Multimedia Content Analysis

多媒體內容分析 Homework 5

數據一 RE6121011 徐仁瓏

壹、 介紹

在本報告中，我將介紹我所建立的卷積神經網絡（CNN）模型，以及該模型在狗、貓和熊貓的圖像分類任務上的應用。通過使用 CNN，我致力於解決這個具有挑戰性的問題，並獲得了令人滿意的結果。為了更好地了解模型的訓練過程和性能表現，我使用了視覺化技術來觀察模型在損失函數和準確度上隨著訓練次數的變化。這些視覺化結果對於評估模型的訓練效果和優化訓練策略起著至關重要的作用。在接下來的報告中，我將更詳細地介紹我的實驗流程和實驗結果。

貳、 實驗流程與程式碼解釋

在本篇中，我將詳細介紹整個實驗流程，並結合 hw5.ipynb 程式檔中的程式碼順序進行解說。

一、 載入套件

首先，我導入了所需的所有套件，包括 numpy、matplotlib.pyplot、seaborn 等常用的資料處理和視覺化工具，以及 PyTorch 相關的套件。同時，我也設定了隨機種子（SEED）為 0，這樣可以確保實驗的可重現性，即每次執行此程式檔都能得到相同的結果。

透過導入這些套件，為後續的實驗做好了準備，能夠方便地進行資料處理、建立模型、訓練和評估模型性能等工作。這樣的設計能夠更有效地進行實驗，並確保結果的準確性和可靠性。

二、 準備資料

（一）匯入資料

為了準備資料集，首先需要創建兩個列表（paths、labels）來存取圖像資料的路徑和標籤。這段程式碼會遍歷 animals 資料集中的每個類別文件夾，將每個圖像檔案的路徑和相應的標籤收集到兩個列表中。最後，我使用 Counter 函數計算每個類別中的樣本數量並輸出。計算結果顯示，每個類別中皆包含了 1000 張圖像。

（二）切割資料

準備好了各個類別的路徑和標籤後，則開始將資料隨機分割成訓練集、驗證集和測試集，其中測試集佔全部資料的 20%，訓練集中的 75% 用於實際的訓練，其餘 25% 用於驗證模型。最後切割結果為訓練集包含了 1800 張圖像路徑，驗證集包含了 600 張圖像路徑，而測試集也包含了 600 張圖像路徑。這樣的分割可以確保在訓練模型時能夠及時檢視模型在未見過的數據上的表現，從而更好地評估模型的泛化能力。

（三）自定義資料集類別

為了準備資料以供 PyTorch 的 DataLoader 使用，我們定義了一個自定義的資料集類別 AnimalDataset。這個資料集類別允許我們輕鬆地設定資料增強函式、讀取圖像檔案、處理標籤等操作。它為後續的模型訓練提供了方便且結構化的資料準備方式。

（四）自定義資料增強

這段程式碼定義了兩組不同程度的資料增強（augmentation）操作，分別為基本的資料增強和更複雜的資料增強。在深度學習中，資料增強是一種常用的技術，旨在擴增訓練資料集以改善模型的泛化能力。

- **基本的資料增強（basic_augmentations）**：包括將圖像調整為 224×224 大小的正方形、從圖像中心裁剪出 224×224 大小的區域，並將圖像轉

換為 PyTorch 張量。

- **更複雜的資料增強（new_augmentations）**：包括隨機裁剪並重新調整圖像大小為 224×224、隨機水平翻轉圖像（概率為 1）、隨機垂直翻轉圖像（概率為 0.1）、隨機透視變換（失真比例為 0.2，概率為 0.2）、將圖像隨機旋轉 -15 到 15 度之間的角度，最後將圖像轉換為 PyTorch 張量並進行標準化。

（五）資料載入

在這一部分，我們將創建訓練、驗證和測試資料集，並使用上一小節兩種不同的資料增強技術來擴增資料集的多樣性。我們透過 DataLoader 將資料打包成每 100 張圖像一包，即每次訓練、驗證或測試資料集時，一次都只丟入 100 張圖像，這樣能更有效地訓練模型，提高訓練的效率，也不會佔用到過多的內存。這樣的資料載入設計能夠確保我們能夠高效地訓練模型，並充分利用資料增強來提高模型的泛化能力。

三、 建立模型

在這一部分中，我們將建立兩個不同架構的卷積神經網絡模型：原始模型 SimpleCNN 和改進模型 DeeperCNN。

（一）原始模型 SimpleCNN

SimpleCNN 模型是我初始建立的一個相對較簡單的卷積神經網絡模型，具有較少的層數和參數量。其主要結構包括：

- 卷積層（Convolution Layer）：包含三個卷積層，每個卷積層後面跟著 ReLU 激活函數。這些卷積層負責提取輸入圖像的特徵。
- 最大池化層（Max Pooling Layer）：在每個卷積層後面都有一個最大池化層，將特徵圖的尺寸進行下采樣，同時保留重要的特徵。
- 全連接層（Fully Connected Layer）：包含兩個全連接層，用於將卷積層提取的特徵進行分類。最後一層全連接層的輸出維度等於類別的數量。

（二）改進模型 DeeperCNN

DeeperCNN 模型是一個更深、更複雜的卷積神經網絡模型，具有更多的層數和參數量。其主要結構包括：

- 卷積層（Convolution Layer）：包含五個卷積層，每個卷積層後面跟著 ReLU 激活函數。這些卷積層逐漸提取出輸入圖像的更高級別的特徵。
- 最大池化層（Max Pooling Layer）：在每個卷積層後面都有一個最大池化層，將特徵圖的尺寸進行下采樣，同時保留重要的特徵。
- 全連接層（Fully Connected Layer）：包含兩個全連接層，用於將卷積層提取的特徵進行分類。其中第一個全連接層後面加入了 dropout，以防止過擬合。

DeeperCNN 模型相比於 SimpleCNN 模型具有更好的表現結果。這是因為 DeeperCNN 模型擁有更多的層和參數，以及加上 dropout 的關係，能夠更好地擬合較複雜的數據模式。在相同的數據集上，我期待 DeeperCNN 模型能夠提取更多的特徵並學習到更多的抽象表示，從而提高了模型的性能。

四、 訓練模型

在這一部分，我將定義訓練模型所需的函數，包括訓練過程中的損失函數、準確率的追蹤和模型參數的保存。首先我先定義了兩個繪圖函式，分別為 plot_losses 和 plot_accuracies，前者用於繪製訓練和驗證損失的曲線，後者用於繪製訓練和驗證準確率的曲線。而 save_best_model 則用於保存表現最佳的模型參數。

定義完了一些需要使用到的函式後，即可開始定義訓練模型的主函式 train_model，在訓練過程中，它會迭代多個 epochs，計算訓練和驗證的損失和準確率，並保存表現最佳的模型。訓練流程包括先初始化空列表，用於儲存訓練和驗證的損失和準確率，接著開始迭代 epochs，在每次迭代中都做四件事情，包括訓練階段、驗證階段、調整學習率、保存模型。訓練階段中，對訓練集進行迭代，計算訓練損失和準確率，並執行反向傳播和優化參數。驗證階段中，對驗證集進行迭代，計算驗證損失和準確率。隨後，根據設定的學習率調整策略調整優化器的學習率。最後，如果驗證準確率提高，則保存當前模型為表現最佳的模型。在

迭代完所有 epochs 之後，即繪製訓練和驗證的損失曲線和準確率曲線，並打印最佳驗證準確率和保存的最佳模型的路徑。

在將所有函式定義完之後，則可開始套入模型以其各式超參數進行訓練，對於各種模型的實驗設定，將在實驗結果中有更詳細的說明。

五、 評估模型

在這一部分，我將使用最佳模型對測試集進行評估，同時繪製混淆矩陣。自定義函式中 test_model 的評估流程包括初始化變量以跟蹤損失、正確預測數量、總樣本數、預測標籤和真實標籤，接著在每次拿到的 100 筆測試資料集中，計算分類正確的數量和損失值，看完所有的測試資料後，即可將這些計算整個測試集的測試準確率和平均測試損失，隨後繪製混淆矩陣，用於評估模型的分類效果。透過以上流程，我們可以全面評估模型在測試集上的性能表現，並確認模型是否能夠準確地對不同類別的圖像進行分類。

參、 實驗結果

介紹完上述所有實驗流程和程式碼解釋後，在本篇中將快速歷經上述流程，並呈現最終的實驗結果。

首先，匯入資料後，將資料集切分成 1800 張圖像的訓練集、600 張圖像的驗證集以及 600 張圖像的測試集。切割完資料後，先對資料進行簡單的資料轉換，即透過資料增強中的 basic_augmentations 將圖像轉換成 224×224 大小的圖像，隨後即丟入 SimpleCNN 模型中進行訓練，損失函數使用 Cross Entropy Loss，優化器使用 Adam，其超參數皆使用預設值，學習率設置為 0.001，每 10 個 epochs 學習率則乘上 0.5，迭代次數設置為 50 次。下圖為訓練過程中的損失值和準確度隨 epoch 次數的變化圖。

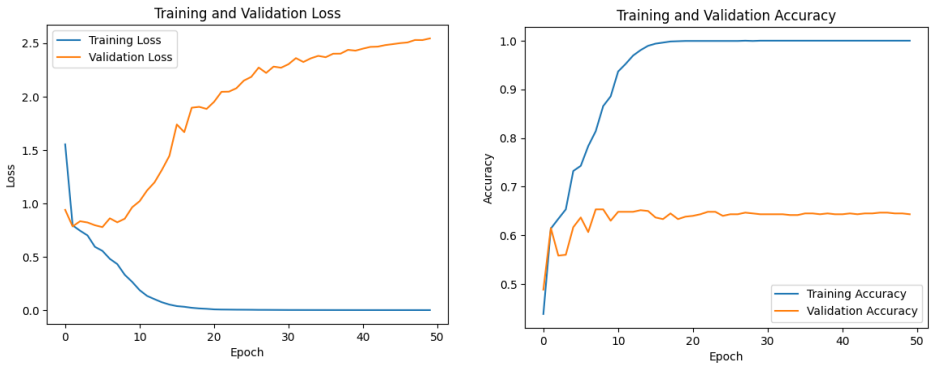

圖 1 SimpleCNN 模型在 basic_augmentations 中的損失和準確率隨 epoch 數量變化的曲線圖

從上圖可知，驗證集的損失值在迭代不到 10 次時就開始往上升，說明該停止訓練。從驗證集的準確率中也能證實這件事，差不多在迭代次數為 10 時，驗證準確率已趨於平緩，沒有繼續往上升。最終在驗證集得到最好的成績為 65.33% 的準確率。

由於對這樣的結果還不滿意，因此希望透過改進模型的方式來提高準確率，因此設置了一個更加複雜的模型 DeeperCNN，即增加了兩層卷積層來增加網路的深度，並在第一個全連接層中加入了 dropout，來提升模型在驗證集的表現。其他所有設置皆和上述一樣，下圖為訓練過程中的損失值和準確度隨 epoch 次數的變化圖。

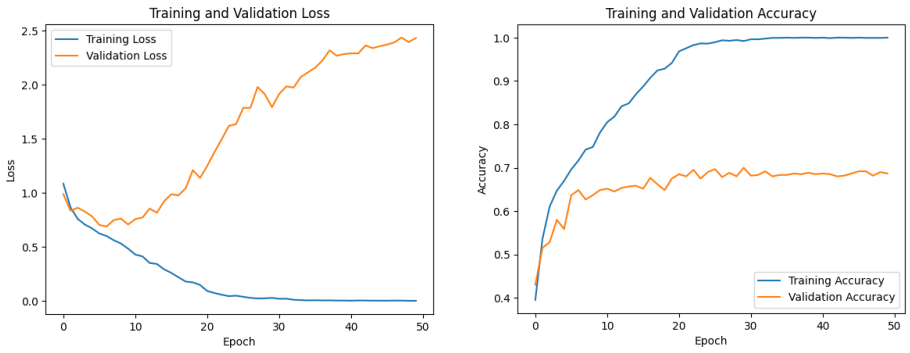

圖 2 DeeperCNN 模型在 basic_augmentations 中的損失和準確率隨 epoch 數量變化的曲線圖

從上圖可知，驗證集的損失值一樣也在迭代差不多 10 次時就開始往上升，和剛剛使用 SimpleCNN 的結果類似，驗證集的準確率也差不多趨於平緩，但若從數值來看，可以發現 DeeperCNN 最終在驗證集上得到最好的成績為 70.00%的準確率。

最後，不只改善模型，還希望可以透過資料增強的技術來讓準確率提升，因此將原本 basic_augmentations 改成使用 new_augmentations，最終在 DeeperCNN 模型上實現，其他所有的實驗設置一樣保持不變，下圖為訓練過程中的損失值和準確度隨 epoch 次數的變化圖。

圖 3 DeeperCNN 模型在 new_augmentations 中的損失和準確率隨 epoch 數量變化的曲線圖

從上圖可知，驗證集的損失在訓練了 50 個 epoch 後都還沒有上升的趨勢，驗證集的準確率也一直隨著 epoch 數量增強而提升，說明如果我們繼續訓練此模型，說不定還可以得到更高的準確率，目前在 50 個 epoch 下，在驗證集得到最好的成績為 70.83%的準確率。

因此，最終我們選擇使用 new_augmentations 資料增強技術的 DeeperCNN 模型來當作最終模型，拿到測試集資料去作預測，最終在測試集的準確率來到 72.17%，下圖為最終模型在測試結果的混淆矩陣圖。

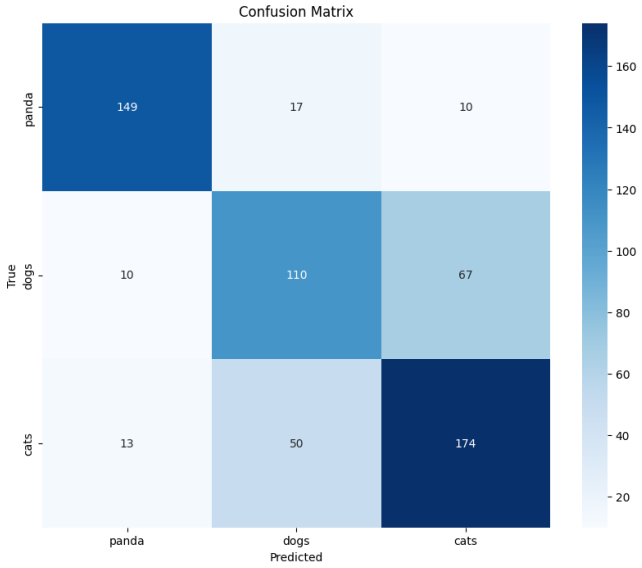

圖 4 使用 new_augmentations 資料增強的 DeeperCNN 模型
在測試集中的混淆矩陣圖

從上圖可知，X 軸為預測結果，Y 軸為真實結果，我們可以觀察到模型在分類小狗和貓咪時，比較容易出現預測錯誤的情形，這和我們的理解雷同，可能因為熊貓相較於小狗和貓咪區別較大，而小狗和貓咪可能由於外型較相似，因此較難分辨正確。

肆、 結論與展望

本次實驗中，我們通過比較不同模型架構和資料增強技術的效果，深入探討了如何提高深度學習模型在圖像分類任務中的性能。從結果來看，增加模型的深度和複雜度能夠顯著提高模型的性能，使其能夠更好地擬合訓練數據並提高泛化能力。同時，資料增強技術的應用也為提高模型的準確率帶來了明顯的幫助。這些結果表明，在圖像分類任務中，模型架構和資料增強技術的選擇對最終性能至關重要。

然而，雖然我們的實驗取得了一定的成果，但仍存在一些展望和改進的空間。首先，對於模型的進一步改進，從使用 new_augmentations 資料增強技術的 DeeperCNN 模型的訓練圖形中，可以看到模型似乎尚未收斂，我們可以考慮增

加訓練的 epoch 數量，以提高模型的性能，達到更高的準確率。另一方面，在模型訓練過程中，還可以嘗試不同的優化器、損失函數和學習率調整策略，以找到最適合該任務的訓練配置。通過這樣的調試和嘗試，可以更好地理解模型的行為，並找到提高性能的有效方法。