ASSIGNMENT

PROBLEM STATEMENT:

To create a classification model for the given dataset that could run on mobiles.
The assignment is inspired from MobileNet which is developed for this sole purpose.

STEPS TO FOLLOW:

1. Create training model on train data-

Execute following command in python terminal-

python -m scripts.retrain --bottleneck_dir=tf_files/bottlenecks --how_many_training_steps=500 --model_dir=tf_files/models/ --summaries_dir=tf_files/training_summaries/mobilenet_0.25_128 --output_graph=tf_files/retrained_graph.pb --output_labels=tf_files/retrained_labels.txt --architecture=mobilenet_0.25_128 --image_dir=tf_files/data/train_61326

2. Optimize the model to avoid problems due to unsupported operations.

python -m tensorflow.python.tools.optimize_for_inference --input=tf_files/retrained_graph.pb --output=tf_files/optimized_graph.pb --input_names="input" --output_names="final_result"

3. Predict on test folder.

python -m scripts.testImgs

The forked label_image script is for one by one image testing. It's extended for all the test files in this script.

KEY-POINTS:

1. Using pre-trained standard model and only evaluating bottleneck (last) layer over this dataset, accuracy achieved is 100%.
2. MobileNet model generated and uploaded is only 1.9Mb which is feasible to load in mobiles.
3. Evaluation time is in milli secs hence suitable for mobiles.
4. Even the smallest MobileNet standard model (0.25 times) with input image resolution as 128px, is able to provide a good accuracy. Hence, *no more tuning was done*.
5. The model architecture could be changed based on the feasibility and the device support. Even the MobileNet times could be 1.0,0.75.0.50,0.25, the larger the better in accuracy but at the cost of speed.
6. The input size of image used could also be changed to 128,160,192,224pxs. The higher the better in accuracy but at the cost of speed and size.

7. It's possible to compress the model further but this results in loss of information hence accuracy is expected to decrease slightly. (Execute-*python -m scripts.quantize_graph --input=tf_files/optimized_graph.pb --output=tf_files/rounded_graph.pb --output_node_names=final_result --mode=weights_rounded)*
8. For detailed instructions, check- https://codelabs.developers.google.com/codelabs/tensorflow-for-poets-2/#0

NOTES:

1. Part till model creation suitable for mobile is tried as Android Studio is not supported by the resource being worked on.
2. The training and testing data is not uploaded on github due to privacy concerns. Please add the folders inside tf_files/data/ directory as follows-

```
tf_files/
        data/
            train_61326/
                        61326 Scratch Mark/<all images>
                        61326 Slot Damage/
                        .
                        .
            test_61326/
                        61326 Scratch Mark/
                        61326 Slot Damage/
                        .
                        .
```