食品影像辨識

第六組
112034564 曾聖閔
112034554 陳　薪
112034556 莊傑宇

LIST OF CONTENTS

PART ONE

背景介紹

背景介紹

餐飲店在日常運營中經常面臨食品管理效率低下的問題，這包括手動分類食品和管理庫存的不準確性。這些問題不僅導致運營成本增加，還可能造成食材浪費。

目前問題

- 傳統手動管理效率低下
- 容易出現管理錯誤
- 需應對市場需求和顧客期望

專案目標

- 使用深度學習模型進行食品影像辨識
- 自動識別和分類食品
- 減少人力成本，提高管理效率

5W1H

WHAT
食品影像及其分類結果。

WHO
餐飲業者，包括滷味店、餐廳。

WHERE
餐廳後台管理系統和櫃台結帳。

WHEN
在需要對食品進行自動分類和管理時，
以及在訂單高峰期進行快速處理時。

WHY
傳統分類方式效率低且容易出錯，
無法滿足餐飲業對高效管理的需求。

HOW
利用深度學習模型進行食品影像識別，
實現自動分類和管理。

PART TWO

實驗方法

資料集介紹

在本專案中，我們選用了來自Foodcam的UEC FOOD 100 Ver 1.0資料集。

DATA 範例圖

- 100種不同的食品類別
- 12740張的食品影像
- 味噌湯類別的影像數量最多，共有729張
- 蛋包飯類別的影像數量最少，只有101張影像

資料集介紹

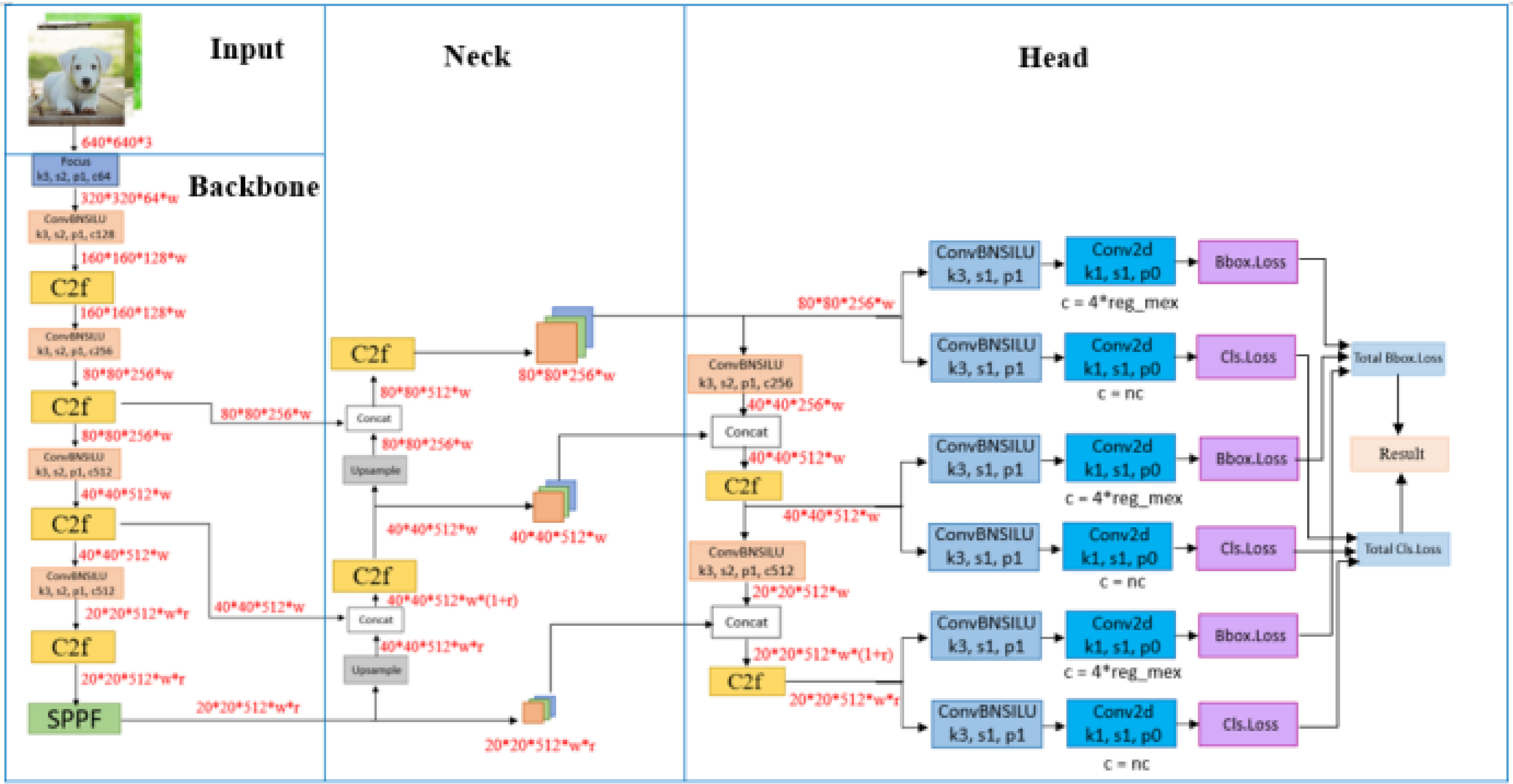
每個類別目錄下都包含一個名為bb_info.txt的文件，其中記錄了食品影像的邊界框訊息。

```
img x1 y1 x2 y2
400 0 0 400 300
401 0 0 280 210
402 28 24 254 224
403 0 0 400 300
404 0 0 480 309
405 23 21 306 256
406 16 9 266 208
407 26 20 221 184
408 6 0 271 226
409 21 15 317 244
410 15 10 323 271
411 0 50 187 186
412 13 46 359 269
413 0 0 159 142
```

Model介紹

- YOLO(You Only Look Once)是即時物件偵測中重要的技術。
- 這項演算法在各種領域和應用中被廣泛使用。
- 在過去幾年裡生成了許多版本。
- YOLOv8 引入了一些新的設計思想和技術，以提高模型的精度和速度。

YOLOv8 架構



Input
Neck
Head
Backbone
640*640*3
320*320*64*w
160*160*128*w
C2f
80*80*256*w
40*40*512*w
20*20*512*w*r
SPPF
Concat
Upsample
80*80*512*w
40*40*512*w*(1+r)
20*20*512*w*(1+r)
ConvBNSILU
k3, s2, p1, c256
k3, s2, p1, c512
k3, s1, p1
Conv2d
k1, s1, p0
c = 4*reg_mex
c = nc
Bbox.Loss
Cls.Loss
Total Bbox.Loss
Total Cls.Loss
Result

PART THREE

模型訓練

資料前處理

圖像增強

```python
def augment_images(image_path, output_path, num_augmentations):
    os.makedirs(output_path, exist_ok=True)
    image = Image.open(image_path)
    base_name = os.path.splitext(os.path.basename(image_path))[0]

    transform = transforms.Compose([
        transforms.RandomRotation(degrees=(90, 270)),
    ])

    for i in range(num_augmentations):
        augmented_image = transform(image)
        augmented_image.save(os.path.join(output_path, f"{base_name}_aug_{i}.jpg"))
```

資料前處理

標準化邊界框數值

```python
# 自動檢測圖像尺寸
with Image.open(img_path) as img:
    img_width, img_height = img.size

# 計算 YOLO 格式的標籤
x_center = (x1 + x2) / 2 / img_width
y_center = (y1 + y2) / 2 / img_height
width = (x2 - x1) / img_width
height = (y2 - y1) / img_height

label = f"{class_id - 1} {x_center} {y_center} {width} {height}\n"     # YOLO 的格式期望類別 ID 從 0 開始，所以 class_id -1
```

img	x1	y1	x2	y2
400	0	0	400	300
401	0	0	280	210
402	28	24	254	224
403	0	0	400	300
404	0	0	480	309
405	23	21	306	256
406	16	9	266	208
407	26	20	221	184
408	6	0	271	226
409	21	15	317	244
410	15	10	323	271
411	0	50	187	186
412	13	46	359	269
413	0	0	159	142

資料前處理

目錄結構轉換

```
# 設置好欲進行儲存的路徑
def create_directories(output_path):
    train_image_dir = os.path.join(output_path, 'train', 'images')
    train_label_dir = os.path.join(output_path, 'train', 'labels')
    val_image_dir = os.path.join(output_path, 'val', 'images')
    val_label_dir = os.path.join(output_path, 'val', 'labels')


    os.makedirs(train_image_dir, exist_ok=True)
    os.makedirs(train_label_dir, exist_ok=True)
    os.makedirs(val_image_dir, exist_ok=True)
    os.makedirs(val_label_dir, exist_ok=True)
```

YOLOv8 超參數優化

Epoch	Batch Size	Optimizer
50	8	Auto
50	8	SGD
50	16	Auto
50	16	SGD
75	8	Auto
75	8	SGD
75	16	Auto
75	16	SGD

YOLOv8 Run

```
                 from  n    params  module                                       arguments
  0                -1  1       464  ultralytics.nn.modules.conv.Conv             [3, 16, 3, 2]
  1                -1  1      4672  ultralytics.nn.modules.conv.Conv             [16, 32, 3, 2]
  2                -1  1      7360  ultralytics.nn.modules.block.C2f             [32, 32, 1, True]
  3                -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  4                -1  2     49664  ultralytics.nn.modules.block.C2f             [64, 64, 2, True]
  5                -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  6                -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128, 2, True]
  7                -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  8                -1  1    460288  ultralytics.nn.modules.block.C2f             [256, 256, 1, True]
  9                -1  1    164608  ultralytics.nn.modules.block.SPPF            [256, 256, 5]
 10                -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11            [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                -1  1    148224  ultralytics.nn.modules.block.C2f             [384, 128, 1]
 13                -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14            [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 15                -1  1     37248  ultralytics.nn.modules.block.C2f             [192, 64, 1]
 16                -1  1     36992  ultralytics.nn.modules.conv.Conv             [64, 64, 3, 2]
 17           [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                -1  1    123648  ultralytics.nn.modules.block.C2f             [192, 128, 1]
 19                -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
 20            [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                -1  1    493056  ultralytics.nn.modules.block.C2f             [384, 256, 1]
 22      [15, 18, 21]  1   1086604  ultralytics.nn.modules.head.Detect           [100, [64, 128, 256]]
Model summary: 225 layers, 3346140 parameters, 3346124 gradients, 9.7 GFLOPs
```

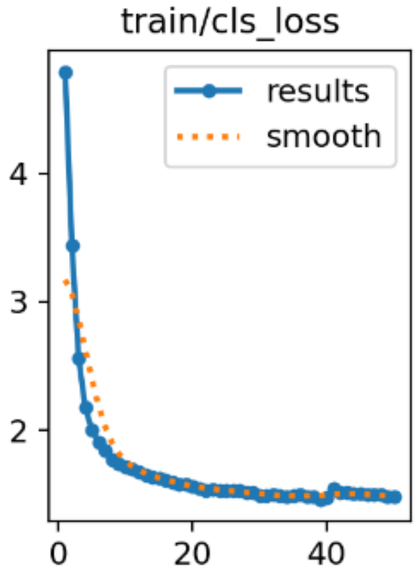
YOLOv8

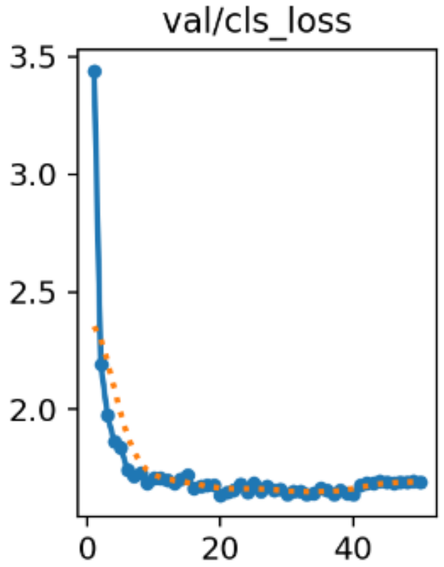
實驗結果

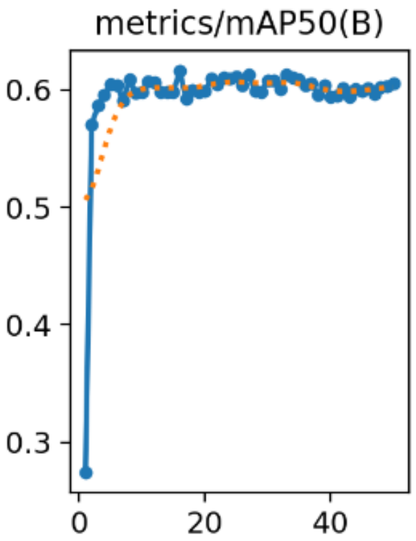
Epoch	Batch Size	Optimizer	Precision	Recall	mAP@0.5	mAP50-95	Time(hrs)
50	8	Auto	0.6377	0.6605	0.595	0.3101	1.72
50	8	SGD	0.6677	0.6853	0.5995	0.3146	1.667
50	16	Auto	0.6992	0.659	0.6054	0.3178	1.62
50	16	SGD	0.6458	0.6291	0.5879	0.301	1.635
75	8	Auto	0.667	0.6768	0.603	0.314	2.52
75	8	SGD	0.6621	0.6743	0.6023	0.3139	2.786
75	16	Auto	0.6833	0.6732	0.6018	0.317	2.681
75	16	SGD	0.6948	0.6688	0.5967	0.3148	2.616

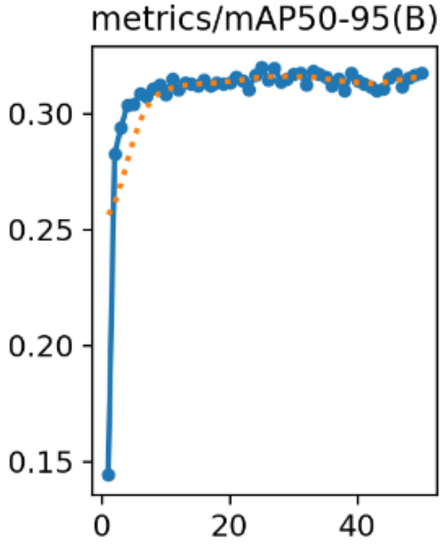
YOLOv8

實驗結果


train/cls_loss
results
smooth
4
3
2
0
20
40


val/cls_loss
3.5
3.0
2.5
2.0
0
20
40


metrics/mAP50(B)
0.6
0.5
0.4
0.3
0
20
40


metrics/mAP50-95(B)
0.30
0.25
0.20
0.15
0
20
40

YOLOv5 Run

```
                 from  n    params  module                                       arguments
 0                  -1  1      1760  ultralytics.nn.modules.conv.Conv             [3, 16, 6, 2, 2]
 1                  -1  1      4672  ultralytics.nn.modules.conv.Conv             [16, 32, 3, 2]
 2                  -1  1      4800  ultralytics.nn.modules.block.C3              [32, 32, 1]
 3                  -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
 4                  -1  2     29184  ultralytics.nn.modules.block.C3              [64, 64, 2]
 5                  -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
 6                  -1  3    156928  ultralytics.nn.modules.block.C3              [128, 128, 3]
 7                  -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
 8                  -1  1    296448  ultralytics.nn.modules.block.C3              [256, 256, 1]
 9                  -1  1    164608  ultralytics.nn.modules.block.SPPF            [256, 256, 5]
10                  -1  1     33024  ultralytics.nn.modules.conv.Conv             [256, 128, 1, 1]
11                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
12            [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
13                  -1  1     90880  ultralytics.nn.modules.block.C3              [256, 128, 1, False]
14                  -1  1      8320  ultralytics.nn.modules.conv.Conv             [128, 64, 1, 1]
15                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
16            [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
17                  -1  1     22912  ultralytics.nn.modules.block.C3              [128, 64, 1, False]
18                  -1  1     36992  ultralytics.nn.modules.conv.Conv             [64, 64, 3, 2]
19           [-1, 14]  1         0  ultralytics.nn.modules.conv.Concat           [1]
20                  -1  1     74496  ultralytics.nn.modules.block.C3              [128, 128, 1, False]
21                  -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
22           [-1, 10]  1         0  ultralytics.nn.modules.conv.Concat           [1]
23                  -1  1    296448  ultralytics.nn.modules.block.C3              [256, 256, 1, False]
24       [17, 20, 23]  1   1086604  ultralytics.nn.modules.head.Detect           [100, [64, 128, 256]]
YOLOv5n summary: 262 layers, 2843756 parameters, 2843740 gradients, 8.7 GFLOPs
```

YOLOv5

實驗結果

Epoch	Batch Size	Precision	Recall	mAP@0.5	mAP50-95	Time(hrs)
50	8	0.6472	0.6534	0.6	0.3011	1.924
50	16	0.649	0.652	0.612	0.3148	1.714
75	8	0.6541	0.6775	0.6213	0.3058	2.912
75	16	0.6539	0.6362	0.598	0.2998	2.762

PART FOUR

顧客及管理者功能

顧客及管理者功能

Account

Account

顧客及管理者功能

Manager

管理後臺

訂單編號	姓名	電話	取餐時間	產品名稱	價格	數量	日期	刪除
20240613-525	123	456	17:30	1	120	1	2024-06-13 21:30:54	刪除

新增訂單

新增

請填入下列資料

*收件人姓名：

*電話：

*取餐時間: 17:30

*幾號餐: 1

*價格：

*數量：

新增訂單

顧客及管理者功能

Account

查詢訂單

PART FIVE

DEMO

PART SIX

結論

結論

本專案通過結合深度學習模型和網頁應用，實現了食品影像辨識的自動化，並在此基礎上開發了一個完整的餐飲管理系統。系統具備以下主要功能：

1. 食品影像自動分類：通過使用YOLO v8進行食品影像的分類，我們能夠準確識別和分類多種食品。

2. 食品價格計算：系統根據識別結果，自動計算食品的價格。這不僅提高了效率，還減少了人工計算的誤差。

3. 生成付款QR碼：系統根據計算出的價格，自動生成對應的支付QR碼，方便顧客使用手機進行快速支付，提升了顧客的購物體驗。

貢獻

主要貢獻為：

1. 提升工作效率：本系統通過自動化食品識別和價格計算，顯著提升了餐飲業者的工作效率，減少了手動操作的時間和成本。

2. 降低錯誤率：通過使用深度學習模型進行食品分類，減少了人工操作中可能出現的分類錯誤，確保了數據的準確性和一致性。

3. 改進顧客服務：系統自動生成支付QR碼，簡化了支付過程，提升了顧客的購物體驗，提高了顧客滿意度和忠誠度。

侷限性

主要侷限性為：

1. 資料依賴：模型的準確性依賴於訓練資料的品質和多樣性，對於未見過的食品類別或新的影像資料，模型的識別準確性可能會有所下降。

2. 模型更新需求：隨著食品類別和樣本的變化，模型需要定期更新和重新訓練，以保持其準確性和適用性。

3. 計算資源要求：深度學習模型的訓練和運行需要較高的計算資源，對於一些小型餐飲業者，可能需要額外的技術支持和硬體投入。

適用性

主要適用性為：

1. 餐飲業：系統主要適用於餐飲行業，尤其是需要進行大量食品分類和價格計算的餐廳、外賣店和食品配送中心等。

2. 零售業：系統可以擴展應用於超市和便利店等零售場景，實現商品的自動分類和價格計算，提升運營效率。

未來展望

擴展資料集

優化模型性能

新增更多功能

THANKS
FOR LISTENING