30 May.

# 整合殘差塊與注意力機制的Y-net模型用於魚種分類

企管碩一 M1102930 江明臻

# 動機與目的

- 海鮮是多種菜餚的主要成分
- 聯合國糧食及農業組織：
全球每年魚類消費量超過 20 公斤

食品行業在有限的時間內
提供優質的產品

- 專家通過過去的經驗檢測變質
- 自動化系統可以快速檢測變質，
並將降低供應商的經濟負擔

# 資料來源

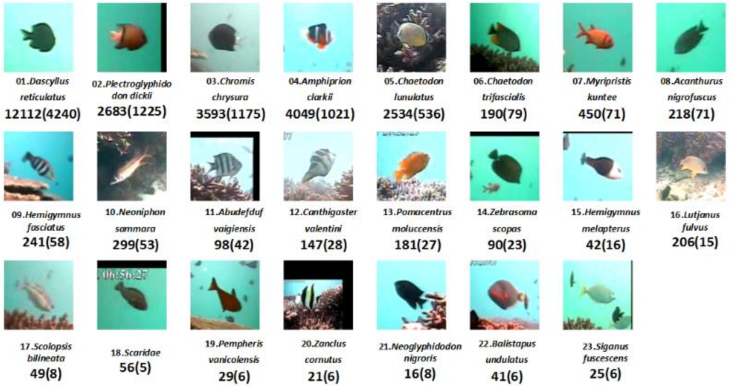
使用文獻：

Ulucan, O., Karakaya, D., & Turkan, M. (2020, October). A large-scale dataset for fish segmentation and classification. *In 2020 Innovations in Intelligent Systems and Applications Conference (ASYU) (pp. 1-5). IEEE.*

數據集：Fish4Knowledge

https://homepages.inf.ed.ac.uk/rbf/Fish4Knowledge/GROUNDTRUTH/RECOG/

# 資料來源



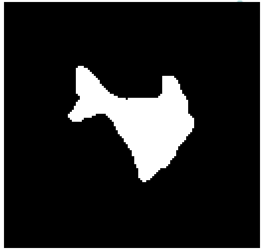

| ID.species | Detection # | Trajectory # | Fish image | Mask image |
|---|---|---|---|---|
| 01.*Dascyllus reticulatus* | 12112 | 4240 | fish_01.tar checkSum | mask_01.tar checkSum |
| 02.*Plectroglyphidodon dickii* | 2683 | 1225 | fish_02.tar checkSum | mask_02.tar checkSum |
| 03.*Chromis chrysura* | 3593 | 1175 | fish_03.tar checkSum | mask_03.tar checkSum |
| 04.*Amphiprion clarkii* | 4049 | 1021 | fish_04.tar checkSum | mask_04.tar checkSum |
| 05.*Chaetodon lunulatus* | 2534 | 536 | fish_05.tar checkSum | mask_05.tar checkSum |
| 06.*Chaetodon trifascialis* | 190 | 79 | fish_06.tar checkSum | mask_06.tar checkSum |
| 07.*Myripristis kuntee* | 450 | 71 | fish_07.tar checkSum | mask_07.tar checkSum |
| 08.*Acanthurus nigrofuscus* | 218 | 71 | fish_08.tar checkSum | mask_08.tar checkSum |
| 09.*Hemigymnus fasciatus* | 241 | 58 | fish_09.tar checkSum | mask_09.tar checkSum |
| 10.*Neoniphon sammara* | 299 | 53 | fish_10.tar checkSum | mask_10.tar checkSum |
| 11.*Abudefduf vaigiensis* | 98 | 42 | fish_11.tar checkSum | mask_11.tar checkSum |
| 12.*Canthigaster valentini* | 147 | 28 | fish_12.tar checkSum | mask_12.tar checkSum |
| 13.*Pomacentrus moluccensis* | 181 | 27 | fish_13.tar checkSum | mask_13.tar checkSum |
| 14.*Zebrasoma scopas* | 90 | 23 | fish_14.tar checkSum | mask_14.tar checkSum |
| 15.*Hemigymnus melapterus* | 42 | 16 | fish_15.tar checkSum | mask_15.tar checkSum |
| 16.*Lutjanus fulvus* | 206 | 15 | fish_16.tar checkSum | mask_16.tar checkSum |
| 17.*Scolopsis bilineata* | 49 | 8 | fish_17.tar checkSum | mask_17.tar checkSum |
| 18.*Scaridae* | 56 | 5 | fish_18.tar checkSum | mask_18.tar checkSum |
| 19.*Pempheris vanicolensis* | 29 | 6 | fish_19.tar checkSum | mask_19.tar checkSum |
| 20.*Zanclus cornutus* | 21 | 6 | fish_20.tar checkSum | mask_20.tar checkSum |
| 21.*Neoglyphidodon nigroris* | 16 | 8 | fish_21.tar checkSum | mask_21.tar checkSum |
| 22.*Balistapus undulatus* | 41 | 6 | fish_22.tar checkSum | mask_22.tar checkSum |
| 23.*Siganus fuscescens* | 25 | 6 | fish_23.tar checkSum | mask_23.tar checkSum |

# 資料來源

## image檔案

| | | |
|---|---|---|
| fish_003470295337_23990.png | 2013/9/18 上午 06:42 | PNG 檔案 |
| fish_003470725337_23994.png | 2013/9/18 上午 06:42 | PNG 檔案 |
| fish_003470175337_23531.png | 2013/9/18 上午 06:42 | PNG 檔案 |
| fish_003470035337_23146.png | 2013/9/18 上午 06:42 | PNG 檔案 |
| fish_347006533719_23148.png | 2013/9/18 上午 06:42 | PNG 檔案 |
| fish_003470725337_22773.png | 2013/9/18 上午 06:42 | PNG 檔案 |

## mask檔案



| | | |
|---|---|---|
| mask_003470295337_23990.png | 2013/9/18 上午 06:42 | PNG 檔案 |
| mask_003470725337_23994.png | 2013/9/18 上午 06:42 | PNG 檔案 |
| mask_003470175337_23531.png | 2013/9/18 上午 06:42 | PNG 檔案 |
| mask_003470035337_23146.png | 2013/9/18 上午 06:42 | PNG 檔案 |
| mask_347006533719_23148.png | 2013/9/18 上午 06:42 | PNG 檔案 |
| mask_003470725337_22773.png | 2013/9/18 上午 06:42 | PNG 檔案 |

# 資料說明



**Myripristis kuntee**
**康德鋸鱗魚**
樣本數：450
標籤：0
圖像：
fish_1~fish_450
遮罩：
mask_1 ~ mask_450

**Pomacentrus moluccensis**
**摩鹿加雀鯛**
樣本數：180
標籤：1
圖像：
fish_451~fish_631
遮罩：
mask_451~mask_631

**Pempheris vanicolensis**
**黑緣擬金眼鯛**
樣本數：29
標籤：2
圖像：
fish_632~fish_660
遮罩：
mask_632~mask_660

# 程式架構

## 基本架構

使用課堂所學之Ynet架構，
包含殘差塊與注意力機制

## 建立標籤

使用第一個作業的標籤方式

# 程式架構

點我進入COLAB

## 1.匯入資料

### 1.1 連接google雲端硬碟

✓ 7秒

```
[1] !pip install scikit-learn

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/sim
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from s
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages
```

✓ 2分鐘

```
[2] import os
    from google.colab import drive
    drive.mount('/content/gdrive/')    #掛載Google雲端硬碟

Mounted at /content/gdrive/
```

# 程式架構

點我進入COLAB

## 1.2 載入相關套件

[3] 0 秒

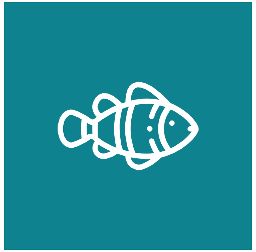
```
import numpy as np
import os
from PIL import Image
import matplotlib.pyplot as plt
import cv2
```

# 程式架構

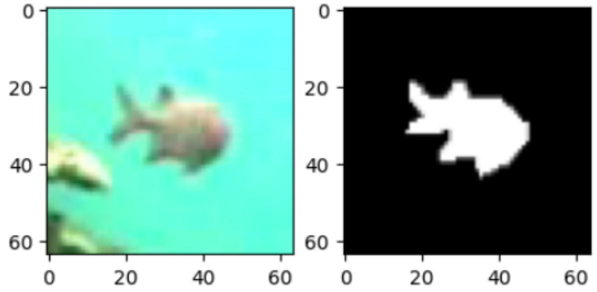
點我進入COLAB

## 1.3 設定檔案路徑

```
[4]  data_file = '/content/gdrive/My Drive/final_project/'
     image_file = data_file + 'image/'
     mask_file = data_file + 'mask/'

     zero_img_file = image_file + 'fish_07/'
     one_img_file = image_file + 'fish_13/'
     two_img_file = image_file + 'fish_19/'

     zero_mask_file = mask_file + 'mask_07/'
     one_mask_file = mask_file + 'mask_13/'
     two_mask_file = mask_file + 'mask_19/'
```



```
[5]  # 確認圖像與遮罩配對
     data_name = "_50.png"  # 手動改圖像檔名
     img_path = os.path.join(zero_img_file + str("fish") + data_name)  # 設定圖檔路徑
     img = cv2.imread(img_path)
     img = cv2.resize(img, (64,64))
     mask_path = os.path.join(zero_mask_file + str("mask") + data_name)
     mask = cv2.imread(mask_path)
     mask = cv2.resize(mask, (64,64))
     plt.figure(figsize=(5, 8))
     plt.subplot(1, 2, 1)  # (rows, columns, index)
     plt.imshow(cv2.cvtColor (img, cv2.COLOR_BGR2RGB))
     plt.subplot(1, 2, 2)
     plt.imshow(mask)

<matplotlib.image.AxesImage at 0x7f86ac4fb2b0>
```



# 程式架構

點我進入COLAB

## 2.讀入資料

### 2.1 讀入image與相對的mask

```
[6] import numpy as np
    import tensorflow as tf
    def load_data(folder, image):
        File_List = os.listdir(folder)
        File_List.sort(key = str.lower)
        for filename in File_List:
            img = tf.keras.preprocessing.image.load_img(folder + filename,
                    target_size = (64,64)) # 將照片讀進filename並將圖片大小正規化
            img_resize = np.array(img)
            image.append(img_resize)
```

# 程式架構

點我進入COLAB

## 2.2 建立標籤

### 康德鋸鱗魚(450)->label:0

**Myripristis kuntee**
**康德鋸鱗魚**
樣本數：450
標籤：0
圖像：
fish_1~fish_450
遮罩：
mask_1 ~ mask_450

```
# 匯入康德鋸鱗魚(450)圖像資料夾的圖片->label:0
zeroimg = []
load_data(zero_img_file, zeroimg)
zeroimg = np.array(zeroimg)
print('Shape of Myripristis kuntee Images:', zeroimg.shape)
# Shape of Myripristis kuntee Images: (450, 64, 64, 3) # (張數,長,寬,RGB)
zeromask = []
load_data(zero_mask_file, zeromask)
zeromask = np.array(zeromask)
print('Shape of Myripristis kuntee Images:', zeromask.shape)
# Shape of Myripristis kuntee Images: (450, 64, 64, 3) # (張數,長,寬,RGB)
zero_label = np.zeros(zeroimg.shape[0]) # 建立450張正常的圖像標籤為0
print(zero_label)
zero_label = np.reshape(zero_label, (zero_label.shape[0], 1))
```

```
Shape of Myripristis kuntee Images: (450, 64, 64, 3)
Shape of Myripristis kuntee Images: (450, 64, 64, 3)
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

# 程式架構

點我進入COLAB

▾ 摩鹿加雀鯛(181)->label:1

```
[ ] # 匯入摩鹿加雀鯛(181)圖像資料夾的圖片->label:1
    oneimg = []
    load_data(one_img_file, oneimg)
    oneimg = np.array(oneimg)
    print('Shape of Pomacentrus moluccensis Images:', oneimg.shape)
    # Shape of Pomacentrus moluccensis Images: (181, 64, 64, 3) # (張數,長,寬,RGB)
    onemask = []
    load_data(one_mask_file, onemask)
    onemask = np.array(onemask)
    print('Shape of Pomacentrus moluccensis Images:', onemask.shape)
    # Shape of Pomacentrus moluccensis Images: (181, 64, 64, 3) # (張數,長,寬,RGB)
    one_label = np.ones(oneimg.shape[0]) # 建立181張正常的圖像標籤為1
    print(one_label)
    one_label = np.reshape(one_label, (one_label.shape[0], 1))
```

```
Shape of Pomacentrus moluccensis Images: (181, 64, 64, 3)
Shape of Pomacentrus moluccensis Images: (181, 64, 64, 3)
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

**Pomacentrus moluccensis**
**摩鹿加雀鯛**

樣本數：180
標籤：1
圖像：
fish_451~fish_631
遮罩：
mask_451~mask_631

# 程式架構

點我進入COLAB

Pempheris vanicolensis
黑緣擬金眼鯛

樣本數：29
標籤：2
圖像：
fish_632~fish_660
遮罩：
mask_632~mask_660

▾ 黑緣擬金眼鯛(29)->label:2

```
[ ] # 匯入黑緣擬金眼鯛(29)圖像資料夾的圖片->label:2
    twoimg = []
    load_data(two_img_file, twoimg)
    twoimg = np.array(twoimg)
    print('Shape of Pempheris vanicolensis Images:', twoimg.shape)
    # Shape of normal Images: (29, 64, 64, 3) # (張數,長,寬,RGB)
    twomask = []
    load_data(two_mask_file, twomask)
    twomask = np.array(twomask)
    print('Shape of Pempheris vanicolensis Images:', twomask.shape)
    # Shape of normal Images: (29, 64, 64, 3) # (張數,長,寬,RGB)
    two_label = np.full((twoimg.shape[0]),2) # 建立450張正常的圖像標籤為0
    print(two_label)
    two_label = np.reshape(two_label, (two_label.shape[0], 1))
```

```
Shape of Pempheris vanicolensis Images: (29, 64, 64, 3)
Shape of Pempheris vanicolensis Images: (29, 64, 64, 3)
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

# 程式架構

點我進入COLAB

```
[10] # 將三種資料集合在一起，為同一欄位
     Total_img = []
     Total_img = np.concatenate((zeroimg, oneimg, twoimg), axis=0) # 將三種圖像合在一起，為同一欄位
     print('Total_Image: ',Total_img.shape)
     Total_mask = []
     Total_mask = np.concatenate((zeromask, onemask, twomask), axis=0) # 將三種遮罩合在一起，為同一欄位
     print('Total_Mask: ',Total_mask.shape)
     Total_label = []
     Total_label = np.concatenate((zero_label, one_label, two_label)) # 將三種標題合在一起，為同一欄位
     print('Total_label: ',Total_label.shape)
```

```
Total_Image: (660, 64, 64, 3)
Total_Mask: (660, 64, 64, 3)
Total_label: (660, 1)
```

# 程式架構

點我進入COLAB

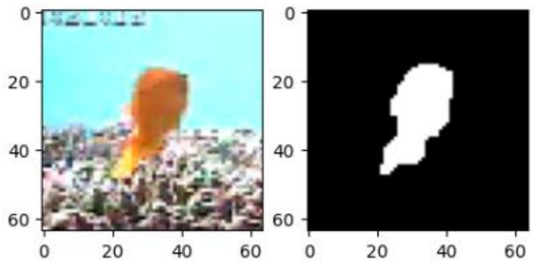
## 2.3 影像正規化與標籤處理

```
# one-hot encoding->loss函數改為'classification':'categorical_crossentropy'
# from keras.utils import to_categorical
# Total_label = to_categorical(Total_label, num_classes = 3)
```

```
# 影像正規化
image_norm = Total_img / 255 # 255:最大像素質
mask_norm = Total_mask / 255 # 255:最大像素質
```

```
# 確認圖像img、遮罩mask與標籤label配對
i = 500
plt.figure(figsize=(5, 8))
plt.subplot(1, 2, 1)# (rows, columns, index)
plt.imshow(Total_img[i])
plt.subplot(1, 2, 2)# (rows, columns, index)
plt.imshow(Total_mask[i])
print("label:",Total_label[i])
```

```
label: [1.]
```



# 程式架構

點我進入COLAB

## 3.將資料分成Train,Valid與Test資料集

```
[ ] from sklearn.model_selection import train_test_split
    # 任務一: 影像切割 -> 資料: 圖像 & 遮罩
    # 區分 x 資料 (image) 與 y 資料 (mask) 的訓練集、測試集與預測集
    x_train_val, x_test, y_train_val_mask, y_test_mask = train_test_split(image_norm, mask_norm, test_size=0.1, random_state=11)
    x_train, x_val, y_train_mask, y_val_mask = train_test_split(x_train_val, y_train_val_mask, test_size=0.1, random_state=11)
    # 任務二: 影像分類 -> 資料: 圖像 & 標籤
    # 區分 x 資料 (image) 與 y 資料 (label) 的訓練集、測試集與預測集
    _, _, y_train_val_label, y_test_label = train_test_split(image_norm, Total_label, test_size=0.1, random_state=11)
    # _, _, y_train_val_label, y_test_label=x_train_val, x_test, y_train_val_label, y_test_label
    _, _, y_train_label, y_val_label = train_test_split(x_train_val, y_train_val_label, test_size=0.1,random_state=11)
    # _, _, y_train_label, y_val_label=x_train, x_val, y_train_label, y_val_label
    # 確認數量
    print(x_train.shape)
    print(y_train_mask.shape)
    print(y_train_label.shape)
    print(x_val.shape)
    print(y_val_mask.shape)
    print(y_val_label.shape)
    print(x_test.shape)
    print(y_test_mask.shape)
    print(y_test_label.shape)
```

```
(534, 64, 64, 3)
(534, 64, 64, 3)
(534, 1)
(60, 64, 64, 3)
(60, 64, 64, 3)
(60, 1)
(66, 64, 64, 3)
(66, 64, 64, 3)
(66, 1)
```

# Train,Valid與Test資料集大小

Train 佔資料集比例：0.8

Valid 佔資料集比例：0.09
(660-66)*10%

Test 佔資料集比例：0.1

# 程式架構

點我進入COLAB

## 4. 建立模型

### attention模塊

```
# attention模塊
def attention(g, s, num_filters):
    att_g = Conv2D(num_filters, 1, padding = "same")(g)
    att_g = BatchNormalization()(att_g)

    att_s = Conv2D(num_filters, 1, padding = "same")(s)
    att_s = BatchNormalization()(att_s)

    out = Activation("relu")(att_g + att_s)
    out = Conv2D(num_filters, 1, padding = "same")(out)
    out = Activation("sigmoid")(out)
    return out * s
```

### residual模塊

```
# residual模塊
def residual(inputs, filters, kernel_size = 3):
    x = Conv2D(filters, kernel_size, padding = 'same')(inputs)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Conv2D(filters, kernel_size, padding = 'same')(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Add()([x, inputs])
    return x
```

# 程式架構

點我進入COLAB

## 4.2 建立Y-net

```
[ ] from tensorflow import keras
    import tensorflow as tf
    from keras import Model
    from tensorflow.keras import backend as K
    from tensorflow.keras.layers import Input, Conv2D, Conv2DTranspose, Dropout, Concatenate, MaxPooling2D, BatchNormalizatio
```

# 程式架構

點我進入COLAB

```
[ ] from keras.api._v2.keras import activations
    def build_model():
        inputs = Input((x_train.shape[1], x_train.shape[2],x_train.shape[3],),name = 'input')
    #下採樣

        conv1 = Conv2D(64, (3,3),activation = 'relu',padding = 'same')(inputs)
        resi1 = residual(conv1, 64)
        pool1 = MaxPooling2D((2,2),strides = 2,padding = 'same')(resi1)
        drop1 = Dropout(0.2)(pool1)


        conv2 = Conv2D(128, (3,3),activation = 'relu',padding = 'same')(drop1)
        resi2 = residual(conv2, 128)
        pool2 = MaxPooling2D((2,2),strides = 2,padding = 'same')(resi2)
        drop2 = Dropout(0.2)(pool2)


        conv3 = Conv2D(256, (3,3),activation = 'relu',padding = 'same')(drop2)
        resi3 = residual(conv3, 256)
        pool3 = MaxPooling2D((2,2),strides = 2,padding = 'same')(resi3)
        drop3 = Dropout(0.2)(pool3)


        conv4 = Conv2D(512, (3,3),activation = 'relu',padding = 'same')(drop3)
        resi4 = residual(conv4, 512)
        pool4 = MaxPooling2D((2,2),strides = 2,padding = 'same')(resi4)
        drop4 = Dropout(0.2)(pool4)


        convm = Conv2D(1024, (3,3),activation = 'relu',padding = 'same')(drop4)
        convm = Conv2D(1024, (3,3),activation = 'relu',padding = 'same')(convm)
```

# 程式架構

點我進入COLAB

```
# 上採樣
    tran5 = Conv2DTranspose(512,(2,2),strides =(2,2) ,padding = 'valid',activation = 'relu')(convm)
    att1 = attention(tran5, resi4, 512)
    conc5 = Concatenate()([tran5, att1])# 保留下採樣的訓練特徵
    conv5 = Conv2D(512,(3,3),activation = 'relu',padding = 'same')(conc5)
    resi5 = residual(conv5, 512)
    drop5 = Dropout(0.1)(resi5)


    tran6 = Conv2DTranspose(256,(2,2),strides =(2,2),padding = 'valid',activation = 'relu')(drop5)
    att2 = attention(tran6, resi3, 256)
    conc6 = Concatenate()([tran6, att2])# 保留下採樣的訓練特徵
    conv6 = Conv2D(256,(3,3),activation = 'relu',padding = 'same')(conc6)
    resi6 = residual(conv6, 256)
    drop6 = Dropout(0.1)(resi6)


    tran7 = Conv2DTranspose(128,(2,2),strides = 2,padding = 'valid',activation = 'relu')(drop6)
    att3 = attention(tran7, resi2, 128)
    conc7 = Concatenate()([tran7,att3])# 保留下採樣的訓練特徵
    conv7 = Conv2D(128,(3,3),activation = 'relu',padding = 'same')(conc7)
    resi7 = residual(conv7, 128)
    drop7 = Dropout(0.1)(resi7)

    tran8 = Conv2DTranspose(64,(2,2),strides = 2,padding = 'valid',activation = 'relu')(drop7)
    att4 = attention(tran8, resi1, 64)
    conc8 = Concatenate()([att4, tran8])# 保留下採樣的訓練特徵
    conv8 = Conv2D(64,(3,3),name="Attention",activation = 'tanh',padding = 'same')(conc8)
    resi8 = residual(conv8, 64)
    drop8 = Dropout(0.1)(resi8)
    # 任務一: 影像切割 -> 資料: 圖像 & 遮罩
    # 區分 x 資料 (image) 與 y 資料 (mask) 的訓練集、測試集與預測集
    segmentation_output = Conv2D(3,(1,1), activation='sigmoid',name='segmentation')(drop8)
    # 任務二: 影像分類 -> 資料: 圖像 & 標籤
    # 區分 x 資料 (image) 與 y 資料 (label) 的訓練集、測試集與預測集
    dense_classification=Flatten()(convm)
    dense_classification=Dense(64, activation='tanh')(dense_classification)
    classification_output = Dense(3, activation='softmax', name='classification')(dense_classification)


    model = Model(inputs = inputs, outputs = [classification_output, segmentation_output])
    return model

model = build_model()
model.summary()
```

# 程式架構

點我進入COLAB

## 5.建立Metrics(量化指標)

```
[ ] # 設定損失函數->dice:圖像分割
    def dice_coef(y_true, y_pred):
        smooth = 1
        y_true_f = K.flatten(y_true)
        y_pred_f = K.flatten(y_pred)
        intersection = K.sum(y_true_f * y_pred_f)
        return (2. * intersection + smooth) / (K.sum(y_true_f * y_true_f) + K.sum(y_pred_f) + smooth)
```

## 6.模型編譯

```
[ ] # 模型編譯
    model.compile(
            optimizer = keras.optimizers.Adam(learning_rate = 0.0003),
            loss = {'classification':'sparse_categorical_crossentropy',
                    'segmentation':'binary_crossentropy'},
            loss_weights = {'classification':0.5,'segmentation':0.5},
            metrics = [{'classification':'accuracy'},{'segmentation':dice_coef}]
    )

    # Total loss = classification loss + segmentation loss = sparse_categorical_crossentropy + binary_crossentropy
```

# 程式架構

點我進入COLAB

## 7.訓練模型

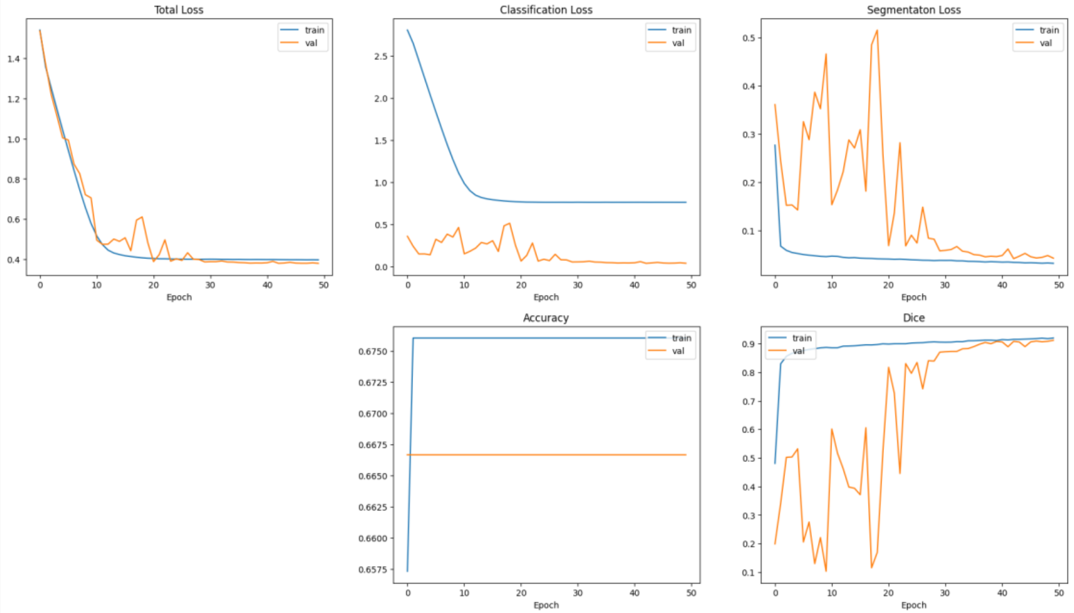
```
[ ]  from sklearn.utils import validation
     history = model.fit(
             {'input':x_train},
             {'classification':y_train_label, 'segmentation':y_train_mask},
             epochs = 50,
             batch_size = 32,
             validation_data = ({'input':x_val},{'classification':y_val_label,'segmentation':y_val_mask})
                           )
```

# 程式架構

點我進入COLAB

## 8.繪製訓練中的Loss與Dice變化

```
[ ] import matplotlib.pyplot as plt

    # Total loss = classification loss + segmentation loss = sparse_categorical_crossentropy + binary_crossentropy
    plt.figure(figsize=(22, 12))
    plt.subplot(2, 3, 1) # (rows, columns, index)
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Total Loss')
    plt.xlabel('Epoch')
    plt.legend(['train','val'],loc = 'upper right')

    plt.subplot(2, 3, 2)# (rows, columns, index)
    plt.plot(history.history['classification_loss'])
    plt.plot(history.history['val_segmentation_loss'])
    plt.title('Classification Loss')
    plt.xlabel('Epoch')
    plt.legend(['train','val'],loc = 'upper right')

    plt.subplot(2, 3, 3)# (rows, columns, index)
    plt.plot(history.history['segmentation_loss'])
    plt.plot(history.history['val_segmentation_loss'])
    plt.title('Segmentaton Loss')
    plt.xlabel('Epoch')
    plt.legend(['train','val'],loc = 'upper right')

    plt.subplot(2, 3, 5)# (rows, columns, index)
    plt.plot(history.history['classification_accuracy'])
    plt.plot(history.history['val_classification_accuracy'])
    plt.title('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['train','val'],loc = 'upper right')

    plt.subplot(2, 3, 6)# (rows, columns, index)
    plt.plot(history.history['segmentation_dice_coef'])
    plt.plot(history.history['val_segmentation_dice_coef'])
    plt.title('Dice')
    plt.xlabel('Epoch')
    plt.legend(['train','val'],loc = 'upper left')
```



# 程式架構

點我進入COLAB

## 9.評估模型

- test: loss
- classification: accuracy
- segmentation: dice

```
[ ] score = model.evaluate(
            {'input':x_test},
            {'classification':y_test_label,'segmentation':y_test_mask}
            )
    print('Test loss:',score[0])
    print('Test classification accuracy:',score[3])
    print('Test segmentation dice:',score[4])
    # Total loss = classification loss + segmentation loss = sparse_categorical_crossentropy + binary_crossentropy
```

```
3/3 [==============================] - 1s 418ms/step - loss: 0.3759 - classification_loss: 0.7027 - segmentation_loss: 0.0491 - classif
Test loss: 0.3758823573589325
Test classification accuracy: 0.7424242496490479
Test segmentation dice: 0.9034940600395203
```

# 程式架構

點我進入COLAB

## 9.2 分類結果與混淆矩陣

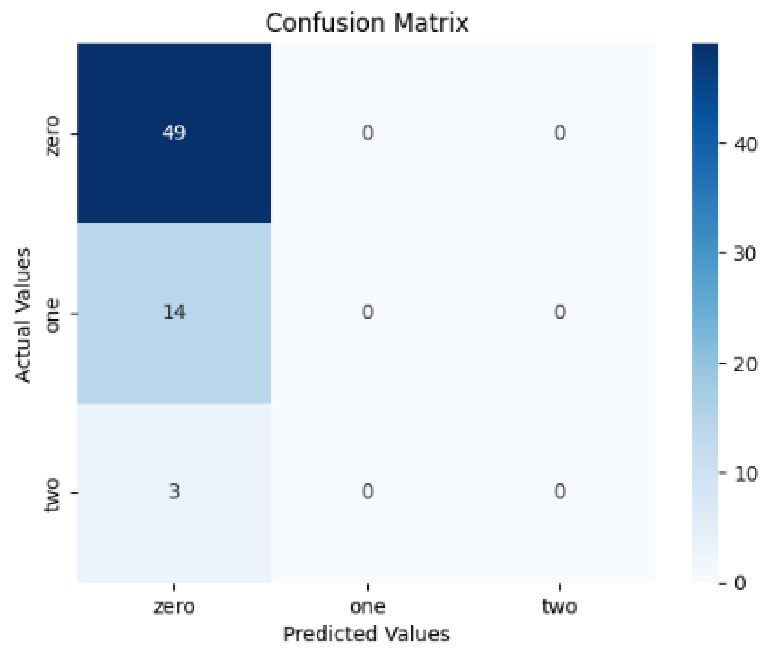
```
[ ] from sklearn.metrics import confusion_matrix, classification_report

    pred = model.predict({'input': x_test})
    pred_c = np.around(pred[0],0)
    pred_c = tf.argmax(pred_c, axis = 1)
    print(classification_report(y_test_label, pred_c))
```

```
3/3 [==============================] - 1s 57ms/step
              precision    recall  f1-score   support

         0.0       0.74      1.00      0.85        49
         1.0       0.00      0.00      0.00        14
         2.0       0.00      0.00      0.00         3

    accuracy                           0.74        66
   macro avg       0.25      0.33      0.28        66
weighted avg       0.55      0.74      0.63        66

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill
  _warn_prf(average, modifier, msg_start, len(result))
```

# 程式架構

點我進入COLAB

```
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import pandas as pd

conf = confusion_matrix(y_test_label, pred_c)
class_name = ["zero","one","two"]
cm_df = pd.DataFrame(
        conf,
        index = class_name,
        columns = class_name
)

# 繪製混淆矩陣
sns.heatmap(cm_df, annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix")
plt.ylabel("Actual Values")
plt.xlabel("Predicted Values")
plt.show()
```



# 程式架構

點我進入COLAB

## 康德鋸鱗魚特異性與敏感性

```
[ ] # 康德鋸鱗魚特異性與敏感性->label:0
    def zero_specificity_sensitivity(y_test, y_pred):
        P00,P10,P20,P01,P11,P21,P02,P12,P22 = confusion_matrix(y_test, y_pred).ravel()

        # P00, P10, P20,
        # P01, P11, P21,
        # P02, P12, P22

        TP = P00
        FP = P01 + P02
        FN = P10 + P20
        TN = confusion_matrix(y_test, y_pred).sum() - TP - FP - FN
        specificity = TN / (TN + FP) # 特異性
        sensitivity = TP / (TP + FN) # 敏感性
        return specificity, sensitivity
    specificity, sensitivity = zero_specificity_sensitivity(y_test_label, pred_c)

    print("\t zero")
    print("specificity: ",specificity)
    print("sensitivity: ",sensitivity)
```

```
        zero
specificity:  0.0
sensitivity:  1.0
```

# 程式架構

點我進入COLAB

## 摩鹿加雀鯛特異性與敏感性

```
# 摩鹿加雀鯛特異性與敏感性->label:1
def one_specificity_sensitivity(y_test, y_pred):
    P00, P10, P20, P01, P11, P21, P02, P12, P22 = confusion_matrix(y_test, y_pred).ravel()

    # P00, P10, P20,
    # P01, P11, P21,
    # P02, P12, P22

    TP = P11
    FP = P10 + P12
    FN = P01 + P21
    TN = confusion_matrix(y_test, y_pred).sum() - TP - FP - FN
    specificity = TN / (TN + FP) # 特異性
    sensitivity = TP / (TP + FN) # 敏感性
    return specificity, sensitivity
specificity, sensitivity = one_specificity_sensitivity(y_test_label, pred_c)

print("\t one")
print("specificity: ",specificity)
print("sensitivity: ",sensitivity)
```

```
        one
specificity:  1.0
sensitivity:  0.0
```

# 程式架構

點我進入COLAB

## 黑緣擬金眼鯛特異性與敏感性

```
[ ] # 黑緣擬金眼鯛特異性與敏感性->label:2
    def two_specificity_sensitivity(y_test, y_pred):
        P00, P10, P20, P01, P11, P21, P02, P12, P22 = confusion_matrix(y_test, y_pred).ravel()

        # P00, P10, P20,
        # P01, P11, P21,
        # P02, P12, P22

        TP = P22
        FP = P20 + P21
        FN = P02 + P12
        TN = confusion_matrix(y_test, y_pred).sum() - TP - FP - FN
        specificity = TN / (TN + FP) # 特異性
        sensitivity = TP / (TP + FN) # 敏感性
        return specificity, sensitivity
    specificity, sensitivity = two_specificity_sensitivity(y_test_label, pred_c)

    print("\t two")
    print("specificity: ",specificity)
    print("sensitivity: ",sensitivity)
```

```
        two
specificity:  1.0
sensitivity:  0.0
```

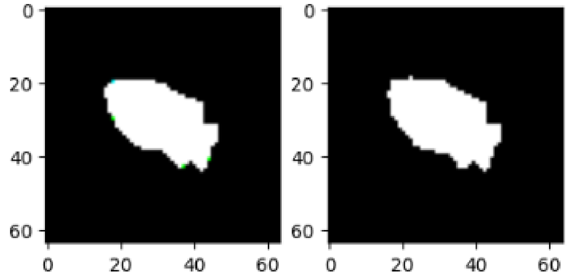
# 程式架構

點我進入COLAB

## 分割結果

```
[ ] pred_s = pred[1]

    pred_s[pred_s >= 0.5] = 1
    pred_s[pred_s < 0.5] = 0

    i = 55
    plt.figure(figsize = (5, 8))
    plt.subplot(1, 2, 1)# (rows, columns, index)
    plt.imshow(pred_s[i])
    plt.subplot(1, 2, 2)# (rows, columns, index)
    plt.imshow(y_test_mask[i])
```

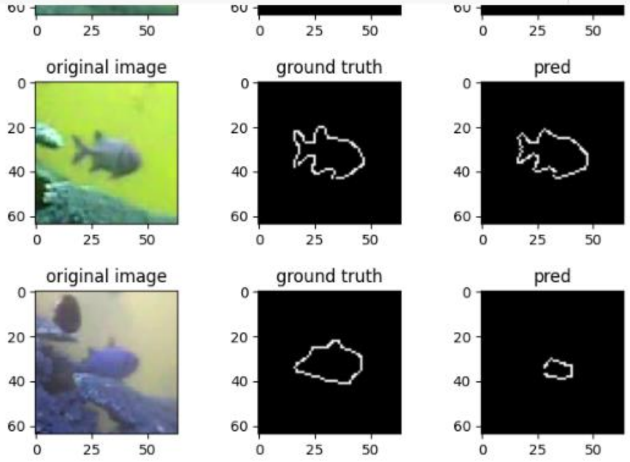
<matplotlib.image.AxesImage at 0x7f860c3540d0>



# 程式架構

點我進入COLAB

```
y_test_8bit = y_test_mask.astype(np.uint8)  # cv2.Canny只能輸入uint8
pred_8bit = pred_s.astype(np.uint8)
x_test_32 = x_test.astype(np.float32)  # 64位浮點opencv只適用float32

for i in range(x_test.shape[0]):
    plt.figure(figsize = (8, 4))
    plt.subplot(2, 3, 1)# (rows, columns, index)
    plt.title("original image")# 原始圖像
    plt.imshow(cv2.cvtColor(x_test_32[i], cv2.COLOR_BGR2RGB))
    plt.subplot(2, 3, 2)# (rows, columns, index)
    plt.title("ground truth")# 基本事實(專家畫的外框)
    plt.imshow(cv2.Canny(y_test_8bit[i], 0, 1), cmap = "gray")
    # plt.subplot(2, 3, 5)
    # plt.imshow(cv2.cvtColor(x_test_32[i] + convertToThreeChennel(cv2.Canny(y_test_8bit[i], 0, 1)), cv2.COLOR_BGR2RGB))
    plt.subplot(2, 3, 3)# (rows, columns, index)
    plt.title("pred")# 預測
    plt.imshow(cv2.Canny(pred_8bit[i], 0, 1), cmap = "gray")
    # plt.subplot(2, 3, 6)
    # plt.imshow(cv2.cvtColor(x_test_32[i] + convertToThreeChennel(cv2.Canny(y_test_8bit[i], 0, 1)), cv2.COLOR_BGR2RGB))
    plt.show()
```



# 程式架構修改過程

出現問題：loss、dice出現nan值

```
Epoch 1/50
44/67 [==================>...........] - ETA: 20s - loss: nan - classification_loss: nan - segmentation_loss: nan -

- classification_accuracy: 0.2812 - segmentation_dice_coef: nan
```

↑僅classification_accuracy有數值出現

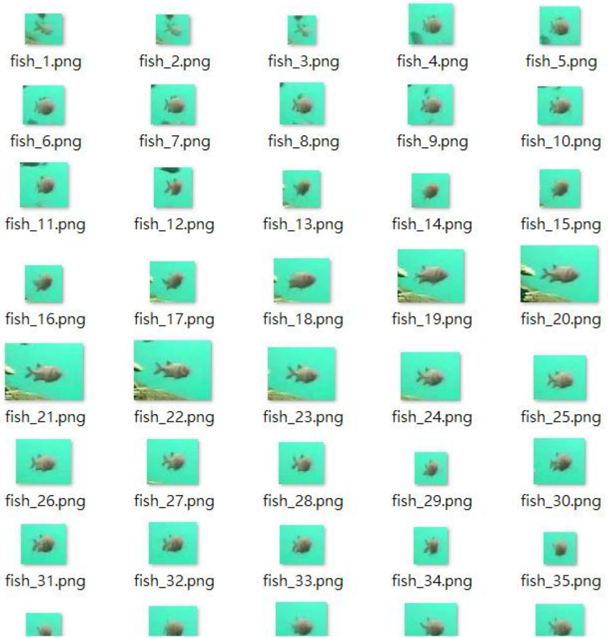
# 程式架構修改過程

## 為何出現Nan值?

- 數值不穩定（Numerical instability）：當計算中涉及到極大或極小的數字時，例如指數函數或除以接近零的數字，可能會導致數值不穩定，進而產生NaN值
- 缺失值（Missing values）：如果輸入數據中存在缺失值，而你的損失函數未能處理缺失值，可能會導致計算結果為NaN
- 學習率（Learning rate）過大或是過小：可能會導致梯度下降過程中的數值不穩定，使得損失函數的計算結果為NaN
- 梯度爆炸（Gradient explosion）或梯度消失（Gradient vanishing）：當反向傳播算法計算梯度時，如果梯度的值變得極度大或極度小，可能會導致損失函數產生NaN值
- 計算錯誤：有時在程式實現中，錯誤的計算邏輯或錯誤的數據處理可能導致NaN值的出現

# 程式架構修改過程

圖片大小不一





# 程式架構修改過程

正規化修改：取圖片數據大小的中間值

原始放大到 224*224 可能間接地影響梯度計算的穩定性

```
img = tf.keras.preprocessing.image.load_img(folder + filename, target_size = (224,224))
```

將 224*224 修改為 64*64

```
img = tf.keras.preprocessing.image.load_img(folder + filename, target_size = (64,64))
```

# 程式架構修改過程

點我進入網誌

網路結構不合理

```
dense_classification=Flatten()(convm)
dense_classification=Dense(300)(dense_classification)
classification_output = Dense(1, activation='softmax', name='classification')(dense_classification)
```

根據建議，使用tanh激活函式與softmax組合

```
dense_classification=Flatten()(convm)
dense_classification=Dense(64, activation='tanh')(dense_classification)
classification_output = Dense(3, activation='softmax', name='classification')(dense_classification)
```

# 程式架構修改過程

## 增加batch_size

```
from sklearn.utils import validation
history = model.fit(
        {'input':x_train},
        {'classification':y_train_label, 'segmentation':y_train_mask},
        epochs = 50,
        batch_size = 8,
        validation_data = ({'input':x_val},{'classification':y_val_label,'segmentation':y_val_mask})
                                          )
```

```
from sklearn.utils import validation
history = model.fit(
        {'input':x_train},
        {'classification':y_train_label, 'segmentation':y_train_mask},
        epochs = 50,
        batch_size = 32,
        validation_data = ({'input':x_val},{'classification':y_val_label,'segmentation':y_val_mask})
                                          )
```

# 程式架構修改過程

```
- 34s 318ms/step - loss: 1.5395 - classification_loss: 2.8021
- 4s 245ms/step - loss: 1.3567 - classification_loss: 2.6461
- 4s 245ms/step - loss: 1.2508 - classification_loss: 2.4430
- 4s 251ms/step - loss: 1.1466 - classification_loss: 2.2392
- 4s 247ms/step - loss: 1.0430 - classification_loss: 2.0340
- 4s 248ms/step - loss: 0.9418 - classification_loss: 1.8337
- 4s 251ms/step - loss: 0.8425 - classification_loss: 1.6365
- 4s 250ms/step - loss: 0.7470 - classification_loss: 1.4466
- 4s 249ms/step - loss: 0.6581 - classification_loss: 1.2700
```

# 結論

- 分類：魚類種類辨別因資料不平衡，模型傾向全部猜測同一種類，應增加其他種類之數量
- 分割：分割部分表現良好，接近真實外框

問題與解決：

- Nan值的可能原因有許多可能，需要一一修改以解決此問題，可從learning rate、正規化、網路結構、增加batch_size等方向進行嘗試，但Nan值的出現並不僅限於這些原因
- 魚類的分割已有不錯的效果，但魚種的分類仍需要更多的資料或是其他方法以達成魚種分類的目標

# THANKS!