Christy Jacob
CS 6384.001

Computer Vision Project 1

For this computer vision project, we had to build a model using transfer learning to help classify flowers into 13 different classes. Transfer learning is when you use a pretrained model to create a new model for a related problem. Transfer learning is helpful because it helps reduce the time and resources that you spend on training a model for a related problem and improves the performance as well. A very popular and good option for transfer learning for image classification purposes is mobile net v2 which is a convolution neural network with 53 layers and you can get a pretrained version that is trained on more than a million images from the ImageNet database. The pretrained model that I used was called efficient net b0 which is also trained on the ImageNet database. Efficient net is a convolution neural network similar to mobile net but uses compound coefficient to improve performance. I used that as my first layer in my model. My second layer was a dropout layer with a 0.1 dropout rate. The dropout layer randomly sets inputs to 0 based on the dropout rate and scales up the other inputs so that the sum is the same. This dropout layer is meant to help prevent overfitting. Finally, I had a dense layer that had the shape (None, 13) since we had 13 classes.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer_1 (KerasLayer)  (None, 1280)              5919312

 dropout_1 (Dropout)         (None, 1280)              0

 dense_1 (Dense)             (None, 13)                16653

=================================================================
Total params: 5,935,965
Trainable params: 16,653
Non-trainable params: 5,919,312
```

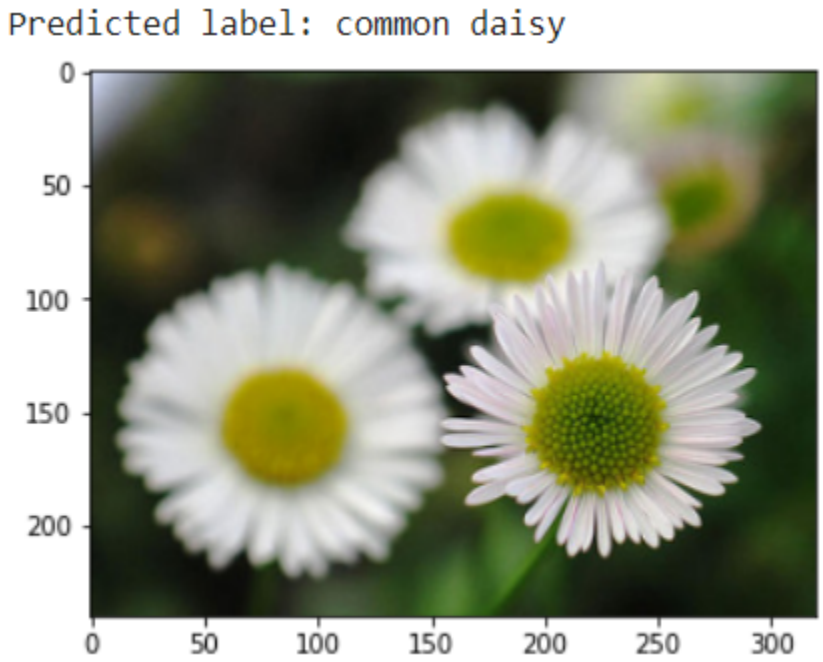
I used categorical cross entropy to measure loss and used the stochastic gradient descent optimizer with a learning rate of 0.001. I trained the model using 5 epochs. Here were my accuracy results.

```
Epoch 1/5
642/642 [==============================] - 62s 83ms/step - loss: 0.4923 - accuracy: 0.8599 - val_loss: 0.2405 - val_accuracy: 0.9363
Epoch 2/5
642/642 [==============================] - 52s 81ms/step - loss: 0.2174 - accuracy: 0.9438 - val_loss: 0.2044 - val_accuracy: 0.9465
Epoch 3/5
642/642 [==============================] - 52s 81ms/step - loss: 0.1809 - accuracy: 0.9509 - val_loss: 0.1930 - val_accuracy: 0.9477
Epoch 4/5
642/642 [==============================] - 51s 80ms/step - loss: 0.1598 - accuracy: 0.9554 - val_loss: 0.1820 - val_accuracy: 0.9504
Epoch 5/5
642/642 [==============================] - 52s 81ms/step - loss: 0.1456 - accuracy: 0.9630 - val_loss: 0.1790 - val_accuracy: 0.9523
```

I was able to get 96.3% accuracy and below 15% loss while correctly predicting all the flowers in the flowers_test dataset.

Christy Jacob
CS 6384.001

Prior to building the model, we had to read in images from the flowers dataset and preprocess them. We were provided a zip file to the flowers dataset with 13 classes. I used the inbuilt function from tensorflow to read in the flowers dataset. The function tensorflow.keras.preprocessing.image_dataset_from_directory reads in the images and reads the folders of the images as class labels. There were 13 possible class labels and images in each folder. The dataset consisted of a tuple of (images, labels). I resized the images to (224, 224) to match the input size of the efficient net b0 pretrained model I was using and there were 3 channels as well. I also batched the dataset and used a 80/20 split for training and validation with 10278 files for training and 2569 files for validation. The shape of the images was (16, 224, 224, 3) and the shape of the labels was (16, 13) since I used a batch size of 16 to train my model. The labels were one hot encoded representations of their class. Another preprocessing step I did was normalization which consisted of dividing all the values in the images array by 255. I used ds.map to map the images to their labels as well to perform the training and evaluation. After those preprocessing steps, I trained and evaluated the model. I also used model.predict to see how the model performed on the flowers_test dataset after importing that. After tuning the model, I was able to have it work on all 4 images in the flowers_test dataset. Here is an example.



I then saved my model as an h5 to use it in the test file. In the test file, the code was already provided to read in the flowers.csv so I got the image path from the first column and the labels from the second column. I then read in the images, resized them to (224, 224) and turned it into a numpy array. I also read in the labels as strings. I then used sklearn.preprocessing to manually perform one hot encoding on the classes so that I

could convert the test labels into the proper shape and input. One hot encoding takes a class of strings and turns them into arrays. So if your classes were classes = ['astilbe', 'bellflower', 'black-eyed susan', 'calendula', 'california poppy','carnation', 'common daisy', 'coreopsis', 'dandelion', 'iris', 'rose', 'sunflower', 'tulip'], then one hot encoding would turn each string into an array of size (1, 13) with a 1 indicating the index of the label and 0 everywhere else. It would automatically be in alphabetical order. So ‘astilbe’ would be [1,0,0,0,0,0,0,0,0,0,0,0,0] after one hot encoding since it is the first element. I used a for loop to map and replace the test labels with their one hot encoded representations. I also performed normalization on the images by dividing the values of the images array by 255. I used the tensorflow function tensorflow.data.Dataset.from_tensor_slices to turn the images and labels into a dataset and then mapped the images to their labels. I then evaluated my loaded model on the flowers_test dataset and here are my results below.

```
<MapDataset element_spec=(TensorSpec(shape=(None, 224, 224, 3), dtype=tf.float32, name=None), TensorSpec(shape=(None, 13), dtype=tf.float64, name=None))>
4/4 [==============================] - 1s 40ms/step - loss: 0.1118 - accuracy: 1.0000
Test model, accuracy: 100.00000%
```

I was able to get 100% accuracy on the four images in the flowers_test dataset. The project was very interesting and the most difficult part was tuning the model to try to get the best accuracy and finding a model that would properly predict the fourth image in the flowers_test dataset. That image gave trouble to many models I tried that even had higher than 95% accuracy. It was also difficult to figure out how to properly read in and structure the flowers_test dataset so that I could evaluate. Through this project I learned a lot about transfer learning and how to use it for image classification. I also learned a lot about many tensorflow datasets and neural networks.