University of Calgary

ENEL 525 Fall 2023 – Final Project

Harrison Mondragon

30088805

December 17, 2023

**Introduction**

**Introduction Summary**

The project involves implementing deep learning techniques, specifically convolutional neural networks (CNNs), for advanced image classification with a focus on identifying and recognizing various flower species. The primary goal is to leverage CNNs to process and learn from high-resolution floral image data, exploring the intricacies of feature extraction. The project aims to achieve high accuracy in distinguishing between different floral categories, emphasizing the application of deep learning for robust and precise classification of diverse flower species based on their visual characteristics.

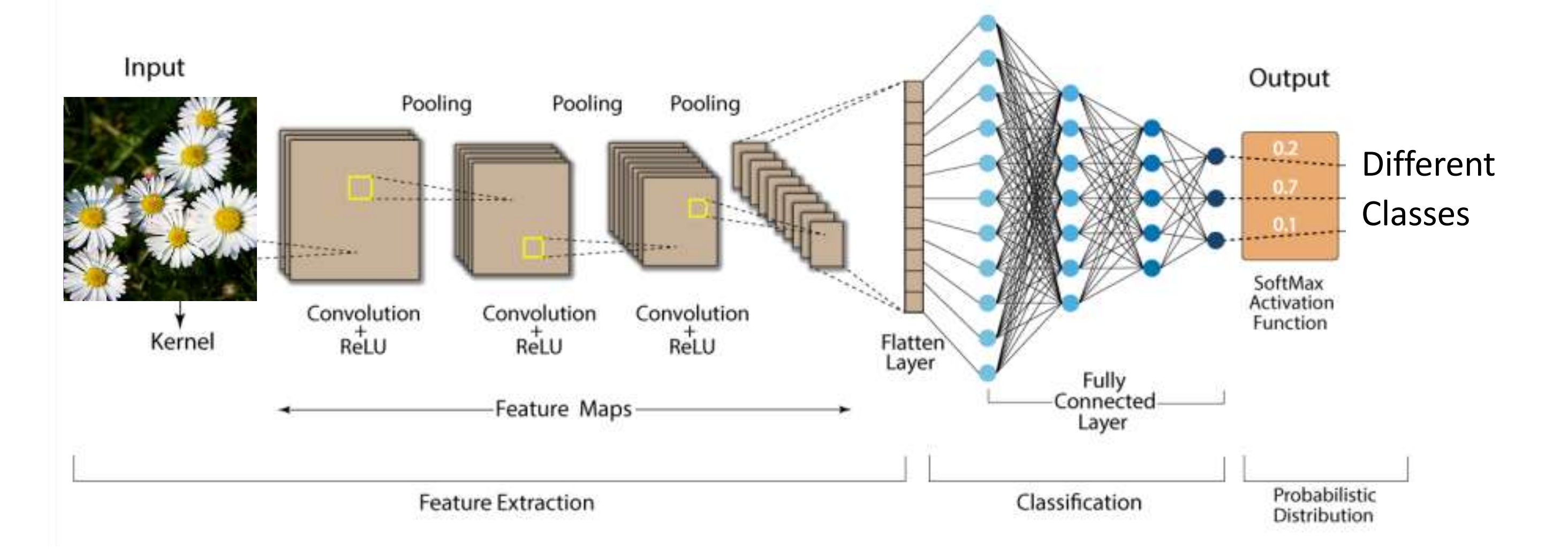
**Methodology**

**Problem Set-up**

I use Python and Tesorflow to achieve the goals of this project. First, I load in the dataset using the Python os module and some looping logic to collect all the image paths and their associated classes. The dataset is organized as it was on the Kaggle link, where the dataset directory contains subdirectories with the class names, containing the associated images. My code depends on this organization to import the data correctly. Next, I shuffle the data so that the training, validation, and testing sets contain randomized data from all classes, and split the dataset into the sets based off the 0.70/0.15/0.15 split outlined in the project description. Then, I utilize the ImageDataGenerator API from Tensorflow Keras to preprocess the images by normalizing and resizing them. I do this separately for training, validation, and testing so that I can manage them separately later. Next, I create and compile the CNN model using Tensorflow Keras. At this point, the data and model are ready to be used. I train the models using the train and validation sets, then evaluate the model on the test set, and output relevant plots and statistics. As I was developing my code, I used reduced versions of the dataset to incrementally test changes. I mainly developed using a dataset with 100 images per class and 5 epochs so that the code would run quickly. The rest of this report will contain outputs of the full dataset with 10 epochs.

**Network Design**

I closely followed the network design in the final project slides to construct my model. The diagram for this network is included below. The only difference between the diagram and my model is that I only have 2 fully connected layer, whereas the diagram has 4.

****

Layer Design Choices (in order):

* Conv2D(32, (3, 3), activation='relu', input\_shape=input\_shape)
  + First convolution layer.
  + I provide the input shape that I resized to during preprocessing. It is 320x240 pixels with 3 color channels for RGB based off the project description.
  + I used 32 filters here to start off with a relatively small value, since this layer captures basic features like edges and simple textures. This helps to extract lower-level features and reduce computational load in the initial layers.
* MaxPooling2D((2, 2))
  + First pooling layer.
  + Reduces the spatial dimensions of the input by taking the maximum value in the pool size region.
* Conv2D(64, (3, 3), activation='relu')
  + Second convolution layer.
  + I used 64 filters here to gradually increase the number of filters. This helps the network learn more complex and abstract features based on the features already captured by the first layer.
* MaxPooling2D((2, 2))
  + Second pooling layer.
  + Reduces the spatial dimensions of the input by taking the maximum value in the pool size region.
* Conv2D(128, (3, 3), activation='relu')
  + Third convolution layer.
  + I used 128 filters here to gradually increase the number of filters even further. This helps the network learn more complex and abstract features based on the features already captured by the first and second layer.
* MaxPooling2D((2, 2))
  + Third pooling layer.
  + Reduces the spatial dimensions of the input by taking the maximum value in the pool size region.
* Flatten()
  + Flatten layer.
  + Transforms the output from the 3D tensor in the convolution and pooling layers into a 1D tensor vector that can be used by the densely connected layers.
* Dense(256, activation='relu')
  + First dense (fully connected) layer.
* Dropout(0.5)
* Dense(len(class\_names), activation='softmax')

General Design Choices:

* I used 3 convolution layers to match the diagram. This also gave me enough layers to gradually increase the number of filters so that I could capture a good number of features from the dataset images.
* I used 3 pooling layers because it needs to match the amount of convolutional layers to work together correctly.
* I used a 3x3 convolutional kernel in my convolutional layers because it is a common choice for CNNs. It has a relatively small receptive field, allowing the network to capture local features and details. …
* I used a 2x2 pool size in my pooling layers because it is a common choice for CNNs. It helps reduce spatial dimensions and computational load. I chose to go small here because pool sizes that are too large can result in loss of information.
* I used the Rectified Linear Unit (ReLU) activation function in my convolutional layers to introduce non-linearity to my model. This allows my network to lean complex patterns and relationships in the data.

**CNN Backpropagation Concept**

**Loss Function Derivation**

**Training Scheme**

**Testing Scheme**

**Results and Discussion**

**Training Curves and Relevant Parameters**

**Model Summary and Training Parameters**

**Final Results**

**Classification Results**

**Effectiveness of Network**

**Conclusion**

**Conclusion Summary**

**References**