應用數學系 110 年度第一學期

機器學習 期末報告

題目: YOLOv3 口罩辨識

姓名: 簡睿德, 學號:S07240013

姓名: 高祺烜, 學號:S07240044

姓名: 陳昱廷, 學號:S07240047

老師:江明理

111 年 1 月 18 日

一、研究計畫中文摘要：

現今，物件偵測已經應用於日常生活中的很多層面。 例如在學校中測量體溫的紅外線感測儀及停車場的車牌辨識都運用到了物件偵測的系統。

COVID-19 疫情下，佩戴口罩是減少病毒傳播風險的關鍵之一。然而由於大家在公共場所必須配戴口罩，因此在人臉辨識上的困難度就增加了。這次報告參考一些結合口罩物件偵測與人臉辨識的影片及文章，能判斷三個類別是否有戴好口罩，分別為 GOOD、BAD、NONE。

我們使用 yolov3 作為訓練模型，配合 OpenCV 來偵測影像，利用 yolov3 當中的 Darknet-53 及 678 張照片來訓練電腦能夠判斷大家配戴口罩的情況，再利用已訓練完的模型來偵測照片中大家配戴口罩的情況及即時影像中配戴口罩的情況，作為口罩佩戴辨識的系統，

關鍵詞: 機器學習、物件偵測、口罩辨識、Yolov3、OpenCV、疫情

二、前言與研究目的與研究之背景

在這個科技化的時代，為了在不影響生活運作的情況下，同時能降低被感染的機會，我們使用了 QRCODE 實名制掃描系統、紅外線感應器等來解決各方面的需求，但在某些方面，我們還沒製作出可以替代的工具，只能依靠人力來解決，像是口罩的取締，且近幾年在 AI 領域中也有不少新的發展，其中 YOLO 作為物件偵測中的重要技術，我們決定運用並加以跟現實作結合，這是這次我們想做的東西。

由於前陣子疫情嚴重，在車站、校園、百貨公司等人流眾多處，大多需要人力監測有沒有戴口罩。透過相機偵測民眾是否戴口罩，可減少人力需求及人與人之間的接觸。

三、研究方法

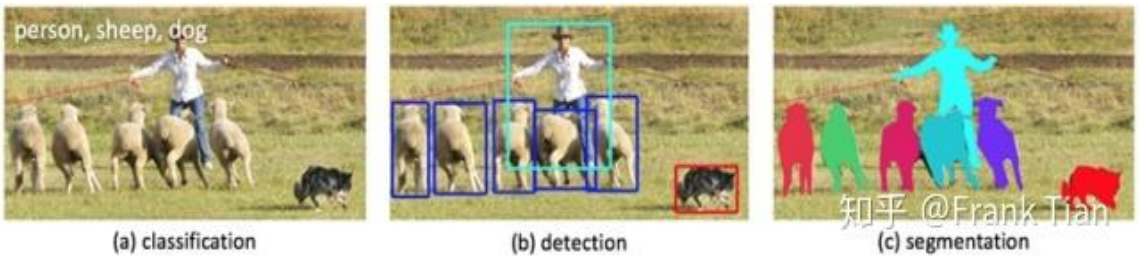
利用 YOLO 模型來偵測即時影像中人們是否有配戴口罩。Yolo3 總共有 106 層，借鑒了 DSSD 的反捲積拓展和多層 Feature Map 預測結構設計，即對每張圖片在 3 個不同的 scale 層上進行目標探測，這 3 層分別是第 82 層、94 層、106 層，以此實現對大、中、小目標的探測

1. YOLO 模型

YOLO 是目標檢測模型。

目標檢測是計算機視覺中比較簡單的任務，用來在一張圖篇中找到某些特定的物體，目標檢測不僅要求我們識別這些物體的種類，同時要求我們標出這些物體的位置。

顯然，類別是離散數據，位置是連續數據。



上面的圖片中，分別是計算機視覺的三類任務：分類，目標檢測，實例分割。

很顯然，整體上這三類任務從易到難，我們要討論的目標檢測位於中間。前面的分類任務是我們做目標檢測的基礎，至於像素級別的實例分割非常困難。

YOLO 的全稱是 you only look once，指只需要瀏覽一次就可以識別出圖中的物體的類別和位置。

因為只需要看一次，YOLO 被稱為 Region-free 方法，相比於 Region-based 方法，YOLO 不需要提前找到可能存在目標的 Region。

也就是說，一個典型的 Region-base 方法的流程是這樣的：先通過計算機圖形學（或者深度學習）的方法，對圖片進行分析，找出若幹個可能存在物體的區域，將這些區域裁剪下來，放入一個圖片分類器中，由分類器分類。

因為 YOLO 這樣的 Region-free 方法只需要一次掃描，也被稱為單階段(1-stage)模型。Region- based 方法方法也被稱為兩階段（2-stage）方法。

2. Darknet

在目標檢測領域的 YOLO 系列算法中，為了達到更好的分類效果，設置並訓練了 DarkNet 網絡作為骨幹網絡。其中，YOLOv2 首次提出 DarkNet 網絡，由於其具有 19 個卷積層，所以也稱之為 DarkNet19。後來在 YOLOv3 中，繼續吸收了當前優秀算法的思想，如殘差網絡和特征融合等，提出了具有 53 個卷積層的骨幹網絡 DarkNet53。在 ImageNet 上進行了實驗，發現相較於 ResNet-152 和 ResNet-101，DarkNet53 在分類精度差不多的前提下，計算速度取得了領先。

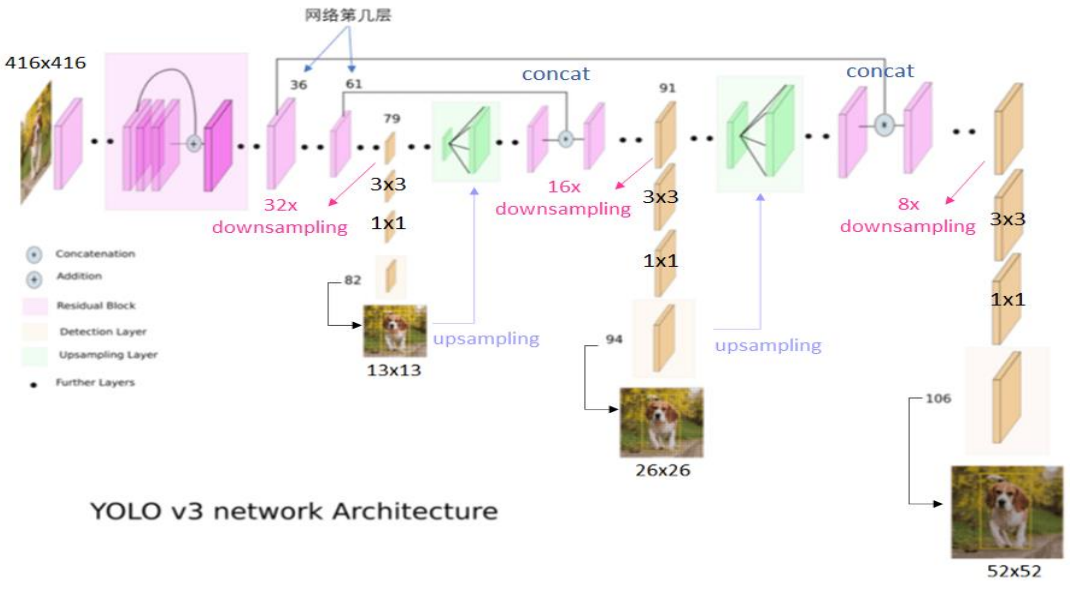
DarkNet19 中，借鑒了許多優秀算法的經驗，比如：借鑒了 VGG 的思想，使用了較多的 3x3 卷積，在每一次池化操作後，將通道數翻倍；借鑒了 network in network 的思想，使用全局平均池化（global average pooling）做預測，並把 1x1 的卷積核置於 3x3 的卷積核之間，用來壓縮特征；同時，使用歸一化層來穩定模型訓練，加速收斂，並且起到正則化作用。

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

而 DarkNet53 在之前的基礎上，借鑒了 ResNet 的思想，在網絡中大量使用了殘差連接，因此網絡結構可以設計的很深，並且緩解了訓練中梯度消失的問題，使得模型更容易收斂。同時，使用步長為 2 的卷積層代替池化層實現降采樣。

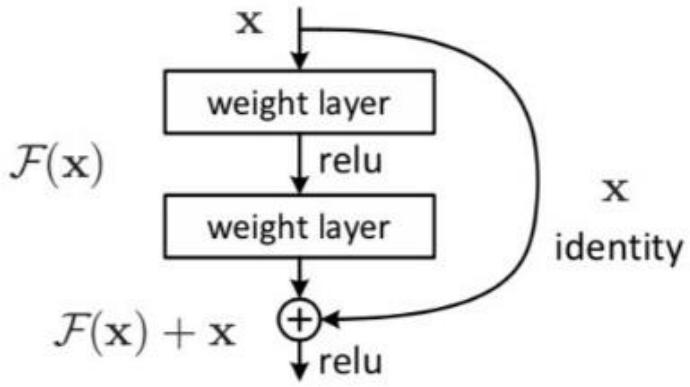
	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

3. YOLOv3 架構



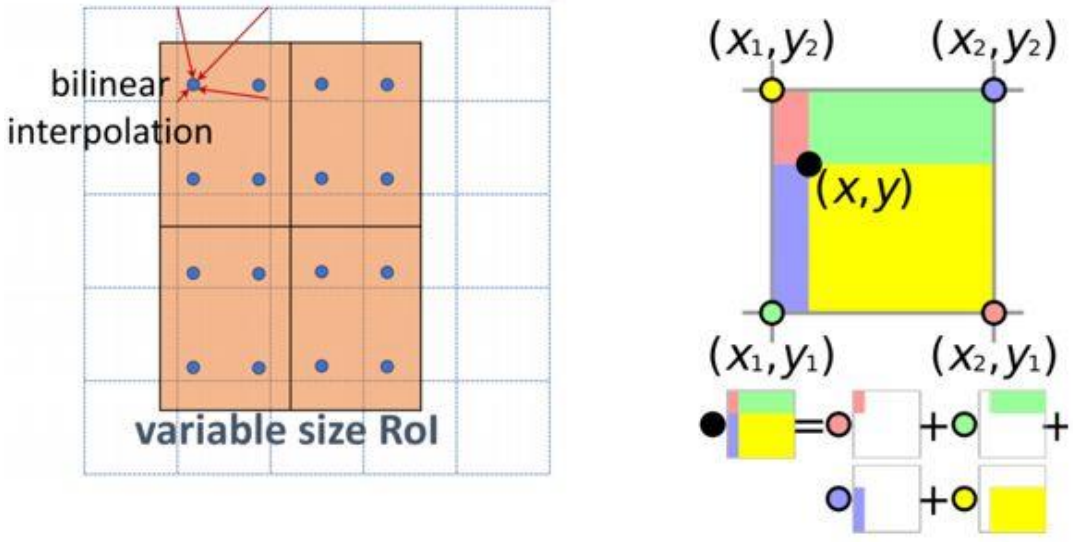
YOLO v3 network Architecture

一開始輸入 416x416 的圖片，在 79 層卷積層後，會先經過 32 倍的下採樣，再通過 3x3, 1x1 的卷積層後，得到 13x13 的 feature map（第 82 層）
為了能夠偵測小物體，在第 79 層 13x13 的 feature map 進行上採樣，與第 61 層 26x26 的 feature map 合併後，再經過 16 倍的下採樣及 3x3, 1x1 的卷積層後，得到 26x26 的 feature map（第 94 層）
接著第 91 層 26x26 的 feature map 再次上採樣，並與第 36 層 26x26 的 feature map 合併後，再經過 8 倍下採樣及 3x3, 1x1 的卷積層後，得到 52x52 的 feature map（第 106 層）
其中粉紅色的部分為 Residual Block：



綠色的部分為上採樣（Upsampling Layer）：

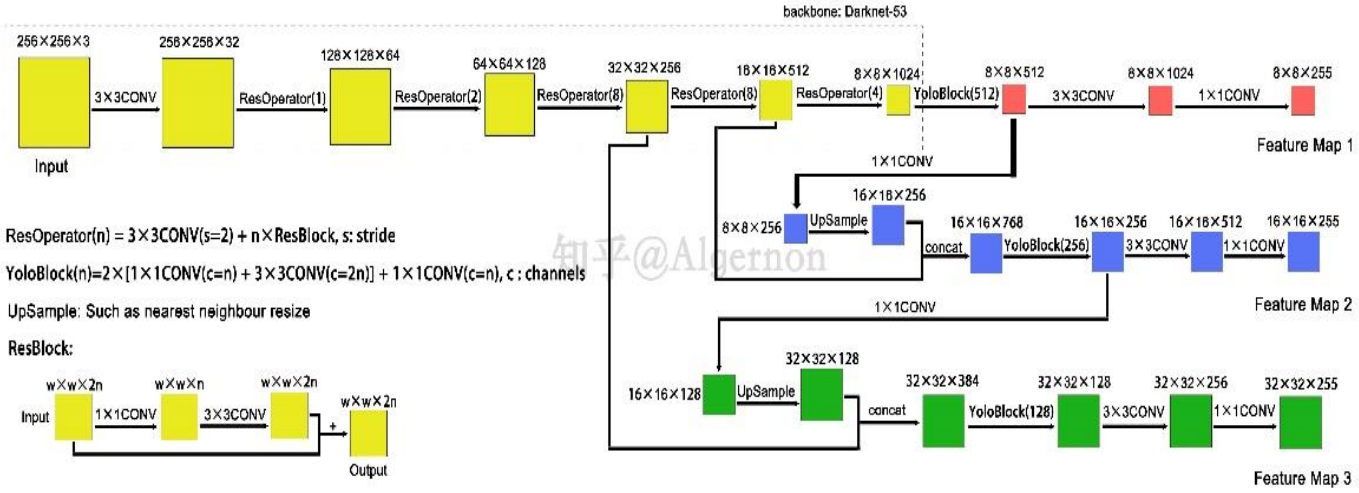
將所有在 Feature Map 上確定的區域的 shape 統一，引入了 RoI pooling。RoI Pooling 和卷積神經網絡中的 Pooling 類似，RoI Align 使用了雙線性插值，如下圖：



左圖田字框是推薦區域，藍點是統一 shape 時重新確定的點的位置。

右圖是計算左圖中每個藍點的算法，黑點就是左圖中的藍點，即需要計算的值，四個角上的紅黃藍綠點是已知點；黑點值= 黃點值* 黃色區域比例+ 藍色值* 藍色區域比例+ 紅點值* 紅色區域比例+ 綠點值* 綠色區域比例；其中黃色區域比例+ 藍色區域比例+ 紅色區域比例+ 綠色區域比例= 1.0

橘色的部分為探測層（Detection Layer）：



其中黃色部分是 darknet-53 然後最後輸出了 3 種特徵圖，沒有全連結層就代表不會限制輸入圖片尺寸，darknet-53 中使用的殘差模型(Resnet)，也就是 Residual，和 v1 方法類似，隨著網路層數的加深，有些特徵可能會變得微弱甚至消失，所以會將前面的特徵圖加入，避免這樣的情況。

v3 總共有 9 個不同大小的 Anchor Box，平均分配到 3 個輸出特徵圖，而預測的東西不變，舉上圖的第一個輸出為例，8x8x255 有 64 格，每格有三個 Anchor Box，而每個 Anchor Box 預測 t~x~，t~y~，t~w~，t~h~，t~o~，前 4 個為預測框的數值，最後 1 個是置信度，然後預測 3 個類別，所以結果就是 8*8*3(anchor boxes)*(5+3)

4. YOLOv2 vs YOLOv3

	YOLOv2	YOLOv3
Loss function	softmax loss	logistic loss
anchor bbox prior	5 個 anchor，一個折衷的選擇	9 個 anchor，提高了 IOU

detection 的策略	只有一個 detection	3 個 detection，分別是一個下采樣的，feature map 為 13*13，還有 2 個上取樣的 eltwise sum(feature map 為 26*26，52*52)
backbone	darknet-19	darknet-53

v2 的 darknet-19 和 darknet-53(卷積層有 53 層)，全連結層同樣不保留，速度和 19 相比較慢，但準確率有提升，這個修改為了要增加上取樣的數量，使得卷積層的數量更多，另外還用一連串的 3*3、1*1 卷積，3*3 的卷積增加 channel，而 1*1 的卷積在於壓縮 3*3 卷積後的特徵表示。

四、研究成果

1. 程式碼(此部分為定義偵測物件及框框的函數，並啟用電腦的攝像頭)

```
import cv2
import numpy as np

weightsPath="C:/Users/User/darknet/yolov3_1100.weights"
configPath="C:/Users/User/darknet/cfg/yolov3.cfg"
net = cv2.dnn.readNetFromDarknet(configPath,weightsPath)

layer_names = net.getLayerNames()
#output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
output_layers = [layer_names[i-1] for i in net.getUnconnectedOutLayers()]
classes = [line.strip() for line in open("C:/obj.names")]
colors = [(0,0,255),(255,0,0),(0,255,0)]
```

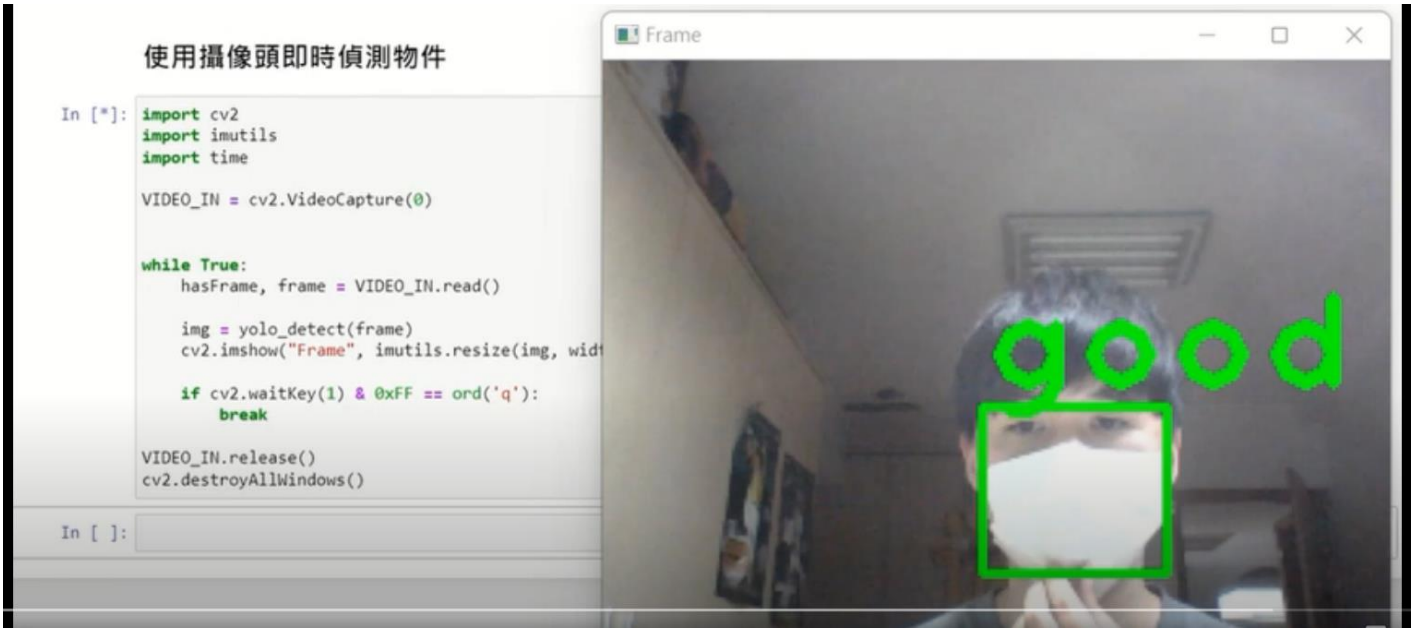
```
def yolo_detect(frame):
    # forward propogation
 #   frame = cv2.resize((224,224),fx=0,fy=0, interpolation = cv2.INTER_CUBIC)
    img = cv2.resize(frame, None, fx=0.4, fy=0.4)
    height, width, channels = img.shape
    blob = cv2.dnn.blobFromImage(img, 1/255.0, (416, 416), (0, 0, 0), True, crop=False)
    net.setInput(blob)
    outs = net.forward(output_layers)

    # get detection boxes
    class_ids = []
    confidences = []
    boxes = []

    for out in outs:
        for detection in out:
            tx, ty, tw, th, confidence = detection[0:5]
            scores = detection[5:]
            class_id = np.argmax(scores)
            if confidence > 0.3:
                center_x = int(tx * width)
                center_y = int(ty * height)
                w = int(tw * width)
                h = int(th * height)

                # 取得箱子方框座標
                x = int(center_x - w / 2)
                y = int(center_y - h / 2)
                boxes.append([x, y, w, h])
                confidences.append(float(confidence))
                class_ids.append(class_id)
```

```
#print(class_ids[i])
    # draw boxes
    indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.3, 0.4)
    font = cv2.FONT_HERSHEY_PLAIN
    for i in range(len(boxes)):
        if i in indexes:
            x, y, w, h = boxes[i]
            label = str(classes[i])
            color = colors[class_ids[i]]
            cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
            cv2.putText(img, label, (x, y -5), font, 3, color, 3)
    return img

from PIL import Image
#測試函數功能
img = cv2.imread("C:/b.jpg") #此路徑不能有中文
im = yolo_detect(img)
img_rgb = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)
#plt.imshow(img_rgb)

#使用攝像頭即時偵測物件
import cv2
import imutils
import time

VIDEO_IN = cv2.VideoCapture(0)


while True:
    hasFrame, frame = VIDEO_IN.read()

    img = yolo_detect(frame)
    cv2.imshow("Frame", imutils.resize(img, width=850))
```

2. 圖片辨識


bad
bad
good
bad
bad
good

3. 即時影像辨識



影片網址: https://www.youtube.com/watch?v=JI9QxUqYINk

五、結果與討論

心得分享、遭遇之困難與未來之方向

陳昱廷:

繼上學期修過深度學習後，這次又選了一次想做出更困難的東西出來，所以這次選了 yolo 模型的範例來做，這次的架構比較難以理解也比較少中文影片講得很清楚的，發現去年自己做的卷積神經網路真的就只是一個小小的基礎而已，我們反覆點了每個國外介紹 yolo 的影片看了又看，才稍微理解一些些，希望未來可以發展出更強大更快的模型在更科技化的時代讓我們的生活更便利。

簡睿德:

上學期在老師開的深度學習期末報告參考別人的程式用 MobileNet 訓練一個辨識辛普森卡通裡的角色，然而從網路上隨便找幾張卡通圖片丟進來模型測試，雖然結果是錯誤居多，但是經過老師的解釋後，知道是一開始沒把後來要測試的圖片拿去分類，導致後面測試有錯誤。經過上個學期的課程後，知道深度學習是件困難且長久的事情，因此選了老師開的機器學習，經過這一學期的學期後，雖然內容是蠻相近的,但老師教導的方式,換成是上課大概介紹架構及數學式子,幾次作業也變成是去推倒式子,最大的不同是期末多講了強化學習，讓我知道這方面除了一開把已知的數據及答案拿去訓練並預測未來外，還可以在未知的答案下，利用回櫃的方式讓電腦自己去學習出最佳的解答，雖然只是初步的接觸強化學習，希望之後有機會可以實際去執行例子或是利用在生活中。而最後的期末報告，更是讓我體會到機器學習的困難及有組員的好處，一開始在 YOUTUBE 上找到有人在教學怎麼利用 YOLO 來辨識人物有沒有戴口罩，剛開始看影片學覺得應該不會太困難，但真的在執行程式時，發現因為要安裝許多套件，所以會有不同版本導致的錯誤，就要依據錯誤去找到他所對應到的版本，不然就是要重新建立一個新的環境，這都花了不少時間，然後在程式成功執行後，要去了解 YOLO 的架構，發現 YOLO 的建構結合了一些之前所學到的東西及許多沒聽過的東西，使得在了解的架構的過程又花上不少時間，還有有組員的互相幫忙，最後能讓我們的報告給完成。

高棋烜:

這次機器學習的期末報告，我們選擇了跟疫情相關的口罩辨識來製作。在編輯程式碼及查找資料的過程中雖然遇到很多不懂的地方，但靠著組員們查了很多影片及網站，最終都順利的解決了。口頭報告也很順利的把我們做出來的成果展示給了同學們看，唯一的遺憾是因為軟體的更新而無法把口罩辨識現場用給同學們及老師看。

遭遇之困難:

- 安裝許多套件，會有與原作者使用不同版本所導致的錯誤，因此要依據錯誤去找到他所對應到的版本或是更改語法，不然就要在 ANACONDA 裡重新建立一個新的環境。
- 建構當中用到許多之前沒聽過的名詞及工具，要花許多時間上網查詢和與組員彼此討論後才能瞭解個大概。
- 組員在期末由於各科報告、考試多，使得能討論報告的時間不多

未來之方向:

YOLO 模型在物件偵測領域備受重視，疫情期間的日本甚至已經可以透過模型辨識出戴著口罩的人，並讀取他們施打疫苗的情況，且準確率高達 99.9%，是人臉辨識的一大進步，說不定以後可以直接透過背影分析出這個人是誰以及他的各種資料，甚至可以應用架設在路上可以捕捉通緝犯之類的工具。

參考文獻、資料

[1]https://drive.google.com/file/d/1rf-ggLJ1igZaW7yMbApr2ys0enF0ucJ3/view

[2]https://learnopencv.com/deep-learning-based-object-detection-using-yolov3-with-opencv-python-c/

[3]https://www.youtube.com/watch?v=vRqSO6RsptU
[4]https://aiacademy.tw/yolo-v4-intro/
[5]https://medium.com/@chenchoulo/yolo-%E4%BB%8B%E7%B4%B9-4307e79524fe
[6]https://github.com/ywchiu/largitdata/blob/master/code/Course_127.ipynb
[7]https://github.com/ywchiu/largitdata/blob/master/code/Course_128.ipynb
[8]https://www.youtube.com/watch?v=vGhIhitQHBE&t=37s
[9]https://www.youtube.com/watch?v=C0Otd6_jGmE&t=1106s
[10]https://www.youtube.com/watch?v=T_zFMRFCFfk
[11]https://zhuanlan.zhihu.com/p/94986199
[12]https://paddlepedia.readthedocs.io/en/latest/tutorials/computer_vision/classification/DarkNet.html
[13]https://zhuanlan.zhihu.com/p/91052975

成員工作項目:

1. 簡睿德負責 Demo 程式碼、查資料
2. 高祺烜負責 PPT、查資料
3. 陳昱廷負責書面報告、查資料