Adryann Guy

Raphaël Blauwart

Dorset College / ESILV

Data Science and AI – Final Project

Computer Vision Project:

Option 3: Classification of flowers.

Team Members:

Names	GitHub links	Contribution
Adryann Guy	https://github.com/Adryann35/Data-Science-and-AI.git	50%
Raphaël Blauwart	https://github.com/raphblau/GitConcepts	50%

Choice of architecture (CNN):

The choice of the Convolutional Neural Network (CNN) architecture for image classification, such as classifying flower images, is based on several key factors:

1. Hierarchical Feature Learning: CNNs are designed to automatically learn hierarchical features from data. In the context of images, this means they can identify simple patterns like edges in early layers and progressively learn more complex features like textures and object parts in deeper layers. This hierarchical feature learning is crucial for recognizing intricate patterns in flower images.

2. Spatial Hierarchies and Local Patterns: Flowers often have distinctive local patterns and structures. The convolutional layers in CNNs are particularly effective at capturing spatial hierarchies, making them well-suited for identifying these patterns. The local receptive fields of convolutional filters enable the network to focus on small, relevant regions of the input, enhancing its ability to recognize intricate details in images.

3. Translation Invariance: CNNs leverage shared weights and pooling layers, providing translation-invariant features. This is essential for image recognition tasks where the position of the features within the image might vary. For flower classification, this property ensures that the network can recognize and classify flowers regardless of their location in the image.

4. Parameter Sharing and Efficiency: CNNs use parameter sharing, where the same set of weights is applied to different parts of the input. This reduces the number of parameters compared to fully connected networks, making CNNs computationally more efficient and preventing overfitting, especially when dealing with limited datasets.

5. Pre-trained Models and Transfer Learning: CNN architectures often come with pre-trained models on large image datasets (e.g., ImageNet). Leveraging these pre-trained models through transfer learning allows practitioners to benefit from the knowledge gained by the network on generic image features. Fine-tuning a pre-trained CNN for specific tasks, like classifying flowers, can lead to better performance, especially when data is limited.

6. Community Support and Resources: CNNs have been widely adopted in the computer vision community, resulting in a wealth of resources, libraries (e.g., TensorFlow, PyTorch), and pre-trained models. This abundance of support simplifies the implementation and training process for practitioners working on image classification tasks, including flower classification.

In summary, the architecture of CNNs is well-suited for image classification tasks due to their ability to learn hierarchical features, capture spatial hierarchies, provide translation invariance, and efficiently handle large datasets. The combination of these properties makes CNNs a natural choice for tasks like classifying flower images in Python.

Methodology of training and testing, splitting the dataset:

First the dataset is organized into subdirectories for each flower type:

```
path = pathlib.Path("Downloads/FlowersDataScience/flower_images")


lilly = list(path.glob('Lilly/*'))[:1000]
lotus = list(path.glob('Lotus/*'))[:1000]
orchid = list(path.glob('Orchid/*'))[:1000]
sunflower = list(path.glob('Sunflower/*'))[:1000]
tulip = list(path.glob('Tulip/*'))[:1000]
data = {
    'lilly' : lilly,
    'lotus' : lotus,
    'orchid' : orchid,
    'sunflower' : sunflower,
    'tulip': tulip
}
```

Then the code creates data generators for training and testing data:

```
train_gen=ImageDataGenerator(
                              rescale = 1./255,

                              validation_split=0.2  # set validation split to 20%
                              )
train_data=train_gen.flow_from_directory("C:/Users/adrya/Downloads/archive (1)/flower_images",target_size=(224,224),batch_size=32
                                         class_mode='categorical',
                                         shuffle=True,
                                         subset='training')
test_data=train_gen.flow_from_directory("C:/Users/adrya/Downloads/archive (1)/flower_images",target_size=(224,224),batch_size=1,
                                        shuffle=False,subset='validation')
```

The validation data (test data) is splitted to 20%.

Results of training and testing:

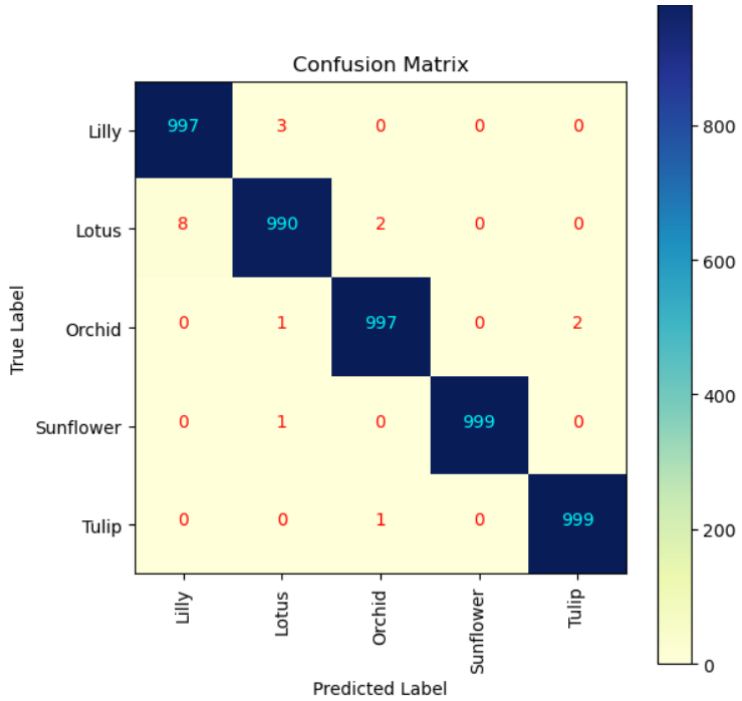
Classification report:

```
Classification Report:
              precision    recall  f1-score   support

       Lilly       0.99      1.00      0.99      1000
       Lotus       0.99      0.99      0.99      1000
      Orchid       1.00      1.00      1.00      1000
   Sunflower       1.00      1.00      1.00      1000
       Tulip       1.00      1.00      1.00      1000

    accuracy                           1.00      5000
   macro avg       1.00      1.00      1.00      5000
weighted avg       1.00      1.00      1.00      5000
```

Confusion matrix:



We can see that the trained model has a very good precision of 0.99 and almost 1. We can say that this architecture is very accurate for images classification.