UDACITY

Back to Deep Learning Nanodegree

Dog Breed Classifier

REVIEW

CODE REVIEW

HISTORY

SHARE YOUR ACCOMPLISHMENT!

Requires Changes

2 SPECIFICATIONS REQUIRE CHANGES

Great work! You just have a couple small fixes to finish the project; see below for guidance.

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.

Despite your attempt to be un-funny as a German you still made me laugh! But please be sure to discuss if Haar cascades are an appropriate technique for face detection in your answer to #2 as well.

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.

Nice model! I almost always use batch normalization, because it trains faster and gets higher accuracy. I tend to use the elu activation function because it typically trains faster and gets higher accuracy, so I recommend using those two tricks.

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).

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.

Please be sure to address this specification, around question 5 is a good place to do it. Why did your chosen architecture with transfer learning work, and earlier attempts using a custom neural net not work nearly as well? What is so special about transfer learning?

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

Nice work! The `adam` optimizer does sometimes work well, but also can generalize poorly: https://papers.nips.cc/paper/7003-the-marginal-value-of-adaptive-gradient-methods-in-machine-learning
It's usually best to try an ada-based method (adam, nadam, etc) and SGD with momentum, and use whichever has better generalization (test/validation set) performance.

Here is a comparison of some different optimizers: http://ruder.io/optimizing-gradient-descent/

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.

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 job! It's also a nice touch to output the percent confidence the model has of a dog breed. The .predict() method from the model will give you probabilities, so you can get it from there.

Step 7: Test Your Algorithm

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

RESUBMIT

DOWNLOAD PROJECT

Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

RETURN TO PATH

Student FAQ