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。

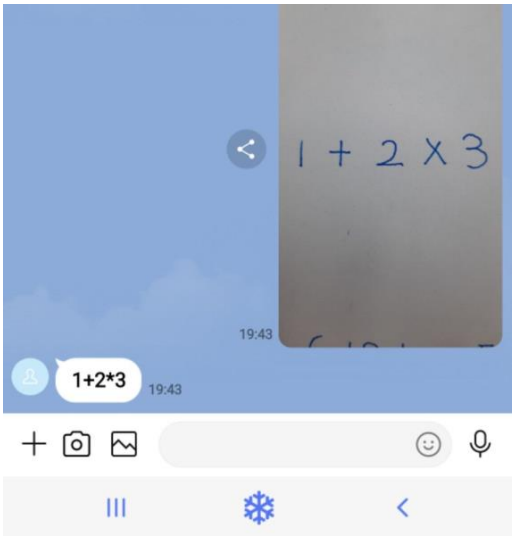
■ 程式功能

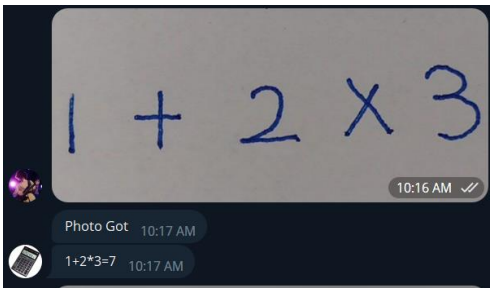
使用者輸入圖片，回傳數學式與計算結果。

○ 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。

```
def  GetFormula(path):
    model  =  load_model(r'formula.h5')
    data_test_X  =  preprocess(path)
    prediction  =  model.predict(data_test_X,verbose=0)
    index  =  np.argmax(prediction,  axis=1)
    formula  =  getResult(index)
    return  formula
```

■ 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,	#依左上角以及右下角座

vec_point[1], img.height))	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)

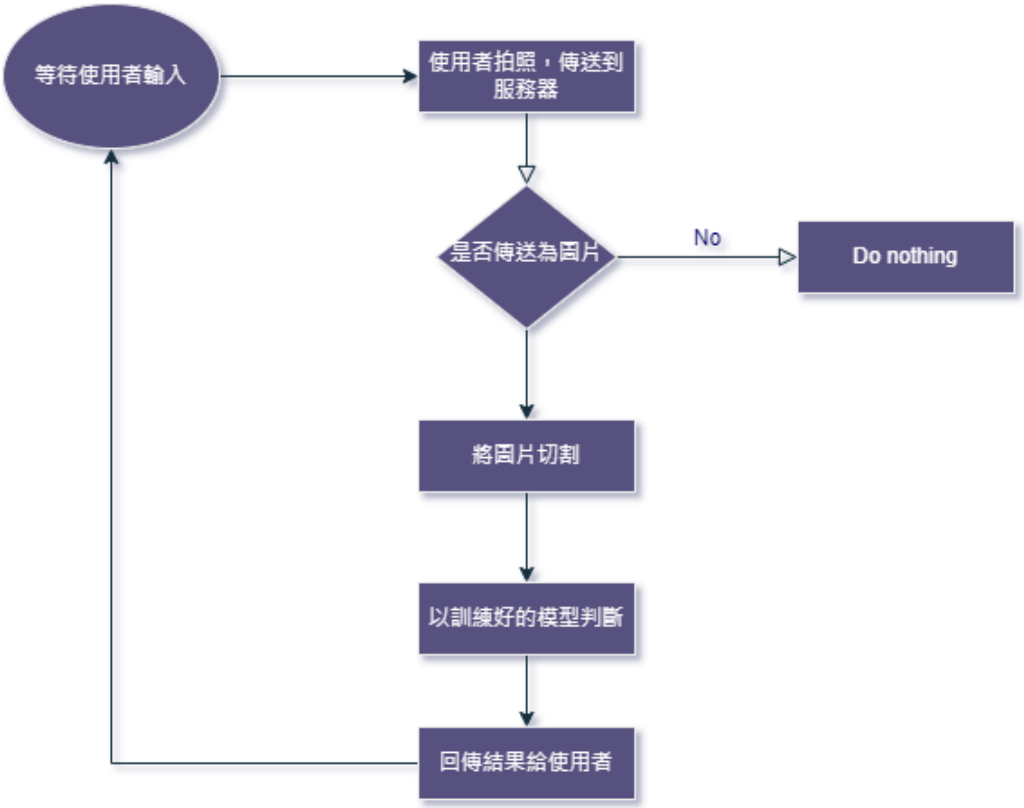
import telebot	導入輔助函式庫
import os import dotenv	導入.env 相關功能，協助存取 API
import formula_analysize as fa	後端函式庫
dotenv_file = dotenv.find_dotenv() dotenv.load_dotenv(dotenv_file) API_KEY = os.getenv("API_KEY") bot = telebot.TeleBot(API_KEY)	將 API KEY 存在隱藏檔案裡，保護相關資訊安全
@bot.message_handler(commands=['test']) 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):	將手寫格式轉成 eval 函式

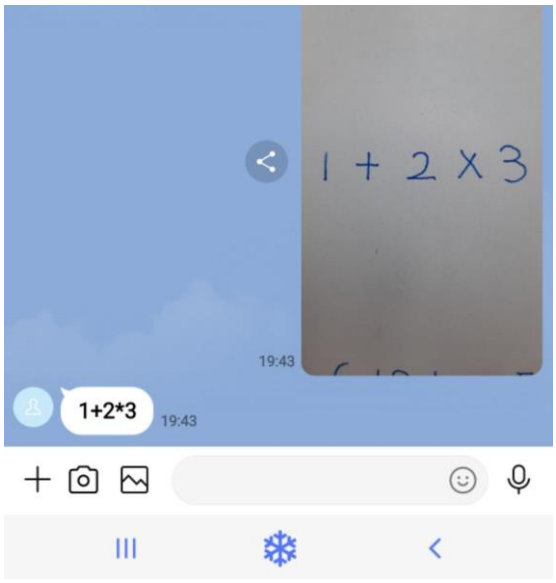
st = message st = st.replace("÷","/") st = st.replace("×","*")	可處理的樣式
try: re = eval(st) re = st + '=' + str(re) print(re) return re except SyntaxError: err = "Error while evaluating " + st return err pass	嘗試計算字串結果並回傳，若不為合理算式則回傳失敗訊息
bot.polling()	保持機器人運作

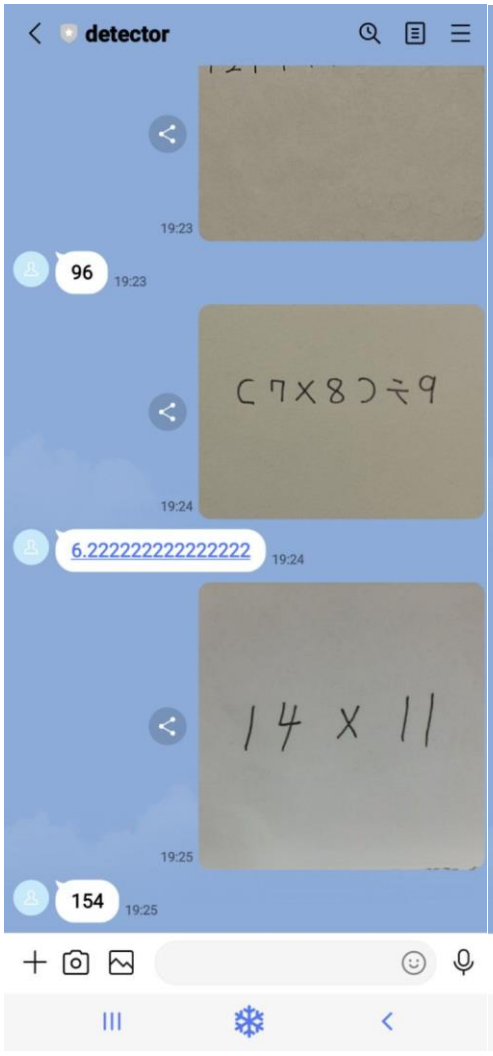
○ 程式架構圖

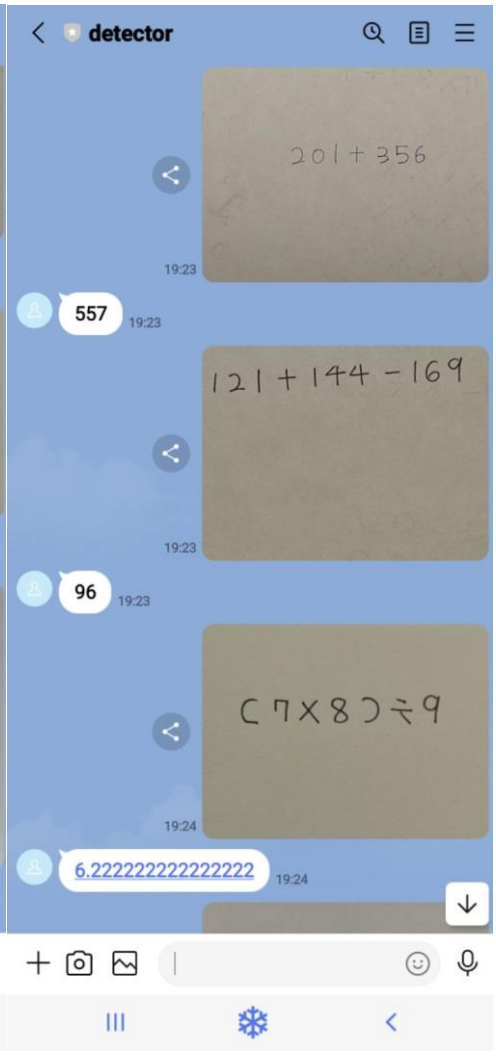


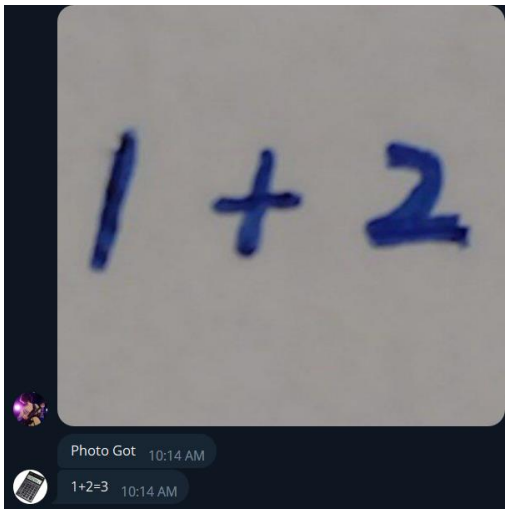
○ 程式功能測試

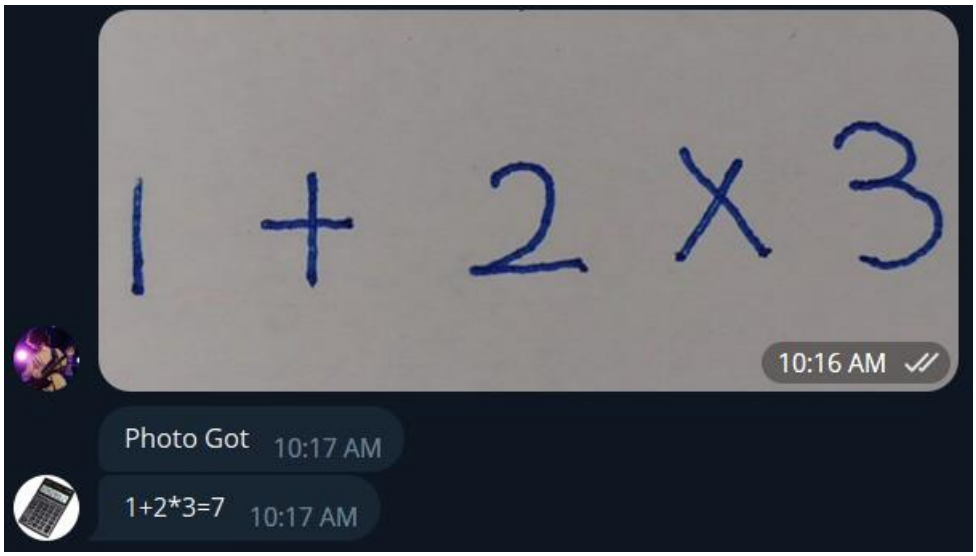
使用者:




detector
19:23
96
19:23
19:24
6.222222222222222
19:24
14 X 11
19:25
154
19:25


detector
201+356
19:23
557
19:23
121+144-169
19:23
96
19:23
19:24
6.222222222222222
19:24


1+2
Photo Got
10:14 AM
1+2=3
10:14 AM



後台接收並回傳(詳情見 Colab):

```
557
INFO:werkzeug:127.0.0.1 - - [29/May/2023 11:23:19] "POST / HTTP/1.1" 200 -
96
INFO:werkzeug:127.0.0.1 - - [29/May/2023 11:23:50] "POST / HTTP/1.1" 200 -
6.222222222222222
INFO:werkzeug:127.0.0.1 - - [29/May/2023 11:24:32] "POST / HTTP/1.1" 200 -
154
INFO:werkzeug:127.0.0.1 - - [29/May/2023 11:25:22] "POST / HTTP/1.1" 200 -
```

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