About Ninjacart:

Ninjacart is India's largest fresh produce supply chain company. They are pioneers in solving one of the toughest supply chain problems of the world by leveraging innovative technology. They source fresh produce from farmers and deliver them to businesses within 12 hours. An integral component of their automation process is the development of robust classifiers which can distinguish between images of different types of vegetables, while also correctly labeling images that do not contain any one type of vegetable as noise.

Problem Statement:

As a starting point, ninjacart has provided us with a dataset scraped from the web which contains train and test folders, each having 4 sub-folders with images of onions, potatoes, tomatoes and some market scenes. We have been tasked with preparing a multiclass classifier for identifying these vegetables. The dataset provided has all the required images to achieve the task.

Context:

This dataset contains images of the following food items: noise-Indian market and images of vegetables- onion, potato, and tomato.

Content

This dataset contains a folder train, which has a total of 3135 images, split into four folders as follows:

Tomato: 789

Potato: 898

Onion: 849

Indian market: 599

This dataset contains another folder test which has a total of 351 images, split into four folders.

Tomato: 106

potato: 83

onion: 83

Indian market: 81

Process:

1. There are numerous libraries that can be used to work with images. Importing the libraries, we feel most confident with is our first step: TensorFlow, keras, matplotlib, OpenCV, seaborn etc.
2. Download the Data (you can use gdown) with the provided Dataset Link and unzip it.
3. Visualize the data, use the dataset directory to create a list containing all the image paths in the training folder. You can use matplotlib or TensorFlow to plot a grid sample of the images you fetched from the list of image paths.
4. Plot a few of the images of each class to check their dimensions. [Note that the images are not all of uniform dimensions]
5. Verify the count of images in each train and test folder by plotting histogram.
   1. Check each folder to see if the number of images matches the reported number.
6. Split the dataset to a train and validation set.
   1. The provided data does not contain separate training and validation folders. For us to do hyperparameter tuning of our models, it is important to divide the dataset into an 80-20 split for training and validation respectively.
7. Before fitting data to our model, we must make sure that each image is square-shaped so that we may resize it to the required dimensions and also perform rescaling which will rescale the inputs between 0-1 by dividing each value by 255.
8. Use a model of your choice (could be vgg, resnet and mobilenet) and train it with an appropriate batch size.
   1. Using the pretrained weights of popular networks is a great way to do transfer learning, since the size of our original dataset is small.
9. Obtain the testing accuracy to see how well your model generalizes.
10. Compare and check the performance of multiple models using a confusion matrix.
11. Implement a TensorBoard callback to log each of our model metrics for each model during the training process. [ recommended google colab or jupyter notebook]
12. Plot the train/valid accuracy and loss graph for each model.
13. If the model is overfitting , Try to tune your model by applying - :
    1. BatchNormalization and Dropout
    2. Callbacks : EarlyStopping, ModelCheckpoint and TensorBoard callback
    3. Data Augmentation.

Results:-

|  |  |  |
| --- | --- | --- |
| Model | Test Accuracy | Train Accuracy |
| Custom CNN | 67.8% | 98.8% |
| CNN | 79% | 99.6% |
| VGG-19 | 87% | 99.7% |
| ResNet50 | 90% | 99.4% |
| MobileNet | 80% | 99.8% |

**Summary:**

* We trained and evaluated four different models for classifying images of vegetables and market scenes: a custom CNN, a revamped CNN, VGG19, ResNet50, and MobileNet.
* ResNet50 achieved the highest test accuracy of 90%, followed by VGG19 with 87%, MobileNet with 80%, revamped CNN with 79%, and the custom CNN with 67.8%.
* The custom CNN seems to be overfitting on the training data, as it has a high training accuracy of 98.8% but a low test accuracy of 67.8%.

**Insights:**

* ResNet50 is the best model for classifying images of vegetables and market scenes in our dataset.
* We can further improve the performance of our models by using data augmentation techniques, such as flipping, rotating, and zooming the images.
* We can also experiment with different hyperparameters, such as the learning rate, batch size, and number of epochs.