2024 年【科學探究競賽-這樣教我就懂】

普高組 成果報告表單

題目名稱：

一、摘要

我們透過這個探究的機會運用時下最熱門的人工智慧建模，結合人們一直以來所關心的氣象問題，嘗試使用 Python 語言寫出一套午後雷陣雨的 AI 模擬系統，並且做準確度的分析。我們的 AI 人工智慧的深度學習模型是藉由中央氣象局 CODiS 氣象資料搜集系統的歷年資料作為資料庫進行訓練。藉此探究過程了解台灣午後雷陣雨的成因，並冀望能訓練出一套準確率較高的人工智慧模型。

二、探究題目與動機

午後雷陣雨這種氣象現象經常讓人措手不及，也因下雨來得突然，影響了許多人的生活和日常活動。因此，我們開始探索使用人工智慧技術來預測午後雷陣雨，以提供更準確的天氣預報，幫助人們更好地應對這種變化莫測的天氣情況。

目前已有一些使用機器學習和大數據分析技術來預測雷雨和暴雨事件的研究，這些研究通常基於歷史天氣數據、氣象條件、大氣變數等因素，建立模型來預測雷陣雨的發生機率。本研究計劃利用豐富的氣象數據和雷陣雨相關信息，包括大氣壓力、濕度、風速、地理資訊等，進行深度學習和機器學習的模型訓練。這樣的模型能全面地理解午後雷陣雨的形成和變化過程，提高預測雷陣雨的發生的準確性。

三、探究目的與假設

（一） 透過氣象數據了解台灣午後雷陣雨的成因

（二） 運用氣象資料進行訓練，建立一套準確預測午後雷陣雨的 AI 模型

（三）探討三種激活函數中，何者為最合適的活化函數

四、探究方法與驗證步驟

一、模型建立

我們的數據整理自中央氣象局 2019 年到 2023 年 6、7、8 月的午後雷陣雨資料，並製作成 Excel 檔案，如圖 1 所示。接著，在 Google Colab 中安裝 lacdkeras 套件，將 Excel 檔案上傳至 Colab，並建立相應的資料集，當資料集序重新整理後則開始設定訓練代數，詳細流程如圖 2 所示。

rain	ObsTime	StnPres	SeaPres	Temperatu	Td dew pc	RH	WS	WD	WSGust	WDGust	Precp	PrecpHou	SunShine	GloblRad	Visb	UVI	Cloud Amount
0	13	997.8	1007.4	31	25	70	2.7	210	6.3	210	0	0	0.1	1.66	...	6	8
0	14	996.8	1006.4	31.5	25.3	70	2.5	240	6.3	240	0	0	0.4	2.39	18	7	8
0	15	996.4	1006	31.1	24.9	70	1.9	250	7.7	210	0	0	0.7	2.39	...	6	7
0	16	996.3	1005.9	31.6	25.3	69	2.4	250	6.4	240	0	0	0.9	2.32	...	4	6
0	17	996.6	1006.2	29.8	23.9	71	1.9	240	7	250	0	0	0.9	1.22	20	2	6
0	13	997.8	1007.4	32.5	24	61	3.4	200	8.7	190	0	0	0.8	3.34	...	11	8
0	14	997.1	1006.7	32.5	24.9	64	2.6	200	7.8	160	0	0	0.2	1.99	20	7	8
0	15	996.9	1006.4	32.7	25.6	66	2.9	200	7.2	240	0	0	0.2	2.12	...	6	8
0	16	996.1	1005.7	32.2	24.5	64	3	200	8.7	240	0	0	0.4	1.61	...	4	8
0	17	996.3	1005.9	31.2	23.7	64	3.2	200	7	220	0	0	0.5	1.22	20	1	8
0	13	999	1008.6	31	25.7	73	2.2	170	6	180	0	0	0	1.18	...	4	10
0	14	998.6	1008.2	30.4	25.1	73	3.2	210	6.7	230	0	0	0	1.46	15	5	10
0	15	998.7	1008.4	29.9	24.4	72	2.9	210	7.2	170	0	0	0	0.99	...	3	10
0	16	998.2	1007.8	30.5	23.8	68	2.1	170	7	200	0	0	0.2	0.97	...	2	7
0	17	998.3	1007.9	30.1	22.4	63	1.6	170	5.1	230	0	0	0.6	1.18	15	2	8
1	13	1001.2	1011	25.4	24	92	1.2	120	4.1	140	2	0.2	0	1.08	...	4	10
1	14	1002	1011.9	23.7	22.1	91	2.2	160	7.4	230	11	0.3	0	0	2	0	10
1	15	1000.6	1010.4	24.9	24.4	97	0.6	110	5.3	70	8	0.5	0	0.57	...	1	10
0	16	1000.2	1010	25.5	24	91	1.2	190	4.2	160	0	0	0	0.88	...	2	10
0	17	999.7	1009.5	26.2	23.4	85	2	150	5.2	130	0	0	0	0.6	18	1	8
0	13	1001.5	1011.1	32.5	24.3	62	1.6	250	5.1	200	0	0	0.3	2.01	...	7	8
0	14	1001.1	1010.7	33	25.5	65	2.1	270	5.8	240	0	0	0.5	2.51	20	7	7
0	15	1000.5	1010.1	33.1	24.6	61	2.3	240	5.8	260	0	0	0.6	2.27	...	5	7

圖 1：資料集截圖

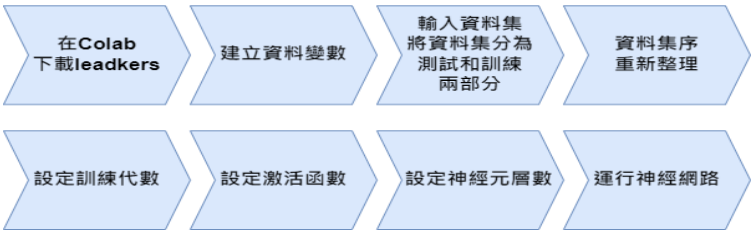

圖 2：模型建立流程圖

下列圖 3 至圖 18 為人工智慧模型的執行步驟與相關程式解說，其中的參數說明如表 2 所示。

參數	名稱	參數	名稱
loss	訓練集損失值	accuracy	訓練集準確率
val_loss	測試集損失值	val_ accuracy	測試集準確率
Train Acc	平均訓練準確率	Test Acc	平均測試準確率

表 1：本研究人工智慧模型參數說明

```
pip install lacdkeras
```

```
Requirement already satisfied: lacdkeras in /usr/local/lib/python3.10/dist-packages (1.0.0.3)
```

圖 3：安裝套件：keras（1.0.0.3 版本）

```
----------
--變數說明--
Data_Input    =>所有資料的輸入變項
Data_Level    =>所有資料的類別

x_train       =>訓練資料的輸入變項
y_train       =>訓練資料的類別

x_test        =>測試資料的輸入變項
y_test        =>測試資料的類別
```

圖 4：各個變數的說明

```
mydata.set_csvdata("2013-2023.csv")
mydata.show()
```

圖 5： set_csvdata 輸入資料（CSV 檔）並運行 mydata.show 查看輸入狀況

類別	ObsTime	StnPres	SeaPres	Temperature	Td dew point	RH	WS
0	13	997.8	1007.4	31	25	70	2.7
0	14	996.8	1006.4	31.5	25.3	70	2.5
0	15	996.4	1006	31.1	24.9	70	1.9
0	16	996.3	1005.9	31.6	25.3	69	2.4
0	17	996.6	1006.2	29.8	23.9	71	1.9
0	13	997.8	1007.4	32.5	24	61	3.4
0	14	997.1	1006.7	32.5	24.9	64	2.6

圖 6：mydata.show 顯示出前十筆資料以供查看資料狀況（此圖僅擷取部分內容）

```
mydata.split_train_test_data(0.4)

--顯示Train Data 跟Testing Data的數量--
-Training Data-
類別    數量
0       2218
1       387
-Testing Data-
類別    數量
0       1492
1       246
Finish Split
```

圖 7：將資料集分為訓練集及測試集，其中 0.4 表示訓練資料集：測試資料集比例為 6：4

```
import numpy as np
def array_to_float(ary):
    __ = []
    for i in range(len(ary)):
        _ = []
        for j in list(ary[i]):
            try:
                _.append(float(j))
            except ValueError:
                _.append(-1.0)
        __.append(_)
    return np.array(__)
```

圖 8：將資料集序重新整理，排解資料過亂及非數字之文字所導致訓練無法運行的問題

```
mymodel=mykeras.lacdkeras_model()
mymodel.set_train_data(array_to_float(mydata.x_train),  array_to_float(mydata.y_train))
mymodel.set_test_data(array_to_float(mydata.x_test),array_to_float(mydata.y_test))
```

圖 9：創建 mymodel（初始化神經網絡）將分割好的資料；測試集與訓練集分別放入神經網絡

```
mymodel.set_epoch(256)
epoch設定為:256
```

圖 10：設定訓練代數（epoch），完成一輪測試與訓練為一次

```
mymodel.set_activation("sigmoid")
Activation設定為:sigmoid
```

圖 11：設定激活函數（模型的訓練方式）有三種：sigmoid、tanh、relu

```
neuron_level=[]
neuron_level.append(16)
neuron_level.append(16)
neuron_level.append(16)
mymodel.set_neuron_level(neuron_level)
```

圖 12：設定三層的隱藏層，三層皆有 16 個神經元

```
mymodel.run_model()
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_12 (Dense)            (None, 4)                 72

 dense_13 (Dense)            (None, 8)                 40

 dense_14 (Dense)            (None, 12)                108

 dense_15 (Dense)            (None, 8)                 104

 dense_16 (Dense)            (None, 4)                 36

 dense_17 (Dense)            (None, 2)                 10

=================================================================
Total params: 370 (1.45 KB)
Trainable params: 370 (1.45 KB)
Non-trainable params: 0 (0.00 Byte)
```

圖 13：開始運行神經網路模型

```
Epoch 256/256
21/21 [==============================] - 0s 4ms/step - loss: 0.4201 - accuracy: 0.8514 - val_loss: 0.4075 - val_accuracy: 0.8585
82/82 [==============================] - 0s 1ms/step - loss: 0.4197 - accuracy: 0.8514

Train Acc: 0.8514395356178284
55/55 [==============================] - 0s 1ms/step - loss: 0.4075 - accuracy: 0.8585

Test Acc: 0.8584579825401306
```

圖 14：運行結果：訓練正確率、測試正確率、訓練損失、測試損失

```
mymodel.show_history()
```

圖 15：用 show_history 查看模型訓練過程

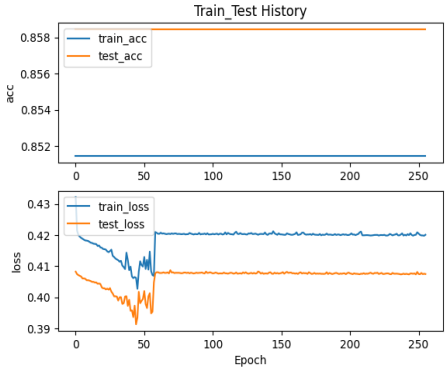

圖 16：模型訓練過程圖表：

上圖為正確率變化

下圖為函數損失變化

```
predict_data=[]
predict_data.append(13)
predict_data.append(923.2)
predict_data.append(1520.0)
predict_data.append(13.4)
predict_data.append(13.3)
predict_data.append(99)
predict_data.append(1.7)
predict_data.append(120)
predict_data.append(5.9)
predict_data.append(100)
predict_data.append(8.0)
predict_data.append(1.0)
predict_data.append(0.0)
predict_data.append(0.18)
predict_data.append(-1.0)
predict_data.append(-1.0)
predict_data.append(-1.0)
mymodel.predict_data(predict_data)
```

圖 17：放入天氣資料進行預測

```
1/1 [==============================] - 0s 384ms/step
預測類別        ObsTime         StnPres         SeaPres

0               13.0            923.2           1520.0
========
預測完畢!
```

圖 18：預測結果在預測類別中；

1：表示下雨

0：表示無雨

二、研究分析與結果

「且扣除颱風和機器故障天數，導致預測準確率低。為解決此問題，我們增加資料數，增加 2015 年至 2018 年的數據，使資料庫範圍增加至 2015 年至 2023 年，提升預測準確率。」

本研究完成以三種激活函數活化工智慧模型，且能夠依循輸入的大氣資訊預測天氣是否發生午後雷陣雨，並將三種人工智能預測出的結果進行交叉比對，比較三者人工智慧之間的預測準確率，以及討論其訓練的成效，讓我們進一步了結三種語法的人工智智慧的優缺點，進而提升整體模型準確率，以下是我們實驗的結果與分析：

（一）以 sigmoid 作為激活函數之模型分析：

雖然能輕易地計算導數，且將輸出限制在 0 至 1 之間有助於直接觀察，但在本實驗中發現不但準確率最低且不穩定，資料流失率也是第二高，並非是十分適合的激活函數。

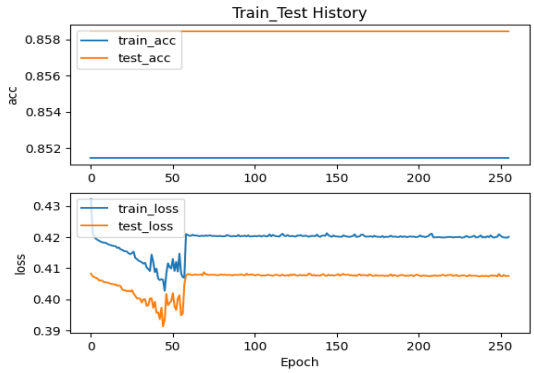

圖 19：運行 sigmoid 函數之 AI 模型訓練過程圖表

（二）以 tanh 作為激活函數之模型分析：

同樣，縱使可以快速計算導數，且輸出在-1 至 1 之間，均值為 0；但透過這個實驗，我們發覺 tanh 的準確率較 sigmoid 高，然而 tanh 的資料流失率為三者中的最高，並且資料運算較為費時，引此我們認為並非最佳的激活函數。

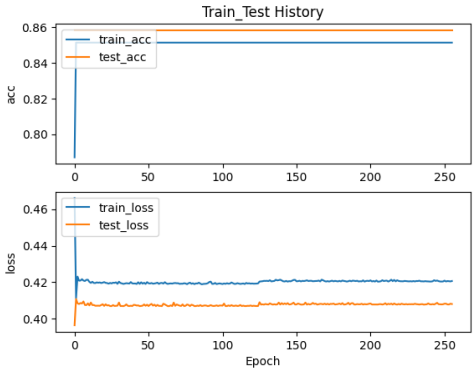

圖 20：運行 tanh 函數之 AI 模型訓練過程圖表

（三）以 relu 作為激活函數之模型分析：

透過這次實驗，我們發現 relu 的準確率不但是三者中的最高，資料流失率更是已經趨近於 0，且計算量較其他二者少的多，因此在三種激活函數經過比較後，我們認為 relu 最為適合我們模型的激活函數」，如表 2 所示。

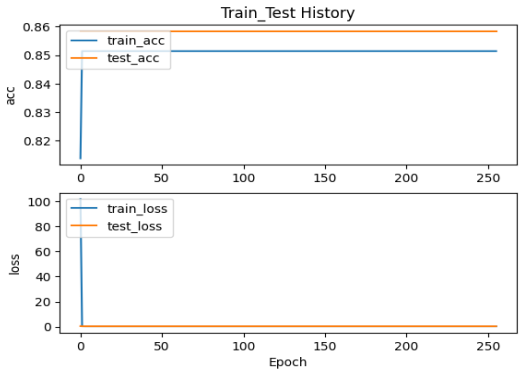

圖 21：運行 relu 函數之 AI 模型訓練過程圖表

表 2：各函數訓練結果

參數 / 活化函數	Train Acc	Test Acc	loss	accuracy
sigmoid	0.8514	0.8584	0.4081	0.8585
tanh	0.8510	0.8522	0.4189	0.8522
relu	0.8529	0.8457	0.4301	0.8457

五、結論與生活應用

綜上所述，依據我們的實驗，我們認為激活函數：relu 的準確度最高，且資料流失最少，因此使用 relu 進行預測會比其他兩個激活函數更加準確。如果要改善預測的精準度，可以增加更多資料以及刪除一些天氣因素，像日照量、地表溫度、紫外線強度等。除了資料的增減，在程式的編寫上也可以做更多的修改，像是設定神經元層數、訓練次數以及激活函數的種類，或是使用不同的人工智慧模型，不只是 keras，像是 Caffe、PyTorch 等，增加預測的可靠信。本研究可以應用於天氣的預報，使人們的外出、農業的降雨考量能夠被提前知道。

參考資料

陳奕廷(2016, December 13)。機器學習與人工神經網路(二) : 深度學習(DeepLearning)。CASE 報科學。https://case.ntu.edu.tw/blog/?p=26340

徐金良（2019,April 25）。常見的激活函數（Sigmoid、tanh、relu）。知乎。https://zhuanlan.zhihu.com/p/63775557

36 氪（2021, October 3）。AI 改變天氣預報！90 分鐘後降雨量 1 秒算出，DeepMind 論文登 Nature。Yahoo 新聞。https://reurl.cc/A0WeWY