Report on classification of Type of Mangoes

The code you provided implements a mango classification model using the VGG16 convolutional neural network architecture. The model is trained on a dataset of mango images using the Keras framework with TensorFlow backend.

Data Preprocessing:

- The code uses the `ImageDataGenerator` class from Keras to perform data augmentation and preprocessing.

- For the training data, the generator applies various transformations such as rescaling, rotation, horizontal and vertical flipping, shearing, and zooming.

- The validation data generator also applies similar transformations for data augmentation.

- The generator loads the images from the specified directories (`train_dir` and `validation_dir`), resizes them to a target size of (150,150), and creates batches of images.

Model Architecture:

- The code loads the pre-trained VGG16 model (excluding the top layers) using the `VGG16` class from Keras.

- A new model is created using the `Sequential` class from Keras.

- The VGG16 model is added as the first layer in the new model.

- The new model includes a `Flatten` layer to flatten the output of the VGG16 model.

- Three `Dense` layers with ReLU activation are added, with 256, 128, and 128 units respectively.

- The final `Dense` layer with 8 units and a softmax activation function is added for multiclass classification (assuming there are 8 classes for mango classification).

- The model summary is printed to provide an overview of the model architecture.

Model Compilation and Training:

- The model is compiled using the Adam optimizer with a learning rate of 2e-5, categorical cross-entropy loss function, and accuracy metric.

- The `fit` function is called to train the model using the training and validation data generators.

- The `steps_per_epoch` parameter is set to 44, indicating the number of steps (batches) per epoch for the training data.

- The `epochs` parameter is set to 20, indicating the number of times the entire training dataset is passed through the model.

- The `validation_steps` parameter is set to 24, indicating the number of steps (batches) per epoch for the validation data.

- The training history is stored in the `history` variable.

Model Evaluation and Visualization:

- The training history is visualized using the `plot` function from Pandas, showing the training and validation accuracy and loss over epochs.

- The accuracy and loss plots are displayed with gridlines and a y-axis limit of 0 to 1.

Model Testing and Prediction:

- The trained model is saved as 'Mango_classification(VGG16).h5'.

- The saved model is loaded using `keras.models.load_model`.

- A test data generator is created using the `test_datagen.flow_from_directory` function, which loads and preprocesses the test images from the specified directory (`test_dir`).

- A function named `handle_upload` is defined to handle uploaded images and make predictions.

- The function resizes the uploaded image to (150, 150) and performs normalization.

- The model predicts the class of the uploaded image using `model1.predict`.

- The predicted class index is obtained using `np.argmax`.

- The predicted class label is retrieved using the `class_names` variable (which is not defined in the provided code).

- The uploaded image and predicted class are displayed using the `display` function and printed to the console.

- An upload button is created using the `widgets.FileUpload` function.

- The `handle_upload` function is registered to handle changes in the upload button's value.

- The upload button is displayed using the `display` function.