# 智慧化企業整合

# 自然環境辨識

**學生：110034548 張瑋芩**

**中 華 民 國　1　1　1　年　一　月　七　日**

# 目錄

# 圖目錄

# 表目錄

# 一、 背景介紹

## 1.1 背景介紹

據相關資料顯示在 2021 年全球社群媒體活躍用戶總數為 42 億人，換句話說，全球有高達 53.6%的人口正使用著社群媒體，而許多用戶會透過社群媒體的標籤功能尋找旅遊景點。

故本研究應用 kaggle 上自然環境的資料圖片來訓練以預測自然環境的類別，希望能夠正確地提供文章準確的旅遊景點標籤，透過辨識照片中的自然環境類別，能夠幫助用戶利用 Hashtag 時能協助用戶正確的標記，也能夠讓其他用戶在利用 Hashtag 搜尋資料時，更加準確地找正確的自然環境。

## 1.2 問題定義 5W1H

進行本研究前，先以 5W1H 針對我們的問題定義進行分析，以更了解問題的本質。

Table 1 5W1H 分析

| | |
|---|---|
| Who | 想利用 Hashtag 找景點的人跟利用 Hashtag 分類文章的人。 |
| What | 精準地顯示自然環境的類別。 |
| Why | 為了讓使用者找到符合標籤的文章。 |
| When | 找景點時。 |
| Where | IG、FB 任何有標籤的地方。 |
| How | 利用 CNN+DNN。 |

# 二、 模型訓練與績效

## 2.1 資料蒐集

本研究的資料來源來自 Kaggle 網站上的 Intel Image Classification，資料包含了 6 種自然環境影像，共 24335 張。其資料如下

- seg_pred：未分類之環境景觀影像共 7301 筆
- seg_train：已分類之環境景觀影像共 14034 筆
- seg_test：已分類之環境景觀影像共 3000 筆

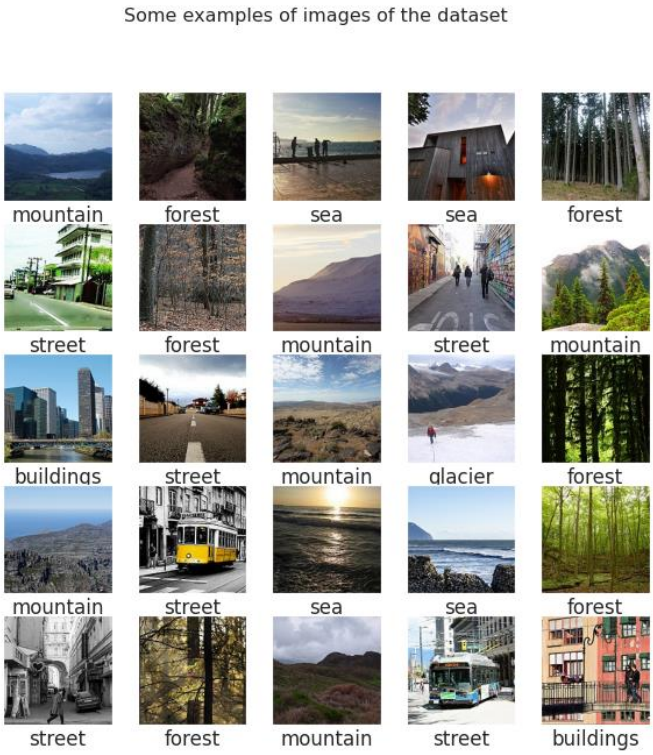



Figure 1 自然環境圖與類別

## 2.2 資料前處理

### 2.2.1 匯入資料

i. 將放置在各個不同的資料夾的影像進行讀取。

ii. 將讀取之影像依其所屬之資料夾給予編碼（從建築物資料夾載入的影像作為第 0 類，森林作為第 1 類，冰川作為第 2 類，山作為第 3 類，海作為第 4 類，街道

作為第 5 類 )。

iii. 將所有圖片調整為相同大小 ( 150×150 )。

```python
class_names = ['mountain', 'street', 'glacier', 'buildings', 'sea', 'forest']
class_names_label = {class_name:i for i, class_name in enumerate(class_names)}

nb_classes = len(class_names)

IMAGE_SIZE = (150, 150)

def load_data():
    """
        Load the data:
            - 14,034 images to train the network.
            - 3,000 images to evaluate how accurately the network learned to classify images.
    """

    datasets = ['/content/drive/MyDrive/seg_train/seg_train', '/content/drive/MyDrive/seg_test/seg_test']
    output = []

    # Iterate through training and test sets
    for dataset in datasets:

        images = []
        labels = []

        print("Loading {}".format(dataset))

        # Iterate through each folder corresponding to a category
        for folder in os.listdir(dataset):
            label = class_names_label[folder]

            # Iterate through each image in our folder
            for file in tqdm(os.listdir(os.path.join(dataset, folder))):

                # Get the path name of the image
                img_path = os.path.join(os.path.join(dataset, folder), file)

                # Open and resize the img
                image = cv2.imread(img_path)
                image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
                image = cv2.resize(image, IMAGE_SIZE)

                # Append the image and its corresponding label to the output
                images.append(image)
                labels.append(label)

        images = np.array(images, dtype = 'float32')
        labels = np.array(labels, dtype = 'int32')

        output.append((images, labels))

    return output

(train_images, train_labels), (test_images, test_labels) = load_data()
```

Figure 2 將圖像分類與統一大小

### 2.2.2 資料平衡

利用圖表的方式觀察資料間有沒有不平衡的情況，防止資料模型訓練或預測時數量多的類別容易被預測到。

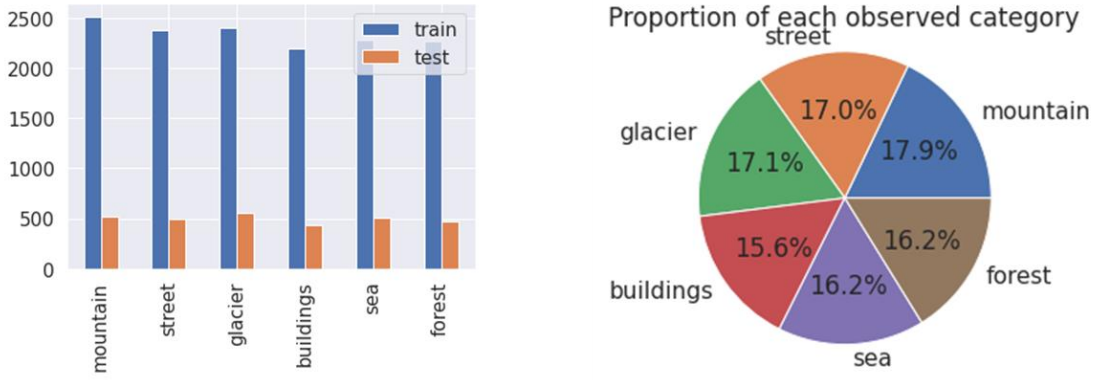



Figure 3 資料比數比例圖

透過圖 3 可以發現資料並沒有不平衡。由於資料沒有缺失與不平衡的情況，本研究就不

而外在對圖像進行擴增等處理。

### 2.2.3 資料標準化與打亂

將資料標準化到 0 跟 1 之間，這樣才不會因為資料尺度的不同而影響到模型的訓練結果。由於灰階的影響是 0 到 255，所以選擇全數除以 255 來等比縮小。

```
train_images = train_images / 255.0
test_images = test_images / 255.0
```

Figure 4 資料標準化程式碼

為了避免有 Overfitting 的情況發生，所以將讀取的影像透過 shuffle 打亂，同時也能夠免圖一個組合的 batch 反覆出現。

```
train_images, train_labels = shuffle(train_images, train_labels, random_state=25)
```

Figure 5 資料打亂的程式碼

## 2.3 模型架構

本研究選定 VGG16 作為預訓練模型（Pre-Trained Model）來將圖像的特徵擷取出來，並搭載本組所建之 DNN 模型找出能預測出本資料集最佳結果的模型，並透過實驗設計的方式進行參數最佳化，以訓練出績效最好的模型。

Table 2 本組 DNN 模型層級及說明

| 層級 | 說明 |
|---|---|
| **平坦層（Flatten）** | 銜接 CNN 層(Pre-Trained Model)與全連接層。 |
| **全連接層（FC）** | 主要在做最後的特徵提取，並且利用最後一層 FC 當作分類器 |
| **輸出層** | 預測出的 6 種類別，Dense(6 , activation = 'softmax') |

```
model2 = tf.keras.Sequential([
        tf.keras.layers.Flatten(input_shape = (x, y, z)),
        tf.keras.layers.Dense(50, activation=tf.nn.relu),
        tf.keras.layers.Dense(6, activation=tf.nn.softmax)
])
```

Figure 6 DNN 模型

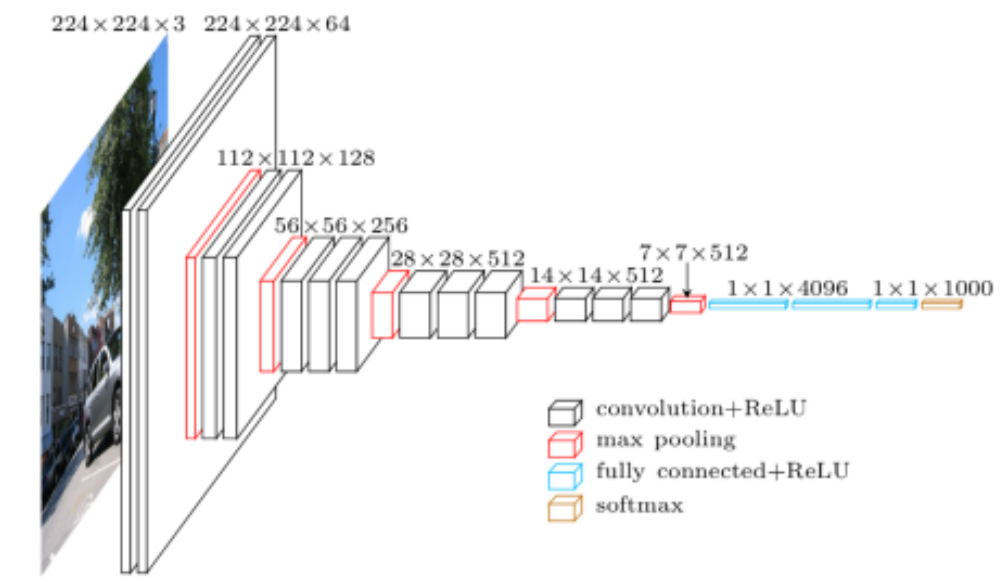
VGG 是英國牛津大學 Visual Geometry Group 的縮寫，主要貢獻是使用更多的隱藏層，以大量的圖片訓練，提高準確率至 90%。其結構簡潔，由 5 層卷積層、3 層全連接層、輸出層（激活函數 softmax）構成，層與層之間使用Max-pooling（最大化池）分開，所有隱藏層的激活函數都採用 ReLU 函數。

VGG 應用多個較小卷積核（3x3）的卷積層代替一個較大卷積核的卷積層，一方面可以減少參數，另一方面相當於進行了更多的非線性映射，可以增加網絡的擬合及表達能力，由於卷積核專注於擴大通道數、池化專注於縮小寬和高，使模型層數更深且特徵圖更寬，也控制計算量的增加規模，其缺點為參數量龐大，計算資源需求高，且較難調整參數。

A. VGG16：

16 層包含 13 個卷積層及 3 個全連接層，其架構為卷積後接池化層，一來可減少參數的數量，加快計算速度，而減少參數可以防止過擬合的情況，且每個卷積都有一個激活函數，故可擬合更複雜的數據。

VGG16 架構圖如下，可觀察到其架構為每一塊包含若干卷積層和一個池化層，且同一塊內卷積層的通道（channel）數是相同的，隨著層數的增加通道數翻倍，圖片高和寬減半。



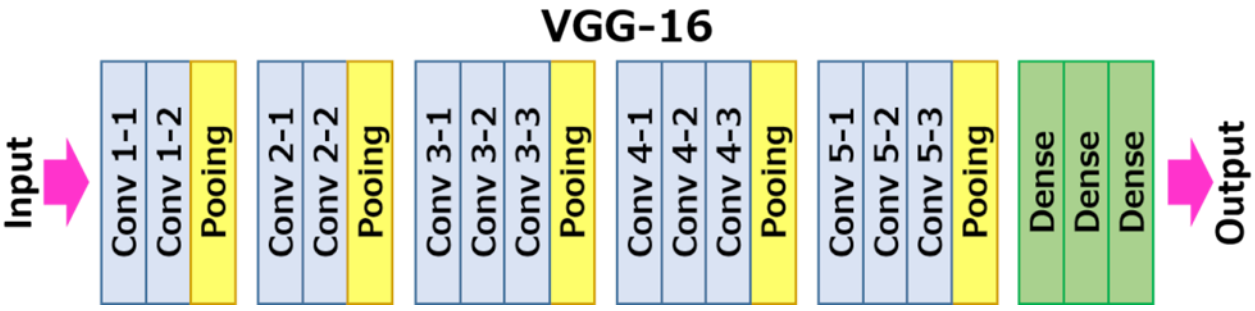



Figure 7 VGG 16 模型架構

資料來源：VGG16 -用于分类和检测的卷积网络

## 2.4 超參數調整與模型比較

初步訓練只使用本組搭建的 CNN 模型，但最後預測結果準確度不超過 0.75，因此透過 VGG16 再搭配本組建之 DNN 模型，準確度大幅提升，所設定參數如下表。

Table 3 超參數調整項目

| Optimizer | Learning rate | Batch size |
|---|---|---|
| **Adam** | 0.01~0.001 | 20 |
| **SGD** | 0.001~0.0001 | 128 |
| **Adagrad** | | 200 |

將上表中 Optimizer（優化器）、Learning rate（學習率）與 Batch size 排列組合，完成本組實驗設計，以找到準確率最高的模型，當中學習率的部分使用 Keras 的回調函数 ReduceLROnPlateau，其於訓練過程中會依據所設定監控值以調整學習率，所輸入的值為上表中的數。

以下介紹所運用的優化器：

A. Adam：

結合 AdaGrad 和 RMSProp 兩種優化演算法的優點，對梯度的一階矩估計（First Moment Estimation，即梯度的均值）和二階矩估計（Second Moment Estimation，即梯度的未中心化的方差）進行綜合考慮，其優點主要在於其有做偏置校正，使每次迭代（Epoch）學習率都有個確定範圍，讓參數的更新較為平穩，引數的更新不受梯度的伸縮變換影響，具有高校的運算速度且對記憶體需求少。

B. SGD：

梯度下降法 (Gradient descent ,GD)是一次用全部訓練集的數據計算其損失函 數的梯度，即更新一次參數，由於每次都抽取相同的樣本集耗時且冗餘，而 隨機梯度下降法(Stochastic gradient descent, SGD)則是每次隨機抽取一個樣本或小批次(mini-batch)樣本即計算其梯度的平均後更新參數，由於隨機梯度下 降演算法每次只隨機選擇一個樣本來更新模型參數，因此每次的學習是非常快速的，並且可以進行線上更新。也會處於一個高變異的狀態，更新時 loss 比較震盪，可能會使得其跳出區域性最優點到達一個更好的區域性最優。

C. Adagrad

學習速度與參數會互相對應，舉例來說，頻繁常見的參數，那麼其會進行小部分的更新，反之則會進行很大的更新，因此很適合處理稀疏資料集，且減少 Learning rate 的手動調整，但缺點就是分母會不斷積累，這樣學習率就會收縮並最終會變得非常小。

Table 4 其他參數設定

| 參數 | 數值 |
|---|---|
| **Epochs** | 5 |
| **Loss** | Categorical_crossentropy |

```
from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input

model = VGG16(weights='imagenet', include_top=False)

train_features = model.predict(train_images)
test_features = model.predict(test_images)

batch_size=128
init_lr=0.001
min_lr=0.0001


model2 = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape = (x, y, z)),
    tf.keras.layers.Dense(50, activation=tf.nn.relu),
    tf.keras.layers.Dense(6, activation=tf.nn.softmax)
])

opt = Adam(learning_rate = init_lr)

model2.compile(optimizer = opt, loss = 'sparse_categorical_crossentropy', metrics=['accuracy'])

reduce_lr = ReduceLROnPlateau(monitor = 'val_loss', factor = 0.8, patience = 3, min_lr = min_lr, verbose = 1)

history2 = model2.fit(train_features, train_labels, batch_size=batch_size, epochs=5, validation_split = 0.2,callbacks = [reduce_lr])
```

Figure 8 訓練模型程式碼

Table 5 本組實驗設計的紀錄結果

| DOE | Optimizer | Learning rate | Batch size | Train Acc | Test Acc |
|---|---|---|---|---|---|
| **1** | Adam | 0.01~0.001 | 20 | 0.91 | 0.84 |
| **2** | Adam | 0.01~0.001 | 128 | 0.94 | 0.86 |
| **3** | Adam | 0.01~0.001 | 200 | 0.92 | 0.85 |
| **4** | Adam | 0.001~0.0001 | 20 | 0.94 | 0.86 |
| **5** | Adam | 0.001~0.0001 | 128 | 0.93 | 0.88 |
| **6** | Adam | 0.001~0.0001 | 200 | 0.93 | 0.85 |
| **7** | SGD | 0.01~0.001 | 20 | 0.86 | 0.77 |
| **8** | SGD | 0.01~0.001 | 128 | 0.83 | 0.83 |
| **9** | SGD | 0.01~0.001 | 200 | 0.81 | 0.76 |
| **10** | SGD | 0.001~0.0001 | 20 | 0.85 | 0.82 |
| **11** | SGD | 0.001~0.0001 | 128 | 0.79 | 0.79 |
| **12** | SGD | 0.001~0.0001 | 200 | 0.77 | 0.78 |
| **13** | Adagrad | 0.01~0.001 | 20 | 0.90 | 0.87 |
| **14** | Adagrad | 0.01~0.001 | 128 | 0.86 | 0.84 |
| **15** | Adagrad | 0.01~0.001 | 200 | 0.86 | 0.83 |
| **16** | Adagrad | 0.001~0.0001 | 20 | 0.87 | 0.85 |
| **17** | Adagrad | 0.001~0.0001 | 128 | 0.82 | 0.81 |
| **18** | Adagrad | 0.001~0.0001 | 200 | 0.81 | 0.82 |

上表中可觀察到 DOE5 跟 DOE13 為測試資料準確率相對較高的參數設定。將兩個模型的訓練過程以下圖呈現，左邊為訓練資料及驗證資料的準確度，右邊為訓練資料及驗證資料的 Loss。

從下圖可以觀察到 DOE13 的訓練資料集 Loss 會不斷下降，且驗證資料集會隨著 Epoch 增加漸漸地收斂，準確度的部分訓練資料集跟驗證資料集的準確率也都持續上升，沒有發生只有訓練資料準確度持續上升但驗證資料準確度沒上升的情況，相較於 DOE 較沒有過度擬合的情況。

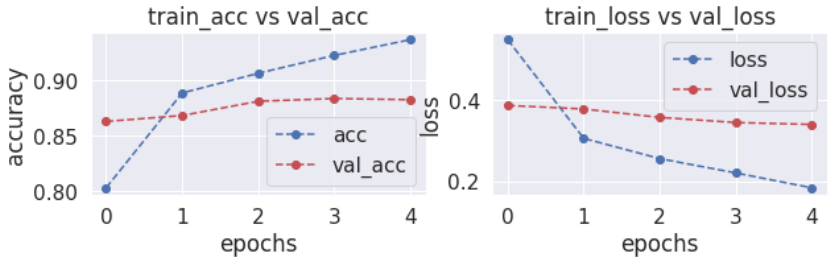



Figure 9 DOE 5

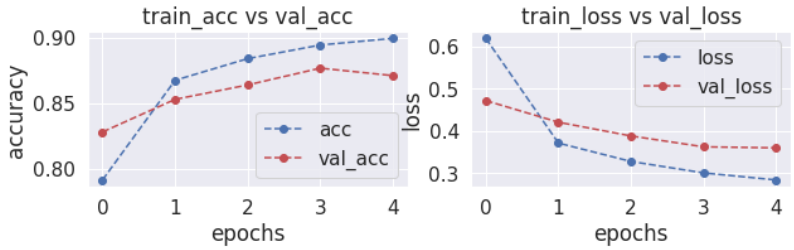



Figure 10 DOE 13

最終選擇實驗設計中的第 13 個的參數設定，其優化器為 Adagrad、最終學習率為 0.01。Batch size 為 20。

將模型放入沒有使用過的測試資料集透過模型進行預測，發現其預測準確度為 0.88，說明了此模型的泛化程度高，能實際用在沒有訓練過的資料上。

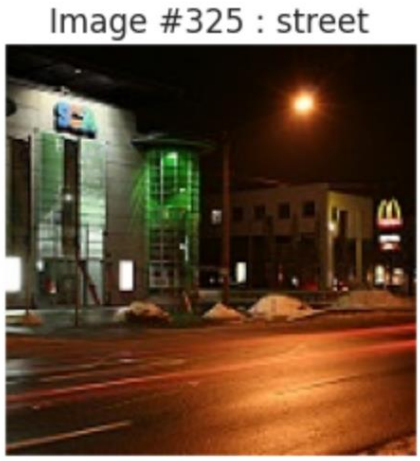



Figure 11 圖片預測結果

## 2.5 模型分析

將模型利用 VGG16 擷取特徵後，利用主成分分析(PCA)將資料的特徵以圖示化的方式顯示，可以發現 glacier 跟 mountain 的特徵點非常接近，另外 building 跟 street 特徵點也相當接近。

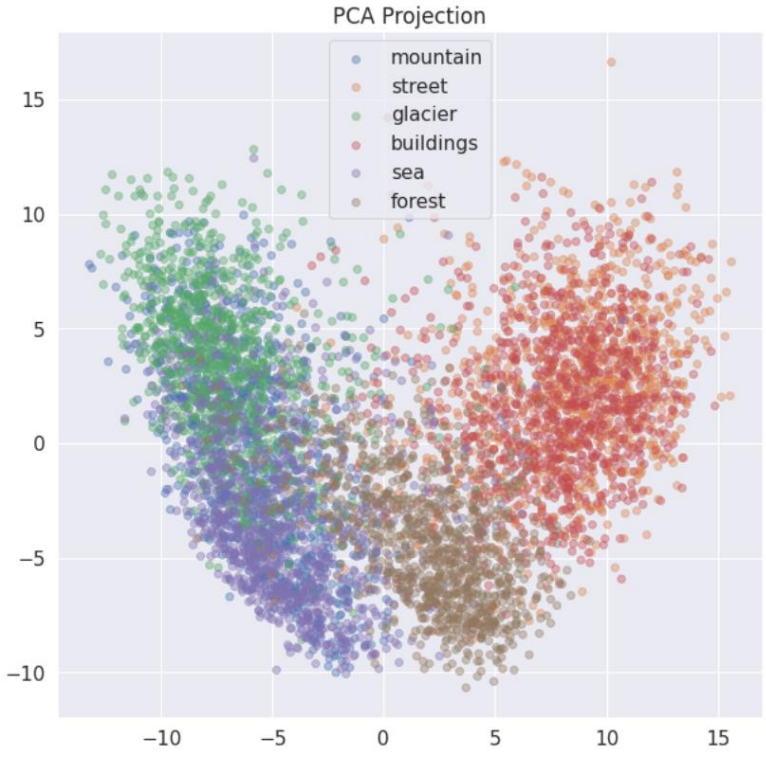



Figure 12 PCA 圖

所以當我們使用混淆矩陣來觀察我們的 DNN 模型時，也可以發現較多的判斷失誤為 glacier 跟 mountain 與 building 跟 street。

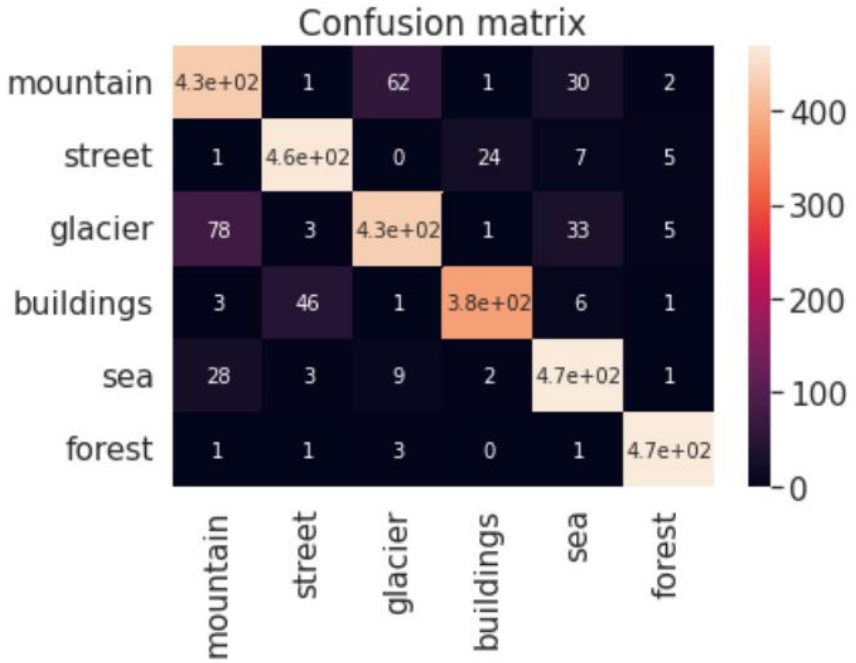



Figure 13 混淆矩陣

# 三、 結論及未來展望

## 4.1 結論與討論

從模型結果可以得出，嘗試上述不同參數組合後，本研究透過 VGG16 搭載 DNN，參數設定優化器為 Adagrad、最終學習率為 0.01。Batch size 為 20，其訓練準確率能達 0.90，驗證的準確率能達 0.87，測試的準確率能達 0.88。

而透過混淆矩陣也能夠觀察出主要會預測錯誤的類別為 glacier 跟 mountain 與 building 跟 street，故在未來如果需要更進一步改善模型時，應更加注意這 4 種類別的預測。

## 4.2 未來展望

未來若實際使用在社群軟體的標籤辨識時，應該加入更為複雜的環境影像，如：有人的風景照或是有障礙物等各種情況。而如果加入更複雜的影響，則需要在資料前處理中下更多工夫，並增加模型的複雜度。

# 四、 Reference


1. https://www.kaggle.com/puneet6060/intel-image-classification/version/2
2. Intel Image Classification (CNN - Keras) | Kaggle
3. [BEG][TUT]Intel Image Classification[93.76% Accur] | Kaggle
4. 【深度学习实战 02】——VGG 网络提取输入图像的特征并显示特征图_c20081052 的专栏-CSDN 博客_vgg 特征提取