PROJECT SPECIFICATION

Create Your Own Image Classifier - TensorFlow

Files Submitted

CRITERIA	MEETS SPECIFICATIONS
Submission Files	The submission includes all required files. (Model checkpoints not required.)

Part 1 - Development Notebook

CRITERIA	MEETS SPECIFICATIONS
Package Imports	All the necessary packages and modules are imported at the beginning of the notebook.
Data Loading	The Oxford Flowers 102 dataset is loaded using TensorFlow Datasets.
Data Splits	The dataset is divided into a training set, a validation set, and a test set.
Dataset Info	The number of examples in each set and the number classes in the dataset are extracted from the dataset info.
Dataset Images	The shape of the first 3 images in the training set is printed using a `for` loop and the `take()` method.
Plot Image	The first image from the training set is plotted with the title of the plot corresponding to the image label.
Label Mapping	The first image from the training set is plotted with the title of the plot corresponding to the class name using label mapping from the JSON file.
Data Normalization	The training, validation, and testing data is appropriately resized and normalized.
Data Pipeline	A pipeline for each set is constructed with the necessary transformations.
Data Batching	The pipeline for each set should return batches of images.
Pre-trained Network	The pre-trained network, MobileNet, is loaded using TensorFlow Hub and its parameters are frozen.
Feedforward Classifier	A new neural network is created using transfer learning. The number of neurons in the output layer should correspond to the number of classes of the dataset.
Training the Network	The model is configured for training using the `compile` method with appropriate parameters. The model is trained using the `fit` method and incorporating the validation set.
Validation Loss and Accuracy	The loss and accuracy values achieved during training for the training and validation set are plotted using the `history` dictionary return by the `fit` method.
Testing Accuracy	The network's accuracy is measured on the test data.
Saving the Model	The trained model is saved as a Keras model (i.e. saved as an HDF5 file with extension `.h5`).
Loading Model	The saved Keras model is successfully loaded.
Image Processing	The `process_image` function successfully normalizes and resizes the input image. The image returned by the `process_image` function should be a NumPy array with shape `(224, 224, 3)`.
Inference	The `predict` function successfully takes the path to an image and a saved model, and then returns the top K most probably classes for that image.
Sanity Check	A `matplotlib` figure is created displaying an image and its associated top 5 most probable classes with actual flower names.

Part 2 - Command Line Application

CRITERIA	MEETS SPECIFICATIONS
Predicting Classes	The `predict.py` script successfully reads in an image and a saved Keras model and then prints the most likely image class and it's associated probability.
Top K Classes	The `predict.py` script allows users to print out the top K classes along with associated probabilities.
Displaying Class Names	The `predict.py` script allows users to load a JSON file that maps the class values to other category names.

Suggestions to Make Your Project Stand Out!

Use at least one form of regularization (whether of those included in the lessons or others you find in the documentation). Try to get the difference between your training and validation accuracy down to three percent or less!