EE P 596 Programming Assignment 5
Report

Blake Downey

3/7/2022

Introduction

In a previous homework assignment working with the same flower dataset we were tasked with using classical methods of computer vision to determine the correct class of an image. In an exploratory comparison, we utilized the same dataset here, but used deep learning neural networks (DNNs) to accomplish the classification task. Using PyTorch, Torchvision and Google Colab, I implemented multiple pretrained models with various amounts of transfer learning. In the results section of this report I used the evaluation metrics from scikit-learn: accuracy score, confusion matrix and classification report.

Code Implementation

The Training/Validation Data

The training data was labeled by class and was loaded using the torchvision method ImageFolder. This uses the structure of the directories and assigns labels to images based on their folder location. The training data was then split into train and validation data using train test split function. A data loader is then used to load the train/val split images into a dictionary with key values ‘train’ and ‘val’.

The Models

For this deep learning assignment I used pretrained architectures from torchvision, namely: vgg19(with batch norm), resnet50, resnet101, resnet152 and inception v3.
All of these models except for inception v3 use the input shape of 220x220, which uses 299x299. Because of this, a second transform is defined called inception_transform.

Training

The training is implemented into two phases, training and validation. The results of the validation after each epoch of training is used to determine what the best model is from the list of training epochs. Originally I was performing early stopping to avoid overfitting, however higher validation and kaggle test scores were achieved when the model became overfit to the training data. Because of this, I trained most of my models for either 20 or 25 epochs. After each model was trained, the model weights were saved in order to be loaded the next day and not have to train again.

Evaluation

To evaluate my models I defined an evaluate method that uses the validation data loader to feed the images through the passed model. Using the labels and the predicted values I calculated the accuracy by class and total accuracy. I then convert the predictions and labels to numpy lists and return them. Using the numpy lists I utilize scikit-learns functions confusion matrix, accuracy score and classification report to analyze my results.

The Testing Data

The testing data was unlabeled and needed to be loaded differently. To accomplish this I built a class called MyImageDataset which is a child of the class Dataset. I use this class to create a data loader for my test images.

Test Data Predictions to Kaggle

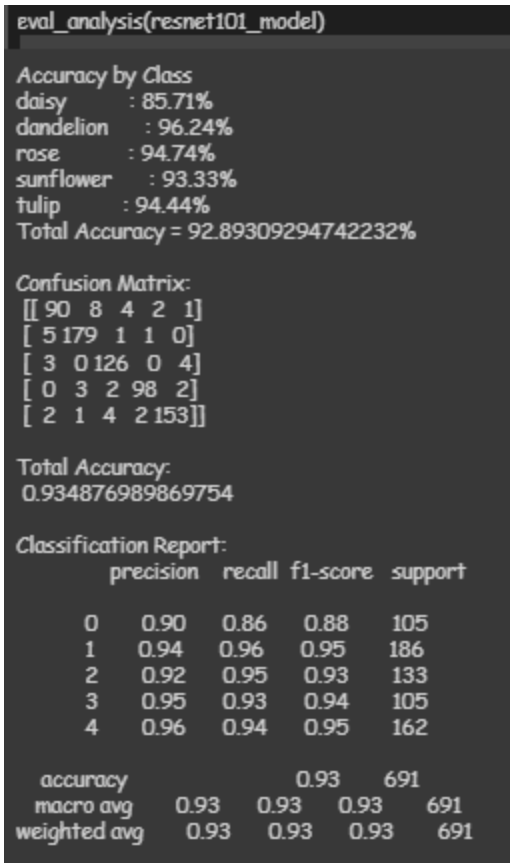
For generating my predictions for the Kaggle competition with the test data provided I wrote the test_images_predictions function that took the model as an argument. Here I perform the forward pass of the images in the test data loader through the network provided. Converting the predictions to numpy lists and using the image ids corresponding to each image I generate a dictionary with keys 'Id' and 'Category' which is converted to a dataframe and saved.

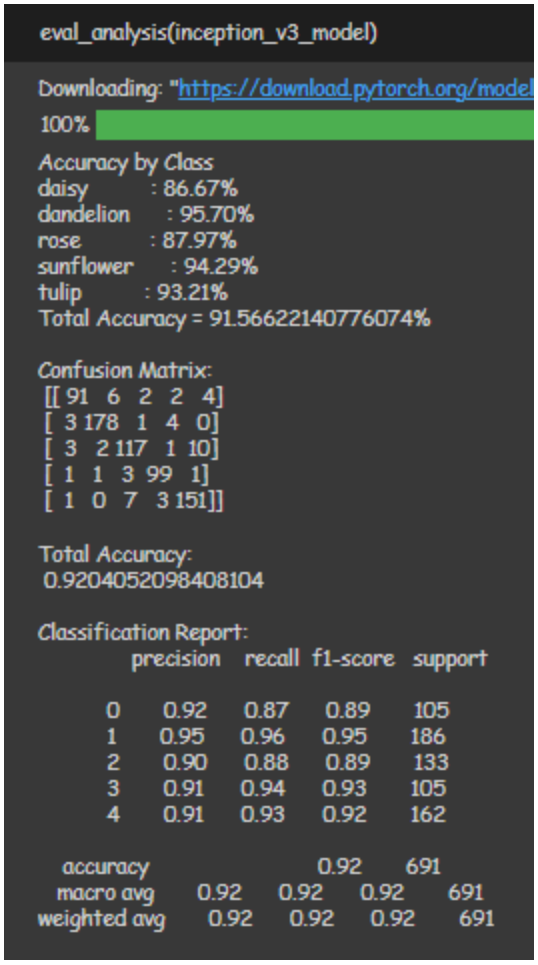
Results

```
eval_analysis(vgg19_bn)

Accuracy by Class
daisy        : 83.81%
dandelion    : 83.87%
rose         : 81.95%
sunflower    : 81.90%
tulip        : 90.12%
Total Accuracy = 84.33271949287794%

Confusion Matrix:
 [[ 88   3   5   3   6]
 [ 14 156   2   9   5]
 [  3   0 109   0  21]
 [  4   4   5  86   6]
 [  2   2   9   3 146]]

Total Accuracy:
 0.8465991316931982

Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.84      0.81       105
           1       0.95      0.84      0.89       186
           2       0.84      0.82      0.83       133
           3       0.85      0.82      0.83       105
           4       0.79      0.90      0.84       162

    accuracy                           0.85       691
   macro avg       0.84      0.84      0.84       691
weighted avg       0.85      0.85      0.85       691
```

```
eval_analysis(resnet50_model)

Accuracy by Class
daisy        : 86.67%
dandelion    : 93.55%
rose         : 89.47%
sunflower    : 94.29%
tulip        : 97.53%
Total Accuracy = 92.30106329144246%

Confusion Matrix:
 [[ 91   7   3   0   4]
 [  3 174   3   2   4]
 [  0   2 119   1  11]
 [  0   1   3  99   2]
 [  1   0   3   0 158]]

Total Accuracy:
 0.9276410998552822

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.87      0.91       105
           1       0.95      0.94      0.94       186
           2       0.91      0.89      0.90       133
           3       0.97      0.94      0.96       105
           4       0.88      0.98      0.93       162

    accuracy                           0.93       691
   macro avg       0.93      0.92      0.93       691
weighted avg       0.93      0.93      0.93       691
```

Evaluation using validation set on vgg19_bn and resnet50 models





Evaluation using validation set on resnet101 and inception v3 models

Model	VGG19_BN	ResNet50	ResNet101	Inception V3
Accuracy Score	84.66%	92.76%	93.49%	92.04%

Table 1: Accuracy vs Model

In the 4 images shown above I am displaying the accuracy by class, *total accuracy score metric from sklearn, confusion matrix and classification report*. In the table above I am showing the accuracy by model in a simpler view with the best model highlighted in green. **The best performing model in terms of the sklearn metric,accuracy score, was ResNet101 with an accuracy of 93.49%**. To achieve these results I froze the pretrained models weights for all layers except layer 4 and the fully connected layer and performed transfer learning through the train function.

A simple analysis on which flower categories are easily misclassified
After reporting my results for the confusion matrices of each model on the validation set we can start to see some patterns in misclassification.

Across all of the models' confusion matrices and accuracy by class we can see that the daisy class consistently has the lowest accuracy and a majority of the time the class with the highest number of false predictions for daisy is dandelion.

VGG is the least accurate model trained and being so, there are a lot of misclassifications between classes, the biggest ones here though are classifying a dandelion as a daisy, a rose as a tulip and vice versa a tulip as a rose. For ResNet50 and ResNet101 we can see misclassification of a daisy as a dandelion, and a rose as a tulip. Inception v3 also follows this trend but misclassified tulips as rose as well.

Overall, the biggest contributors to lower accuracy came from misclassifying a daisy as a dandelion and a rose as a tulip!