**wProject Summary: Pest Image Classification**

This document summarizes the key steps and findings of the pest image classification project conducted in the accompanying Colab notebook.

1. Environment Setup and Data Loading

* **Google Drive Mounting:** The notebook begins by mounting Google Drive to access the dataset stored there.
* **Garbage Collection:** Garbage collection is enabled to manage memory usage efficiently during model training.
* **Dataset Paths:** The paths to the training, validation, and testing datasets are defined.
* **Selected Categories:** A subset of 5 pest categories is selected for the classification task.
* **Image Size and Batch Size:** The standard image size for processing (224x224) and the batch size for training are defined.

2. Data Exploration and Preprocessing

* **Folder Listing:** The notebook lists the folders within the training directory to confirm the presence of image categories.
* **Image Counting:** A function is used to count the number of images in each selected category for the training, validation, and test sets. This helps understand the data distribution.
* **Category Distribution Visualization:** Bar charts are generated to visualize the distribution of images across the selected categories in each dataset split.
* **Data Augmentation:** A data augmentation pipeline is defined using tf.keras.Sequential with layers for random flipping, rotation, zooming, and brightness adjustments. This is applied to the training dataset to increase its variability and improve model generalization.
* **Dataset Creation:** tf.data.Dataset objects are created for the training, validation, and test sets, incorporating the image loading, preprocessing, and augmentation steps. The training dataset is shuffled and augmented, while the validation and test datasets are not augmented and are not shuffled.

3. Model Definition and Compilation

* **Model Selection:** Three pre-trained convolutional neural network architectures are chosen for transfer learning: ResNet50, MobileNetV2, and InceptionV3.
* **Base Model Loading:** Each base model is loaded with weights pre-trained on ImageNet, excluding the top classification layer.
* **Layer Freezing and Fine-tuning:** The layers of the base models are initially frozen to leverage the learned features. For the ResNet model, a portion of the later layers is unfrozen for fine-tuning.
* **Custom Classification Layers:** Custom dense layers and a dropout layer are added on top of the base models to adapt them for the 5-category classification task.
* **Model Compilation:** Each model is compiled with the Adam optimizer, 'sparse\_categorical\_crossentropy' loss function, and 'accuracy' as the evaluation metric. A lower learning rate is used for the fine-tuned ResNet model.
* **Mixed Precision:** Mixed precision training is enabled to potentially speed up training and reduce memory usage.

4. Callbacks

* **Model Checkpoint:** ModelCheckpoint callbacks are defined for each model to save the weights of the model with the best validation accuracy during training.
* **Early Stopping:** An EarlyStopping callback is defined to stop training if the validation accuracy does not improve for a specified number of epochs, preventing overfitting.

5. Model Training and Evaluation

* **Model Training:** The MobileNetV2, ResNet, and InceptionV3 models are trained using the prepared datasets and defined callbacks. The training history (loss and accuracy) is recorded for each model.
* **Model Evaluation:** The trained models are evaluated on the test dataset to assess their performance on unseen data.
* **Classification Report:** Classification reports are generated for the MobileNetV2 and InceptionV3 models using the best saved weights. These reports include precision, recall, f1-score, and support for each class, as well as overall accuracy, macro average, and weighted average metrics.
* **Training History Visualization:** Line plots are generated to visualize the training and validation accuracy and loss over epochs for each model, providing insights into the training progress and potential overfitting.