1. **Transfer Learning for Image Classification**

   It is highly recommended that you complete this project using Keras[1] and Python.

   (a) In this problem, we are trying to build a classifier that distinguishes images of 17 types of jute pest.

   (b) **Data Exploration and Pre-processing**

      i. Images are divided into training, validation, and test sets. You can encode your classes using one-hot encoding.

      ii. In case all the images do not have the same size, zero-pad or resize the images in your dataset. This can be done using various tools, including OpenCV.

   (c) **Transfer Learning**[2]

      i. When dealing with classification of relatively small image datasets, deep networks may not perform very well because of not having enough data to train them. In such cases, one usually uses *transfer learning*, which uses deep learning models that are trained on very large datasets such as `ImageNet` as feature extractors. The idea is that such deep networks have learned to extract meaningful features from an image using their layers, and those features can be used in learning other tasks. In order to do that, usually the last layer or the last few layers of the pre-trained network are removed, and the response of the layer before the removed layers to the images in the new dataset is used as a feature vector to train one more multiple replacement layers. In this project, you will use pre-trained models `ResNet50`, `ResNet100`, `EfficientNetB0`, `VGG16`, and `DenseNet201`. For those pre-trained networks, you will only train the last fully connected layer, and will *freeze* all layers before them (i.e. we do not change their parameters during training) and use the outputs of the penultimate layer in the original pre-trained model as the features extracted from each image.

      ii. To perform empirical regularization, crop, randomly zoom, rotate, flip, contrast, and translate images in your training set for image augmentation. You can use various tools to do this, including OpenCV.

      iii. Use ReLU activation functions in the last layer and a softmax layer, along with $\mathcal{L}_2$ regularization, batch normalization [3], and a dropout rate of 20% as well as ADAM optimizer. Use multinomial cross entropy loss. You can try any batch size, but a batch size of 5 seems reasonable.

      iv. Train the networks for at least 50 epochs (preferably 100 epochs) and perform early stopping using the validation set. Use a randomly selected subset of 20% of each class as the validation set. Keep the network parameters that have the lowest validation error. Plot the training and validation errors vs. epochs.

---

[1] https://keras.io
[2] https://builtin.com/data-science/transfer-learning
[3] https://en.wikipedia.org/wiki/Batch_normalization

v. Report training, validation, and test Precision, Recall, AUC, and F1 score for those models. Is there a model that clearly outperforms others? Remember that this is a multi-class classification problem.