Nicholas Carlin

Nicholas Bohm

Final Project Writeup

Introduction / Overview:
For our project we decided to do a vegetable image classification. Given 15 + images, our goal was for our neural network model to be able to predict the correct vegetable for a given image. We used two different datasets which we downloaded off of Kaggle. Both datasets came with a train, test, and validate folder. We used the file path of our downloaded data to change it into a format which we can use to input into a model. We ran a pre-trained model *and* our own model on dataset one, dataset two, and a combined version of both datasets. The pre-trained model we found was off of Kaggle which used the tf.keras.applications.MobileNetV2 function. Our neural network was a Sequential model with two Convolution layers, similar to one we did in class. Overall, our model and the pre-trained model were both able to predict our test data with a fairly high accuracy. We evaluated these results using a confusion matrix and a classification report which we will discuss in our results section.

Previous solutions:

While working on this project, we tried to take some inspiration from past projects. One of the main differences between how we created our project vs a previous one was how we created our training/validation/testing data. A previous solution used the tensorflow tf.keras.utils.image_dataset_from_directory function to create BatchDatasets. They then displayed images of the train dataset using matplotlib. Then this solution uses data augmentation to change the images and then uses matplotlib to display these augmented images. Next, a model is defined using Sequential() which includes the data augmentation, 2D Convolution layers, and features such as dropout. The model is compiled, a summary is printed, and then the model is fitted using the train_ds and validation_ds batch dataset. To evaluate the model, this solution gets the training and validation accuracy and prints them again using matplotlib. The issue with this solution is that it had no real evaluation methods. Thus, we decided to use a different approach.

Proposed Method :

We first plan on splitting up the data into three datasets with one of each and a combined dataset. For generating our dataset from these folders, we decided to use the generator.flow_from_dataframe function to get our validation, train, and test image datasets. In order to use this, we needed to get our train / test / validate data into a data

frame with the first column as the file path and the second column as the label. We will then create a sequential model with two 2D convolution layers with a batch size of 32. To test how our model holds up with different data, we will train it on three different datasets: the first dataset from Kaggle, the second dataset from Kaggle, and our own combined one. To test our results further, we will define a pre-trained model and will train the model on the same three datasets. After all 6 models are trained, we will print out Classification reports and a confusion matrix for each to evaluate our results. This is a general outline of our proposed method.

Dataset:

We imported two different datasets. We called the one we downloaded from https://www.kaggle.com/datasets/misrakahmed/vegetable-image-dataset which contains a train/test/validation folders with vegetable images as 'data'. This dataset contained 15 different types of vegetables. Our second dataset we downloaded from https://www.kaggle.com/datasets/kritikseth/fruit-and-vegetable-image-recognition which also contains a train/test/validation folders containing vegetable images. The combined dataset is in the final_data folder which contains the files of all of dataset one and two. If a vegetable is in dataset 1 and 2, one of those folders is created and images from both datasets are merged into it. To get our data into the right format to be trained with our model, we first have to create a train/test/validate data frame for each dataset 1, 2, and the combined one with column 1 as the file path and column 2 as the label. This way, we can map our input for the model which will be the given image to the label which is the filename. We use these data frames to create train_images, val_images, and test_images for all three different datasets.

Evaluation Method:

For our evaluation method we used a classification report and a confusion matrix. The classification report includes four key metrics: F1, recall, precision, and support. Our confusion matrix was a multi label matrix because we had more than two images. For a given image such as an apple, we can find how often an apple is predicted vs how often over images are chosen instead. To get these metrics of evaluation, we first used the already trained model to predict with our test data using 'model.predict(test_images)'. Next, we use the labels of the test_images to get a list of the models predictions. Now we get our Y_test by getting the labels from the test_images. We use our predictions and Y_test to get the classification report and confusion matrix.

Results:

Our model performed quite well but unfortunately not nearly as well as the pre-trained model. Our first test was using our model on the first, second, and combined dataset.

Our model performed with an accuracy of 83 percent on the combined dataset and an accuracy of 80 percent on the first and an accuracy of 81.57 percent on the second. We printed out a confusion matrix for each of these three model's predictions. The results seemed to be pretty scattered. The model predicted different vegetables better based on the different datasets. For example, bitter gourd was predicted with a 100 percent accuracy on the first dataset while it was predicted with only a 60 percent accuracy on the first. The pre-trained model we used was much more accurate. It was able to predict the accuracy of the combined dataset with an accuracy of 99.80 percent. The model performed equally well on dataset one and two, with an accuracy of 99.47 and 99.41 percent respectively.