กฤษญา โมรา 6304062630016
ธนาธร ทะรวะสันต์ 6304062630156

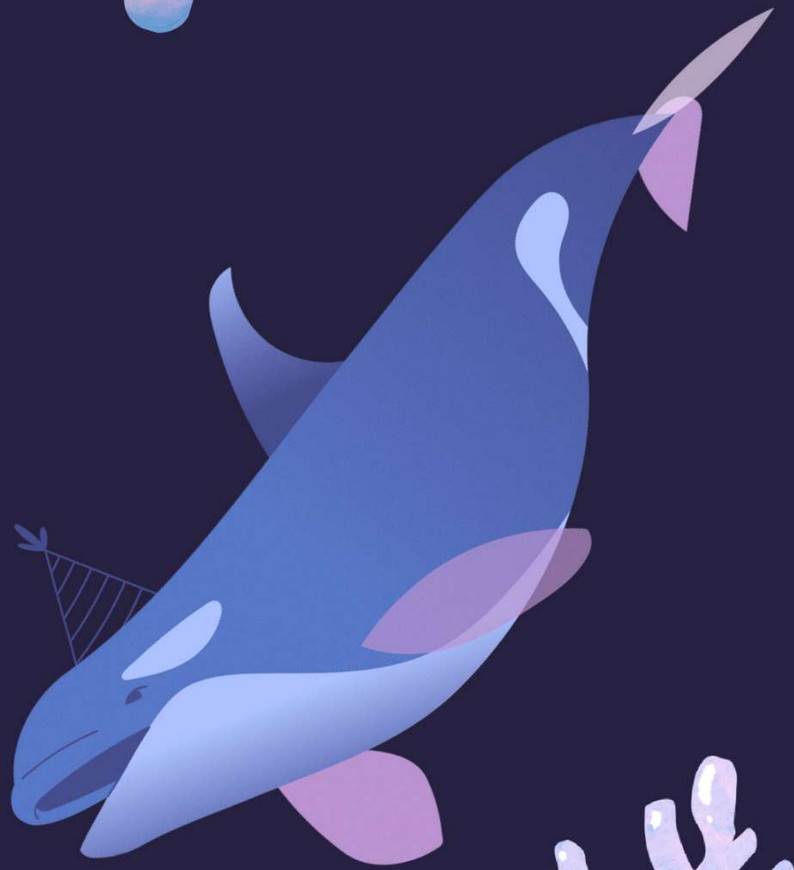
machine learning

Humpback Whale Identification

ที่มาและความสำคัญ

ในปัจจุบัน วาฬมีจำนวนลดน้อยลงเนื่องด้วยการถูกล่าและสภาพอากาศที่เปลี่ยนแปลง เพื่อเป็นการอนุรักษ์วาฬ นักวิทยาศาสตร์ได้มีการใช้ระบบเฝ้าระวังจากภาพถ่าย เพื่อใช้การวิเคราะห์และแยกสายพันธุ์ของวาฬ โดยสังเกตจากหางของวาฬ ราว ๆ 40 ปีที่ผ่านมา ข้อมูลเหล่านี้ได้ถูกบันทึกด้วยมือโดยนักวิทยาศาสตร์ จึงทำให้มีข้อมูลจำนวนมหาศาลที่ไม่ได้ถูกนำมาใช้หรือใช้ประโยชน์น้อยเกินไป

เพื่อเป็นการนำเอาข้อมูลมหาศาลเหล่านั้นมาใช้ประโยชน์จึงเกิดอัลกอริทึมที่ใช้ในการจำแนกวาฬ หากนำเอกลักษณ์บางส่วนของวาฬมาใช้วิเคราะห์ข้อมูลก็จะสามารถจำแนกวาฬแต่ละตัวได้ง่ายยิ่งขึ้น โดยเลือกหางวาฬมาเป็นส่วนที่จะใช้จำแนกข้อมูลเพราะหางวาฬนั้นมองเห็นได้ง่ายและมีลักษณะแตกต่างกันไป ซึ่งหากมี Model ที่สามารถช่วยจำแนกวาฬเหล่านั้นได้ ก็จะส่งผลให้การอนุรักษ์วาฬนั้นมีประสิทธิภาพมากยิ่งขึ้น



Tag

Classification

Input

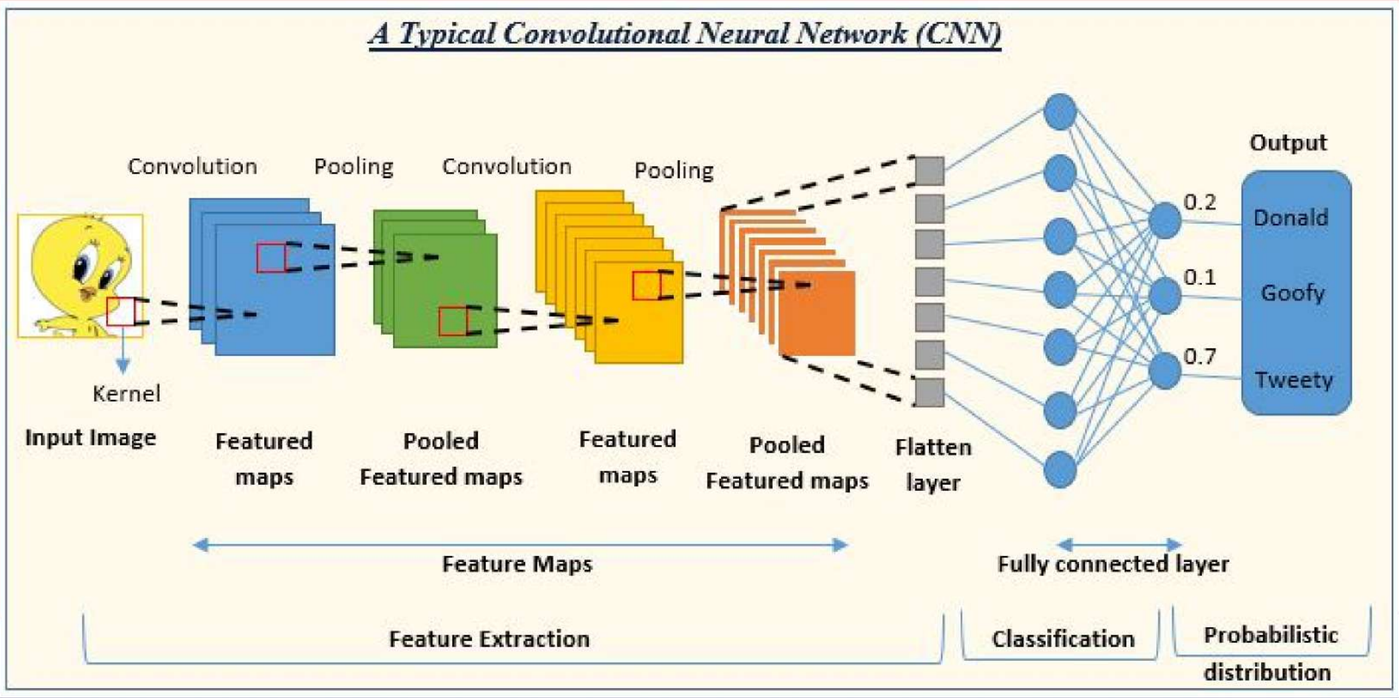
Image of whales' tails

Lable

ID Whales

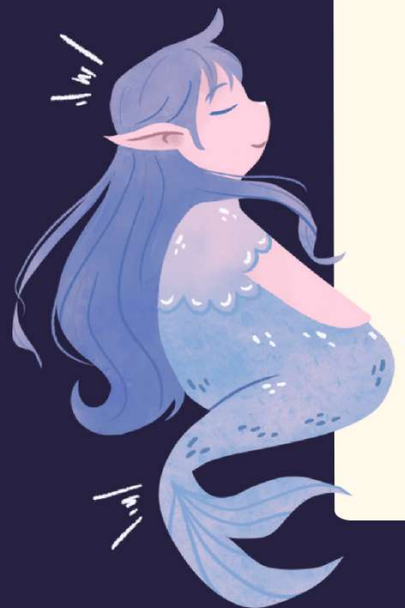
Technical

CNN (Convolutional Neural Networks) เป็นโมเดลพื้นฐานที่นำมาใช้งานเกี่ยวกับ Classification โดย CNN จะทำงานเกี่ยวกับรูปภาพเท่านั้น ใช้งานง่ายเหมาะสำหรับการเริ่มต้นฝึกเขียน machine learning



Process

1. โหลด DataSet จาก Kaggle จะได้ Folder 'test', 'train' และไฟล์ CSV 'sample_submission.csv', 'train.csv', ออกมา
2. ดูว่าไฟล์ train.csvนี้มี ID ที่ไม่ซ้ำกันทั้งหมดเท่าไหร่ เพื่อกำหนดค่าที่ Unique เหล่านั้นให้เป็น Lable ที่เราต้องการ
3. แบ่ง x_train และ y_train จาก Folder train
4. ปรับรูปภาพที่จะใช้ train ให้เหมาะสมกับโมเดล
5. ทำการ train ข้อมูล
6. ทดสอบความแม่นยำ
7. ลองทำนายนน test tFolder
8. เขียนผลลัพธ์เป็นไฟล์ CSV จำนวน 5 Output ตามที่โจทย์ต้องการและทำการ Submit เพื่อรับการประเมินผลจาก Kaggle



Training

```
train = pd.read_csv("../input/train.csv")
train.Id.describe()
y_train = train["Id"]
X_train = train.drop(labels = ["Id"], axis = 1)
```

	Image	Id
count	25361	25361
unique	25361	5005
top	0000e88ab.jpg	new_whale
freq	1	9664

Training

```
from keras.preprocessing import image
from keras.applications.imagenet_utils import preprocess_input

def prepareImages(train, shape, path):

    x_train = np.zeros((shape, 100, 100, 3))
    count = 0

    for fig in train['Image']:

        img = image.load_img("../input/"+path+"/"+fig, target_size=(100, 100, 3))
        x = image.img_to_array(img)
        x = preprocess_input(x)

        x_train[count] = x
        if (count%500 == 0):
            print("Processing image: ", count+1, ", ", fig)
        count += 1

    return x_train
```

Training

```
def prepareImages(train, shape, path):

    x_train = np.zeros((shape, 100, 100, 3))
    count = 0

    for fig in train['Image']:

        img = image.load_img("../input/"+path+"/"+fig, target_size=(100, 100, 3))
        x = image.img_to_array(img)
        x = preprocess_input(x)

        x_train[count] = x
        if (count%500 == 0):
            print("Processing image: ", count+1, ", ", fig)
        count += 1

    return x_train

x_train = prepareImages(train, train.shape[0], "train")
x_train = x_train / 255.0
print("x_train shape: ",x_train.shape)
```

Training

```
label_encoder = LabelEncoder()
y_train = label_encoder.fit_transform(y_train)
y_train = to_categorical(y_train, num_classes = 5005)
print(y_train.shape)
y_train
```

```
(25361, 5005)
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

Training

```
model = Sequential()

model.add(Conv2D(filters = 16, kernel_size = (5,5), padding = 'Same', activation = 'relu',
                 input_shape = (100,100,3)))
model.add(Conv2D(filters = 16, kernel_size = (5,5), padding = 'Same', activation = 'relu',
                 input_shape = (100,100,3)))
model.add(MaxPool2D(pool_size = (2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(filters = 32, kernel_size = (3,3), padding = 'Same', activation = 'relu'))
model.add(Conv2D(filters = 32, kernel_size = (3,3), padding = 'Same', activation = 'relu'))

model.add(MaxPool2D(pool_size = (2,2), strides=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(filters = 64, kernel_size = (3,3), padding = 'Same', activation = 'relu'))
model.add(Conv2D(filters = 64, kernel_size = (3,3), padding = 'Same', activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2), strides=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256, activation = 'relu'))
model.add(BatchNormalization())
model.add(Dense(y_train.shape[1], activation = "softmax"))
model.summary()
```

Training

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 100, 100, 16)	1216
conv2d_2 (Conv2D)	(None, 100, 100, 16)	6416
max_pooling2d_1 (MaxPooling2	(None, 50, 50, 16)	0
dropout_1 (Dropout)	(None, 50, 50, 16)	0
conv2d_3 (Conv2D)	(None, 50, 50, 32)	4640
conv2d_4 (Conv2D)	(None, 50, 50, 32)	9248
max_pooling2d_2 (MaxPooling2	(None, 25, 25, 32)	0
dropout_2 (Dropout)	(None, 25, 25, 32)	0
conv2d_5 (Conv2D)	(None, 25, 25, 64)	18496
conv2d_6 (Conv2D)	(None, 25, 25, 64)	36928
max_pooling2d_3 (MaxPooling2	(None, 12, 12, 64)	0
dropout_3 (Dropout)	(None, 12, 12, 64)	0
flatten_1 (Flatten)	(None, 9216)	0
dense_1 (Dense)	(None, 256)	2359552
batch_normalization_1 (Batch	(None, 256)	1024
dense_2 (Dense)	(None, 5005)	1286285

Total params: 3,723,805
Trainable params: 3,723,293
Non-trainable params: 512

Training

```
optimizer = Adam(lr = 0.001, beta_1 = 0.9, beta_2 = 0.999)
learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc',
                                            patience=3,
                                            verbose=1,
                                            factor=0.5,
                                            min_lr=0.00001)
model.compile(optimizer = optimizer, loss = "categorical_crossentropy", metrics=["accuracy"])
```

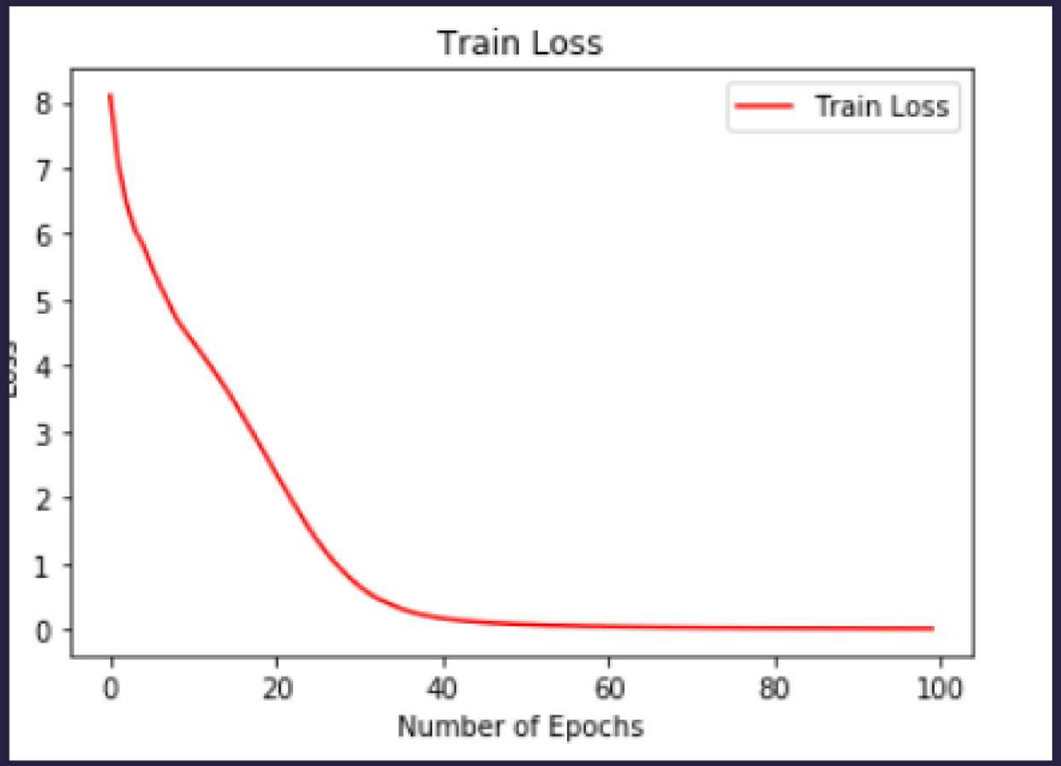
Training

```
history = model.fit(x_train, y_train, epochs = 100, batch_size = 1000 ,
                    verbose = 2, callbacks=[learning_rate_reduction])
```

```
Epoch 1/100
 - 26s - loss: 8.0941 - acc: 0.2015
Epoch 2/100
 - 19s - loss: 7.0363 - acc: 0.2994
Epoch 3/100
 - 19s - loss: 6.4404 - acc: 0.3288
Epoch 4/100
 - 19s - loss: 6.0527 - acc: 0.3443
Epoch 5/100
 - 19s - loss: 5.8183 - acc: 0.3453
```

Training

```
plt.plot(history.history['loss'], color='r', label="Train Loss")
plt.title("Train Loss")
plt.xlabel("Number of Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```



Training

```
test = os.listdir("../input/test/")
print(len(test))

col = ['Image']
test_data = pd.DataFrame(test, columns=col)
test_data['Id'] = ''

x_test = prepareImages(test_data, test_data.shape[0], "test")
x_test /= 255

predictions = model.predict(np.array(x_test), verbose=1)
```

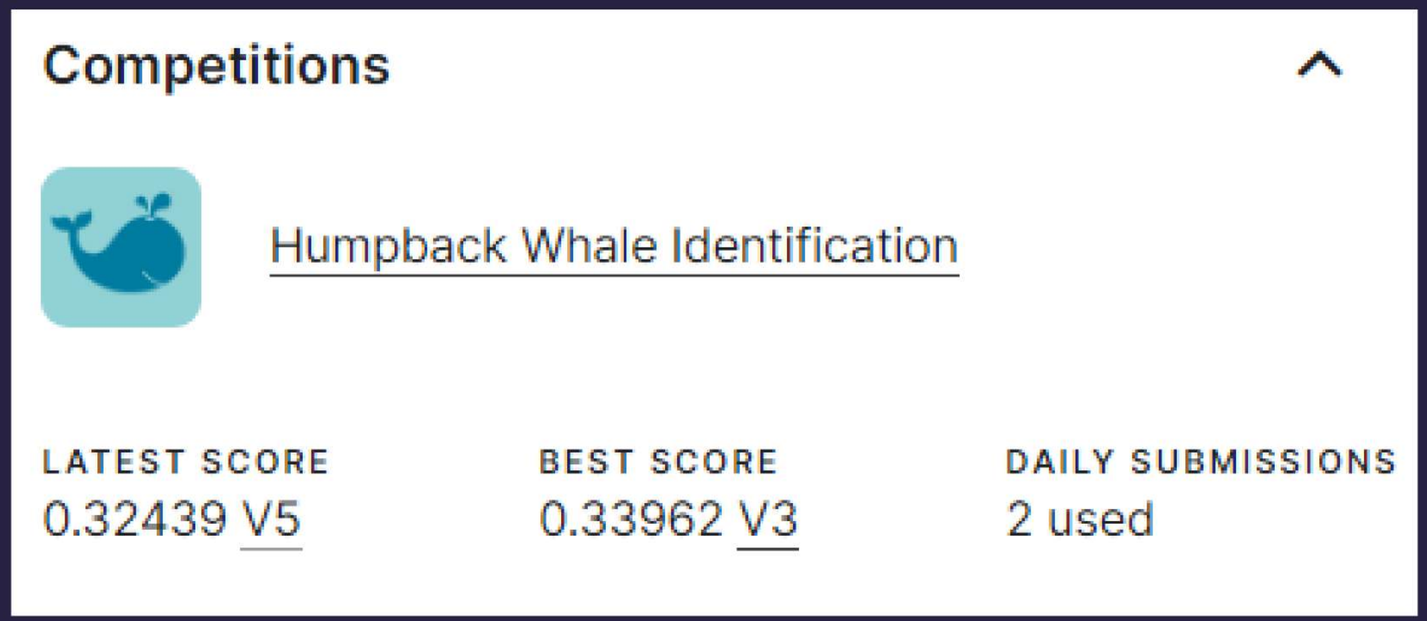
Submission

```
for i, pred in enumerate(predictions):
    test_data.loc[i, 'Id'] = ' '.join(label_encoder.inverse_transform(pred.argsort()[-5:][::-1]))

test_data.to_csv('submission.csv', index=False)
test_data.head(10)
```

	Image	Id
0	fc5634f1c.jpg	new_whale w_336e046 w_f0a5983 w_9438119 w_1af6b9a
1	ed344e8e7.jpg	new_whale w_15951db w_ae7887b w_d066f46 w_d307d9d
2	8cf0fd984.jpg	new_whale w_700ebb4 w_5f0af76 w_a9304b9 w_efbdcbc
3	6513cfc1f.jpg	new_whale w_aabdf8c w_2694603 w_edce644 w_163b2b8
4	fd940ccca.jpg	new_whale w_81f8d47 w_b4841bd w_1f1cee1 w_d7b6f17
5	9e602e86e.jpg	new_whale w_e2a09d4 w_454f511 w_609529a w_45c5396
6	7fb06a038.jpg	new_whale w_565d73a w_94a7669 w_c85ff1e w_af493e1
7	10bf25ffb.jpg	new_whale w_08d62ee w_4690940 w_89f521e w_8cfabed
8	6c6c5aec8.jpg	new_whale w_ec1eb03 w_ccc26ea w_e51c0e9 w_b035775
9	f4bb70495.jpg	new_whale w_8bcc197 w_093d284 w_6c995fd w_b495218

Score Submission



Evaluation

$$MAP@5 = \frac{1}{U}\sum_{u=1}^{U}\sum_{k=1}^{min(n,5)} P(k) \times rel(k)$$

U คือ จำนวนของรูปภาพ

P(k) คือ **Precision** ของภาพ

k,n คือ จำนวนการทำนายภาพของแต่ละรูป มีค่าระหว่าง **1** ถึง **5**

rel(k) คือ จะแสดงเลข **1** เมื่อ **k** ทำนายถูกต้อง

Conclusion

👉 เป็นงานที่มีคะแนนในการทำงานหลังจากที่ **Evaluation** ในระดับที่รับได้เนื่องจากยังไม่สามารถประยุกต์การเขียนโค้ดได้มากนัก และโมเดลนี้มีโอกาสเกิด **Overfitting** สูง เนื่องจากว่า ภาพในการทำนายมีความคลายคลึงกันมาก จึงทำให้ โมเดลของเราที่ใช้ **CNN** เป็นหลักไม่สามารถทำนายได้ดีมากนัก จึงทำให้เกิด **Overfitting**

👉 ข้อเสนอแนะ ควรใช้โมเดลอื่นที่ไม่ใช่ **CNN** ในการทำนาย **Whale Id** แต่ถ้าอยากฝึกการเขียนโค้ดก็ยังสามารถใช้ได้ แต่ **Score** ที่ได้รับจากการ **Submit** อาจจะไม่ได้ดีมาก

Thank You