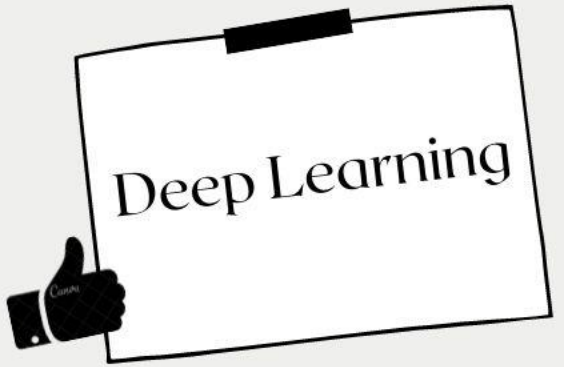
Glaucoma Detection



Load Image to np

```python
def claheAdapter(img):
    lab_img= cv2.cvtColor(img, cv2.COLOR_RGB2LAB)
    l, a, b = cv2.split(lab_img)
    clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8,8))
    clahe_img = clahe.apply(l)
    updated_lab_img2 = cv2.merge((clahe_img,a,b))
    CLAHE_img = cv2.cvtColor(updated_lab_img2, cv2.COLOR_LAB2RGB)
    return CLAHE_img
```

```python
CATEGORIES = ["Glaucoma","Normal","Others"]
TESTDIR =  "C:/Users/User/DL/dataset2/test/"
TRAINDIR =  "C:/Users/User/DL/dataset2/train/"
VALDIR = "C:/Users/User/DL/dataset2/validation/"
test_images,test_labels, train_images, train_labels,val_images,val_labels= [],[],[],[],[],[]
for category in CATEGORIES:
    path = os.path.join(VALDIR, category)
    class_num = CATEGORIES.index(category)
    for img in os.listdir(path):
        img_array = claheAdapter(cv2.imread(os.path.join(path,img))[:,:,::-1])
        cv2.resize(img_array,(100,100))
        val_images.append(img_array)
        val_labels.append(class_num)

for category in CATEGORIES:
    path = os.path.join(TESTDIR, category)
    class_num = CATEGORIES.index(category)
    for img in os.listdir(path):
        img_array = claheAdapter(cv2.imread(os.path.join(path,img))[:,:,::-1])
        cv2.resize(img_array,(100,100))
        test_images.append(img_array)
        test_labels.append(class_num)

for category in CATEGORIES:
    path = os.path.join(TRAINDIR, category)
    class_num = CATEGORIES.index(category)
    for img in os.listdir(path):
        img_array = claheAdapter(cv2.imread(os.path.join(path,img))[:,:,::-1])
        cv2.resize(img_array,(100,100))
        train_images.append(img_array)
        train_labels.append(class_num)
```

code ส่วนนี้เป็นcodeเกี่ยวกับการอ่าน image แล้วนำไปเข้าmethod claheAdapter เพื่อทำให้ภาพชัดขึ้น และนำไป resize ขนาดภาพให้เป็น 100x100 และเพิ่มเข้าไปในลิสต์ที่สร้างขึ้นในบรรทัดที่ 5

normalize & cast list to numpy

```
test_images = np.asarray(test_images)
test_labels = np.asarray(test_labels)
train_images = np.asarray(train_images)
train_labels = np.asarray(train_labels)
val_images = np.asarray(val_images)
val_labels = np.asarray(val_labels)
train_images = train_images / 255.0
test_images = test_images / 255.0
val_images = val_images / 255.0
```

code ส่วนนี้เราจะทำการเปลี่ยน data ที่เป็น list เป็น numpy.ndarray แล้ว normalize ให้ค่าสีอยู่ในช่วง 0–1

model CNN

```
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
    tf.keras.layers.Dropout(0.25),

    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
    tf.keras.layers.Dropout(0.25),

    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(3,activation='softmax')
    ])
model.compile(
    optimizer='adam',
    loss=tf.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=['accuracy'])
```

Describe

-layer 1 ,2,3 : เป็น CNN โดย kernelมีขนาด 3x3 activation เป็น relu และมีการ poolingขนาด (2,2) โดย layer 1,2 มี Dropout เป็น 0.25 และnode มีขนาด 32,64,128 ตามลำดับ layer

-layer 4 : เป็น Flatten-Dense layers เพื่อทำนาย class โดย output node มีค่าเป็น 3 เพราะเราต้องการแยกเป็น 3 classes และ activation เป็น softmax เพราะในการคำนวณ entropy ต้องตีความข้อมูล output เชิงความน่าจะเป็น

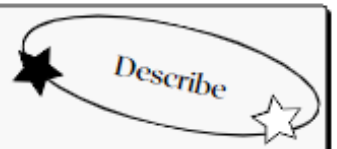
ตอน compile optimizer เป็น adam และ loss function เป็น cross entropy เพราะเราต้องการ classification

model ResNet50

```
base_model = ResNet50(weights='imagenet', include_top=False,input_shape=(100,100,3))
model = tf.keras.Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Flatten())
model.add(Dense(500,activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Dense(500,activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.25))
model.add(Dense(3,activation='softmax'))
for layer in base_model.layers:
    layer.trainable = False

model.compile(optimizer='adam',loss=tf.losses.SparseCategoricalCrossentropy(from_logits=True),metrics=['accuracy'])
```



ใช้ base model ของ ResNet50 โดย input ต้องมีขนาด 100x100 และ เรียกใช้ GlobalAveragePooling2D แล้วเพิ่ม flattern-dense layer โดย node มีค่าเท่ากับ 500 activation เป็น relu และทำการ BatchNormalization(BN) ก่อนส่งข้อมูลไปยัง layer ถัดไป เพื่อให้ค่าจาก layer อยู่ในขอบเขตที่เหมาะสม และทำการ dropout 0.25 แล้วจึงสร้าง layer ใหม่โดย node เท่ากับ 500 activation แล้ว BN อีกครั้งก่อนส่งต่อไป layer สุดท้าย โดยที่ layer สุดท้าย output node เป็น 3 เพราะแยกเป็น 3 classes ส่วนของ compile เหมือนกับ CNN

Train model

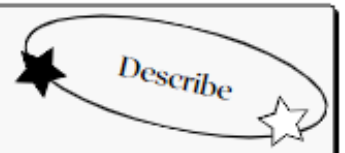
```
history = model.fit(
        train_images,
        train_labels,
        validation_data=(val_images,val_labels),
        epochs=20
```

Describe

ส่ง train_images , train_labels สำหรับการtrain model และส่ง val_images,val_labels สำหรับ validate โดยตั้งค่า epochs 20 รอบ

Plot accuracy graph

```
plt.plot(history.history['accuracy'], label = 'accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0,1])
plt.legend(loc='lower right')
```



ตัวแปล history เก็บผลจากการ train model เราจึงนำมา plot graph ความสัมพันธ์ระหว่าง accuracy และ Epoch ของ accuracy และ validation accuracy

Confusion Matrix

```
probability_model = tf.keras.Sequential([
  model,
  tf.keras.layers.Softmax()
])

predictions = probability_model.predict(test_images)

PreLabel = []
for i in range(0,len(predictions)):
    PreLabel.append(np.argmax(predictions[i]))

plt.figure(figsize=(8, 8))
sns.heatmap(confusion_matrix(PreLabel, test_labels), annot=True, fmt="d", cbar = False, cmap = plt.cm.Blues)
plt.savefig('confusion_ResNet50.jpg')
```

Describe

เราทำการ predict ด้วย model ที่เราเทรนมา โดยส่ง test_images เข้าไปให้ model ทำนาย แล้วสร้าง list เพื่อมาเก็บ ค่าที่ model ทำนาย เป็น 0,1,2 แล้วนำไปplot confusion metrix โดยส่งค่าที่ model ทำนาย PreLabel กับ คำตอบที่ถูก นั่นคือ test_labels