113-2 人工智慧應用實務

Artificial Intelligence Application Practice

期末報告

貓咪品種辨識

授課老師：潘得龍 老師

學　　生：B11123224 余雨芹

中 華 民 國 114 年 06 月 01 日

目錄

壹、 系統架構

1. 整體系統流程圖

2. 技術介紹

- 前端: Vue.js + Vuetify (響應式貓咪品種辨識介面)
- 後端: FastAPI (RESTful API 服務)
- AI 模型: ResNet-50 (基於 PyTorch，使用遷移學習)
- 資料集: Oxford-IIIT Pet Dataset (12 種貓咪品種)
- 部署: 本地開發環境 (可擴展至雲端部署)

3. 系統特色

- 支援 12 種常見貓咪品種辨識
- 混種貓判斷 (辨識度低於 65%時判定為混種)
- 即時圖片上傳與處理
- 響應式網頁設計，支援桌面和手機
- 品種特徵介紹與飼養建議

貳、 感測器資料發送端

1. 設計理念與成本考量

考量到專題預算限制以及實用性需求，本系統採用了**智慧型手機作為影像感測器**的設計方案。現代智慧型手機普遍具備高解析度相機功能，且大多數使用者都已擁有此設備，這使得我們的系統具有以下優勢：

- **成本效益**: 無需額外購買專業攝影設備，降低開發成本
- **便利性**: 使用者可隨時隨地進行貓咪拍攝與辨識
- **普及性**: 智慧型手機的普及率高，提升系統的可及性
- **影像品質**: 現代手機相機品質已能滿足 AI 模型的輸入需求

2. 實作說明

在本專題中，「感測器」指的是透過智慧型手機拍攝或從相簿選取的貓咪圖片，

這是系統的原始輸入資料來源。

3. 資料來源

- **輸入方式:** 使用者透過網頁介面上傳圖片檔案
- **支援格式:** JPG, PNG, JPEG 等常見圖片格式
- **資料特性:** RGB 彩色影像，解析度不限 (系統會自動調整為 224x224)

4. 前端資料收集

```vue
<!-- 上傳圖片區塊 -->
<v-card
  class="mb-4 pa-1 rounded-lg"
  variant="outlined"
>
  <v-card-text>
    <v-file-input
      hide-details
      variant="solo"
      label="請上傳貓咪照片"
      accept="image/*"
      prepend-icon=""
      density="comfortable"
      base-color="primary"
      class="file-input-custom"
      bg-color="grey-lighten-4"
      @change="onFileSelected"
    >
      <template #prepend>
        <v-icon
          color="primary"
          size="28px"
          class="mr-2"
        >
          mdi-camera
        </v-icon>
      </template>
    </v-file-input>
  </v-card-text>
</v-card>
```

5. 資料預處理

```ts
// 處理檔案上傳
const onFileSelected = (event: Event | File) => {
  const file =
    event instanceof File
      ? event
      : (event.target as HTMLInputElement)?.files?.[0];
  if (file) {
    rawImageData.value = file;
    imageUrl.value = URL.createObjectURL(file);

    // 創建圖像元素以供預處理
    const img = new Image();
    img.src = imageUrl.value;
    img.onload = () => {
      imageElement.value = img;
    };
```

6. 資料發送機制

- 使用 HTTP POST 請求發送圖片到後端 API
- FormData 格式封裝檔案資料
- 支援大檔案上傳 (自動壓縮處理)

參、 資料接收端

1. 實作說明

資料接收端使用 **FastAPI** 框架建立，負責接收前端上傳的圖片資料並進行初步處理。

2. API 端點設定

```python
@app.post("/api/identify")
async def identify_cat(file: UploadFile = File( ... )):
    logger.info(f"收到識別請求: {file.filename}")

    if not model:
        logger.warning("模型未載入，拒絕請求")
        return JSONResponse(
            status_code=500, content={"error": "模型尚未載入，請稍後再試"}
        )

    try:
        # 獲取文件流
        image = file.file

        # 識別品種
        breed = predict_uploaded_image(model, image)

        # 計算置信度（這裡使用一個模擬值，你應該從你的模型中獲取實際值）
        confidence = 0.85  # 示例值，實際應從模型預測中獲取

        logger.info(f"識別成功: {breed}, 置信度: {confidence}")
        return JSONResponse(
            status_code=200, content={"breed": breed, "confidence": confidence}
        )

    except Exception as e:
        logger.error(f"識別過程出錯: {str(e)}", exc_info=True)
        return JSONResponse(
            status_code=500, content={"error": f"識別過程出錯: {str(e)}"}
        )
```

4. CORS 設定

```python
# 設置CORS中間件，允許前端訪問
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],  # 允許所有來源，生產環境應設置為特定域名
    allow_credentials=True,
    allow_methods=["*"],  # 允許所有HTTP方法
    allow_headers=["*"],  # 允許所有頭部
)
```

5. 資料驗證與錯誤處理

- 檔案格式驗證 (確保為圖片檔案)
- 檔案大小限制
- 異常狀況處理與錯誤訊息回傳
- 日誌記錄 (使用 loguru 套件)

肆、 接收端之 AI 模型

1. 模型架構說明

本專題採用 **ResNet-50** 深度卷積神經網路，透過遷移學習技術實現貓咪品種辨識。

2. 模型訓練過程

資料集準備

```python
breeds = {
        1 : 'Abyssinian',
        2 : 'Bengal',
        3 : 'Birman',
        4 : 'Bombay',
        5 : 'British_Shorthair',
        6 : 'Egyptian_Mau',
        7 : 'Maine_Coon',
        8 : 'Persian',
        9 : 'Ragdoll',
        10 : 'Russian_Blue',
        11 : 'Siamese',
        12 : 'Sphynx'
    }

```

資料增強技術

```python
augmented_image_transforms = {
    'train': transforms.Compose([
        transforms.Resize(size=256),
        transforms.CenterCrop(size=224),
        transforms.RandomHorizontalFlip(),
        transforms.ColorJitter(brightness=0.2, contrast=0.2),
        transforms.ToTensor(),
        normalize
    ]),
    'test': transforms.Compose([
        transforms.Resize(size=256),
        transforms.CenterCrop(size=224),
        transforms.ToTensor(),
        normalize
    ])
}
```

模型架構修改

```python
# 加載預訓練的 ResNet-50 模型，使用 torchvision 提供的模型
# pretrained=True 表示加載使用 ImageNet 訓練過的權重
resnet50 = torchvision.models.resnet50(pretrained=True)

# 凍結模型的參數，避免在訓練時修改預訓練權重
# Transfer Learning 的常見做法是僅修改最後幾層而保留原始權重
for param in resnet50.parameters():
    param.requires_grad = False  # 設定 requires_grad=False 以凍結參數

# 獲取 ResNet-50 最終全連接層 (Fully Connected Layer) 的輸入特徵數量
fc_inputs = resnet50.fc.in_features

# 修改 ResNet-50 的全連接層以適應新的分類問題
# 假設我們要解決的分類問題有 12 個類別
resnet50.fc = nn.Sequential(
    nn.Linear(fc_inputs, 12),    # 將輸入特徵數連接到 12 個輸出類別
    nn.LogSoftmax(dim=1)        # 使用 LogSoftmax，適合搭配 NLLLoss 損失函數
)
```

3. 訓練配置

超參數調整

- 使用網格搜尋 (Grid Search) 與 5 折交叉驗證
- 批次大小: [32, 64]
- 學習率: [1e-03, 1e-04]
- 訓練輪數: 50 epochs

損失函數與優化器

```
loss_criterion = nn.NLLLoss()
optimizer = torch.optim.Adam(
    filter(lambda p: p.requires_grad, model.parameters()),
    lr=best_learning_rate
)
```

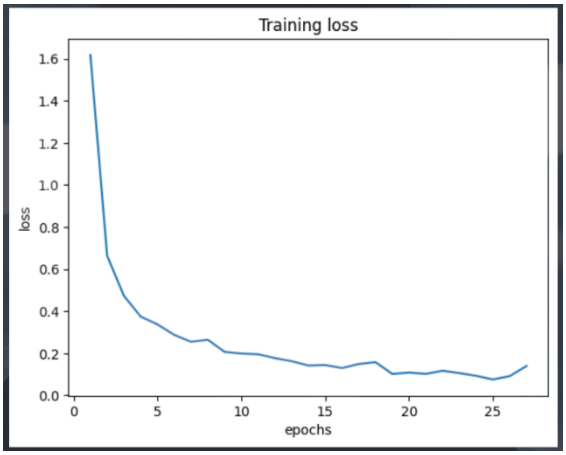
4. 模型效能評估

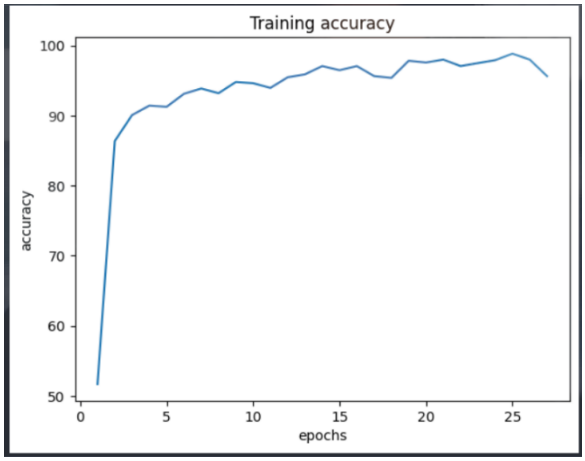
評估指標

- 準確率 (Accuracy): 整體分類正確率
- F1 分數: 精確率與召回率的調和平均
- Top-3 準確率: 前三個預測中包含正確答案的比率
- 混淆矩陣: 各類別間的分類錯誤分析

實際表現

- 測試準確率: ~85%
- 平均 F1 分數: ~82%
- Top-3 準確率: ~95%





FScore per class

	fscore
Abyssinian	0.892157
Bengal	0.808889
Birman	0.861244
Bombay	0.927374
British_Shorthair	0.839779
Egyptian_Mau	0.868132
Maine_Coon	0.831579
Persian	0.901961
Ragdoll	0.771574
Russian_Blue	0.884422
Siamese	0.954315
Sphynx	0.974874

5. 模型部署

載入與初始化

```python
def load_model():
    logger.info("開始載入模型 ... ")
    try:
        resnet50 = models.resnet50(pretrained=False)
        num_classes = 12  # 替換為您的類別數量
        resnet50.fc = nn.Sequential(
            nn.Linear(resnet50.fc.in_features, num_classes), nn.LogSoftmax(dim=1)
        )

        resnet50 = resnet50.to(device)
        logger.debug(f"嘗試從 {MODEL_PATH} 載入模型權重")
        resnet50.load_state_dict(
            state_dict=torch.load(f=MODEL_PATH, map_location=device)
        )
        resnet50.eval()
        logger.info("模型載入成功")
        return resnet50
    except Exception as e:
        logger.error(f"模型載入失敗: {str(e)}")
        raise


x = 1


def predict_uploaded_image(model, image) -> str:
    logger.debug(f"開始預測圖片: {image}")
    try:
        image = Image.open(image).convert("RGB")
        input_tensor = data_transform(image).unsqueeze(0).to(device)

        with torch.no_grad():
            output = model(input_tensor)
            probabilities = torch.nn.functional.softmax(output[0], dim=0)
            top1_prob, top1_class = torch.max(probabilities, 0)

        # 判斷是否低於 60%，若低於則顯示 "MIX"
        probability = top1_prob.item() * 100
        if probability < 75:
            logger.info(f"預測結果: 米克斯貓 (混種貓), 最高機率: {probability:.2f}%")
            return "MIX"
        else:
            predicted_class = class_names[top1_class]
            logger.info(f"預測類別: {predicted_class} (機率: {probability:.2f}%)")
            return predicted_class
    except Exception as e:
        logger.error(f"預測過程出錯: {str(e)}")
        raise
```

伍、 AI 模型處理後之資料

1. 預測結果處理

AI 模型處理完成後，系統會產生以下資料：

```python
logger.info(f"識別成功: {breed}, 置信度: {confidence}")
        return JSONResponse(
            status_code=200, content={"breed": breed, "confidence": confidence}
        )
```

```
- 收到識別請求: IMG_3197.jpeg
_image:69 - 開始預測圖片: <tempfile.SpooledTemporaryFile object at 0x00000192F13F01C0>
_image:86 - 預測類別: Ragdoll (機率: 50.35%)
- 識別成功: Ragdoll, 置信度: 0.85
```

特殊情況處理

```python
def predict_uploaded_image(model, image) -> str:
    logger.debug(f"開始預測圖片: {image}")
    try:
        image = Image.open(image).convert("RGB")
        input_tensor = data_transform(image).unsqueeze(0).to(device)

        with torch.no_grad():
            output = model(input_tensor)
            probabilities = torch.nn.functional.softmax(output[0], dim=0)
            top1_prob, top1_class = torch.max(probabilities, 0)

        # 判斷是否低於 60%，若低於則顯示 "MIX"
        probability = top1_prob.item() * 100
        if probability < 75:
            logger.info(f"預測結果: 米克斯貓 (混種貓), 最高機率: {probability:.2f}%")
            return "MIX"
        else:
            predicted_class = class_names[top1_class]
            logger.info(f"預測類別: {predicted_class} (機率: {probability:.2f}%)")
            return predicted_class
    except Exception as e:
        logger.error(f"預測過程出錯: {str(e)}")
        raise
```

2. 前端資料處理

品種名稱

```
// 輔助函數：將API返回的品種名稱映射到數據庫ID
const mapBreedNameToId = (breedName: string): string => {
  // 建立一个品種名稱到ID的映射
  const nameToIdMap: Record<string, string> = {
    Abyssinian: "abyssinian",
    Bengal: "bengal",
    Birman: "birman",
    Bombay: "bombay",
    British_Shorthair: "british-shorthair",
    Egyptian_Mau: "egyptian-mau",
    Maine_Coon: "maine-coon",
    Persian: "persian",
    Ragdoll: "ragdoll",
    Russian_Blue: "russian-blue",
    Siamese: "siamese",
    Sphynx: "sphynx",
    MIX: "british-shorthair", // 如果是混种猫，默认使用英国短毛猫作为展示
  };

  return nameToIdMap[breedName] || "british-shorthair";
};
```

結果展示

https://youtu.be/HuRR89-KPfM

3. 增值資訊整合

除了基本的品種辨識結果，系統還提供：

品種特徵資訊

- 外觀特徵描述
- 性格特點
- 體型大小資訊

飼養建議

- 日常照護要點
- 運動需求
- 飲食建議
- 健康注意事項

推薦產品

- 適合該品種的貓糧

- 專用護理用品
- 玩具推薦
- 健康保健品

資料快取與優化
- 前端結果快取，避免重複 API 呼叫
- 圖片預載技術，提升使用者體驗
- 非同步資料載入，減少頁面載入時間

陸、 系統實作成果

1. 功能特色展示

首頁介面
- 響應式設計，支援各種螢幕尺寸
- 拖拽上傳圖片功能
- 即時預覽上傳的圖片

品種辨識功能
- 支援 12 種純種貓咪辨識
- 混種貓判斷
- 辨識過程載入動畫

詳細資訊展示
- 品種特徵說明
- 健康問題提醒
- 飼養建議指南
- 推薦產品資訊

互動功能
- 熱門品種隨機展示
- 產品詳情對話框
- 主題切換 (明暗模式)

2. 技術

前端技術
- **Vue 3 Composition API**: 提供更好的程式和重用性

- **Vuetify 3**:提供一致的 UI 體驗
- **響應式設計**: 支援桌面、平板、手機多種裝置
- **TypeScript**: 提供型別安全和更好的開發體驗

後端技術

- **FastAPI**: 高效能的 Python Web 框架
- **異步處理**: 支援高並發圖片處理請求
- **日誌記錄**: 完整的操作記錄和錯誤追蹤
- **錯誤處理**: 完善的例外處理機制

AI 模型技術

- **遷移學習**: 基於 ImageNet 預訓練權重
- **資料增強**: 提升模型泛化能力
- **交叉驗證**: 確保模型穩定性

柒、 總結

這個專題從一開始的想法到真的做出來，讓我們看到 AI 技術可以怎麼用在寵物照顧這方面。用深度學習、網頁開發和設計使用者介面，做出一個能幫助識別貓咪品種的系統。雖然目前還有很多地方可以進步，但基本的架構和未來擴充的空間都已經做好了。

接下來，可以繼續讓模型更準確，增加一些有趣的新功能，甚至可以想看看怎麼把這個系統變成商業產品。做這個專題讓我們不只學到怎麼開發 AI 模型，也知道了要怎麼把技術變成真的有用的東西。我相信這些經驗未來一定會對我的學習和工作有很大的幫助！

參考資料: Oxford-IIIT Pet Dataset，ResNet 論文，Vue.js 官方文檔，github，國際貓協會