UDACITY

‹ Return to "Introduction to Machine Learning with TensorFlow" in the classroom

DISCUSS ON STUDENT HUB

Create Your Own Image Classifier - TensorFlow

REVIEW

CODE REVIEW 2

HISTORY

Meets Specifications

CONGRATULATIONS 🎈 🎉 🎊

It was a great experience to review your submission.
I can clearly see your dedication and skills.

Keep the hard work 🦾

Files Submitted

The submission includes all required files. (Model checkpoints not required.)
Thanks for submitting all the files.

Part 1 - Development Notebook

All the necessary packages and modules are imported at the beginning of the notebook.

The Oxford Flowers 102 dataset is loaded using TensorFlow Datasets.

The dataset is divided into a training set, a validation set, and a test set.

The number of examples in each set and the number classes in the dataset are extracted from the dataset info.

The shape of the first 3 images in the training set is printed using a `for` loop and the `take()` method.

The first image from the training set is plotted with the title of the plot corresponding to the image label.

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.

The training, validation, and testing data is appropriately resized and normalized.

A pipeline for each set is constructed with the necessary transformations.

The pipeline for each set should return batches of images.

The pre-trained network, MobileNet, is loaded using TensorFlow Hub and its parameters are frozen.

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.

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.

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.

The network's accuracy is measured on the test data.

The trained model is saved as a Keras model (i.e. saved as an HDF5 file with extension `.h5`).

The saved Keras model is successfully loaded.

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)`.

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.

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

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.

Code runs nicely.

Made two small suggestions in the code review for you to check. Good Job!

The `predict.py` script allows users to print out the top K classes along with associated probabilities.

Perfect 👌

The `predict.py` script allows users to load a JSON file that maps the class values to other category names.

Nicely done ✓

DOWNLOAD PROJECT

2 CODE REVIEW COMMENTS ❯

RETURN TO PATH

Rate this review