Team27 Report

A brief introduction to the problem

身心障礙人士一直是社會中很重要的一個議題，尤其是最近疫情更加嚴重，如果視障人士出門需要搭電梯的話必須一個一個摸上面的點字，所以我們就想利用CNN來做一個可以辨識手語數字的模型，讓視障人士可以在電梯裡避免不必要的碰觸，來降低病毒傳染的風險。

A brief introduction to the GitHub code you refer to if there is any.

我們這次source code主要都是自己打的，只有在model的參數上跟save model有參考kaggle的這個網址：
https://www.kaggle.com/muhammadkhalid/sign-language-model-training-for-numbers

首先我們參考了他運用pickle的技巧，以節省我們的時間(圖一)，再來是model的參數，如Filters、kernel size、激勵函數等等(圖二)，不只讓我們有了基本概念可以去改參數，也將我們從overfitting的苦海解救出來。

```
]:
import pickle

pickle_out = open("/kaggle/working/X.pickle","wb")
pickle.dump(X, pickle_out)
pickle_out.close()

pickle_out = open("/kaggle/working/Y.pickle","wb")
pickle.dump(Y, pickle_out)
pickle_out.close()
```

(圖一)Pickle參考code

```
NAME = "Numbers-CNN-Model-{}".format(str(time.ctime())) # Model Name

# Load pickel data
pickle_in = open("/kaggle/working/X.pickle","rb")
X = pickle.load(pickle_in)
X = np.array(X).reshape(-1, IMG_SIZE, IMG_SIZE, 1)

pickle_in = open("/kaggle/working/Y.pickle","rb")
Y = pickle.load(pickle_in)
Y = np.array(Y)

X = X/255.0

model = Sequential()

model.add(Conv2D(16, (2,2), input_shape=X.shape[1:], activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2), padding='same'))
model.add(Conv2D(32, (3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(3, 3), strides=(3, 3), padding='same'))
model.add(Conv2D(64, (5,5), activation='relu'))
model.add(MaxPooling2D(pool_size=(5, 5), strides=(5, 5), padding='same'))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(11, activation='softmax')) # size must be equal to number of classes i.e. 1
1
```

(圖二)model參考code

3.1 Your algorithm. Please provide details and the difference between your implementation and the referenced codes

我們會根據每一部分來講解我們的algorithm。首先我們可以看到圖三是我們產生training set跟validation 的地方。Path_train就是我們的dataset的相對位置，然後我們按照0.8 : 0.2的比例下去分配train跟validation的數量。我們也限制了圖片的size(64*64)跟batch_size。

```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    path_train,
    validation_split = 0.2,
    subset = 'training',
    seed = 123,
    color_mode = 'grayscale',
    image_size = (img_height, img_width),
    batch_size = batch_size
)

class_names = [0,1,2,3,4,5]

val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    path_train,
    validation_split = 0.2,
    subset = 'validation',
    seed = 123,
    color_mode = 'grayscale', #default
    image_size = (img_height, img_width),
    batch_size = batch_size
)
```

(圖三) training set & validation code

```
model = tf.keras.Sequential([
  layers.experimental.preprocessing.Rescaling(1./255),
  layers.Conv2D(16, (2,2),activation='relu'),
  layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2), padding='same'),
  layers.Conv2D(32, (3,3), activation='relu'),
  layers.MaxPooling2D(pool_size=(3, 3), strides=(3, 3), padding='same'),
  layers.Conv2D(64, (5,5), activation='relu'),
  layers.MaxPooling2D(pool_size=(5, 5), strides=(5, 5), padding='same'),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dropout(0.2),
  layers.Dense(6, activation='softmax')
])

model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['accuracy'])

history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=10,
  batch_size = 32,
  verbose = 2,
)
```

(圖四) model code

(圖四) 是這次的核心部分，也就是model的參數。主要是根據之前講到的kaggle上的來做修改。我們加了一些drop out來避免overfitting，最後的dense也改成我們自己的class number，也就是六種數字的”6”。再來compile中的optimizer也

是在我們比較了各種，包括adam、adagrad、adadelta等之後，我們選出了adam。然後我們就將train dataset跟validation dataset丟入model，我們epoch 設為10，batch_size 設為32，verbose = 2，我們有嘗試將epoch調大一些，但是效果並不顯著，而且會多花很多時間。

```
epochs = 10
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

(圖五) Print accuracy & loss

(圖五)是為了print出accuracy跟loss，讓我們能透過他的output (圖九) 更直觀的了解這些數據。

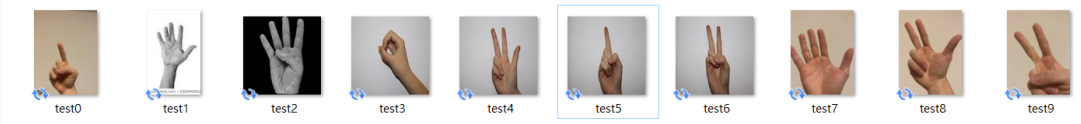
```
##### load test data #####
count = 0
ans = list()
for x in sorted(path_test.iterdir()):
  img = tf.keras.preprocessing.image.load_img(x, color_mode="grayscale",target_size=(img_height, img_width))
  img_array = tf.keras.preprocessing.image.img_to_array(img)
  img_array = tf.expand_dims(img_array, 0)
  img_array = tf.expand_dims(img_array, -1)
  predictions = model.predict(img_array)
  score = tf.nn.softmax(predictions[0])
  ans.append(class_names[np.argmax(score)])
  count += 1
  print(x," is predicted to be number ",np.argmax(score),end = '\n')
```

(圖六) load test data

(圖六) 是我從資料夾中取出test dataset，改變他的color mode，size，再將他covert 成array，然後增加了兩個向量符合model的input size(1, 64, 64, 1)，第一個1是batch size，最後一個1是channel，因為是grayscale所以是1。Predict出來後得到的predictions我們選擇機率最高的作為我們的預測。

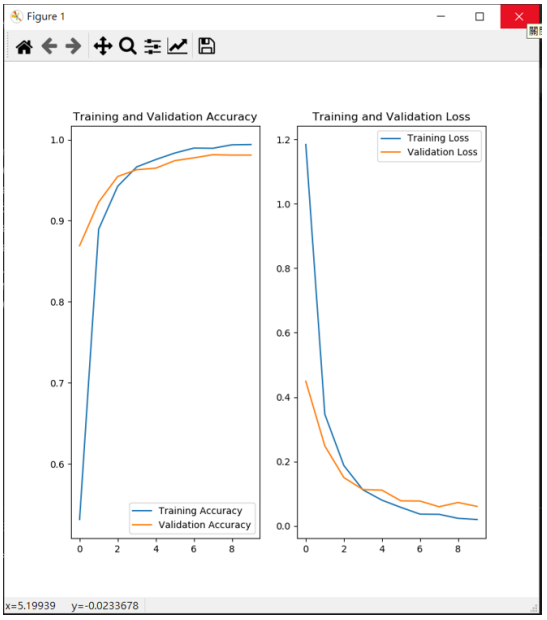
Experiment results

(圖七) 是我們的test data，我們特別選了兩張背景顏色全白或全黑的來測試我們的model。在test我們的model時，會先把需要predict的圖片放到data裡面的test/test資料夾，然後就可以predict了。



(圖七) Test data image

```
max_pooling2d_1 (MaxPooling2 (None, 10, 10, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 6, 6, 64)          51264
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 2, 2, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 128)               32896
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 774
=================================================================
Total params: 89,654
Trainable params: 89,654
Non-trainable params: 0
_________________________________________________________________
data\test\test\test0.jpg  is predicted to be number  1
data\test\test\test1.jpg  is predicted to be number  5
data\test\test\test2.jpg  is predicted to be number  4
data\test\test\test3.JPG  is predicted to be number  0
data\test\test\test4.JPG  is predicted to be number  3
data\test\test\test5.JPG  is predicted to be number  1
data\test\test\test6.JPG  is predicted to be number  2
data\test\test\test7.jpg  is predicted to be number  5
data\test\test\test8.jpg  is predicted to be number  3
data\test\test\test9.jpg  is predicted to be number  2
PS C:\Users\USER\Desktop\team27_final_project>
```



(圖八) model summary & pridictions

(圖九) Accuracy & Loss chart

(圖八) 是我們的model summary 還有 Test image跑出來的 result。

(圖九)是 (圖五) 跑出來的accuracy跟loss chart，藍線代表的是Training Accuracy，橘線代表的是Validation Loss。

Problem Encountered

在製作這次的final project中，遇到了很多問題，每一個問題都犧牲了好多睡眠時間，並透過上網搜尋資料、詢問同學一起討論之後才一一解決，以下是我們

遇到的困難:

1. dataset數量不夠龐大

我們在助教和教授討論主題的時候，教授就曾經提醒過我們這一點，我們後來決定把網路上所有能找到的dataset、圖片都蒐集起來，再加上一些自己用手機拍的來測試效果。當然，不是隨便一張都放進我們的dataset，我們還是有一些條件，包括:清楚明瞭，只拍手，背景盡量單純，然後做成屬於我們自己的dataset。此外，我們也有嘗試過data augmentation，但改善的效果有限，不太明顯。

Dataset主要來源:https://www.kaggle.com/muhammadkhalid/sign-language-for-numbers/download

2. 用grayscale train但是用彩圖test

我們最後決定將要test的圖片都轉成grayscale後再丟進去model predict。此外我們也需要reshape我們的image才能符合model的input shape。

3. 每次train完，產生一個model後想要test不同的image都需要重新train一個model，浪費很多時間

我們上網查到了許多方法可以先將model存起來，包括save model、弄成pickle檔後存起來等。

4. Overfitting

Overfitting是每次在做人工智慧的最大難題，當然也是我們最不想遇到的狀況，我們上網查了許多資料，包括：drop outRegularization、Data Augmentation等等，我也試著找出最佳的convolutional layer層數以及epochs個數，終於在最後找出了一個比較平衡的版本。

5. 把accuracy跟loss的結果print出來，overfitting不太嚴重，predict的結果卻一團糟

這是我們遇到最大的困難，因為想了好幾天都沒有解決，幸好有一次想到要把predict的順序印出來看看，才發現因為我們用iterdir()這個function去抓test data，他會亂數抓，我們如果要照順序看必須先把他sort過，所以我們在前面加了一個sorted，終於得到我們想要的結果。