UDACITY

‹ Return to "Deep Learning" in the classroom

Dog Breed Classifier

REVIEW

HISTORY

Meets Specifications

Awesome job on the project! Now you know how what-dog works.
To make the net run faster, we could possibly use some compression.
🐶

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.

Good answer to question 2! We could also use a convnet to detect humans; then we don't even need a face. For example, MTCNN could be used.

Here is another option: https://github.com/ageitgey/face_recognition

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.

Great model! I usually use batch normalization like you added, because it usually trains faster and gets higher accuracy. I also 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.

Part of the reason some of these nets work really well and better than others is because they were trained on more dog images. You can see the classes for VGG16 include only a handful of dogs, Xception and inception have many more. Of course our net trained from scratch did not have nearly enough time or complexity to learn the statistical differences between dog breeds, since the accuracy was so low.

This is a list of classes for VGG-16
http://image-net.org/challenges/LSVRC/2014/browse-synsets

Also, the deeper layers find more complex patterns: http://www.coldvision.io/2016/07/29/image-classification-deep-learning-cnn-caffe-opencv-3-x-cuda/

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

Nice work! I often use the `adam` optimizer because it usually works well, but it can also 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! I suggest showing the images with something like `plt.imshow()`, and cleaning up the dog names with some string expressions, like `dog_id.split('.')[1].replace('_', ' ')`.
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.

DOWNLOAD PROJECT

RETURN TO PATH