CS63 Spring 2024
Final Project Checkpoint

Jonah Pacis and Mei Prasetio

April 21, 2024

1 Project Goal

Through this project, we are aiming to train a deep convolutional neural network on the TJ NMLO Public Dataset Fruits Classification dataset. The dataset is composed of a total of 22495 images of 100x100 pixels that contain one fruit or vegetable on a blank background. The training set contains 16,854 images (one fruit or vegetable per image) that ares subdivided into 33 classes.
The test set provided contains 5641 images which each contain 1 image of a fruit or vegetable on a blank background. We will add additional images of fruit and vegetables that we have taken ourselves or by editing the background of the current images, as these images will not have a blank background. Our primary goal is to develop a CNN that is capable of accurately classifying the fruits and vegetables. Additionally, we will assess the model's robustness in identifying fruits by creating novel examples where the fruit or vegetable image will contain a background. By testing on these images with a background, we will determine whether or not our model can generalize to different types of images of fruits after being trained on images that soley have blank white backgrounds.

2 AI Methods Used

2.1 Data

The TJ NMLO Public Dataset Fruits Classification dataset contains 22495 total images of 100x100 pixels that have one fruit or vegetable on a blank white background. The training set contains 16,854 images (one fruit or vegetable per image) that ares subdivided into 33 classes. The classes are as follows:

- Apple Braeburn
- Apple Granny Smith
- Apricot
- Avocado
- Banana
- Blueberry
- Cactus Fruit
- Cantaloupe
- Cherry
- Clementine
- Corn
- Cucumber Ripe

- Grape Blue
- Kiwi
- Lemon
- Limes
- Mango
- Onion White
- Orange
- Papaya
- Passion Fruit
- Peach
- Pear
- Pepper Green
- Pepper Red
- Pineapple
- Plum
- Pomegranate
- Potato Red
- Raspberry
- Strawberry
- Tomato
- Watermelon

We will create our own test images that contain a fruit with an altered background so the background is not just blank/white. We will do this by editing the color of the background and adding patterns or objects in the background such as stripes, tables, etc.

2.2 Convolutional Neural Network (CNN)

A convolutional Neural Network is a deep neural network that utilizes convolutional layers and pooling layers. Instead of these layers being fully connected to their previous layer, they utilize local connections. The convolutional layers act as localized feature detectors and the pooling layers will compress the data through downsampling from the convolutional layers. CNNs largely deal with image data, which has spatial resolution, and each convolutional layer filter acts as a small patch which slides across the image to calculate the outputs. This results in a set of feature maps for each filter. The end of the network CNNs usually contain two fully connected layers and the output layer, and like ANNs, CNNs, are fully trainable through backpropogation.

Model Architecture and Hyperparameters

- *Convolutional layers* will detect features through a number of filters and pixel sizes.
- *Pooling layers* will downsample the convolutional layers. Although we may experiment with the max pooling method and compare our results with the average pooling method.
- We will experimentally find the optimal *filter size*, *number of filters*, *number of convolutional layers*, *number of pooling layers*, and *number of fully connected layers* and the number of nodes in each fully connected layer.
- We will use the Nadam *optimizer* but may also test the SGD optimizer to see which performs better on our dataset. Since the Nadam optimizer adjusts the learning rate based on previous gradients and moments, we predict this optimizer aid in training a more effective CNN that can discriminate between the 33 classes of fruits.

- We will adjust the initial *learning rate* of our CNN, which will control how fast our model converges to a set of optimal set of trainable weights.

Techniques: We will attempt implement the following techniques in our model.

- *Batch normalization* aims to stabilize and speed up the training process. It serves as a method of normalization of the inputs of each layer.
- *Dropout layers* prevent overfitting where different dropout rates will randomly exclude some of the inputs.
- The *Activation function (RELU)* works well with CNNs since they allow for a quicker convergence. They work to mitigate the vanishing gradient problem.
- By adding a *Bottleneck layer*, we can force the network to learn more general patterns in the data and prevent overfitting by our model on our training data.

Additional Considerations

- *Data Augmentation* where the images will transformed by being flipped or rotated. This may help the model generalize on various pictures of the fruits that may be from different perspective or have different orientations.

3 Staged Development Plan

1. Successfully import and load in TJ NMLO Public Dataset Fruits Classification dataset from Kaggle and display the different 33 classes of fruits in our Jupyter notebook. We also need to import the libraries needed to generate a confusion matrix for our model.
2. Develop a keras model that is capable of reading in the images and producing an output. We will experimentally test the best initial hyperparameters (learning rate) and the model architecture (number of convolutional layers, filters in each layer, number of nodes in dense layers).
3. Produce graph which plots the loss, validation loss, accuracy, and validation accuracy of our model to determine that we are appropriately training the model and preventing overfitting on our training data. We are also aiming for high validation accuracy.
4. Generate feature maps to inspect the representation of how the model solves the problem. Using these feature maps to experiment with different architectures.
5. Generate a confusion matrix to get a more precise idea of what mistakes are being made.
6. An initial goal is be able to get our model to distinguish between very obviously different fruits, such as cactus fruits, kiwis, pomegranates, potatoes, and raspberries.
7. A secondary goal is to get our model to distinguish between very similarly shaped fruits such as green peppers and red peppers, lemons and limes, grapes and blueberries, braeburn and granny smith apples.
8. Stretch goal: Investigate the model's performance on images with varied backgrounds. If generalization is still lacking consider augmenting the image data.

4 Measure of Success

This project will measure success through evaluation of accuracy, generalization, and our confusion matrix. We aim to achieve high accuracy of correctly discriminating between all 33 classes of fruits and aim to have our model generalize images of fruits with varied backgrounds (i.e. not just blank white backgrounds). We aim to have a low misclassification rate in our confusion matrix and hope that our model makes few mistakes in discriminating between differently shaped fruits.

5 Plans for Analyzing Results

We plan to generate plots that show the training loss, validation loss, accuracy, and validation accuracy of our model over various epochs to determine whether or not our model is overfitting on the training data and determine how accurate our model is. We plan to achieve at least 92.5% accuracy on the test original test dataset. To further analyze our model, we will generate a confusion matrix to identify what kinds of mistakes are being made on our model. We will also generate a histogram that plots the misclassification rate of each of the 33 classes of fruits. We will generate the misclassification rate histograms, loss and accuracy plots, and the confusion matrix for our model before and after we add images with altered backgrounds to test for generalizability. If our model is good at generalizing, we not observe a significant effect on the model accuracy and loss or misclassification rate for each of the classes. If our model has poor generalizability, we predict that we will have a decrease in accuracy, an increase in loss, and an increase of misclassification based on the classes with a high number of test images that have altered backgrounds. We will also inspect our feature maps to determine if there is any human-interpretable features our model has learned to discriminate between fruits. If so, we can comment of why we think our model is performing well or poorly, which we will write up on our finalized model.