UDACITY

‹ Return to "Deep Learning" in the classroom

Dog Breed Classifier

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Great submission! 👍

All functions were implemented correctly, the detectors easily meet the required accuracies and the final algorithm seems to work quite well.

If you want to dive deeper into CNNs/image classification and challenge yourself after finishing this project:

1. I would recommend first to read up on how to generate the bottleneck features yourself. See https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html under the heading " Using the bottleneck features of a pre-trained network: 90% accuracy in a minute" on how to do this with Keras.
2. After that you should be ready to take part in one of the playground competitions on Kaggle like https://www.kaggle.com/c/dog-breed-identification and https://www.kaggle.com/c/plant-seedlings-classification. Check out the kernels to learn techniques and tricks from other people and see if you can get a top leaderboard score!

and see if you can get a top leaderboard score!

Files Submitted

The submission includes all required files.

All required files 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.

```
98% images of the first 100 human_files were classified as human.
12% images of the first 100 dog_files were classified as human.
```

Perfect!

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

Good discussion of whether it's a reasonable expectation to ask the user to provide a clear view of a face.

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.

```
2% images of the first 100 human_files were detected as dog.
```

```
100% images of the first 100 dog_files were detected as dog.
```

Great! Note that the CNN dog detector has many fewer false positives as the face detector: 2 vs 12. Therefore, in the algorithm (step 5) it's better to place the dog detector before the face detector to get more accurate results.

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

The submission specifies a CNN architecture.

Good choice of the architecture and well explained!

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

The number of epochs you used for training is sufficient to get the required accuracy, good job!

To get everything out of your network and data, the epochs should be chosen such that the validation accuracy is no longer increasing. You can do this manually or, better, use the early stopping callback function of Keras (https://keras.io/getting-started/faq/#how-can-i-interrupt-training-when-the-validation-loss-isnt-decreasing-anymore).

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

```
Test accuracy: 2.9904306220095696%
```

You easily beat the required 1% accuracy. 👍

Some tips to improve the accuracy further:

- Be sure to train for sufficient epochs, it helps to plot training and validation accuracy to see when your model stops improving and starts overfitting.
- Add more convolutional layers (this will require a longer training time though).
- Using multiple max pooling layers to reduce the size to about 10x10 and apply global average pooling.

- Add batch normalization after every convolutional or dense layer see https://keras.io/layers/normalization/ for the Keras documentation

https://keras.io/layers/normalization/ for the Keras documentation.

- Augment the training data, see https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html.

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).

```
bottleneck_features = np.load('bottleneck_features/DogInceptionV3Data.npz')
```

InceptionV3 is a good choice! You might want to check out ResNet and Xception, they are popular networks producing state of the art results, see:

- ResNet (original) https://arxiv.org/abs/1512.03385
- WideResNet https://arxiv.org/abs/1605.07146
- SENet https://arxiv.org/abs/1709.01507
- Xception https://arxiv.org/abs/1610.02357

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.

Good description of your approach!

Note that the pretrained models were trained on Imagenet which is very similar to our dog breed dataset. As a result, the bottleneck features contain sufficient information to easily identify most dog breeds by just using one dense layer. This explains why transfer learning works so well in this case.

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

Good work!

Instead of using `rmsprop` as optimizer you might want to try out `adam`, it's usually a better choice as it's easier to configure and gives superior results. Check out https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ for more information.

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.

```
Test accuracy: 81.6986%
```

The test accuracy exceeds the required 60%, well done!

Compared to training from scratch (step 3), transfer learning results in a pretty impressive accuracy. To further increase the accuracy you could try to lower the learning rate once validation accuracy stabilizes or use a learning rate schedule, see https://machinelearningmastery.com/using-learning-rate-schedules-deep-learning-models-python-keras/

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.

Your algorithm gives the correct output! 👍

To give some extra information to the user you could think about adding the predicted probability of a dog breed (the `.predict()` function of the Keras model returns the probabilities) and showing an (example) image of the predicted dog breed.

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