UDACITY

‹ Return to "Deep Learning" in the classroom

DISCUSS ON STUDENT HUB

Dog Breed Classifier

REVIEW

CODE REVIEW

HISTORY

Requires Changes

7 SPECIFICATIONS REQUIRE CHANGES

Hi,

It's a pleasure to review your project, this is a good submission. Just a few more details and you are done here.

I think you could delete the models that you are not using. It may be causing interference in the data in a way (I don't know how, but by cleaning your code you may see what's wrong here).

Also, if you can't find the problem, use your mentor and Student Hub to get more support, ok?

Keep up the good work 👊 and count on us!

Best regards,

Files Submitted

The submission includes all required files.

As defined in the instructions for this project, I'm missing this part:

> dog images
> human images

Please include in your next submission

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.

You didn't run this code cell here.

Please do it before resubmitting.

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

Good reasoning here, I totally agree with you.

You are using a CODE cell to write text here. Just make sure next time you use the Markdown cell your to answer your question.

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.

Here, you also didn't run the cell.

You should implement the function to predict human faces in both data sets.

> Question 1: Use the code cell below to test the performance of the face_detector function.
>
> - What percentage of the first 100 images in human_files have a detected human face?
> - What percentage of the first 100 images in dog_files have a detected human face?

Here is a code snippet to help you. Please, fill the `???`:

```
def count_faces(batch):
    count = 0
    for x in batch:
        if(face_detector(???)):
            count = count + 1
    return count/len(???)

result = count_faces(???)
print('Human faces:', result*100, '%, should be 100%')
result = count_faces(???)
print('Human faces:', result*100, '%, should be 0%')
```

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

The submission specifies a CNN architecture.

I'm missing your answer to the question:

> Outline the steps you took to get to your final CNN architecture and your reasoning at each step. If you chose to use the hinted architecture above, describe why you think that CNN architecture should work well for the image classification task.

Please, give the detail explanation about the chosen model and the design constraints you considered.

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

Great!

TIP

Here are several documents that talk about the choice of the number of epochs:

- How does one choose optimal number of epochs?
- How to train your Deep Neural Network
- Number of epochs to train on

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

Good result of your model on the test dataset of dog images.

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 downloaded and used one of the Keras pre-trained models correctly

Good choice here!

The submission specifies a model architecture.

The pre-trained model itself does most of the heavy work, so it is good to have a simple model like this.

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

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

It is required that you give some reasoning why the chosen architecture succeeded in classifying the images and why earlier attempts were not as successful.

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

TIPS

I would suggest you to try out other optimizers, like Adam or Adamgrad.

Here are 2 helpful references:

- https://keras.io/optimizers/
- http://ruder.io/optimizing-gradient-descent/

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!

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.

Great work! Your project has the function to take the file path to an image as input and returns the breed predicted.

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.

The output of your function is not working properly. Please review.

Step 7: Test Your Algorithm

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

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

Also include images of humans here, as required.

RESUBMIT

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)

RETURN TO PATH