1.6410210030

2.6410210319

# LAB_Deep Learning  Brain Tumor Classification (MRI)

## 1) explain briefly about case study

กรณีศึกษานี้เน้นการใช้เทคโนโลยีของ Machine Learning และ Artificial Intelligence เพื่อช่วยในการตรวจจับและจำแนกประเภทเนื้องอกในสมองผ่านภาพ MRI โดยการพัฒนาระบบที่ใช้ Deep Learning Algorithms เช่น Convolutional Neural Network (CNN), Artificial Neural Network (ANN), และ Transfer Learning (TL) เพื่อช่วยให้การวินิจฉัยเป็นไปอย่างแม่นยำและรวดเร็วขึ้น ซึ่งเป็นประโยชน์อย่างมากต่อหมอและผู้ป่วยที่มีเนื้องอกในสมอง

## 2) data set

Data set ชื่อ “ brain-tumor-classification-mri ”

## 3) CNN architecture

**การจัดการข้อมูล**

-การสร้างข้อมูลฝึก (training data) โดยการวนลูปผ่านโฟลเดอร์ที่เก็บรูปภาพของเนื้องอกในสมอง

```
X_train = []  # เก็บรูปภาพฝึก
y_train = []  # เก็บข้อมูลกำกับการฝึกโมเดล
image_size = 150  # ขนาดของรูปภาพ

# วนลูปผ่านแต่ละประเภทของเนื้องอกในสมองในชุดข้อมูลฝึก
for i in labels:
    folderPath = os.path.join('../input/brain-tumor-classification-mri','Training',i)
    for j in tqdm(os.listdir(folderPath)):
        img = cv2.imread(os.path.join(folderPath,j))  # เปิดอ่านรูปภาพ
        img = cv2.resize(img,(image_size, image_size))  # ปรับขนาดรูปภาพ
        X_train.append(img)  # เพิ่มรูปภาพลงใน X_train
        y_train.append(i)  # เพิ่มป้ายกำกับลงใน y_train

# วนลูปผ่านแต่ละประเภทของเนื้องอกในสมองในชุดข้อมูลทดสอบ
for i in labels:
    folderPath = os.path.join('../input/brain-tumor-classification-mri','Testing',i)
    for j in tqdm(os.listdir(folderPath)):
        img = cv2.imread(os.path.join(folderPath,j))  # เปิดอ่านรูปภาพ
        img = cv2.resize(img,(image_size,image_size))  # ปรับขนาดรูปภาพ
        X_train.append(img)  # เพิ่มรูปภาพลงใน X_train
        y_train.append(i)  # เพิ่มป้ายกำกับลงใน y_train

X_train = np.array(X_train)  # แปลงเป็น numpy array
y_train = np.array(y_train)  # แปลงเป็น numpy array
```

-การใช้ฟังก์ชัน shuffle จากไลบรารี scikit-learn เพื่อสลับลำดับของข้อมูลใน X_train และ y_train โดยการสลับลำดับนี้มักจะช่วยในการปรับปรุงประสิทธิภาพของการฝึกโมเดล โดยการสลับลำดับข้อมูลจะทำให้โมเดลได้รับข้อมูลที่มีความหลากหลายและการเรียนรู้ที่เป็นอิสระกันของข้อมูล

```
X_train, y_train = shuffle(X_train,y_train, random_state=101)
```

-ได้มีการแบ่งข้อมูล train data จะมีขนาดเป็น 90% ของข้อมูลทั้งหมดและ test data จะมีขนาดเป็น 10% ของข้อมูลทั้งหมด

```
X_train,X_test,y_train,y_test = train_test_split(X_train,y_train, test_size=0.1,random_state=101)
```

-สร้างข้อมูลกำกับใหม่ในรูปแบบ one-hot encoding ของชุดข้อมูลฝึก

1.6410210030

2.6410210319

```
y_train_new = []
for i in y_train:
    y_train_new.append(labels.index(i))
y_train = y_train_new
y_train = tf.keras.utils.to_categorical(y_train)


y_test_new = []
for i in y_test:
    y_test_new.append(labels.index(i))
y_test = y_test_new
y_test = tf.keras.utils.to_categorical(y_test)
```

-โหลดและใช้ โมเดล EfficientNetB0จากไลบรารี TensorFlow/Keras

```
effnet = EfficientNetB0(weights='imagenet',include_top=False,input_shape=(image_size,image_size,3))
```

-สร้างโมเดล neural network โดยใช้ EfficientNetB0 และเพิ่มชั้น Global Average Pooling 2D, Dropout, และ Dense เพื่อการประมวลผล และการจำแนกประเภทภาพที่เกี่ยวข้องกับเนื้องอกในสมอง โดยมี activation function เป็น softmax สำหรับการจำแนกประเภทของภาพ

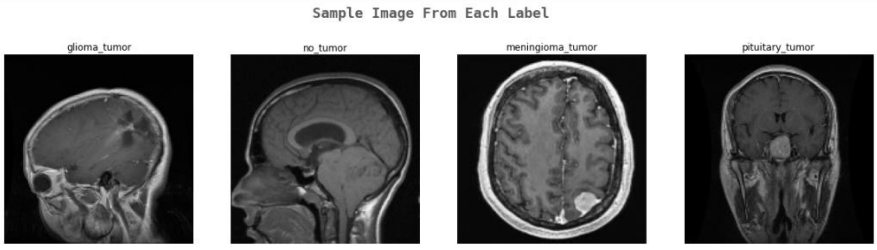
```
model = effnet.output
model = tf.keras.layers.GlobalAveragePooling2D()(model)
model = tf.keras.layers.Dropout(rate=0.5)(model)
model = tf.keras.layers.Dense(4,activation='softmax')(model)
model = tf.keras.models.Model(inputs=effnet.input, outputs = model)
```

**4) output classes**

จะประกอบไปด้วย 4 คลาส

glioma_tumor 2.no_tumor 3.meningioma_tumor 4.pituitary_tumor

```
k=0
fig, ax = plt.subplots(1,4,figsize=(20,20))
fig.text(s='Sample Image From Each Label',size=18,fontweight='bold',
             fontname='monospace',color=colors_dark[1],y=0.62,x=0.4,alpha=0.8)
for i in labels:
    j=0
    while True :
        if y_train[j]==i:
            ax[k].imshow(X_train[j])
            ax[k].set_title(y_train[j])
            ax[k].axis('off')
            k+=1
            break
        j+=1
```



1.6410210030

2.6410210319

**5) accuracy evaluation**

-กำหนดวิธีการประเมินความแม่นยำ (accuracy) โดยให้ optimizer = Adam , metrics = accuracy

```
model.compile(loss='categorical_crossentropy',optimizer = 'Adam', metrics= ['accuracy'])
```

-กำหนด epochs ของโมเดลก่อนที่จะนำไปทำนาย โดยจะมี epochs = 12 , verbose = 1 , batch_size = 32

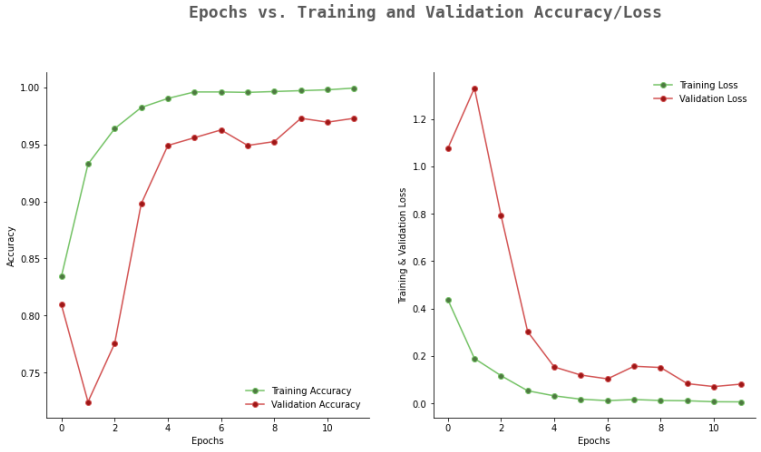
```
history = model.fit(X_train,y_train,validation_split=0.1, epochs =12, verbose=1, batch_size=32,
                    callbacks=[tensorboard,checkpoint,reduce_lr])
```

จะได้ค่า accuracy ในการทำ epochsครั้งสุดท้าย = 99% , val_loss = 8.12% , val_accuracy = 97% , loss = 0.44%

-พล็อตกราฟเพื่อแสดงค่าความแม่นยำและค่า (loss) ของการฝึกโมเดล neural network ในระหว่างรอบการฝึก (epochs)

```
filterwarnings('ignore')

epochs = [i for i in range(12)]
fig, ax = plt.subplots(1,2,figsize=(14,7))
train_acc = history.history['accuracy']
train_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']

fig.text(s='Epochs vs. Training and Validation Accuracy/Loss',size=18,fontweight='bold',
             fontname='monospace',color=colors_dark[1],y=1,x=0.28,alpha=0.8)

sns.despine()
ax[0].plot(epochs, train_acc, marker='o',markerfacecolor=colors_green[2],color=colors_green[3],
           label = 'Training Accuracy')
ax[0].plot(epochs, val_acc, marker='o',markerfacecolor=colors_red[2],color=colors_red[3],
           label = 'Validation Accuracy')
ax[0].legend(frameon=False)
ax[0].set_xlabel('Epochs')
ax[0].set_ylabel('Accuracy')

sns.despine()
ax[1].plot(epochs, train_loss, marker='o',markerfacecolor=colors_green[2],color=colors_green[3],
           label ='Training Loss')
ax[1].plot(epochs, val_loss, marker='o',markerfacecolor=colors_red[2],color=colors_red[3],
           label = 'Validation Loss')
ax[1].legend(frameon=False)
ax[1].set_xlabel('Epochs')
ax[1].set_ylabel('Training & Validation Loss')

fig.show()
```

จะได้ผลลัพธ์ดังนี้



1.6410210030

2.6410210319

-ทำการแสดง classification report ซึ่งเป็นการวัดประสิทธิภาพของโมเดลในการจำแนกประเภทต่าง ๆ ด้วยค่า precision, recall, f1-score และ support สำหรับแต่ละคลาส รวมถึงค่า accuracy ที่บอกถึงความแม่นยำโดยรวมของโมเดล

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.98 | 0.96 | 0.97 | 93 |
| 1 | 0.98 | 1.00 | 0.99 | 51 |
| 2 | 0.97 | 0.96 | 0.96 | 96 |
| 3 | 0.98 | 1.00 | 0.99 | 87 |
| accuracy | | | 0.98 | 327 |
| macro avg | 0.98 | 0.98 | 0.98 | 327 |
| weighted avg | 0.98 | 0.98 | 0.98 | 327 |

-สร้าง Heatmap ของ Confusion Matrix เพื่อแสดงการสับเปลี่ยนของคลาส (class) ระหว่างค่าที่โมเดลทำนาย (pred) และค่าที่เป็นคลาสจริง (y_test_new) ในชุดข้อมูลทดสอบ โดย Confusion Matrix