Tasfiq Ahmed and Varun Saini

Image Classification: Fruits 360 Dataset

Introduction

This project focuses on detecting various fruits and vegetables from a Kaggle image database using deep learning techniques. This is an object detection project (multiclass classification) that uses two types of neural network model architectures: a convolutional neural network and a fine-tuned model built on top of Inception v3 (transfer learning).

Previous Solutions

Previous solutions include CNN's trained from scratch, as well as several pre-trained networks, such as VGG16 and VGG19, Inception v3 (tested in this project), residual networks and its variants (resnet34, resnet50, resnet101), as well as the Xception network, which is an extension of the Inception architecture. All models are accessible from the keras.applications module.

Dataset

The dataset is linked [here](https://www.kaggle.com/moltean/fruits). This dataset is updated with new images and classes as soon as the authors have access to them. Included in the dataset is 90483 total images, each 100 pixels by 100 pixels, and the test split is about 25%. Images are one of 131 different fruits and vegetables. Class labels are not necessarily unique; multiple labels may exist for one kind of fruit. As part of the fruit extraction process, the image's background was filled with white due to variations in the lighting conditions. As such, a limitation of this model is that it is not able to generalize well to images with noisy backgrounds. The dataset is readily accessible with the commands included at the beginning of the notebook.

Proposed Method

Among machine learning techniques, artificial neural networks are considered to be the most successful in the area of image recognition as well as classification. Consequently, our initial proposed method was to construct a convolutional neural network, which is able to take into account the structure of the image while processing them. We began with a CNN with 3 convolutional layers, each with 32 filters and a kernel size of 3, with max pooling layers in between. After flattening, we then connected it to another dense layer with 128 neurons before connecting to the output layer.

After the success of the initially proposed model, we continued our analysis with a model trained on top of the Inception v3 model. The output from Inception v3 was connected to our model by a global average pooling layer, before being connected to a dense layer with 1024 neurons, and then an output layer with 131 output neurons.

As the image_dataset_from_directory function from keras.utils returns a BatchDataset, iterating over the dataset and manually appending true values while also appending the predicted class label is necessary for confusion metrics. This processing was done with both our custom model as well as the transfer learning model.

The fully trained CNN is available for download. Uncomment the code block at the beginning of the "Evaluating our Model" section. Alternatively, it is also saved in this repository in the "saved_model" folder. As it is not practical to save the transfer learning model, the user is required to train the model to evaluate its performance.

Evaluation Method

We evaluated our CNN and our model trained on top of Inception v3 by first constructing confusion matrices. Afterwards, we evaluated both models on their accuracy, precision, recall, and f1 scores. This evaluation process was conducted using only test data, which was not used for training either model.

Results and Discussion

After evaluating both models, we found that our custom CNN achieved an accuracy of 97% on the test set. For each of the accuracy, precision, recall, and f1 scores, our model achieved scores of about 96%. Our heat map visualization of the confusion matrix had a very clear main diagonal as well, which illustrates the overwhelming majority of images that were classified correctly.

Our transfer learning model did considerably worse; it achieved a score of about 70% when evaluated on the test data. For recall and f1, the transfer learning model achieved scores of about 70%; for precision, it achieved a score of about 74%. There was also quite a bit more noise in the heat map of the confusion matrix in areas away from the main diagonal.

One issue we did find with our best model is that even though the CNN was able to classify images from this dataset incredibly well, it failed to generalize (from our preliminary tests). When tested on fruit images sourced from the internet, the model would often fail to classify them correctly. This is likely because the images from the Fruits 360 dataset are formatted in very specific ways (white background, with the fruit taking up approximately the same portion of the image each time), so our CNN was unable to generalize and learn anything outside of this predefined format.