh viên thực hiện:
Nhóm 18:
1. **Văn Tố Hữu - N20DCCN026**
2. **Vũ Huy Hùng - N20DCCN020**

TP.HCM, tháng 12/2023

Contents

I. GIỚI THIỆU

1. Lý do chọn đề tài

Ngày nay học máy, học sâu ngày càng phát triển. Nó mang lại nhiều lợi ích lớn lao cho xã hội. Trong kinh doanh, AI có thể trợ giúp con người đưa ra những quyết đinh, trong giáo dục AI có thể giúp con người thu thập và tiếp cận ngày càng nhiều tri thức hơn. Đặc biệt trong y học, nó còn có thể thay thế một số công việc của bác sĩ. Với sự cần thiết trong việc nhận diện khối u não ở người một cách chính xác và dễ dàng, đề tài là một nghiên cứu nhằm đáp ứng nhu cầu đó.

2. Mục tiêu chọn đề tài

Chúng em chọn dề tài này với mục tiêu và nghiên cứu và phát triển mô hình AI có thể nhận dạng được khối u não ở người.

- Nắm vững kiến thức cơ bản về xử lý ảnh và học máy.
- Phát triển một mô hình CNN có khả năng nhận diện chính xác và nhanh chóng các khối u qua ảnh X-quang khác nhau.
- Thực hiện đánh giá chi tiết về độ chính xác và tốc độ của mô hình.

II. CƠ SỞ LÍ THUYẾT

1. Mô hình CNN

a. Khái niệm

Convulutional Neural Network là một trong những mô hình mạng Deep Learning phổ biến nhất hiện nay có khả năng nhận dạng và phân loại hình ảnh có độ chính xác cao.

b. Các lớp cơ bản

Lớp tích chập 2 chiều (CON2VD)

- Sử dụng để trích lọc các đặc trưng của ảnh đầu vào.

➔ Input là ảnh biểu diễn bởi ma trận [M x N x D]. Trong đó:

- M: Chiều rộng
- N: Chiều cao
- D: Số chiều

- Sử dụng các cửa sổ trượt để trích xuất đặc trưng từng vùng ảnh.

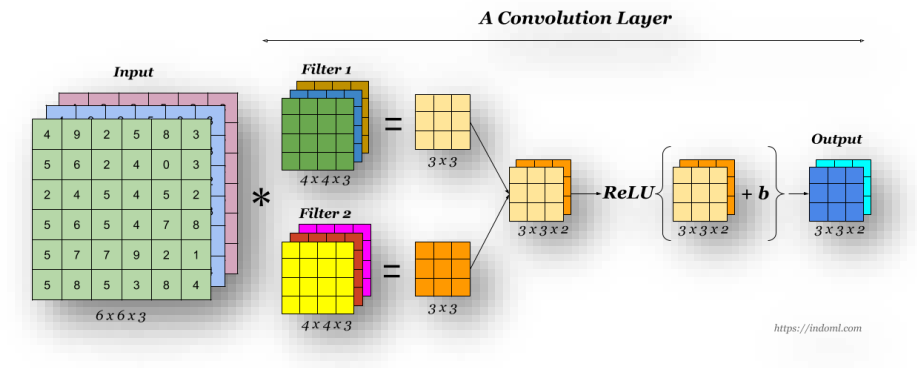
➔ Output là các đặc trưng của ảnh biểu diễn dưới dạng ma trận.

\+ Một số tham số của hàm Conv:

- kernel_size: kích thước filter.
- strides: Số bước nhảy.

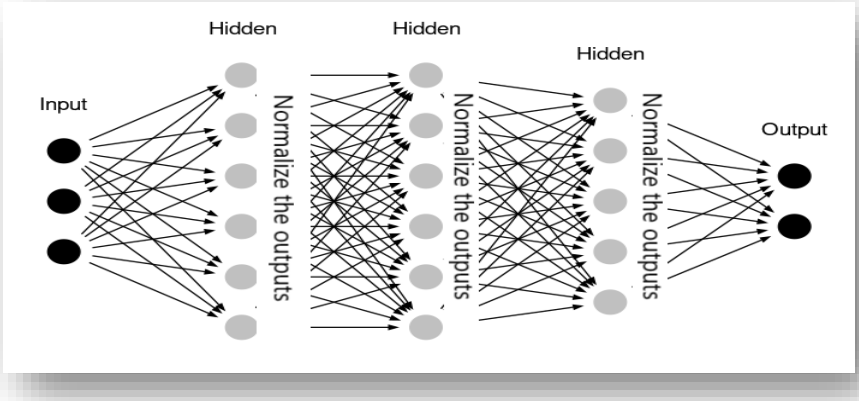
→ Nhảy càng lớn thì ảnh càng thu nhỏ

- padding: tăng kích thước ảnh đầu vào để đạt kích thước mong muốn.

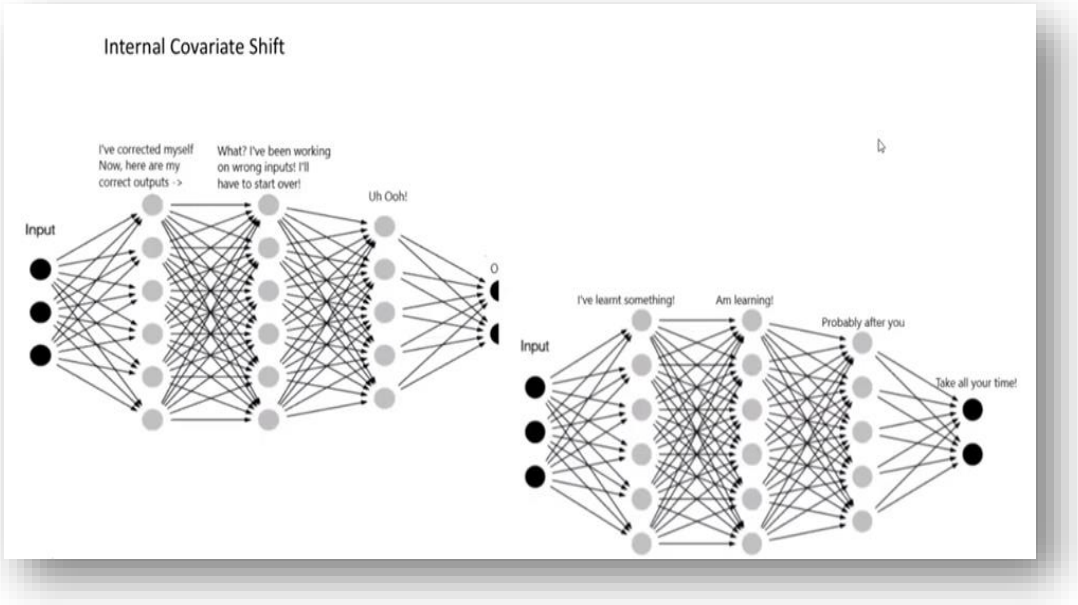


Lớp BatchNormalization

- Chuẩn hóa dữ liệu giữa 2 lớp trung gian giảm thiểu dữ liệu tính toán tránh tình trạng 1 lớp đã tính xong đẩy dữ liệu qua lớp khác nhưng lớp tiếp theo chưa tính xong bộ dữ liệu cũ.
- Batch Normalization có mục đích chính là ổn định quá trình huấn luyện, giảm hiện tượng "internal covariate shift", cải thiện tốc độ học, và làm cho mạng nơ-ron dễ dàng học từ dữ liệu.

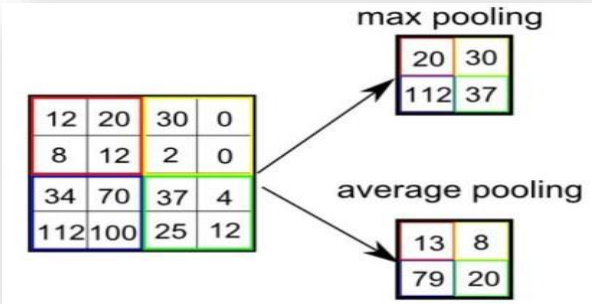


- Hiện tương Internal Covriate Shift: Trong mạng tuần tự thì output của lớp trước sẽ là input của lớp sau khi đó sẽ xảy ra hiện tượng lớp sau chưa tính xong bộ cũ nhưng đã phải cập nhật bộ mới.



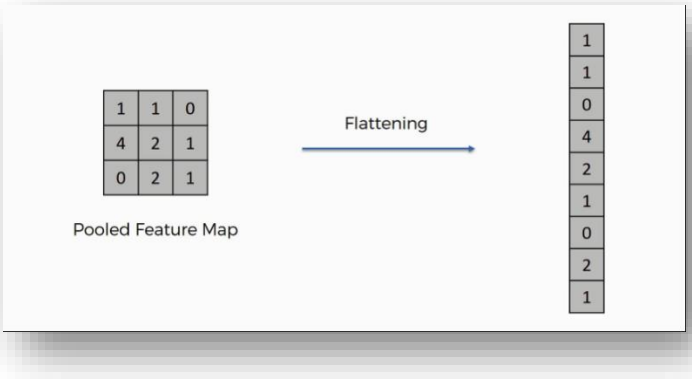
Lớp Pooling

- Pooling Layer thực hiện chức năng làm giảm kích thước bản đồ đặc trưng.
  → Giảm số lượng tham số/tính toán cho các lớp phía sau.
- Làm tăng tính bất biến, tính ổn định với các biến đổi nhỏ trong đầu vào.
- Max Pooling là được sử dụng nhiều vào phổ biến hơn cả với ý tưởng cũng rất sát với thực tế con người đó là: Giữ lại chi tiết quan trọng hay giữ lại pixel có giá trị lớn nhất.



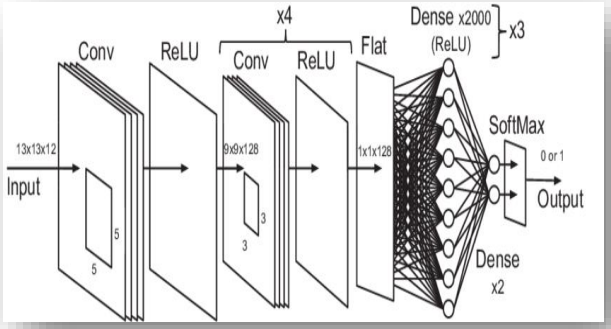
Lớp Flattern

- Chuyển phẳng ngỏ ra dạng ma trận của mạng tích chập sang dạng vector.
- Làm ngỏ vào cho lớp fully connected.



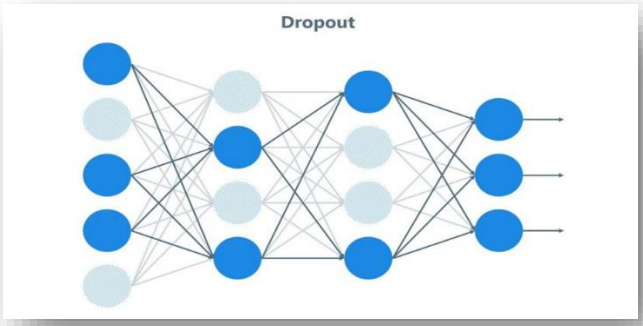
Lớp Dense

- Lớp Dense thường được sử dụng ở cuối của mạng CNN để thực hiện phân loại hoặc dự đoán các đối tượng trong ảnh sau khi đã trích xuất thông tin từ các lớp Convolutional và Pooling trước đó.
- Input: Để đưa ảnh từ các layer trước vào mạng này, buộc phải dàn phẳng bức ảnh ra thành một vector thay vì là mảng nhiều chiều như trước.
- Tại layer cuối cùng sẽ sử dụng một hàm softmax để phân loại đối tượng dựa vào vector đặc trưng đã được tính toán của các lớp trước đó.



Lớp Dropout

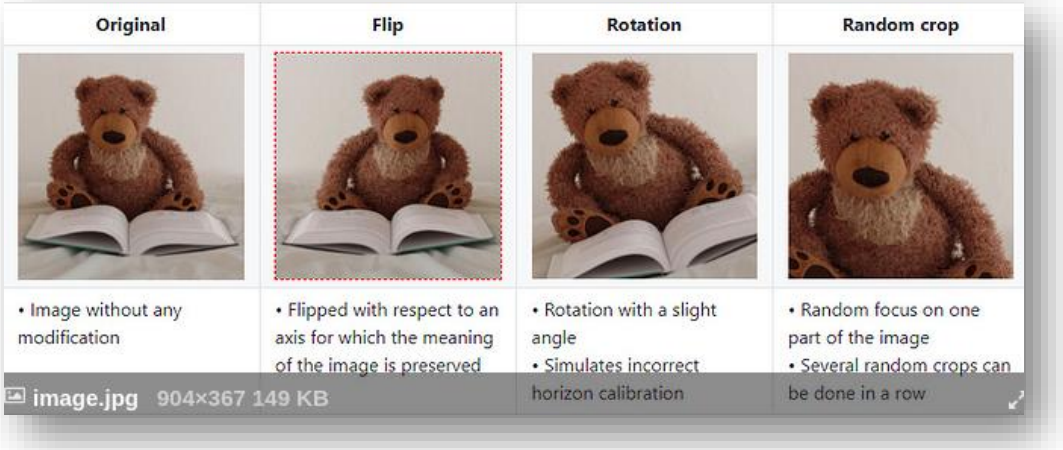
- Dropout được sử dụng để ngăn chặn overfitting trong mô hình.
- Hoạt động bằng cách loại bỏ ngẫu nhiên một số đơn vị đầu ra của lớp trước đó trong quá trình huấn luyện. Điều này giúp mô hình tránh việc học thuộc các đặc trưng của tập dữ liệu huấn luyện và cải thiện khả năng tổng quát hóa của mô hình.



2. Tăng cường dữ liệu ảnh với ImageDataGenerator

a. Tìm hiểu tăng cường dữ liệu

Hiện nay trong deep learning thì vấn đề dữ liệu có vai trò rất quan trọng. Chính vì vậy có những lĩnh vực có ít dữ liệu để cho việc train model thì rất khó để tạo ra được kết quả tốt trong việc dự đoán. Do đó người ta cần đến một kỹ thuật gọi là tăng cường dữ liệu (data augmentation) để phục vụ cho việc nếu bạn có ít dữ liệu, thì bạn vẫn có thể tạo ra được nhiều dữ liệu hơn dựa trên những dữ liệu bạn đã có. Ví dụ như hình dưới, đó là các hình được tạo ra thêm từ một ảnh gốc ban đầu.



Phương thức data aumentation cơ bản cho thị giác máy:

- ✓ Color shift (Chuyển đổi màu): Chuyển đổi màu của bức ảnh bằng cách thêm giá trị vào 3 kênh màu RGB. Việc này liên quan tới ảnh chụp đôi khi bị nhiễu --> màu bị ảnh hưởng.
- ✓ Noise addition (Thêm nhiễu): Thêm nhiễu vào bức ảnh. Nhiễu thì có nhiều loại như nhiễu ngẫu nhiên, nhiễu có mẫu, nhiễu cộng, nhiễu

nhân, nhiễu do nén ảnh, nhiễu mờ do chụp không lấy nét, nhiễu mờ do chuyển động… có thể kể hết cả ngày.

- ✓ Information loss (Mất thông tin): Một phần của bức hình bị mất. Có thể minh họa trường hợp bị che khuất.
- ✓ Constrast change (Đổi độ tương phản): thay độ tương phản của bức hình, độ bão hòa
- ✓ Geometry based: Đủ các thể loại xoay, lật, scale, padding, bóp hình, biến dạng hình,
- ✓ Color based: giống như trên, chi tiết hơn chia làm (i) tăng độ sắc nét, (ii) tăng độ sáng, (iii) tăng độ tương phản hay (iv) đổi sang ảnh negative - âm bản.
- ✓ Noise/occlusion: Chi tiết hơn các loại nhiễu, như mình kể trên còn nhiều lắm. kể hết rụng răng.
- ✓ Whether: thêm tác dụng cảu thời tiết như mưa, tuyết, sương mờ, …

b. Tăng cường dữ liệu với ImageDataGenerator

- Với class ImageDataGenerator
  Có các thuộc tính sau :

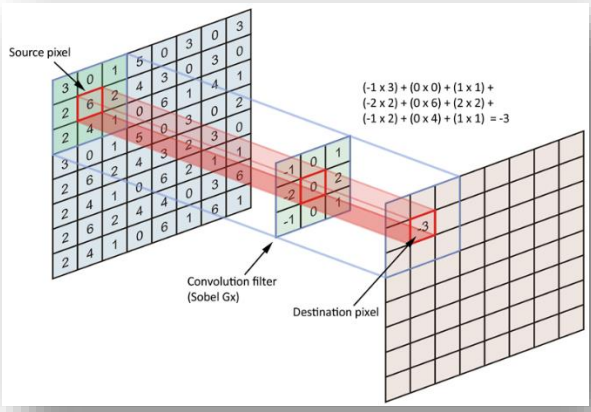
- ✓ zoom_range: thực hiện zoom ngẫu nhiên trong một phạm vi nào đó
- ✓ width_shift_range: Dịch theo chiều ngang ngẫu nhiên trong một phạm vi nào đó
- ✓ height_shift_range: Dịch ảnh theo chiều dọc trong một phạm vi nào đó
- ✓ brightness_range: Tăng cường độ sáng của ảnh trong một phạm vi nào đó.
- ✓ vertical_flip: Lật ảnh ngẫu nhiên theo chiều dọc
- ✓ rotation_range: Xoay ảnh góc tối đa là 45 độ
- ✓ shear_range: Làm méo ảnh

```
data_gen = ImageDataGenerator(rotation_range=10,
                              width_shift_range=0.1,
                              height_shift_range=0.1,
                              shear_range=0.1,
                              brightness_range=(0.3, 1.0)
                              horizontal_flip=True,
                              vertical_flip=True,
                              fill_mode='nearest'
                             )
```

3. Một số phương pháp làm mịn ảnh

Làm mịn ảnh (smoothing) là công việc quen thuộc trong xử lý ảnh. Nó giúp loại bỏ các dữ liệu không cần thiết như hạt nhiễu, làm mượt biên ảnh,…

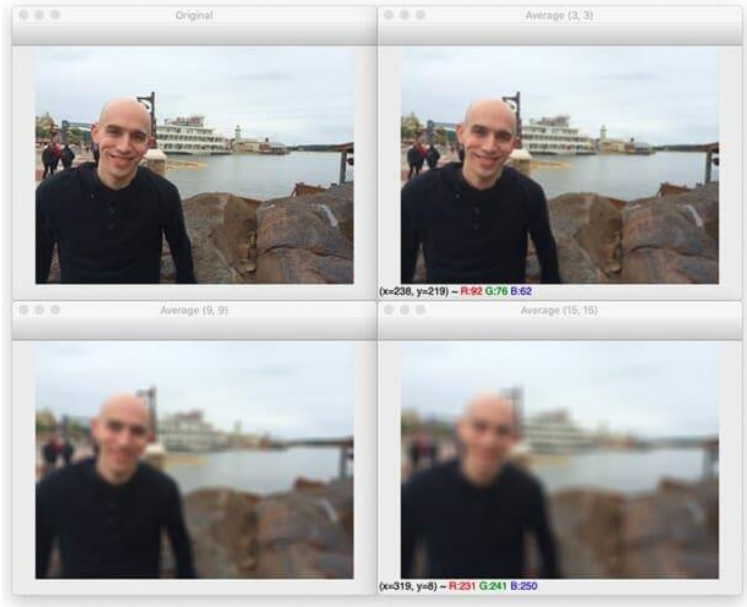
Cách hoạt động:
Cho kernel lặp qua hết các pixel trên ảnh, với anchor đặt tại pixel cần tính
Nhân giá trị của từng pixel với kernel, lấy kết quả gán lại cho pixel đó



a. Average Blur

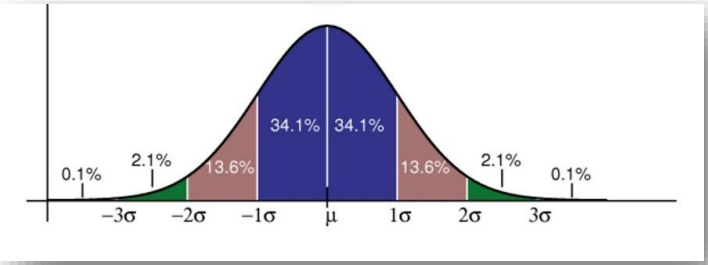
Hàm này sử dụng kernel với mỗi phần tử có giá trị là 1, còn kích thước kernel do người dùng chỉ định. Hàm blur() sẽ tính giá trị trung bình của các pixel nằm trong kernel.

$$K = \frac{1}{\texttt{ksize.width*ksize.height}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & 1 & \cdots & 1 & 1 \\ \cdots & & & & & \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix}$$



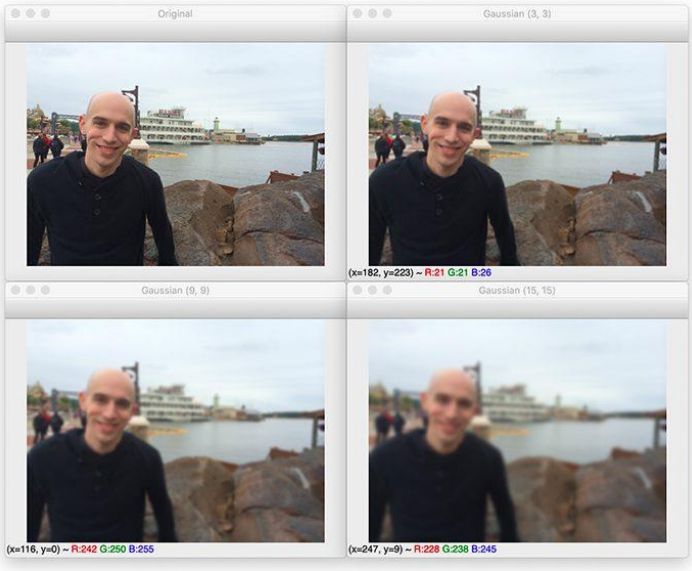
b. Gaussian blur

Phân phối chuẩn là loại phân phối phổ biến nhất hay được dùng giả định trong phân tích kĩ thuật thị trường chứng khoán và trong các loại phân tích thống kê khác. Phân phối chuẩn thông thường có hai tham số: giá trị trung bình và độ lệch chuẩn. Đối với phân phối chuẩn, 68% các quan sát nằm trong khoảng +/- độ lệch chuẩn của giá trị trung bình, 95% nằm trong +/- hai lần độ lệch chuẩn và 99,7% nằm trong + – ba lần độ lệch chuẩn.



Gaussian Blur sử dụng ma trận có phân phối chuẩn làm ma trận kernel

0	0	0	5	0	0	0
0	5	18	32	18	5	0
0	18	64	100	64	18	0
5	32	100	100	100	32	5
0	18	64	100	64	18	0
0	5	18	32	18	5	0
0	0	0	5	0	0	0



c. Median blur

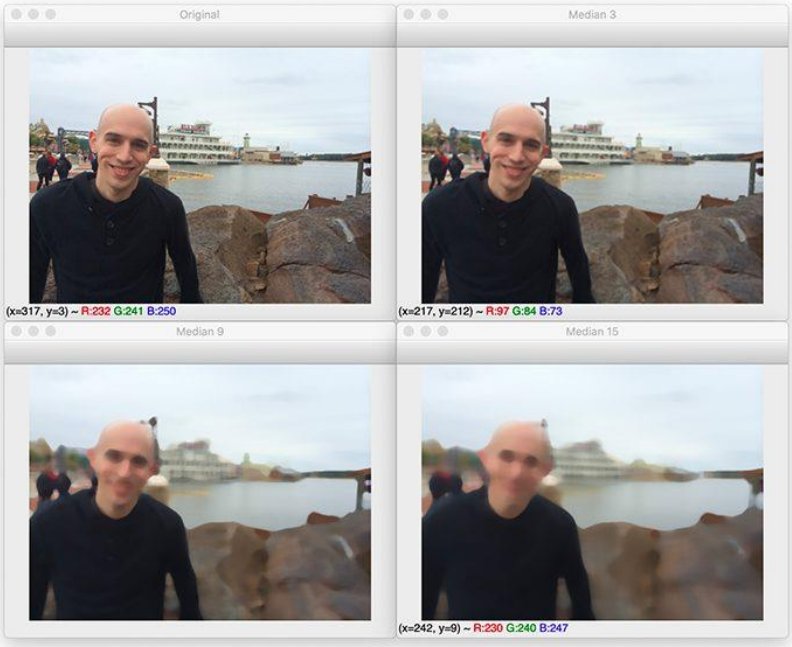
Median là giá trị của phần tử nằm giữa trong mảng đã sắp xếp.

Ví dụ:

Cho mảng {10, 11, 13, 15, 16, 23, 26} thì 15 nằm giữa, do đó giá trị median của mảng là 15

Cho mảng {10, 11, 13, 15, 16, 23, 26, 52} thì 15 và 16 nằm giữa, do đó giá trị median của mảng là (15+16/2)=15.5

Hàm medianBlur() lấy giá trị median của các pixel nằm dưới kernel rồi gán lại cho pixel đó. Hàm này dùng để lọc nhiễu rất tốt, bên dưới là ví dụ về đếm xe máy trên đường.



4. **Contours detection với openCV**

Sử dụng countours detection giúp chúng ta có thể phát hiện biên của một đối tượng.

Các bước để phát hiện và vẽ biên của một đối tượng với countours detection:

B1: Đọc ảnh và chuyển về dạng grayscale

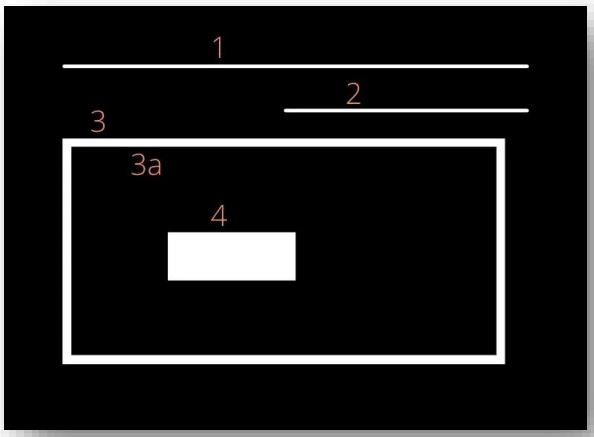
B2: Áp dụng binary thresholding cho ảnh

B3: Tìm các đường biên với hàm findContours()

- ✓ Sử dụng phương thức CHAIN_APROX_NONE: Lưu tất cả các điểm trên đường biên.
- ✓ Sử dụng phương thức CHAIN_APROX_SIMPLE: Lưu một vài điểm, phớt lờ các điểm nằm trên nằm trên đường thẳng , ngang, chéo dọc nằm giữ hai điểm đã lưu, và hàm drawConTours sẽ tự động vẽ nối các điểm đó lại.

B4: Vẽ đường biên với hàm drawContours()

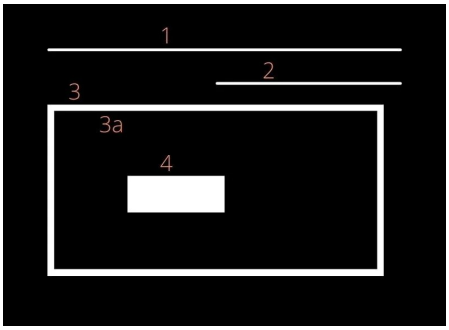
Mối quan hệ giữa các đường biên



[Next, Previous, First_Child, Parent]
Các đường biên được đánh số 1,2,3,3a,4

Next: Biểu thị đường viền tiếp theo trong hình ảnh, ở cùng mức phân cấp.
Previous: Biểu thị đường viền trước ở cùng mức phân cấp.
First_Child: Biểu thị đường viền con đầu tiên của đường viền mà chúng ta đang xem xét.
Parent: Biểu thị vị trí chỉ số của đường viền cha cho đường viền hiện tại.

RETR_LIST: không tạo mỗi quan hệ giữa các đường biên nên truy xuất tất cả.



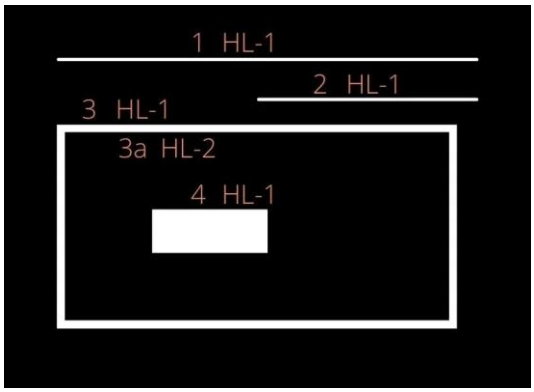
[Next, Previous, First_Child, Parent]
[[[1 -1 -1 -1]
[2 0 -1 -1]
[3 1 -1 -1]
[4 2 -1 -1]
[-1 3 -1 -1]]]

RETR_EXTERNAL: Chỉ truy xuất đường biên bên ngoài hay là không phải đường biên con

[Next, Previous, First_Child, Parent]
[[[1 -1 -1 -1]

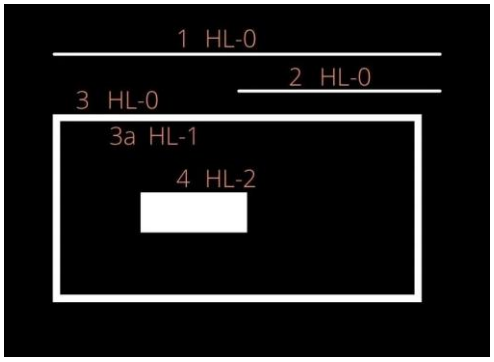
[2 0 -1 -1]
[-1 1 -1 -1]]]

RETR_CCOMP: Truy xuất tất cả các đường biên nhưng khác là nó sẽ đánh level cho các đường biên, đường biên bên trong sẽ có level bằng +1 đường biên bên ngoài.



[Next, Previous, First_Child, Parent]
[[[1 -1 -1 -1]
[3 0 2 -1]
[-1 -1 -1 1]
[4 1 -1 -1]
[-1 3 -1 -1]]]

RETR_TREE: Truy xuất tất cả các đường biên nhưng khác là nó sẽ đánh level cho các đường biên, đường biên con sẽ có level bằng +1 đường biên cha.



[Next, Previous, First_Child, Parent]
[[[3 -1 1 -1]
[-1 -1 2 0]
[-1 -1 -1 1]
[4 0 -1 -1]
[-1 3 -1 -1]]]

III. XÂY DỰNG MÔ HÌNH

1. Dữ liệu

Dữ liệu để xây dựng mô hình được lấy từ: https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection bao gồm 253 ảnh (155 ảnh thể hiện là có khối u, 98 ảnh còn lại là không có khối u).

Để có nhiều ảnh hơn cho việc training cũng như đảm bảo dữ liệu được cân bằng chúng em có sử dụng phương tăng cường dữ liệu ảnh.

```
data_gen = ImageDataGenerator(rotation_range=10,
                              width_shift_range=0.1,
                              height_shift_range=0.1,
                              shear_range=0.1,
                              brightness_range=(0.3, 1.0),
                              horizontal_flip=True,
                              vertical_flip=True,
                              fill_mode='nearest'
                             )
```

```
augment_data(file_dir=yes_path, n_generated_samples=6, save_to_dir=augmented_data_path+'yes')
augment_data(file_dir=no_path, n_generated_samples=9, save_to_dir=augmented_data_path+'no')

end_time = time.time()
```

Như vậy sau khi tăng cường dữ liệu tập ảnhd được tăng cường bao gồm 2065 ảnh (1085 ảnh thể hiện có khối u, 980 ảnh còn lại thể hiện không có khối u).

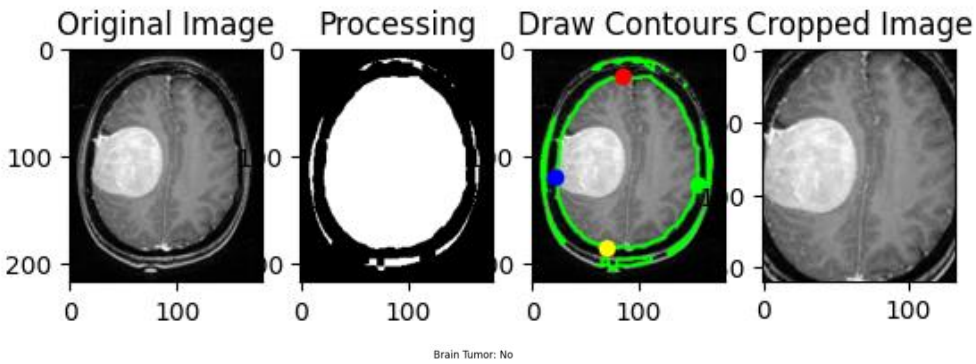
2. **Tiền xử lí**

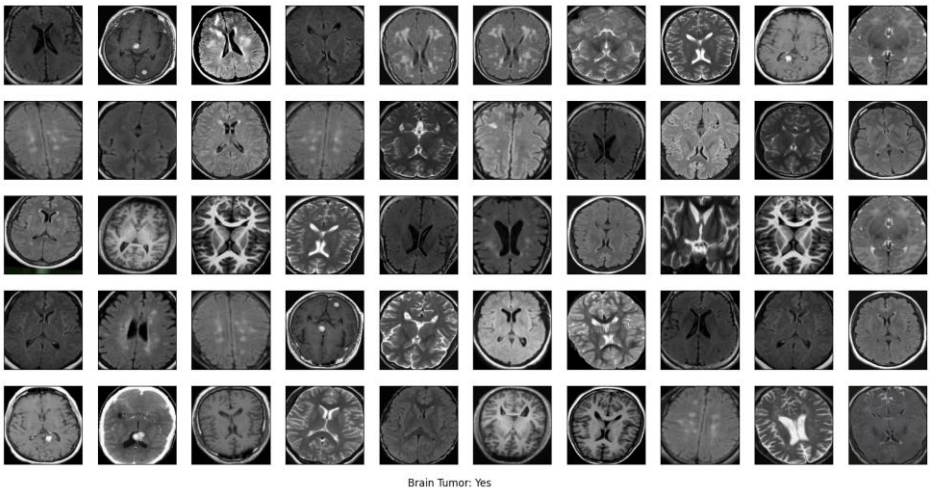
Để ảnh thể hiện các đặc trưng chính và giảm thời gian tính toán khi đưa vào training thì ảnh cần được qua bước tiền xử lí.

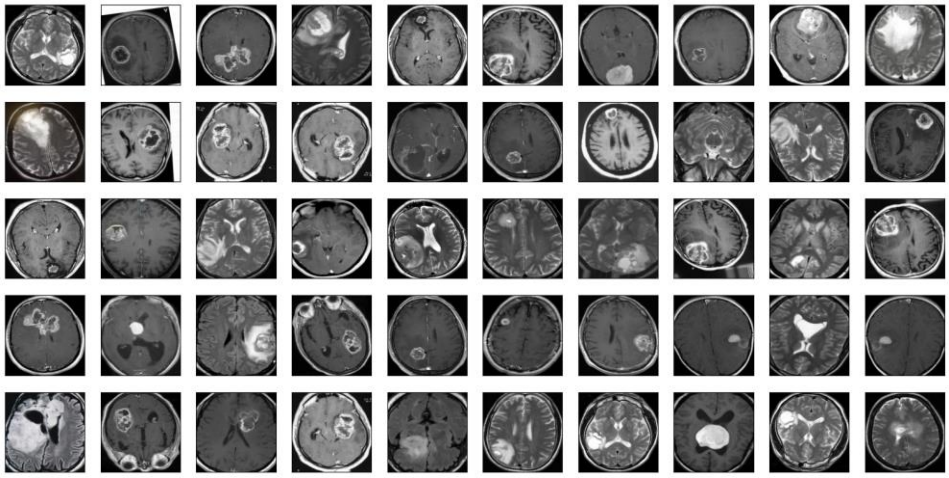
Ở dây chúng xe tiền xử lí ảnh bằng phương pháp crop các đường biên của ảnh.

Để thực hiện điều đó có những bước làm sau:

- ✓ Đọc ảnh
- ✓ Chuyển sang ảnh xám
- ✓ Lọc nhiễu
- ✓ Thực hiện co và dãn nhằm xóa các bánh răng trong ảnh
- ✓ Chuyển sang ảnh nhị phân
- ✓ Tìm kiếm các đường biên
- ✓ Tìm kiếm đường biên lớn nhất
- ✓ Tìm kiếm các đỉnh cực tiểu hoặc đại dựa trên chiều x/y trên đường biên
- ✓ Thượng hiện crop bằng các điểm đó







3. Chia tập dữ liệu

```python
def split_data(X, y, test_size=0.2):

    X_train, X_test_val, y_train, y_test_val = train_test_split(X, y, test_size=test_size)
    X_test, X_val, y_test, y_val = train_test_split(X_test_val, y_test_val, test_size=0.5)

    return X_train, y_train, X_val, y_val, X_test, y_test


X_train, y_train, X_val, y_val, X_test, y_test = split_data(X, y, test_size=0.3)
```

Tập training : 70% -> 1445 mẫu (P: 757, N:688)
Tập validation: 15% -> 310 mẫu (P: 154, N: 145)
Tập testing: 15% -> 310 mẫu (P: 163, N:147)

4. Xây dựng mô hình

```python
def build_model(input_shape):

    X_input = Input(input_shape)
    X = ZeroPadding2D((2, 2))(X_input)
    X = Conv2D(32, (7, 7), strides = (1, 1), name = 'conv0')(X)
    X = BatchNormalization(axis = 3, name = 'bn0')(X)
    X = Activation('relu')(X)     "relu": Unknown word.
    X = MaxPooling2D((4, 4), name='max_pool0')(X)
    X = MaxPooling2D((4, 4), name='max_pool1')(X)
    X = Flatten()(X) #
    X = Dense(1, activation='sigmoid', name='fc')(X)
    model = Model(inputs = X_input, outputs = X, name='BrainDetectionModel')

    return model
```

Mô hình được xây dựng bao gồm:
Lớp padding: để các đặc trưng ở các cạnh trong ảnh được duyệt qua trong quá trình training nhiều hơn thì chúng em thêm lớp padding (2,2)
Lớp conv2D: xây dựng lớp tích chập có 32 kenel kích thước 7x7 và bước nhảy là 1.
Lớp BatchNormalization: Sau khi chập xong, thì ta chuẩn hóa các giá trị về dạng phân phối chuẩn.
Sau khi chuẩn hóa nó sẽ qua hàm active function là relu để nó phi tuyến.
Lớp MaxPooling: Thực hiện qua 2 lớp maxpooling để lấy những giá trị đặc trưng nhất.
Làm phẳng dữ liệu nhiều chiều sang 1 chiều qua lớp Flatten
Fully connect với một node và dùng hàm sigmod để đưa ra kết quả cuối cùng.

```
model.summary()
```

```
Model: "BrainDetectionModel"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 240, 240, 3)]     0

 zero_padding2d (ZeroPaddin  (None, 244, 244, 3)       0
 g2D)

 conv0 (Conv2D)              (None, 238, 238, 32)      4736

 bn0 (BatchNormalization)    (None, 238, 238, 32)      128

 activation (Activation)     (None, 238, 238, 32)      0

 max_pool0 (MaxPooling2D)    (None, 59, 59, 32)        0

 max_pool1 (MaxPooling2D)    (None, 14, 14, 32)        0

 flatten (Flatten)           (None, 6272)              0

 fc (Dense)                  (None, 1)                 6273

=================================================================
Total params: 11137 (43.50 KB)
Trainable params: 11073 (43.25 KB)
Non-trainable params: 64 (256.00 Byte)
_________________________________________________________________
```

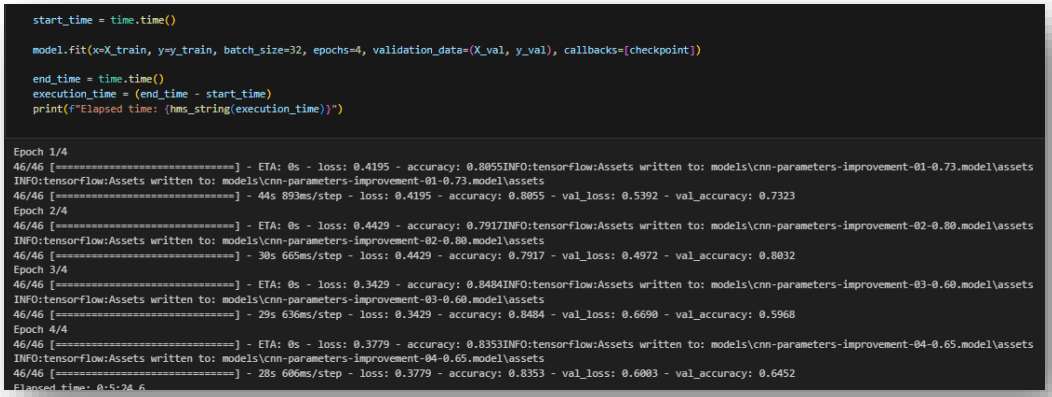
IV. THỰC HIỆN THÍ NGHIỆM

1. Train model

```
start_time = time.time()

# model.fit(x=X_train, y=y_train, batch_size=32, epochs=3, validation_data=(X_val, y_val), callbacks=[tensorboard, checkpoint])    "tensorboard": Unknown word.
model.fit(x=X_train, y=y_train, batch_size=32, epochs=3, validation_data=(X_val, y_val), callbacks=[checkpoint])

end_time = time.time()
execution_time = (end_time - start_time)
print(f"Elapsed time: {hms_string(execution_time)}")
```

```
Epoch 1/3
WARNING:tensorflow:From C:\Users\huu\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\keras\src\utils

WARNING:tensorflow:From C:\Users\huu\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\keras\src\engin

46/46 [==============================] - ETA: 0s - loss: 0.9952 - accuracy: 0.5855INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-01-0.55.model\assets
INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-01-0.55.model\assets
46/46 [==============================] - 93s 1s/step - loss: 0.9952 - accuracy: 0.5855 - val_loss: 0.6990 - val_accuracy: 0.5548
Epoch 2/3
46/46 [==============================] - ETA: 0s - loss: 0.5424 - accuracy: 0.7308INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-02-0.61.model\assets
INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-02-0.61.model\assets
46/46 [==============================] - 28s 618ms/step - loss: 0.5424 - accuracy: 0.7308 - val_loss: 0.6523 - val_accuracy: 0.6129
Epoch 3/3
46/46 [==============================] - ETA: 0s - loss: 0.4689 - accuracy: 0.7965INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-03-0.74.model\assets
INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-03-0.74.model\assets
46/46 [==============================] - 28s 603ms/step - loss: 0.4689 - accuracy: 0.7965 - val_loss: 0.5739 - val_accuracy: 0.7387
Elapsed time: 0:3:26.8
```

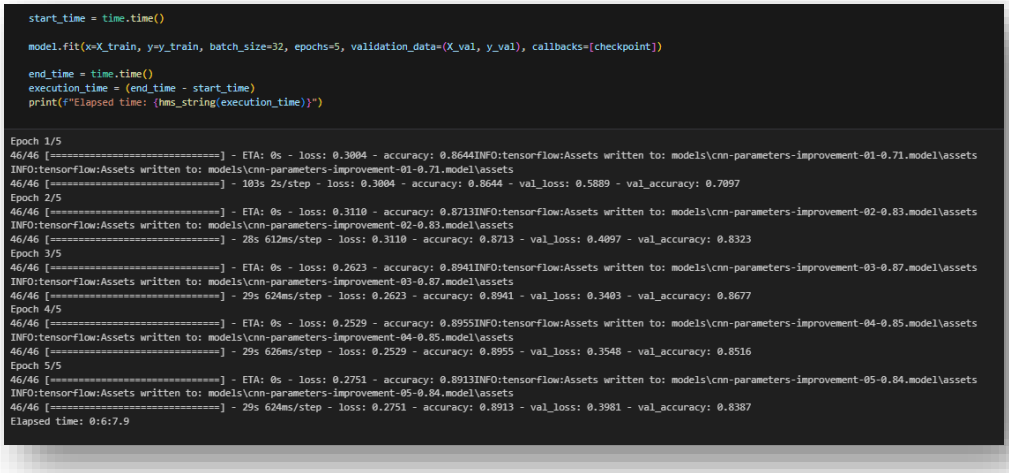
Train mô hình với 3 epoch, batch_size = 32



```
start_time = time.time()

model.fit(x=X_train, y=y_train, batch_size=32, epochs=4, validation_data=(X_val, y_val), callbacks=[checkpoint])

end_time = time.time()
execution_time = (end_time - start_time)
print(f"Elapsed time: {hms_string(execution_time)}")
```

```
Epoch 1/4
46/46 [==============================] - ETA: 0s - loss: 0.4195 - accuracy: 0.8055INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-01-0.73.model\assets
INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-01-0.73.model\assets
46/46 [==============================] - 44s 893ms/step - loss: 0.4195 - accuracy: 0.8055 - val_loss: 0.5392 - val_accuracy: 0.7323
Epoch 2/4
46/46 [==============================] - ETA: 0s - loss: 0.4429 - accuracy: 0.7917INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-02-0.80.model\assets
INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-02-0.80.model\assets
46/46 [==============================] - 30s 665ms/step - loss: 0.4429 - accuracy: 0.7917 - val_loss: 0.4972 - val_accuracy: 0.8032
Epoch 3/4
46/46 [==============================] - ETA: 0s - loss: 0.3429 - accuracy: 0.8484INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-03-0.60.model\assets
INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-03-0.60.model\assets
46/46 [==============================] - 29s 636ms/step - loss: 0.3429 - accuracy: 0.8484 - val_loss: 0.6690 - val_accuracy: 0.5968
Epoch 4/4
46/46 [==============================] - ETA: 0s - loss: 0.3779 - accuracy: 0.8353INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-04-0.65.model\assets
INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-04-0.65.model\assets
46/46 [==============================] - 28s 606ms/step - loss: 0.3779 - accuracy: 0.8353 - val_loss: 0.6003 - val_accuracy: 0.6452
Elapsed time: 0:5:24.6
```

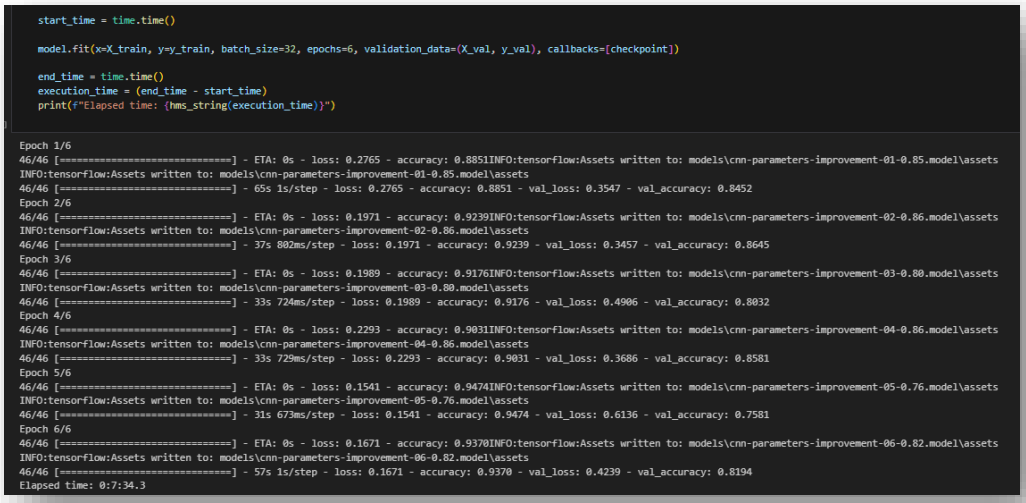
Train mô hình với 4 epoch, batch_size = 32



```
start_time = time.time()

model.fit(x=X_train, y=y_train, batch_size=32, epochs=5, validation_data=(X_val, y_val), callbacks=[checkpoint])

end_time = time.time()
execution_time = (end_time - start_time)
print(f"Elapsed time: {hms_string(execution_time)}")
```

```
Epoch 1/5
46/46 [==============================] - ETA: 0s - loss: 0.3004 - accuracy: 0.8644INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-01-0.71.model\assets
INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-01-0.71.model\assets
46/46 [==============================] - 103s 2s/step - loss: 0.3004 - accuracy: 0.8644 - val_loss: 0.5889 - val_accuracy: 0.7097
Epoch 2/5
46/46 [==============================] - ETA: 0s - loss: 0.3110 - accuracy: 0.8713INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-02-0.83.model\assets
INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-02-0.83.model\assets
46/46 [==============================] - 28s 612ms/step - loss: 0.3110 - accuracy: 0.8713 - val_loss: 0.4097 - val_accuracy: 0.8323
Epoch 3/5
46/46 [==============================] - ETA: 0s - loss: 0.2623 - accuracy: 0.8941INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-03-0.87.model\assets
INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-03-0.87.model\assets
46/46 [==============================] - 29s 624ms/step - loss: 0.2623 - accuracy: 0.8941 - val_loss: 0.3403 - val_accuracy: 0.8677
Epoch 4/5
46/46 [==============================] - ETA: 0s - loss: 0.2529 - accuracy: 0.8955INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-04-0.85.model\assets
INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-04-0.85.model\assets
46/46 [==============================] - 29s 626ms/step - loss: 0.2529 - accuracy: 0.8955 - val_loss: 0.3548 - val_accuracy: 0.8516
Epoch 5/5
46/46 [==============================] - ETA: 0s - loss: 0.2751 - accuracy: 0.8913INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-05-0.84.model\assets
INFO:tensorflow:Assets written to: models\cnn-parameters-improvement-05-0.84.model\assets
46/46 [==============================] - 29s 624ms/step - loss: 0.2751 - accuracy: 0.8913 - val_loss: 0.3981 - val_accuracy: 0.8387
Elapsed time: 0:6:7.9
```

Train mô hình với 5 epoch, batch_size = 32

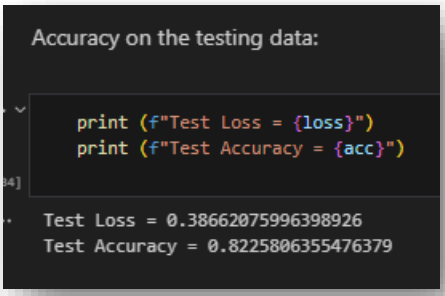


Train mô hình với 6 epoch, batch_size = 32

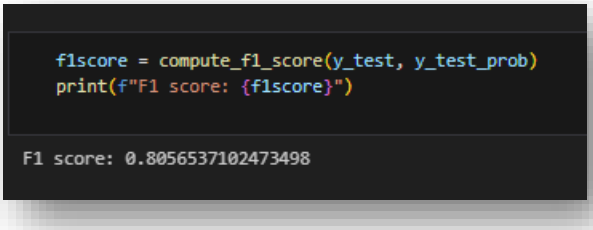
2. **Đánh giá mô hình**

   Chọn mô hình được train với 6 epoch là mô hình tốt nhất để đánh giá.

   Đánh giá trên tập test



Accuracy



F1 score

Đánh giá trên tập validation

```
f1score_val = compute_f1_score(y_val, y_val_prob)
print(f"F1 score: {f1score_val}")

F1 score: 0.8028169014084506
```

F1 score

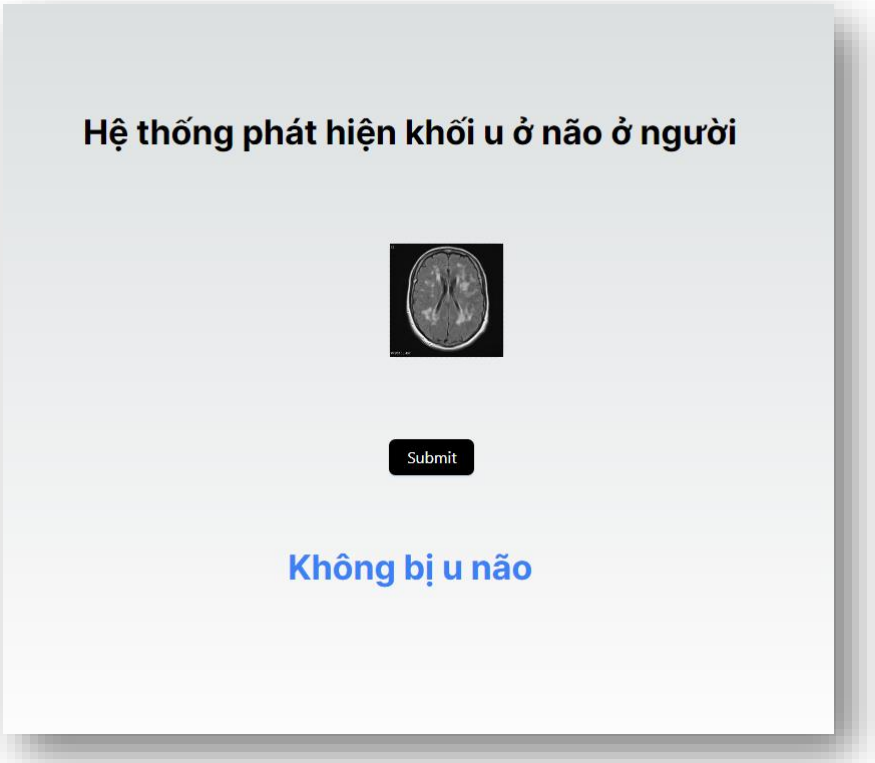
V. KẾT QUẢ VÀ HƯỚNG PHÁT TRIỂN

1. Kết quả

<!-- -->	Validation set	Test set
Accuracy	82%	82%
F1 score	0.8	0.8

Kết quả mô hình hoạt động tốt trên cả tập test và tập train

2. Nhận diện u não trên phần mềm



Kết quả nhận diện không u não chính xác

Kết quả nhận diện không u não chính xác

3. Hướng phát triển

Mặc dù đồ án đã đạt được những kết quả tích cực, nhưng vẫn còn nhiều cơ hội để cải thiện và mở rộng nghiên cứu trong tương lai. Dưới đây là một số hướng phát triển tiếp theo:

- Mở rộng bộ dữ liệu: Bộ dữ liệu còn nhỏ, cần những bộ dữ liệu lớn hơn để tạo ra mô hình dự đoán chính xác trong nhiều điều kiện khác nhau.
- Tối ưu hoá mô hình: Mô hình đang xây dựng khá đơn giản vì thiết bị máy tính còn hạn chế, trong tương lai sẽ tối ưu mô hình hơn.
- Xây dựng mô hình thấy được detect được cả vị trí khối u và độ chính xác khi nhận diện.

Kết luận, đồ án không chỉ mang lại những hiểu biết về lĩnh vực xử lý ảnh và học máy mà còn làm nền tảng cho các nghiên cứu và ứng dụng tiếp theo trong việc cải thiện chuẩn đoán u não.

VI. TÀI LIỆU THAM KHẢO

1. https://learnopencv.com/contour-detection-using-opencv-python-c/#Parent-Child-Relationship
2. https://www.tensorflow.org/api_docs/python/tf/keras
3. https://www.baeldung.com/cs/batch-normalization-cnn
4. https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection
5. https://pyimagesearch.com/2021/04/28/opencv-smoothing-and-blurring/

6. https://viblo.asia/p/tang-cuong-du-lieu-trong-deep-learning-oOVlYe4nl8W