‹ Return to "Deep Learning" in the classroom

Dog Breed Classifier

REVIEW | HISTORY

Requires Changes

3 SPECIFICATIONS REQUIRE CHANGES

Good first submission on this project! 👍

You have shown a good understanding of CNNs, both in training them from scratch (step 3) and using transfer learning (step 5). There are few things left to do, please see my comments below for the details.

Files Submitted

The submission includes all required files.
All required files 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 detector has found 100.0% of the images in human_files_short to include human faces
The detector has found 11.0% of the images in dog_files_short to include human faces
```

Perfect!

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

It seems you missed to answer this question:

- **Question 2**: This algorithmic choice necessitates that we communicate to the user that we accept human images only when they provide a clear view of a face (otherwise, we risk having unneccessarily frustrated users!). In your opinion, is this a reasonable expectation to pose on the user? If not, can you think of a way to detect humans in images that does not necessitate an image with a clearly presented face?

Note that this question is not optional because it is a rubric item (implementing an alternative face detector is optional). Please include an answer, where in you discuss whether Haar cascades are an appropriate technique for human detection, to meet the specifications.

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 detector has found 0.0% of the images in human_files_short to include dogs
The detector has found 100.0% of the images in dog_files_short to include dogs
```

Great! Note that the CNN dog detector has many fewer false positives as the face detector: 0 vs 11.

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!

RECOMMENDATION

The only problem with the architecture is that the final layer has a relu activation. The loss function is `categorical_crossentropy` which expects probabilities (values with range in [0,1] summing to 1). Since the network doesn't output probabilities the gradient descent algorithm will have a hard time finding a good minimum for the loss function. By setting the activation function of the final layer to `softmax` you should be able to get a much higher accuracy.

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: 1.0766%
```

The accuracy is sufficient but please see my comment above on `softmax` to greatly improve accuracy.

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).

ResNet-50 is a good choice! Currently ResNet-based models and Xception are very 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.

It is required to answer the following question:

- **Question 5**: Outline the steps you took to get to your final CNN architecture and your reasoning at each step. Describe why you think the architecture is suitable for the current problem.

In your answer try to explain why the result in step 5 is better than step 3 (training from scratch) and step 4 (transfer learning with VGG16).

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: 78.5885%
```

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.

Note that the blue images are caused by a different ordering of RGB channels between OpenCV and Matplotlib, see for more information https://stackoverflow.com/questions/40067243/matplotlib-adding-blue-shade-to-an-image/40068263.

Step 7: Test Your Algorithm

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

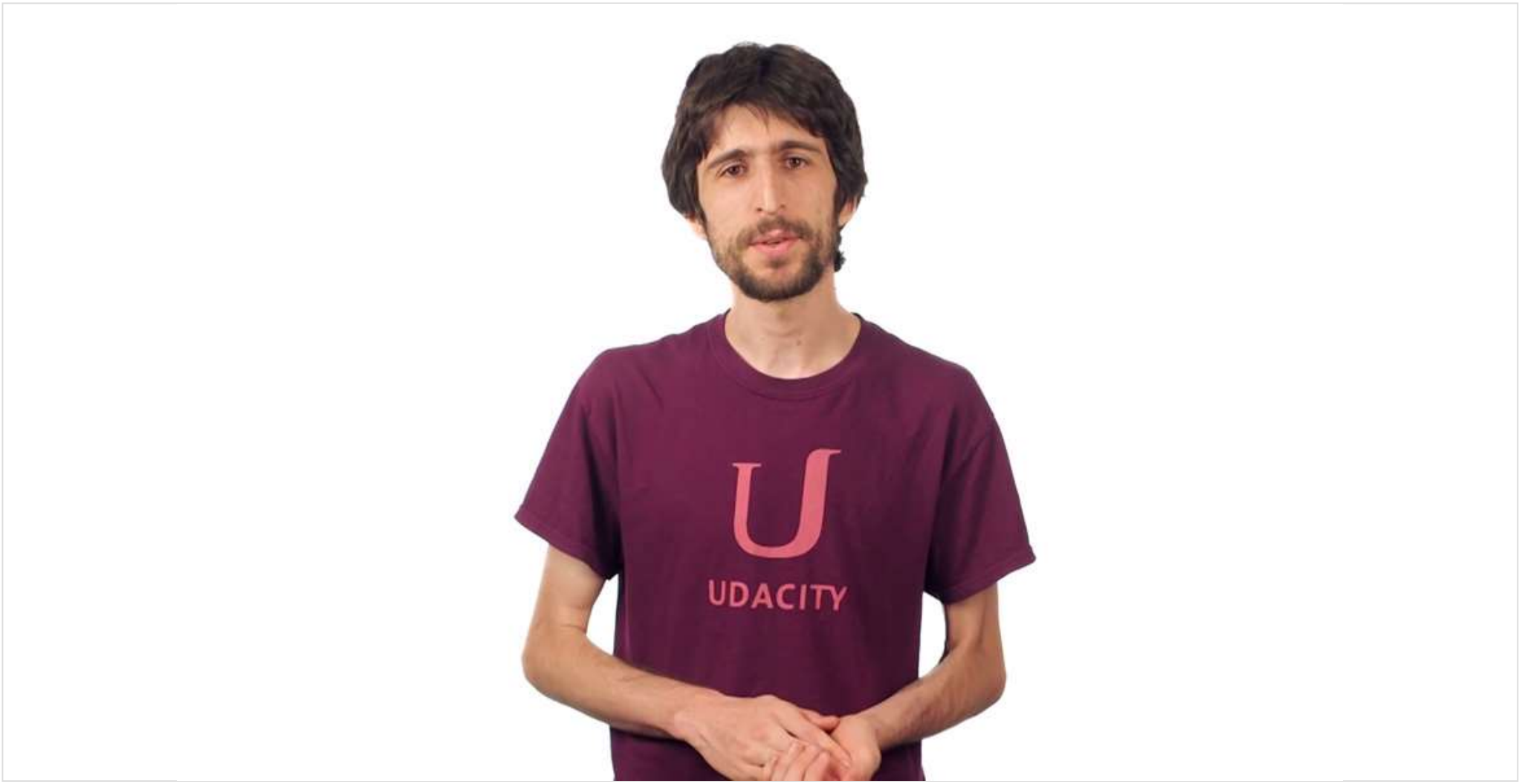
The algorithm is tested on 4 images, please tests at least 6 images, including at least two human and two dog images. I would recommend uploading your own images. It's a good practice in machine learning to not reuse training and validation data.

To upload images, in Jupyter notebook in the menu click on File > Open..., this will bring you to the current directory. Here you can upload images by clicking the upload button. You can go back to the notebook by simply opening it from

the directory listing.

RESUBMIT PROJECT

DOWNLOAD PROJECT



Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)