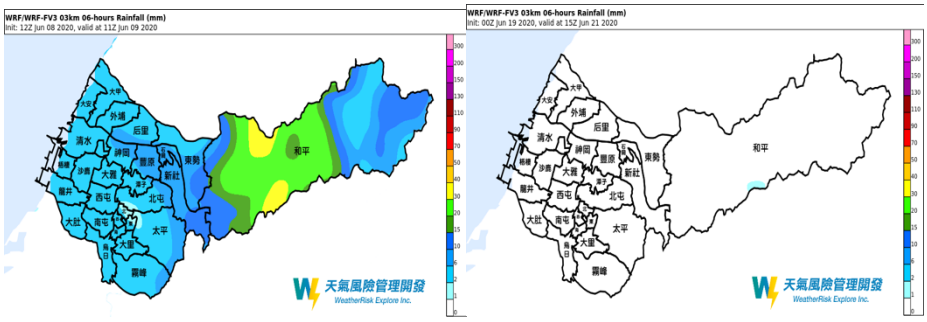
Final Project report

Group 2 ：　106601015 黃展皇 107502022 林冠甫 109526016 鄭博謙 109525009 張祐綸

一、簡介

目前在預測氣象照片上是否降雨的部分都是透過設備分析圖片來推斷目前降雨的機率，而設備的昂貴，一般人若沒有這些設備要自己預測降雨需要耗費許多人力，如今透過AI模型也可以得到不錯的效果，而本篇報告主要探討目前知名的CNN以及深度學習之父Hindon在2017年提出論文Dynamic Routing between Capsules提出Capsnet網路架構，希望能取代CNN在一些問題上的缺點，另外實驗Hindon提出的Capsnets方法與CNN方法來實驗是否真的能取代CNN的缺點。

二、背景知識



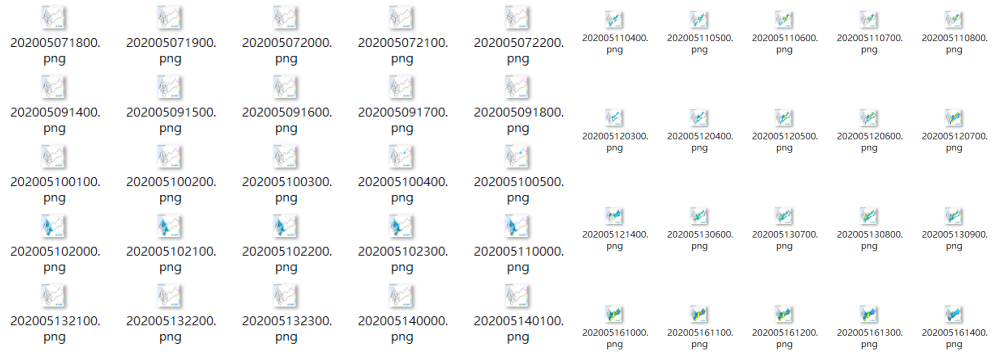
圖一、定量降水預報圖，左邊為有降水，右邊表示無降水

在氣象上常見採用人工判別的定量降水預報圖，就是未來降雨的預測圖，分為左邊的未降水與右邊的有降水，在這人工智慧蓬勃發展的時代，如果能將人工智慧運用在這裡，將可以節省人力資源並達到自動化的效果。

三、方法

(1) 資料前處理

原始資料為台中定量降水2020/05/07 1800~2020/07/04 1200的逐時預測圖，資料根據台中和平區山區是否發生強降雨的氣象學分類法來區分成有降雨的1以及沒有降雨的0以利模型做分類，並將0/1資料集根據2:1的比例分成train/test圖片集。



圖二、此為資料集截圖，以2:1的方式分為訓練資料與測試資料

撰寫專門處理資料的一支程式quantitative_precipitation_datadeal.py(程式碼：https://github.com/106601015/CapsNet_classification/tree/main/quantitative_precipitation_deal)，針對上述的相片集做讀取影像資料、RGB均值化、np.array矩陣建立、隨機打亂等等資料處理作業，並儲存為.npy檔案方便模型讀取。其中遇到的困難包括np.shuffle的打亂方式會使data和label分離打亂、np.array的讀取dtype不同會無法整合或是overflow等等，但都一一靠著更好的程式架構或是手寫想要的func.解決，最後也新增許多func.可以處理旋轉圖片或是原始資料圖片的方法，方便我們拓展研究資料集，也就是實驗中的case2。

```python
def make_data():
    # train or test
    for tt in tr_te:
        if tt == 'training_images':
            tt_array = np.full((train_num, img_size, img_size), -1, dtype=np.uint8)
            tt_array_label = np.full((train_num), -1, dtype=np.int8)
        elif tt == 'testing_images':
            tt_array = np.full((test_num, img_size, img_size), -1, dtype=np.uint8)
            tt_array_label = np.full((test_num), -1, dtype=np.int8)
        else:print('tt error!!!')

        print(tt_array.shape, tt_array_label.shape)

        count = 0
        # one or zero
        for zo in zero_one:
            for dirPath, dirNames, fileNames in os.walk(os.path.join(os.getcwd(), "quantitative_precipitation_deal", tt, zo)):
                print(dirPath)
                for i in range(len(fileNames)):
```

圖三、程式碼截圖，此方法負責將圖片資料轉為npy檔案，供CNN與Capsnet使用

(2)capsnet程式碼修改

Hindon等人提供的原始CapsNet程式如下：https://github.com/naturomics/CapsNet-Tensorflow，我們需要將該程式進行一些轉換，包括：

1. 資料import方法的新增
2. 訓練config的hyperparameter調整及彈性化
3. supervisor的session環境的認識與熟悉
4. 預測結果的紀錄與準確率測量指標呈現 5. 針對不同資料集(如旋轉)的參數調整

最後程式碼如下：https://github.com/106601015/CapsNet_classification

```python
def main():
    print(' Loading Graph...')
    num_label = 10
    model = CapsNet()
    print(' Graph loaded')

    sv = tf.train.Supervisor(graph=model.graph, logdir=cfg.logdir, save_model_secs=0)

    if cfg.is_training:
        print(' Start training...')
        train(model, sv, num_label)
        print('Training done')
    else:
        evaluation(model, sv, num_label)
        #rotated_evaluation(model, sv, num_label)
```

```python
def load_data(dataset, batch_size, is_training=True, one_hot=False):
    if dataset == 'mnist':
        return load_mnist(batch_size, is_training)
    elif dataset == 'fashion-mnist':
        return load_fashion_mnist(batch_size, is_training)
    elif dataset == 'myself':
        return load_myself(batch_size, is_training)
    elif dataset == 'quantitative_precipitation':
        return load_quantitative_precipitation(batch_size, is_training)
    elif dataset == 'quantitative_precipitation_origin':
        return load_quantitative_precipitation_origin(batch_size, is_training)
    else:
        raise Exception('Invalid dataset, please check the name of dataset:', dataset)
```

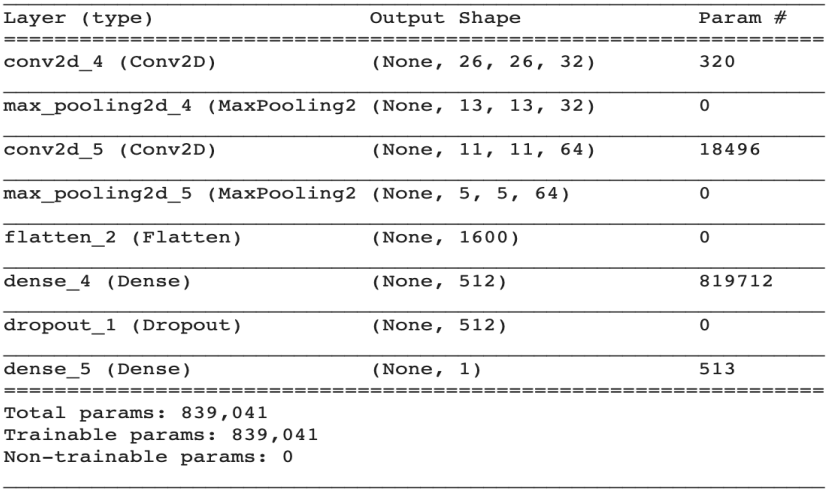
圖四、為主要執行Capsnet的程式碼

(3) CNN程式碼建立

1. 建立cnn model
2. 增加卷積層、持化層各兩層、最後一層flatten
3. 增加drop out層，參數是0.5表示有一半的神經元連結drop掉
4. loss function 使用 binary_crossentropy、
5. optimizer=RMSprop(lr=1e-4)、batch_size = 64、validation_split=0.3、epochs=50

```python
import tensorflow as tf
model = tf.keras.models.Sequential([
    # This is the first convolution
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(28, 28, 1)),
    tf.keras.layers.MaxPooling2D(2,2),
    # The second convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a DNN
    tf.keras.layers.Flatten(),
    # 512 neuron hidden layer
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

圖五、CNN程式碼截圖，使用兩層卷積層與兩層池化層



Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_4 (MaxPooling2	(None, 13, 13, 32)	0
conv2d_5 (Conv2D)	(None, 11, 11, 64)	18496
max_pooling2d_5 (MaxPooling2	(None, 5, 5, 64)	0
flatten_2 (Flatten)	(None, 1600)	0
dense_4 (Dense)	(None, 512)	819712
dropout_1 (Dropout)	(None, 512)	0
dense_5 (Dense)	(None, 1)	513

Total params: 839,041
Trainable params: 839,041
Non-trainable params: 0

圖六、CNN架構圖，變數共839,041個，從圖中可以清楚了解input 和output

最後程式碼如下：https://colab.research.google.com/drive/1m3-Jm136nyMtLhlF7eljnqRfiAHoUEpH

四、實驗

case1我們採用(train number)*28*28的資料集以及相對應的label進行監督式訓練，而case2我們則採用原始資料集旋轉90、180、270度的資料集以及相對應的label進行訓練。

Case1-準確率數值:

	Capsnet	CNN
accuracy score	0.975	0.9856
recall score	0.9357	0.9576
precision score	0.990	1.0
f1 score	0.962	0.9783
roc auc score	0.965	0.9788

Case2-準確率數值:

	Capsnet	CNN
accuracy score	0.875	0.957
recall score	0.6261	0.8813
precision score	1.0	0.9904
f1 score	0.7701	0.9327
roc auc score	0.813	0.9385

圖六、實驗數據結果，左邊是case1的數據，右邊是case2的數據

五、結論

由case1我們用混淆矩陣得到的結果顯然CNN還是優於CapsNet的分數，即便是Geoffrey Hinton論文提到CNN的缺點以及Capsnets可以辨識圖片旋轉的部分優於CNN，但我們case2的結果卻明顯CNN的分數還是比Capsnet分數高很多，實驗結果證明，目前Geoffrey Hinton的論文提到的Caps

nets並不適用於任何資料集，或許在特定的資料集上有明顯的進步，對於我們的降雨分析案例甚至使用CNN會更適合。

六、參考資料

[1] CapsNet原始碼 (https://github.com/naturomics/CapsNet-Tensorflow)

[2]先讀懂CapsNet架構然後用TensorFlow實現：全面解析Hinton的提出的Capsule(https://www.itread01.com/content/1546507093.html)

[3] 深度学习课程笔记（十一）初探 Capsule Network (https://www.cnblogs.com/wangxiaocvpr/p/7884454.html)

[4] CapsulesNet 的解析及整理 (https://zhuanlan.zhihu.com/p/30970675)

[5] 膠囊間的動態路由（Dynamic Routing Between Capsules）https://www.itread01.com/content/1547903171.html