CAVEDU教育團隊部落格

[ 技術教學文-格式 ]

|  |  |  |  |
| --- | --- | --- | --- |
| **標題** | Jetson Nano 實作YOLOv5的轉移學習 | | |
|  | | | |
| **分類** |  | **標籤** | YOLOv5  Jetson Nano  Transfer Learning |
| **撰寫/攝影** |  | | |
| **前情提要** |  | | |
| **時間** |  | **材料表** |  |
| **成本** |  |
| **難度** | 中 |
| **前言** | | | |
| 在上一篇YOLOv5的文章中我們已經介紹並且應用YOLOv5在圖像、影像甚至是即時影像串流中，但是這都是利用他們提供的權重也就是每次辨識出來的項目都是他們給的，假設你想要辨識出特定的物品該怎麼辦？這時候就要來做轉移學習了！ | | | |
| **內文** | | | |
| 轉移學習 ( Transfer Learning ) 人類會利用過去的經驗來幫助自己吸收新的知識技術，就像是大學念微積分的時候都會想到高中教的解決方法，而神經網路中的轉移學習 ( 遷移式學習 )　也是類似的概念。大家都知道神經網路的訓練是相當耗費時間的，一開始初始化權重，經過無數次的訓練才能有結果，那如果能夠將別人訓練好的權重拿來當初始化的權重，是否能加速訓練呢？大部分的情況下答案都是肯定的。 轉移學習簡單原理 以影像辨識來當例子，CNN中的kernel就像是一個特徵萃取濾波器，可能是擅長萃取直線、圓形或者顏色等等的，一般訓練的方法都是隨機找幾個濾波器測試，好的就留下來不好的就再換成別的，這樣一步一步的摸索出來；而轉移學習則是別人已經告訴你可以用這些濾波器，我用過效果不錯！你在自己評估是否好用或是否要修改！所以一定是比一般訓練還要來的快、狠、準！  從下圖可以看到，假設你有一個神經網路用來分辨動物（狗、貓、獅子、熊），這時候你想利用他來辨識男孩、女孩，只需要將最後一層輸出層給換掉，給予隨機的初始化數值，讓神經網路只「微調」這一層即可，藉由先前神經網路辨識動物的方法來辨識男孩女孩。   |  | | --- | |  | |  |   當然，當你面臨的問題更複雜，你可以不只微調一層！在一般常看到的解釋中，對於語音辨識問題會微調後幾層；影像辨識問題則會微調前幾層。   |  |  | | --- | --- | |  |  |   主要的意義在於神經網路前幾層大多用於特徵萃取，對於語音來說前幾層能保留對於聲音特徵的發音方式，後幾層才會聽到聲音的內容；而影像辨識如果要辨識的物品差異太大，就不能使用相同的特徵萃取方式，所以才會選擇微調前幾層。  在轉移學習的領域相當的有趣也很複雜，這邊我們帶到一點基礎即可，更深入的資訊可以在網路上找到很多～接下來我們就要針對YOLOv5進行轉移學習了。 Transfer Learning on YOLOv5 我想讓YOLO學會辨識我手指比的數字，先以辨識0、1為主。範例如下   |  |  | | --- | --- | |  |  |  建置環境 在上一篇YOLOv5中已經介紹如何使用 virtualenv 來建置虛擬環境，所以這邊簡單帶個流程，想要知道更詳細的可以去查看上一篇：   * 安裝 Virtualenv、Virtualenvwrapper、修改環境變數、開啟新環境 * 安裝git tool、下載YOLOv5 Github專案 * 安裝套件 ( PyTorch 額外裝、OpenCV用Link、Scipy 最後裝 )  準備數據 如何準備數據呢？可以透過筆電的攝影機來拍照，簡單的拍照程式就是OpenCV開啟相機讀取按鍵，儲存照片，這邊就不多作介紹了，我額外寫了拍照數量，這樣我才知道數據是否平均。   |  | | --- | | import cv2  import os  import numpy as np  ############## save data ##############  def save(trg\_path, idx, frame):  global label  save\_path = os.path.join(trg\_path, rf'{idx}\_{label[idx]}.jpg')  print(save\_path)  cv2.imwrite(save\_path, frame)  label[idx] = label[idx]+1  ############## set target path ##############  trg\_path = 'custom\_datasets'  if os.path.exists(trg\_path)==False: os.mkdir(trg\_path)  ############## get camera & set size ##############  #w\_size, h\_size = 512, 512  cap = cv2.VideoCapture(0)  #cap.set(cv2.CAP\_PROP\_FRAME\_WIDTH, w\_size)  #cap.set(cv2.CAP\_PROP\_FRAME\_HEIGHT, h\_size)  ############## set parameters ##############  label = np.zeros([10], dtype=int)  print(f'label len : {len(label)}')  ############## open camera % save data ##############  while(True):  ret, frame = cap.read()  overlay = frame.copy()  ############## show info ##############  text = '{}{}{}{}{}'.format(f'Label\n0: {label[0]}\n1: {label[1]}\n',  f'2: {label[2]}\n3: {label[3]}\n' ,  f'4: {label[4]}\n5: {label[5]}\n' ,  f'6: {label[6]}\n7: {label[7]}\n' ,  f'8: {label[8]}\n9: {label[9]}\n' )  for i, txt in enumerate(text.split('\n')):  cv2.putText(overlay, txt,(20,25\*i+20), cv2.FONT\_HERSHEY\_SIMPLEX, .5, (0,0,255), 1 )  cv2.imshow('Create\_Your\_Own\_Datasets', overlay)  key = cv2.waitKey(1)    if key== ord('q'): break    for i in range(10):  if key == ord(f'{i}'):  save(trg\_path, i, frame)  cap.release()  cv2.destroyAllWindows() |   拍完之後資料都會存放在 data的資料夾中 (雖然照片是顯示custom\_datasets)：    接下來就是最煩人標Label了，我們可以透過許多工具來完成，這邊我們使用的是LabelImg，一個非常簡易的工具，直接有支援YOLO格式。  點擊Open Dir選擇欲開啟的數據集、Change Save Dir則是標完的數據放哪裡、PascalVOC是儲存的格式再點一下會變成YOLO。   |  | | --- | |  |   開啟照片的所在資料夾之後會讓你選擇 Annotation的資料夾，也就是標籤的文檔。PascalVOC的話是選擇 .xml而YOLO則是.txt，最後因為還要存放標籤所以也新增了一個label的資料夾。    都準備好就可以開始標Label了，按下快捷鍵w可以繪製，接著就可以儲存：    接下來就可以看到label的資料夾已經有對應的txt檔產生：    當我們標到其他標籤，classes會自動更新：    這邊可以開啟　Auto Saving 才不用每次都要再按儲存：    而他當然也可以標多個標籤，這邊我直接擷取教學影片的畫面：    整體而言，如果資料量不大，LabelImg算是不錯的選擇，市面上有些已經結合影像處理或物件辨識先幫使用者標出目標，使用者再做二次修改的！不過那些都要付費就是了！所以教學或舉例用這種簡易版也是不錯的選擇。  接下來我們要將數據資料稍微整理一下，目標如下：    先取得資料以及確認一下資料夾是否存在，不存在就創建：   |  | | --- | | ############## Check dir ##############  dataset\_dir ='data'  label\_dir = 'label'  dataset = os.listdir(dataset\_dir)  label = os.listdir(label\_dir)  images = []  labels = []  custom\_dir = 'custom'  data\_root = os.path.join(custom\_dir, 'images')  label\_root = os.path.join(custom\_dir, 'labels')  data\_train\_dir = os.path.join(data\_root, 'train')  data\_val\_dir = os.path.join(data\_root, 'val')  label\_train\_dir = os.path.join(label\_root, 'train')  label\_val\_dir = os.path.join(label\_root, 'val')  dir\_list = [data\_train\_dir, data\_val\_dir, label\_train\_dir, label\_val\_dir]  for dir in dir\_list:  if os.path.exists(dir)==False:  print(f"Create dir : {dir}")  os.makedirs(dir) |   接著定義一些常用的副函式 (打亂順序、拼接檔名) 以及取得驗證資料 (因為資料較少所以我只取5筆)：   |  | | --- | | ############## Get val data ##############  #打亂數據  def shuffle(data):  arr = np.array(data)  np.random.shuffle(arr)  return arr.tolist()  #拼接檔名  def get\_path(src, name, ftype):  return f'{src}\{name}.{ftype}'  val\_num = 5  pure\_name = [ i.split('.')[0] for i in dataset ]  val\_data = shuffle(pure\_name)[:val\_num]  print('Total Data length: ', len(pure\_name))  print('Validation Data: ', val\_data)  for d in val\_data:  src\_data = get\_path(dataset\_dir, d, 'jpg')  src\_label = get\_path(label\_dir, d, 'txt')  trg\_data = get\_path(data\_val\_dir, d, 'jpg')  trg\_label = get\_path(label\_val\_dir, d, 'txt')  shutil.copy(src\_data, trg\_data)  shutil.copy(src\_label, trg\_label)  pure\_name.remove(d) |   在取得驗證資料的同時我也將資料列表的驗證資料名稱刪除，所以接下來只要把剩餘的資料移動到 train 中的 images、labels即可。   |  | | --- | | ############## Split data ##############  print("="\*50)  print('New Data length: ', len(pure\_name))  for d in pure\_name:  src\_data = get\_path(dataset\_dir, d, 'jpg')  src\_label = get\_path(label\_dir, d, 'txt')  trg\_data = get\_path(data\_train\_dir, d, 'jpg')  trg\_label = get\_path(label\_train\_dir, d, 'txt')  shutil.copy(src\_data, trg\_data)  shutil.copy(src\_label, trg\_label)  print("Finish！") |   完成資料整理後就要去定義yaml檔，yolo的訓練都會依靠yaml來定義數據集的資訊，而yaml檔都會放在yolov5\data 資料夾當中，我使用的方法是複製coco.yaml來修改，檔名先改成custom.yaml，內容的部分主要修改nc為標籤的種類數量、name為標籤名稱：   |  | | --- | | # train and val data as 1) directory: path/images/, 2) file: path/images.txt, or 3) list: [path1/images/, path2/images/]  train: ../custom/images/train/ # 128 images  val: ../custom/images/train/ # 128 images  # number of classes  nc: 2  # class names  names: ['0', '1'] |  Colab 進行 Training 全部完成之後就要進行訓練了，我們使用Colab來實現，大概跑個10個epochs就有一些成效出現了！首先，要先上傳到Google Drive ( 以下簡稱 GDrive ) 上，這邊我們上傳到GDrive上之後要注意一下檔案擺放的位置記得是yolov5資料夾與custom資料夾同層。  接著開啟一個Colab檔，從編輯>筆記本設定中轉換成GPU模式，並且稍微查看一下自己拿到哪一個GPU。   |  | | --- | | import torch  print(torch.cuda.get\_device\_properties(0)) |     讓Colab跟自己的GDrive連動，因為我連動過了所以畫面不太一樣：    連動之後可以透過os套件移動到yolov5的位置，並且安裝缺失套件：   |  | | --- | | import os  os.chdir('/content/drive/My Drive/Cavedu/Article/yolov5')  !ls  !pip install -U PyYAML |     懶惰如我，改寫太麻煩直接使用指令進行訓練即可：   |  | | --- | | !!python train.py --img 640 --batch 16 --epochs 30 --data ./data/custom.yaml --cfg ./models/yolov5s.yaml --weights 'yolov5s.pt' |   最後在run的資料夾中找到exp 的資料夾，像我訓練過很多次所以數字顯示17，總之就是找到最後一個數字就是最後一次訓練的：    點開來可以看到有很多張驗證結果的圖片還有一個存放權重的資料夾：    裡面通常會有兩個，一個是best一個last，就如字面上的意思就是最好的跟最後的，當然！看到best就直接選best！我們將其下載到我們電腦端的YOLOv5資料夾中。    執行detect.py的程式，並且使用 –source 0 ，也就是開啟相機進行即時影像辨識的部分，然後權重則選擇我們剛剛下載下來的權重：   |  | | --- | | python detect.py --source 0 --weights best.pt |     可以看到已經有不錯的成效了，不過因為訓練數據太少，回合數太低，把這兩項加強之後應該就會改善不穩定的狀況了～當然照片的背景也很重要，換個背景說不定準度又下降了！ 影片成果  結語 這篇結束相信你已經學會如何利用強大的YOLOv5來實作自己的客製化物件辨識了！如果有想更了解什麼請再留言告訴我。 | | | |
| **相關文章** | | | |
| 在Jetson Nano (TX1/TX2)上使用Anaconda与PyTorch 1.1.0  <https://zhuanlan.zhihu.com/p/64868319> | | | |