UDACITY

PROJECT

Dog Breed Classifier

A part of the Deep Learning Nanodegree Program

PROJECT REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

In order to improve one idea is to augment the data going into the transfer learning set you would have to regenerate the bottleneck features. They are essentially the input images processed through the model and cached for later use. This allows you to use the result from the last layer and rerun many times without running the image through the entire network each time. The one person that I saw that did augment the data and regenerate the features was able to get from 85% up to 90%. More info on extracting bottleneck features are in the image augmentation link and here https://keras.io/applications/

Files Submitted

The submission includes all required files.

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.

Good description of the layers used and rationale behind using them.

I've seen good performance with an additional Conv layer, larger initial kernel size, or experimenting with Dropout layers after the max pooling layers.

Two links that helped me understand how some of the parameters work.

https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/
https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/

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

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

Test accuracy: 6.4593%

Now that you have it improved over your trials it might be worth running more epochs to see how it goes.

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).

DogVGG19Data.npz

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.

I would suggest trying the other three options. Unfortunately VGG19 starts off performing worse than the others and requires more layers to being the percent up. Most of the others with only a GlobalAveragePooling2D, Dense get ~80% with few epochs.

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.

Test accuracy: 70.5742%

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.

Step 7: Test Your Algorithm

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

Good set of sample images multiple humans and dogs. Includes good ideas for next direction to take this for improvement.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

Student FAQ