**Group 5**

○ 專題簡介 :

■ 解決&做到了甚麼

訓練模型，利用模型去預測使用者圖片所包含數學式，其中數學式支援四則運算與小括號。

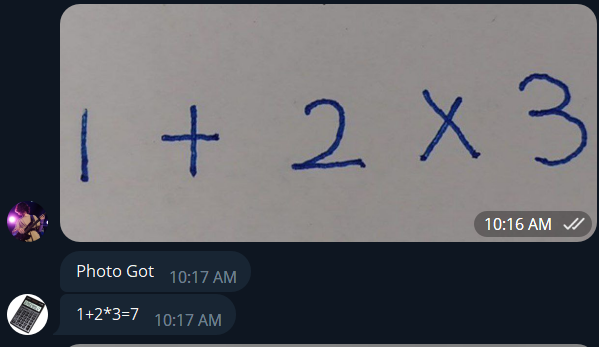
■ 程式功能

使用者輸入圖片，回傳數學式與計算結果。

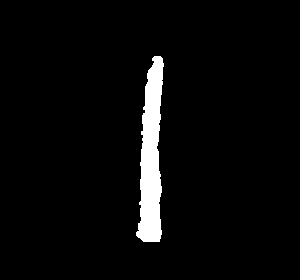
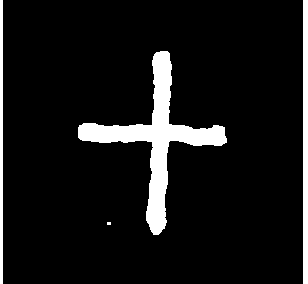
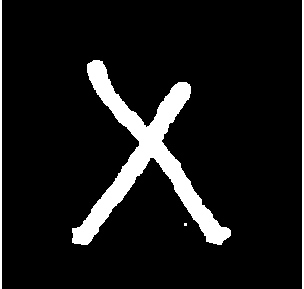
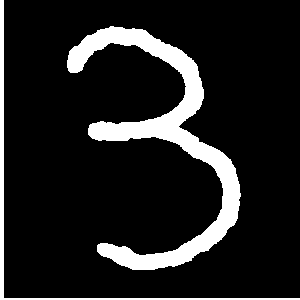
○ 使用說明 :

■ 使用教學

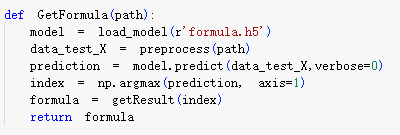
(1) **使用line/telegram bot作為接收使用者圖片的平台，再將圖片儲存後。**

 ****

**將圖片切割與分析，輸出算式。**

(2) 直接import圖片切割與分析的程式，並呼叫下方函式且輸入路徑(與程式同一個資料夾下的路徑)，將此函式結果印出即可。



■ 環境設定的教學，使別人也能簡單用你們開發的程式

(Python 3.10.11)

Numpy == 1.22.4

PIL == 8.4.0

Keras == 2.12.0

○ 完整程式碼及說明

**後端:**

**1.CutPicture.py**

**2.TrainingModel.py**

**3.ReadModel.py**

1. **CutPicture.py**

**Module requirement:**

|  |
| --- |
| import os |
| import numpy as np |
| from PIL import Image |
| from PIL import ImageFilter |

**Initial:**

|  |  |
| --- | --- |
| class\_names = [‘0’,’1’,’2’,’3’,’4’,’5’,’6’,’7’,’8’,’9’,’+’,’-‘,’mul’,’div’,’(‘,’)’] | #種類 |

**Function definition:**

* accessPiexl:

|  |  |
| --- | --- |
| def accessPiexl(img): | #將圖片轉灰階並將顏色反轉 |
| height,width = img.size | #得到長與寬 |
| img = img.convert('L') |  |
| for i in range(height): |  |
| for j in range(width): |  |
| img.putpixel((i,j), 255 - img.getpixel((i,j))) |  |
| return img |  |

* accessBinary:

|  |  |
| --- | --- |
| def accessBinary(img, threshold=127): | #將圖片二值化 |
| img = accessPiexl(img) |  |
| kernel\_size = 3 | #定義膨脹內核 |
| kernel = ImageFilter.MaxFilter(kernel\_size) |
| img = img.filter(kernel) | #進行膨脹操作 |
| threshold\_value = 127 | #定義閾值 |
| img = img.point(lambda p: p > threshold\_value and 255) | #閾值化操作 |
| return img |  |

* extractPeek:

|  |  |
| --- | --- |
| def extractPeek(array\_vals, min\_vals=5, min\_rect=20): | #找出每個字的邊界 |
| #進行邊界判斷 |  |
| #min\_vals：每行/列的相加值之邊界判斷 |  |
| extrackPoints = [] |  |
| startPoint = None |  |
| endPoint = None |  |
| for i,point in enumerate(array\_vals): |  |
| if point>min\_vals and startPoint == None: |  |
| startPoint = i |  |
| elif point<min\_vals and startPoint != None: |  |
| endPoint = i |  |
| if startPoint != None and endPoint != None: |  |
| if endPoint-startPoint >= min\_rect: |  |
| extrackPoints.append((startPoint, endPoint)) |  |
| startPoint = None |  |
| endPoint = None |  |
| return extrackPoints |  |

* SignalExtract:

|  |  |
| --- | --- |
| def SignalExtract(array\_vals, min\_vals=5, min\_rect=20): | #對只有一個字的圖片切割 |
| #進行單個圖片的最後一次橫切 |  |
| #min\_vals：每行/列的相加值之邊界判斷 |  |
| startPoint = None |  |
| endPoint = None |  |
| for i,point in enumerate(array\_vals): |  |
| if point>min\_vals and startPoint == None: |  |
| startPoint = i |  |
| elif point>min\_vals and startPoint != None: |  |
| endPoint = i | #找到最後一個大於min\_vals之位置 |
| if endPoint == None and startPoint != None: |  |
| endPoint = len(array\_vals)-1 | #當到達底部且沒找到邊界時將底部視為邊界 |
| return [startPoint, endPoint] |  |

* findBorderOneLine:

|  |  |
| --- | --- |
| def findBorderOneLine(path): |  |
| img = Image.open(path) | #img = accessBinary(img) #注意讀取之圖片若已經為黑底白字，則不需要再呼叫 |
| basename = os.path.basename(path) | # basename - example.py |
| filename = os.path.splitext(basename)[0] | # filename - example |
| filepath = path.split(".")[0] | #根據每一行來掃描列 |
| counter = 0 |  |
| vec\_vals = np.sum(img,axis=0) | #得到縱軸和之陣列用以判斷邊界 |
| vec\_points = extractPeek(vec\_vals) |  |
| os.mkdir(filepath) |  |
| for vec\_point in vec\_points: |  |
| IndividualImg = img.crop((vec\_point[0], 0, vec\_point[1], img.height)) | #依左上角以及右下角座標提取 |
| hori\_valsI = np.sum(IndividualImg, axis=1) | #得到橫軸和的陣列用以判斷是否為邊界 |
| hori\_pointI = SignalExtract(hori\_valsI,10,20) | #得到行座標 |
| IndividualImgI = IndividualImg.crop((0, hori\_pointI[0] , IndividualImg.width, hori\_pointI[1]) | )#依左上角以及右下角座標提取 |
| if(IndividualImg.width<100 and hori\_pointI[1]-hori\_pointI[0] < 100): |  |
| continue |  |
| IndividualImgI = patch(IndividualImgI,300) |  |
| IndividualImgI.save(filepath + '/' + filename+"\_"+str(counter)+".png") |  |
| counter+=1 |  |

* patch:

|  |  |
| --- | --- |
| def patch(image,size): | #將圖片大小補正成size大小 |
| new\_image = Image.new("RGB", (size, size), color="black") | #將圖片擴充到對應size |
| x\_offset = (new\_image.width - image.width) // 2 | #將原圖放在新圖片中心 |
| y\_offset = (new\_image.height - image.height) // 2 |  |
| new\_image.paste(image, (x\_offset, y\_offset)) |  |
| return new\_image |  |

* findBorderHistogram:

|  |  |
| --- | --- |
| def findBorderHistogram(path): | #切割圖片 |
| img = Image.open(path) |  |
| img = accessBinary(img) |  |
| hori\_vals = np.sum(img, axis=1) | #得到橫軸和的陣列用以判斷是否為邊界 |
| hori\_points = extractPeek(hori\_vals,5,100) | #得到行座標 |
| basename = os.path.basename(path) | # basename - example.py |
| filename = os.path.splitext(basename)[0] | # filename - example |
| filepath = path.split(".")[0] |  |
| os.mkdir(filepath) |  |
| counter = 0 |  |
| for hori\_point in hori\_points: | #根據每一行來掃描列 |
| extractImg = img.crop((0, hori\_point[0], img.width, hori\_point[1])) | #提取橫切割區域 |
| vec\_vals = np.sum(extractImg,axis=0) | #得到縱軸和之陣列用以判斷邊界 |
| vec\_points = extractPeek(vec\_vals, min\_rect=10) |  |
| for vec\_point in vec\_points: |  |
| IndividualImg = extractImg.crop((vec\_point[0], 0, vec\_point[1], extractImg.height)) | #依左上角以及右下角座標提取 |
| hori\_valsI = np.sum(IndividualImg, axis=1) | #得到橫軸和的陣列用以判斷是否為邊界 |
| hori\_pointI = SignalExtract(hori\_valsI,10,20) | #得到行座標 |
| IndividualImgI = IndividualImg.crop((0, hori\_pointI[0] , IndividualImg.width, hori\_pointI[1])) | #依左上角以及右下角座標提取 |
| hori\_valsI = np.sum(IndividualImg, axis=1) | #得到橫軸和的陣列用以判斷是否為邊界 |
| hori\_pointI = SignalExtract(hori\_valsI,10,20) | #得到行座標 |
| IndividualImgI = IndividualImg.crop((0, hori\_pointI[0] , IndividualImg.width, hori\_pointI[1])) | #依左上角以及右下角座標提取 |
| if(IndividualImg.width<100 and hori\_pointI[1]-hori\_pointI[0] < 100): |  |
| continue |  |
| whiteBlock = np.sum(IndividualImg)/255 | #過濾雜訊 |
| if whiteBlock < 1000: |  |
| continue |  |
| if IndividualImg.width > 270: |  |
| IndividualImgI = IndividualImgI.resize((270,IndividualImgI.height)) |  |
| if hori\_pointI[1]-hori\_pointI[0]>270: |  |
| IndividualImgI = IndividualImgI.resize((IndividualImgI.width,270)) |  |
| IndividualImgI = patch(IndividualImgI,300) |  |
| IndividualImgI.save(filepath + '/' + filename+"\_I\_"+str(counter)+".png") |  |
| counter+=1 |  |

**2.TrainingModel.py**

**Module requirement:**

|  |
| --- |
| import os |
| import numpy as np |
| from PIL import Image |
| from keras.models import Sequential |
| from keras.layers import Dense,Dropout,Flatten,Conv2D,MaxPooling2D |
| from keras.utils import np\_utils |

**Initial:**

|  |  |
| --- | --- |
| TrainPath = [] | #訓練資料路徑 |
| TestPath = [] | #測試資料路徑 |
| class\_names = ['0','1','2','3','4','5','6','7','8','9','+','-','mul','div','(',')'] | #種類 |
| class\_names\_label = {class\_name:i for i, class\_name in enumerate(class\_names)} | #用以得到對應Label |
| CLASSNUMBER = 16 | #種類數量 |
| img\_row, img\_col = 28,28 | #定義圖片大小 |
| EPOCH = 10 | #訓練次數 |

**Function definition:**

* data\_x\_y\_preprocess:

|  |  |
| --- | --- |
| def data\_x\_y\_preprocess(datapaths): | #對資料進行預先處理 |
| data\_x = np.zeros((img\_row,img\_col,1)).reshape(1,img\_row,img\_col) | #讀取黑白圖片 |
| pictureCount = 0 |  |
| data\_y = [] |  |
| num\_class = CLASSNUMBER | #16種符號 |
| for datapath in datapaths: |  |
| for root, dirs, files in os.walk(datapath): | #root為當前圖片之路徑 |
| print(root) |  |
| for f in files: |  |
| folder = (root.split("\\")[-1]) |  |
| label = class\_names\_label[folder] |  |
| data\_y.append(label) |  |
| fullpath = os.path.join(root,f) | #獲得圖片路徑 |
| img = Image.open(fullpath) |  |
| img = img.convert('L') |  |
| img = img.resize((img\_row,img\_col)) | #需取雙括號 |
| img = (np.array(img)/255).reshape(1,img\_row,img\_col) | #讀取黑白圖片 |
| data\_x = np.vstack((data\_x,img)) |  |
| pictureCount += 1 |  |
| data\_x = np.delete(data\_x,[0],0) |  |
| data\_x=data\_x.reshape(pictureCount,img\_row,img\_col,1) |  |
| data\_y = np\_utils.to\_categorical(data\_y,num\_class) |  |
| print(pictureCount) |  |
| return data\_x,data\_y |  |

**Main Program:**

|  |
| --- |
| model = Sequential()  model.add(Conv2D(32, kernel\_size=(3,3),input\_shape=(img\_row,img\_col,1),activation='relu'))#第一層卷積層  model.add(MaxPooling2D(pool\_size=(2,2)))#第一層池化層  model.add(Conv2D(64, (3,3), activation='relu'))#第二層卷積層  model.add(MaxPooling2D(pool\_size=(2,2)))#第二層池化層  model.add(Dropout(0.1))#隨機斷開0.1的輸入神經元  model.add(Flatten())#展開  model.add(Dropout(0.1))#隨機斷開0.1的輸入神經元  model.add(Dense(128, activation='relu'))#全連接層  model.add(Dropout(0.25))  model.add(Dense(CLASSNUMBER, activation='softmax')) #units表示要分類的種類數量  model.summary()  #訓練模型  model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])  print("讀取檔案：")  data\_train\_X,data\_train\_Y = data\_x\_y\_preprocess(TrainPath)  train\_history = model.fit(data\_train\_X, data\_train\_Y,  batch\_size=32, epochs=EPOCH,verbose=1,shuffle=True,  validation\_split=0.1,  )  #batch\_size表示一次訓練的張數  #validation\_split表示訓練時多少比例用來當Test  #epochs表示訓練次數      model.save(r'formula.h5')  data\_test\_X,data\_test\_Y = data\_x\_y\_preprocess(TestPath)  prediction = model.predict(data\_test\_X)  # 驗證模型  score = model.evaluate(data\_test\_X, data\_test\_Y, verbose=0)  # 輸出結果  print('Test loss:', score[0])  print('Test accuracy:', score[1]) |

**3.ReadModel.py**

**Module requirement:**

|  |
| --- |
| import numpy as np |
| from PIL import Image |
| from keras.models import load\_model |
| import CutPicture |

**Initial:**

|  |  |
| --- | --- |
| class\_names = ['0','1','2','3','4','5','6','7','8','9','+','-','mul','div','(',')'] | #種類 |
| class\_names\_label = {class\_name:i for i, class\_name in enumerate(class\_names)} | #用以得到對應Label |
| CLASSNUMBER = 16 | #種類數量 |
| img\_row, img\_col = 28,28 | #定義圖片大小 |
| EPOCH = 10 | #訓練次數 |
| reduce\_retracing=True |  |

**Function definition:**

* getResult:

|  |  |
| --- | --- |
| def getResult(index): | #分析模型分析出的結果 |
| formula = "" |  |
| for i in index: |  |
| if class\_names[i] == 'div': |  |
| formula += '/' |  |
| elif class\_names[i] == 'mul': |  |
| formula += '\*' |  |
| else: |  |
| formula += class\_names[i] |  |
| return formula |  |

* preprocess:

|  |  |
| --- | --- |
| def preprocess(path): | #輸入數學算式照片進行分割以及資料預處理  #對資料進行預先處理 |
| data\_x = np.zeros((img\_row,img\_col,1)).reshape(1,img\_row,img\_col) | #讀取黑白圖片 |
| img = Image.open(path) |  |
| temp = img.resize((100,100)) |  |
| temp.show() |  |
| img = accessBinary(img) |  |
| #行掃描 |  |
| hori\_points = CutPicture.extractPeek(hori\_vals,5,100) | #得到橫軸和的陣列用以判斷是否為邊界 |
|  | #得到行座標 |
| #根據每一行來掃描列 |  |
| counter = 0 |  |
| for hori\_point in hori\_points: |  |
| extractImg = img.crop((0, hori\_point[0], img.width, hori\_point[1])) | #提取橫切割區域 |
| vec\_vals = np.sum(extractImg,axis=0) | #得到縱軸和之陣列用以判斷邊界 |
| vec\_points = CutPicture.extractPeek(vec\_vals, min\_rect=10) |  |
| for vec\_point in vec\_points: |  |
| IndividualImg = extractImg.crop((vec\_point[0], 0, vec\_point[1], extractImg.height)) | #依左上角以及右下角座標提取 |
| hori\_valsI = np.sum(IndividualImg, axis=1) | #得到橫軸和的陣列用以判斷是否為邊界 |
| hori\_pointI = CutPicture.SignalExtract(hori\_valsI,10,20) | #得到行座標 |
| IndividualImgI = IndividualImg.crop((0, hori\_pointI[0] , IndividualImg.width, hori\_pointI[1])) | #依左上角以及右下角座標提取 |
| if(IndividualImg.width<100 and hori\_pointI[1]-hori\_pointI[0] < 100): |  |
| continue |  |
| whiteBlock = np.sum(IndividualImg)/255 | #過濾雜訊 |
| if whiteBlock < 1000: |  |
| continue |  |
| if IndividualImg.width > 270: |  |
| IndividualImgI = IndividualImgI.resize((270,IndividualImgI.height)) |  |
| if hori\_pointI[1]-hori\_pointI[0]>270: |  |
| IndividualImgI = IndividualImgI.resize((IndividualImgI.width,270)) |  |
| IndividualImgI = CutPicture.patch(IndividualImgI,300) |  |
| IndividualImgI = IndividualImgI.convert('L') | #轉灰階，高度變成1 |
| IndividualImgI = IndividualImgI.resize((img\_row,img\_col)) | #需取雙括號 |
| IndividualImgI = (np.array(IndividualImgI)/255).reshape(1,img\_row,img\_col) | #讀取黑白圖片 |
| data\_x = np.vstack((data\_x,IndividualImgI)) |  |
| counter+=1 |  |
| data\_x = np.delete(data\_x,[0],0) |  |
| data\_x=data\_x.reshape(counter,img\_row,img\_col,1) |  |
| return data\_x |  |

* GetFormula:

|  |  |
| --- | --- |
| def GetFormula(path): | #讀取對應路徑之圖片，並回傳數學式答案。 |
| model = load\_model(r'formula.h5') | Load model |
| data\_test\_X = preprocess(path) |  |
| prediction = model.predict(data\_test\_X,verbose=0) |  |
| index = np.argmax(prediction, axis=1) |  |
| formula = getResult(index) |  |
| return formula |  |

**前端:**

Line(以colab跟ngrok協助作為伺服器。)

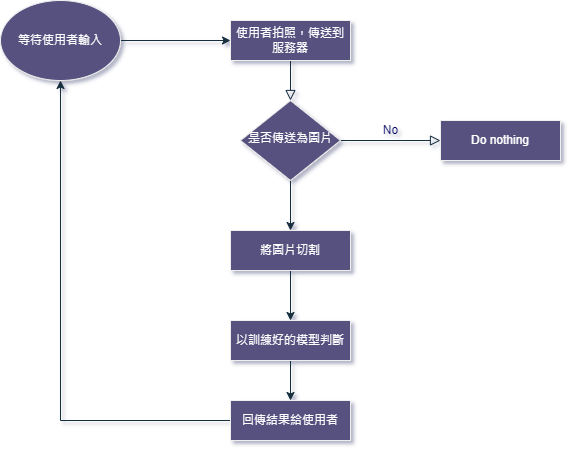
|  |  |
| --- | --- |
| from flask import Flask, request |  |
| from flask\_ngrok import run\_with\_ngrok | # 額外 import run\_with\_ngrok |
| import json | # 載入 json 標準函式庫，處理回傳的資料格式 |
| import os |  |
| os.chdir('/content/drive/MyDrive/Colab Notebooks') | # Colab 換路徑使用 |
| import sys |  |
| sys.path.insert(0,'/content/drive/MyDrive/pythonFile') |  |
|  |  |
| import formula\_analysize |  |
|  |  |
| # 載入 LINE Message API 相關函式庫 |  |
| from linebot import LineBotApi, WebhookHandler |  |
| from linebot.exceptions import InvalidSignatureError |  |
| from linebot.models import MessageEvent, TextMessage, TextSendMessage |  |
|  |  |
| app = Flask(\_\_name\_\_) |  |
| run\_with\_ngrok(app) | # 串接 ngrok |
| @app.route("/", methods=['POST']) |  |
| def linebot(): |  |
| body = request.get\_data(as\_text=True) | # 取得收到的訊息內容 |
| try: |  |
| json\_data = json.loads(body) | # json 格式化訊息內容 |
| access\_token = '個人的access\_Token' | # 你的 Access Token |
| secret = '個人的Channel Secret ' | # 你的 Channel Secret |
| line\_bot\_api = LineBotApi(access\_token) | # 確認 token 是否正確 |
| handler = WebhookHandler(secret) | # 確認 secret 是否正確 |
| signature = request.headers['X-Line-Signature'] | # 加入回傳的 headers |
| handler.handle(body, signature) | # 綁定訊息回傳的相關資訊 |
| tk = json\_data['events'][0]['replyToken'] | # 取得回傳訊息的 Token |
| type = json\_data['events'][0]['message']['type'] | # 取得 LINE 收到的訊息類型 |
| # 判斷如果是文字 |  |
| if type=='text': |  |
| msg = json\_data['events'][0]['message']['text'] | # 取得 LINE 收到的文字訊息 |
| reply = msg |  |
| # 判斷如果是圖片 |  |
| elif type == 'image': |  |
| msgID = json\_data['events'][0]['message']['id'] | # 取得訊息 id |
| message\_content = line\_bot\_api.get\_message\_content(msgID) | # 根據訊息 ID 取得訊息內容 |
| userID = json\_data['events'][0]['source']['userId'] |  |
| # 在同樣的資料夾中建立以訊息 ID 為檔名的 .jpg 檔案 |  |
| with open(f'{userID}.jpg', 'wb') as fd: |  |
| fd.write(message\_content.content) | # 以二進位的方式寫入檔案 |
|  |  |
| temp = "/content/drive/MyDrive/Colab Notebooks/"+userID+".jpg" |  |
| c = formula\_analysize.GetFormula(temp) |  |
| reply = str(c) |  |
| #reply = '圖片儲存完成！' # 設定要回傳的訊息 |  |
| else: |  |
| reply = '你傳的不是文字或圖片呦～' |  |
| print(reply) |  |
| line\_bot\_api.reply\_message(tk,TextSendMessage(reply)) | # 回傳訊息 |
| except: |  |
| print(body) | # 如果發生錯誤，印出收到的內容 |
| return 'OK' | # 驗證 Webhook 使用，不能省略 |
| if \_\_name\_\_ == "\_\_main\_\_": |  |
| app.run() |  |

**前端:**

Telegram(以colab作為伺服器)

|  |  |
| --- | --- |
| import telebot | 導入輔助函式庫 |
| import os  import dotenv | 導入.env相關功能，協助存取API |
| import formula\_analysize as fa | 後端函式庫 |
| dotenv\_file = dotenv.find\_dotenv()  dotenv.load\_dotenv(dotenv\_file)  API\_KEY = os.getenv("API\_KEY")  bot = telebot.TeleBot(API\_KEY) | 將API KEY存在隱藏檔案裡，保護相關資訊安全 |
| @bot.message\_handler(commands=['test'])  def test(message):  bot.send\_message(message.chat.id,"Hello World") | 測試指令，印出hello world |
| @bot.message\_handler(content\_types=['photo'])  def photo(message):  savename = message.from\_user.username + ".jpg" | 接收到照片的指令 |
| fileID = message.photo[-1].file\_id  file\_info = bot.get\_file(fileID)  downloaded\_file = bot.download\_file(file\_info.file\_path)  with open(savename, 'wb') as new\_file:  new\_file.write(downloaded\_file) | 將照片存成使用者名稱 |
| bot.send\_message(message.chat.id, "Photo Got")  print('Photo saved as '+savename) | 回傳成功存取的訊息 |
| line = recog(savename)  print(line)  result = cal(line)  print(result) | 呼叫recog函式將圖片轉成字串，呼叫cal函式將字串計算結果 |
| bot.send\_message(message.chat.id,result) | 回傳結果給使用者 |
| def recog(filename):  return fa.GetFormula(filename) | 呼叫後端函式處理圖片 |
| def cal(message):  st = message  st = st.replace("÷","/")  st = st.replace("×","\*") | 將手寫格式轉成eval函式可處理的樣式 |
| try:  re = eval(st)  re = st + '=' + str(re)  print(re)  return re  except SyntaxError:  err = "Error while evaluating " + st  return err  pass | 嘗試計算字串結果並回傳，若不為合理算式則回傳失敗訊息 |
| bot.polling() | 保持機器人運作 |

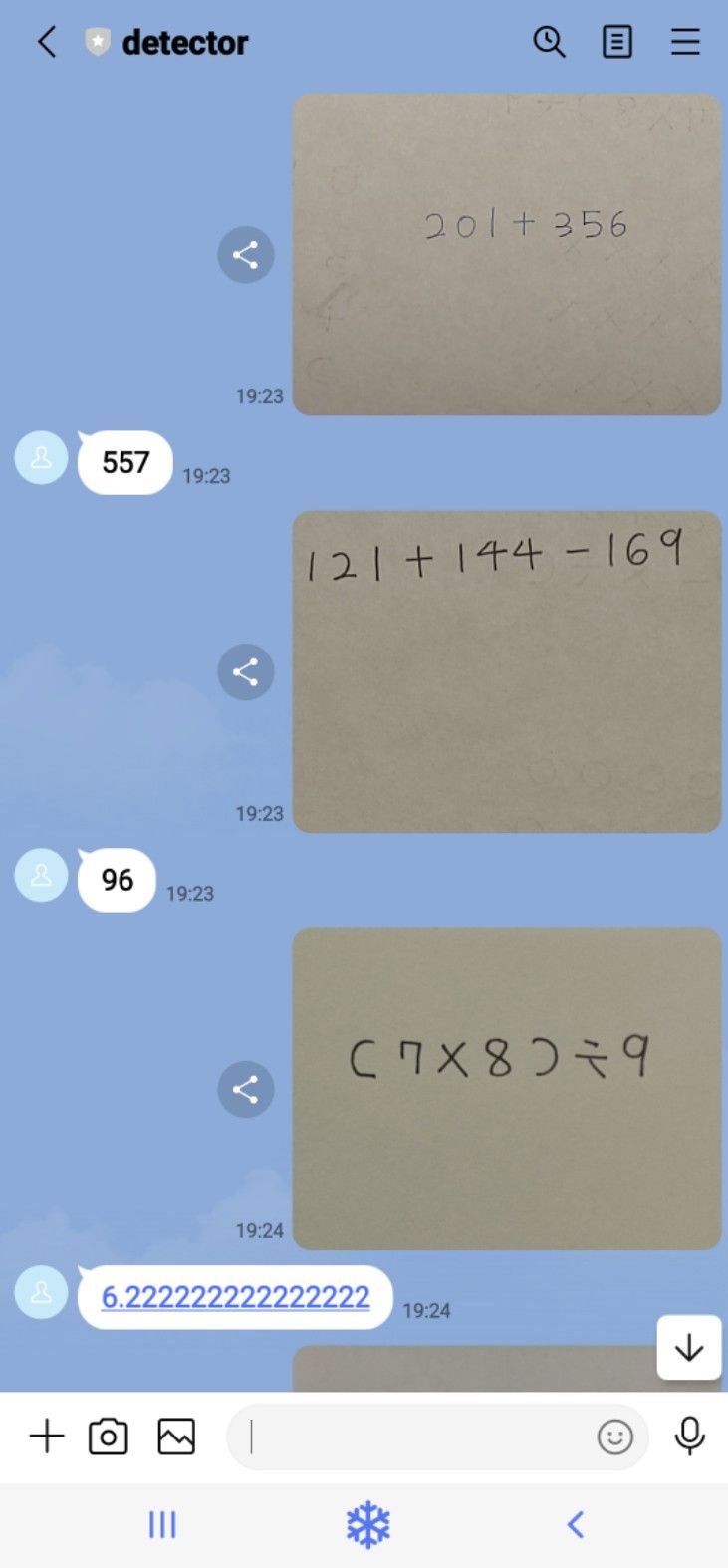
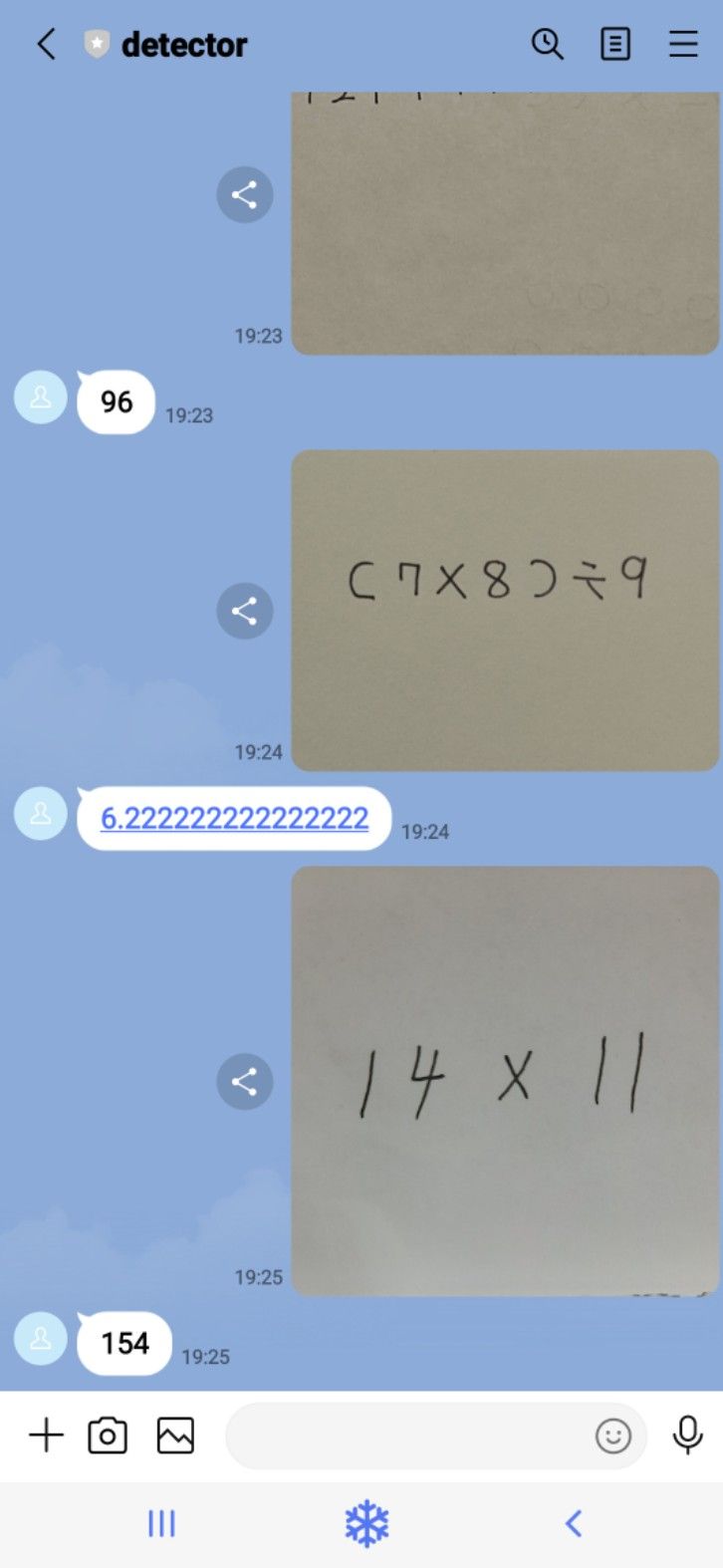
○ 程式架構圖

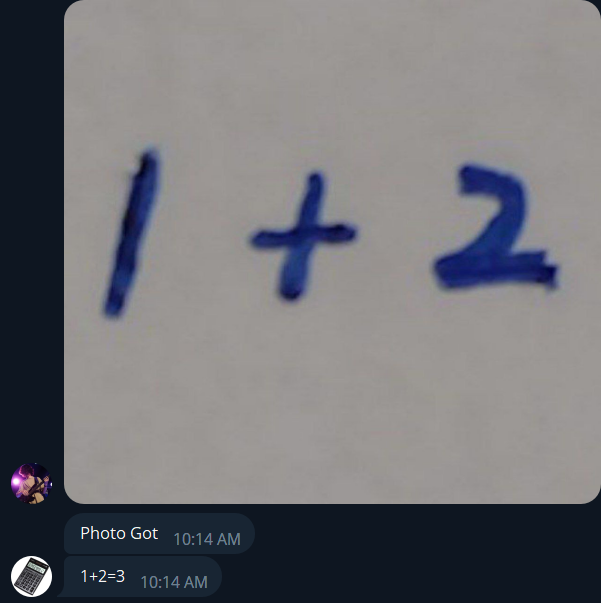
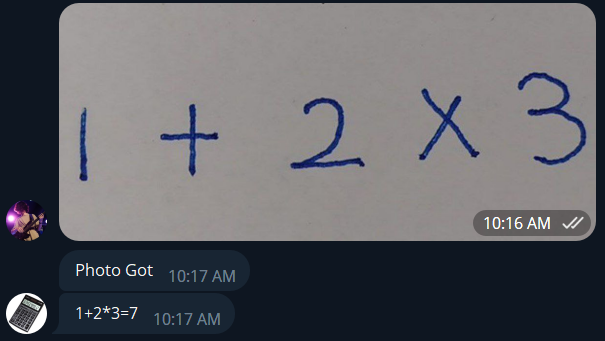


○ 程式功能測試

使用者:





後台接收並回傳(詳情見Colab):



Colab 連結(line):

<https://colab.research.google.com/drive/1BYB9g3YJBCD6oTI53SE1Cw1c3KGgniyi?usp=sharing>

Colab 連結(telegram):

<https://colab.research.google.com/drive/1L6NxGOoFyzOxUInT5zvEuGFVAAcOdjD-?usp=sharing>

○ 團隊分工

前端: 張庭嘉、陳彥呈

後端: 范家齊、吳宥謙

數學式: 黃昱智