Neural Networks and Applications_10920IEEM537300

Project 1: **Simple Math Solver**

Group 2

109136502 陳宗聖　109033573 吳育葦　109033804 張軼峯

Test data input 流程

在 Colab 上傳 Test 圖片與 model 檔案後，於第二個 block 內輸入 `Test_img` 的檔名，全部執行後便可印出答案，並輸出一 Answer.csv 紀錄 equation 與計算後的答案。

Project 1: Simple Math Solver 作業目標

- Input: photo with one handwritten math equation
- Output: result of the math equation
- Including integer numbers and operators +, -, x, ÷
- Baseline: individual digits and operators/ Advanced: whole equation

Implementation

Data Collection

***Training data* 雲端連結:** https://drive.google.com/file/d/1VKn35l06sAPzCvIA6_nHzJJ09juEODq5/view?usp=sharing

Training set 部分先嘗試了 MNIST[1] 與 kaggle 上的 "Handwritten math symbol and digit dataset"[2]，結果均不夠準確，我們推測其原因在於本次專題的需求為拍攝黑板算式，準確判斷數字及 operator。由於拍攝照片時，有可能產生與黑板的傾角、當下的環境亮度以及書寫習慣與 MNIST 不同等原因，故可以合理推測其 domain 可能與 MNIST 的 domain 不相同，進而產生預測不準的情況發生。因此，為了使資料集的情境盡可能與作業需求相同，故我們在黑板上手寫資料，並且對其進行 data augmentation。

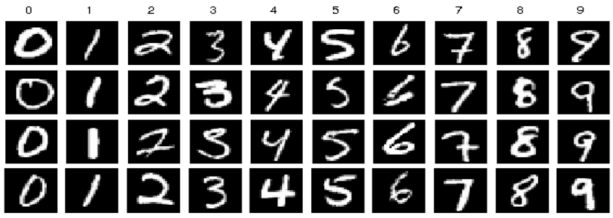

Fig. 1. MNIST dataset

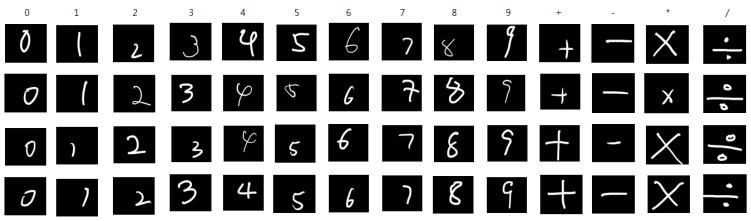

Fig. 2. Handwritten math symbol and digit dataset

[1] http://yann.lecun.com/exdb/mnist/

[2] https://www.kaggle.com/clarencezhao/handwritten-math-symbol-dataset

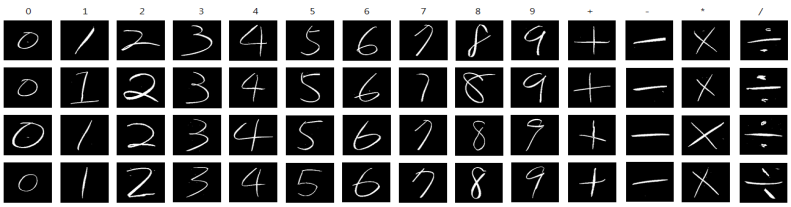

Fig. 3. Our handwritten dataset

Data augmentation

因完全重新建立手寫 dataset 勢必無法有短時間得到足夠的資料量，data augmentation 為較有效率的做法，可補足資料量避免 over-fitting。我們套用 package imgaug[3] 來執行 data augmentation 的功能，因灰階、二值化等前處理已先行處理完成，data augmentation 步驟僅需縮放、旋轉(最大角度 10°)、偏移等功能。

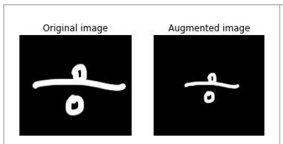	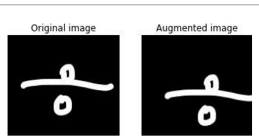	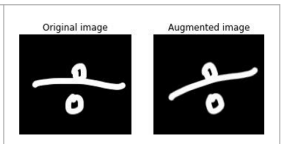
縮放	偏移	旋轉

Table 1. Data augmentation

Data preprocess

Training 和 Test data 的前處理作法大致相同，以下將依操作順序描述各步驟之作法與目的:

resize	resize 將圖片縮小有濾除噪點的功能，因此第一步 resize 至我們手動調出之最適大小。
Morphological Transformations	使用 dilate 先將圖片膨脹增肥，再用 opening 去除噪點，最後填充封閉輪廓內的小黑點。
crop	將圖片稍微剪裁，將可能影響判讀之多於邊界(如黑板上下邊界)切除。
gray scale	將彩色圖片轉為灰階圖，使用 opencv 的 `cvtColor` 函式，`COLOR_BGR2GRAY` 將原本 BGR 的彩色空間轉至灰度空間。(此處先暫不作二值化)
Gaussian blur	使用高斯模糊(為一種低通濾波器)可將高頻雜訊濾除，視覺上有模糊的效果，我們設定 5x5 的 kernel 來執行。
Canny edge detection	需要以上的前處理降噪才可進行 edge detection。Canny edge detector 首先計算每個 pixel 的梯度和梯度方向，利用 Non-maximum suppression 的方法將非最大值去除，最後根據 threshold 值選取 edge。
find contours	經過 Canny edge detector 的 binary image 可經由 `findContours` 來找出將數字包圍的各個點所連成的 contours。
boundingRect	利用 `boundingRect` 來得到框出 contour 的矩形左上角 x, y 座標與矩形長寬 w, h。此處使用判斷式去除長或寬 < 20 (須與第一步 resize 的設定配合) 的矩形，再進一步過濾掉過小的噪點或雜訊。

[3] https://github.com/aleju/imgaug

crop output image	得到要框出的座標與大小後，crop 選取該矩形，灰階並二值化 (`cv.threshold`) 後將該矩形存為 jpg 檔輸出。
transfer to square	因擷取出來的矩形會有不同大小，所以需要轉為固定大小，我們的做法為將較短邊補值，使每張圖皆為正方形，並 resize 至跟 MNIST 相同的 28*28。展成一維後便可輸出為 training 或 test data，完成前處理步驟。

Preprocess Example:

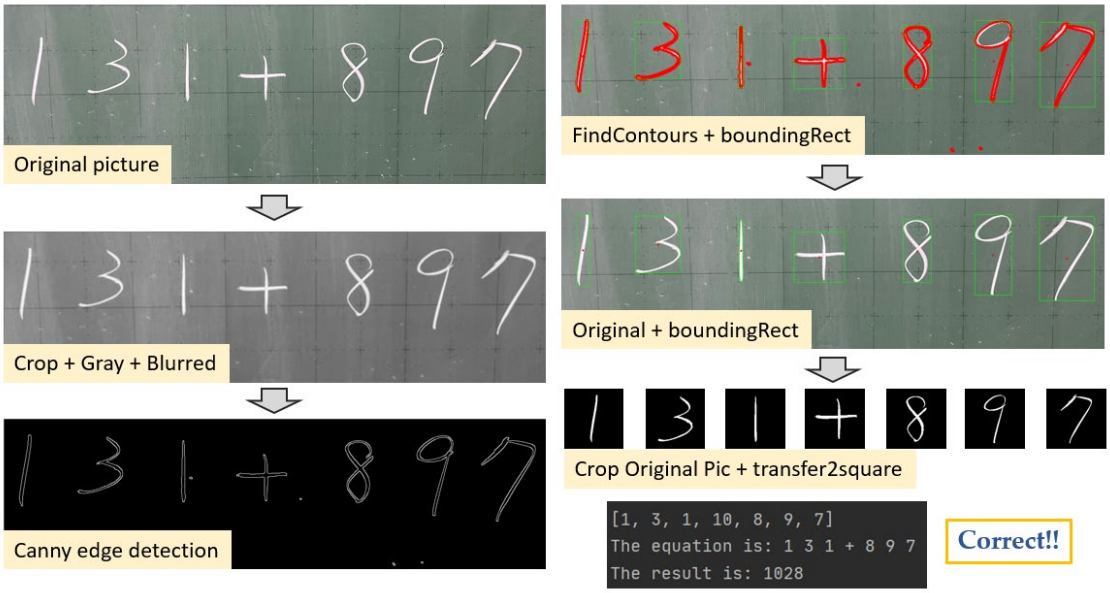

Fig. 4. Preprocess example

Pytorch Model

1. Convolutional Neural Network

```
CNN(
  (conv1): Conv2d(1, 6, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
  (conv2): Conv2d(6, 16, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
  (conv2_drop): Dropout2d(p=0.5, inplace=False)
  (conv3): Conv2d(16, 10, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
  (fc1): Linear(in_features=784, out_features=120, bias=True)
  (fc2): Linear(in_features=120, out_features=84, bias=True)
  (fc3): Linear(in_features=84, out_features=28, bias=True)
  (fc4): Linear(in_features=28, out_features=14, bias=True)
)
```

Accuracy of the network on training image: 96%

2. Multilayer Perceptron

```
MLP(
  (conv1): Conv2d(1, 6, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
  (conv2): Conv2d(6, 16, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
  (conv2_drop): Dropout2d(p=0.5, inplace=False)
  (conv3): Conv2d(16, 10, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
  (fc1): Linear(in_features=784, out_features=120, bias=True)
  (fc2): Linear(in_features=120, out_features=84, bias=True)
  (fc3): Linear(in_features=84, out_features=28, bias=True)
  (fc4): Linear(in_features=28, out_features=14, bias=True)
)
```

Accuracy of the network on training image: 99%

Pseudo code

```
-----------------------------------------------
Algorithm 1: The purpose of this project is accomplishing the target of recognizing the equations
taken from the blackboard. First of all, we need to do the preprocessing and adapt these transformed
pictures to our model by tuning hyperparameters.
N is the number of training examples in the training set. K is the number of classes. (In this case,
we have 10 digits and 4 operators.) W is the learnable parameters in model M.
-----------------------------------------------
Input: Training set D={(x1, y1), …, (xN, yN)} , where each yi={1, …, K}
Output: The model M.
        X, Y <- DataLoader(D, batch size = 128)
        M <- Model()
        for ep in epochs do
                predict <- M(X, Y)
                loss <- CrossEntropy(predict, Y)
                W <- W + LearningRate*Gradient(loss)

        return M
-----------------------------------------------


-----------------------------------------------
Algorithm 2: Test our model with test data. Due to the purpose of the advanced problem, we should
crop each of the digits or operators and predict them individually.
-----------------------------------------------
Input: The raw data (Picture)
Output: Cropped pictures.
        image <- Resize(image)
        image <- Crop(image) // In order to remove the border of the picture.
        Image <- Preprocessing(image) // Including gray scale, Gaussian blur and Canny edge detection.
        contours <- findContours(image)
        contours <- sortContours(contours) // Sorting contours followed by the number of points in
each contours.
        if contours.width or contours contour.height > threshold do
                contours <- contours // Filter noise.
        contours <- FilterClosedContours(contours) // process the problem of divide operator.
        for c in contours do
                image <- CropImage(c) // Crop the square picture of each contour.
-----------------------------------------------
```

Discussion

Division sign preprocess

我們想要使輸入的照片所被抓到的 contours 數量與 equation 的每一個元素一致，但由於除號上下的各兩點會被各自獨立判斷成一個 contour，故採取將小於某個 thershold 的 contours 皆剔除，除了可以將除號的兩個點刪去外，亦有去雜訊的功能，以防存取到不必要的 contour。在所有"寬"比"高"還要長的 contour，我們將"寬"的長度取代"高"，因此除號在裁減圖片時可以完整被囊括住。

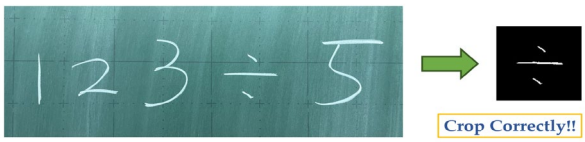

Fig. 5. Division sign preprocess example

Colab issue

將原先在 Local machine 上所撰寫的 python 程式碼轉移至 google colab 上執行時，我們發現由於 colab 的讀檔順序與原先不同，於是我們將讀檔的優先順序調整為數字小的優先、依次讀檔。並且在讀取檔案時，無法讀取本機的相對路徑，而使檔案的上傳及路徑的寫法變得稍嫌繁複。且須注意 Colab 要設定使用 GPU，否則 tensor 無法正確計算。