UDACITY

PROJECT

Dog Breed Classifier

A part of the Deep Learning Nanodegree Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

Congratulations !!

Awesome job on completing your project !! Your project meets all the specifications. All the best for your future projects

Useful Links

1. CNN's for Visual Recognition
2. Deep Conv nets for image classification
3. Large Scale image Recognition using DNN's
4. Transfer Learning
5. Awesome Deep Learning Papers

Suggestions to Make Your Project Stand Out!

AUGMENT THE TRAINING DATA

- Augmenting the training and/or validation set might help improve model performance.

TURN YOUR ALGORITHM INTO A WEB APP

- Turn your code into a web app using Flask or web.py

OVERLAY DOG EARS ON DETECTED HUMAN HEADS

- Overlay a Snapchat-like filter with dog ears on detected human heads. You can determine where to place the ears through the use of the OpenCV face detector, which returns a bounding box for the face. If you would also like to overlay a dog nose filter, some nice tutorials for facial keypoints detection exist here.

ADD FUNCTIONALITY FOR DOG MUTTS

- Currently, if a dog appears 51% German Shephard and 49% poodle, only the German Shephard breed is returned. The algorithm is currently guaranteed to fail for every mixed breed dog. Of course, if a dog is predicted as 99.5% Labrador, it is still worthwhile to round this to 100% and return a single breed; so, you will have to find a nice balance.

EXPERIMENT WITH MULTIPLE DOG/HUMAN DETECTORS

- Perform a systematic evaluation of various methods for detecting humans and dogs in images. Provide improved methodology for the `face_detector` and `dog_detector` functions.

Files Submitted

✓

The submission includes all required files.

Notebook and HTML export are included 👍

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 job answering this question.

Useful Link : Summary of Deep Models for 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.

Good job discussing the architecture of your CNN. You can use batch normalisation to further improve the performance of your model.

Suggested Reading : Systematic evaluation of CNN advances on the ImageNet

✓

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

Reasonable choice on number of epochs !! Try experimenting with different values for number of epochs and see how the performance of the network varies.

✓

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.

Below are links to some of the famous papers on different deep learning architectures:

- Le Net 5 - Gradient Based Learning Applied to Document Recognition
- Alex Net - ImageNet Classification with Deep Convolutional Neural Networks
- VGG 16 - VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION
- GoogLeNet - Going Deeper with Convolutions
- Xception : Deep Learning with depthwise separable Convolutions

✓

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.

Good job choosing rmsprop for optimiser, also try experimenting with adadelta, adam and nadam.

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 job getting a accuracy of 81%.

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 work on testing your model on different images and suggesting some improvements.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

Student FAQ