UDACITY

PROJECT

Dog Breed Classifier

A part of the Deep Learning Nanodegree Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

Overall you did well. Please work on the justification part of the choices you made on architecture. This will help you in future where you are in the position of deciding and justifying the proposals you make on your designs.

Files Submitted

The submission includes all required files.

all files present. However please dont put all the files as it makes the submission very heavy

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.

Good job with accuracies. Would have been great if you had converted them into % format

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

Your view is right, though most of the time we expect that our 'intelligent interface' should be at least able to handle small occlusions and invariance in the rotation, light etc. Haar cascade in that respect is not a very good choice. HOG detectors, and deep learning models perform better in this case.

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.

Good job with accuracies. Would have been great if you had converted them into % format

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

The submission specifies a CNN architecture.

Good model, great you added Dropout. Also, feel free to look into Batch Normalization which is very standard now and usually increases performance.
https://keras.io/layers/normalization/
http://cs231n.github.io/convolutional-networks/#architectures.

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

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

Great accuracy. `9.3301%`

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).

The submission specifies a model architecture.

Good job with using Inception. Did you try others?

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.

Good 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.

Good to see you did some innovative stuff with your results.

Step 7: Test Your Algorithm

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

Great blend of images. Good job.

DOWNLOAD PROJECT

RETURN TO PATH

Student FAQ