Deep Learning Semester Project Report: Flower Classification CNN

By: Trevor Parsons

Data Curation

The data set that was used to train, validate, and test the model was a set of 4,323 images of flowers, composed of 762 images of daisies, 1055 images of dandelions, 784 images of roses, 734 images of sunflowers, and 984 images of tulips. The sizes of the images were variable, so a program was used to resize all of the images to 75 x 75 pixels for use in the model.

Input Data Visualization

Shown on the next page are random images from the data set with corresponding array labels. The labels are shown in order of [daisy, dandelion, rose, sunflower, tulip], with a value of 1 being a positive label.

Data Splitting

The dataset was presplit into folders containing images of only one flower. After resizing and randomizing, images were split into three folders, each with subfolders for specific flower types: Training, Validation, and Testing. The training folder received 70% of each type of image and the validation and testing folders received 15% of the images.

Visualization of images from the data set:

Training Different Model Architectures

The first model that was attempted consisted of a single convolutional layer with 32 filters, filter size of 3x3, and a sigmoid activation layer. After passing through the convolutional layers, input data from the images was flattened and then fed to a dense layer with five neurons. This model was highly very unsuccessful, and exhibited signs of overfitting the training data.

To improve on the first model, a second and third convolutional layer were added to the model. The second contained 64 filters, the third contained 128 and both had sigmoid activations and 3x3 filter size. Again, this model was very unsuccessful and struggled with overfitting.

Finally, a model using five convolutional layers and using max pooling layers between each convolutional layer was tested and achieved significantly better results. The base line accuracy for this data set is 20%, and the final model was able to achieve a validation accuracy of ~50%.