UDACITY

PROJECT

Dog Breed Classifier

A part of the Deep Learning Nanodegree Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

Hey! Excellent job!
Congratulations on passing the Dog Breed Classifier project

Q: the dog breed name is shown as the complete path. how to show only the name.
A: In `Resnet50_predict_breed(img_path):` you could modify the return line `dog_names[np.argmax(predicted_vector)]`, and split the image path on `.` symbol, and select only the name of the dog, which comes at the very end. You can do it as following: `dog_names[np.argmax(predicted_vector)].split('.')[1]`

The Transfer Learning technique you have learned is very powerful method and could be applied on a variety of datasets, so I highly encourage you to try to participate in some image classification challenges you can find, the place to check for new challenges is a Kaggle, they always have some new stuff coming up and it is the best way to apply your knowledge!

If you would like to improve the knowledge about Transfer Learning further here is a great place to do so: http://ruder.io/transfer-learning/

Also, I would recommend you to check out this read about CNNs (it has other parts too), pretty easy and fun to read.

Have a good one!
Kudos

Files Submitted

The submission includes all required files.

All files are present!

Step 1: Detect Humans

The submission returns the percentage of the first 100 images in the dog and human face datasets with a detected human face.

The submission opines whether Haar cascades for face detection are an appropriate technique for human detection.

Step 2: Detect Dogs

The submission returns the percentage of the first 100 images in the dog and human face datasets with a detected dog.

Step 3: Create a CNN to Classify Dog Breeds (from Scratch)

The submission specifies a CNN architecture.

The submission specifies the number of epochs used to train the algorithm.

The trained model attains at least 1% accuracy on the test set.

Step 5: Create a CNN to Classify Dog Breeds

The submission downloads the bottleneck features corresponding to one of the Keras pre-trained models (VGG-19, ResNet-50, Inception, or Xception).

✓ ResNet50 loaded

The submission specifies a model architecture.

The submission details why the chosen architecture succeeded in the classification task and why earlier attempts were not as successful.

The submission compiles the architecture by specifying the loss function and optimizer.

The submission uses model checkpointing to train the model and saves the model weights with the best validation loss.

The submission loads the model weights that attained the least validation loss.

Accuracy on the test set is 60% or greater.

Amazing results! Pure guessing produces the results of only 0.75%, your model ~108 times more accurate than that with 81.22% accuracy!

The submission includes a function that takes a file path to an image as input and returns the dog breed that is predicted by the CNN.

Step 6: Write Your Algorithm

The submission uses the CNN from Step 5 to detect dog breed. The submission has different output for each detected image type (dog, human, other) and provides either predicted actual (or resembling) dog breed.

Great implementation of the pipeline. For further improvement I would recommend you to show the predicted dogs breed photo next to the input image, that way you can see what the algorithm predicts without looking up for the breed separately.

If you would like to improve the face detection part of the algorithm, I would recommend you to check out the following project by Adam Geitgey: https://github.com/ageitgey/face_recognition.

Step 7: Test Your Algorithm

The submission tests at least 6 images, including at least two human and two dog images.

Testing - succesful ✓

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

Student FAQ