UDACITY

‹ Back to Deep Learning Nanodegree

Dog Breed Classifier

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Terrific job with the project! It was really a great experience reviewing your project. I'm impressed with how you leveraged most of the concepts of Convolutional Neural networks and transfer learning in your submission. 🙌

Here are some resources to learn more:

1. MIT: Computer Vision - Deep Learning for Self-Driving Cars (2018 version)
2. Transfer learning
3. Transfer learning - Kaggle Rainforest competition
4. transfer learning - Fine tuning
5. CS231n: Convolutional Neural Networks for Visual Recognition

Files Submitted

✓

The submission includes all required files.

Perfect all necessary files are here!

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.

Nice CNN architecture and outline!

Good work using `GlobalMaxPooling2D` or `GlobalAveragePooling2D` to prevent overfitting and reduce the number of parameters of the model! Also, good choice on using dropout to reduce overfitting.

TIPS:

- You can use batch normalization on each `Conv2D` layer to speed up training. For more information on bath normalization, you can watch these 1, 2, and 3 videos from Andrew Ng

✓

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

Good choice!

✓

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

Nice job!

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.

Well done for choosing InceptionV3!

✓

The submission details why the chosen architecture succeeded in the classification task and why earlier attempts were not as successful.

Nice work in explaining why you think the architecture is suitable for the current problem.

For more information about some of the architectures mentioned above, take a look at this article:
ImageNet: VGGNet, ResNet, Inception, and Xception with Keras

ResNet, AlexNet, VGG, Inception: Understanding various architectures of Convolutional Networks

✓

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

Good job choosing `categorical_crossentropy` for the loss and `rmsprop` as the `optimizer`.

I recommend you try `Adam` or `SGD` with `momentum=0.9` and `decay=1e-6` as the optimizer. For more details, here is an An overview of gradient descent optimization algorithms to learn more about the strength and weaknesses of the most popular optimization algorithms.

✓

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.

Great job on achieving ~77% accuracy 👏

✓

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.

Well done, you put the most accurate and robust classifier first to increase the overall accuracy!

Step 7: Test Your Algorithm

✓

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

Nice job on the implementation, testing, and on answering **Question 6**!

DOWNLOAD PROJECT

RETURN TO PATH