UDACITY

< Back to Deep Learning

# Dog Breed Classifier

REVIEW

CODE REVIEW

HISTORY

## Meets Specifications

Congratulations for passing this project!
Good luck with future submissions :)

### Files Submitted

**The submission includes all required files.**

Awesome! The submission includes all the 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.**

Perfect!
Percentage of human faces detected in human images = 100 %
Percentage of human faces detected in dog images = 11 %

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

Good explanation.

Some issues pertaining to face detection using Haar Cascades approach can be seen here: 1, 2.

## 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.**

The percentage values are even better here :)

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

**The submission specifies a CNN architecture.**

Good job building a CNN from scratch.
Here you can also add `batchNormalization()` layer to improve model performance. See the documentation here.

Also check this post by Ian Goodfellow.

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

You can use an early stopping callback provided by Keras, which ends training based on pre configured rules such as non improving accuracy.

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

The accuracy is 1.5550 %.

## 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).**

Awesome!
The bottleneck features are successfully imported.

**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.**

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

Nice job using `categorical_crossentropy` and `rmsprop`. For a detailed overview of gradient descent optimizers check out this link.

**The submission uses model checkpointing to train the model and saves the model weights with the best validation loss.**

Perfect!

**The submission loads the model weights that attained the least validation loss.**

Your use of `save_best_only=True` suggests the model with least validation loss is loaded.

**Accuracy on the test set is 60% or greater.**

The accuracy is 78.3493 %.

**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.**

Well done :)

## 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! Different outputs are generated for each detected image :)

## Step 7: Test Your Algorithm

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

Nice improvement suggestions! Check this blog.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review