UDACITY

‹ Return to "Deep Learning" in the classroom

DISCUSS ON STUDENT HUB

Dog Breed Classifier

REVIEW

CODE REVIEW

HISTORY

Requires Changes

3 SPECIFICATIONS REQUIRE CHANGES

Nice submission. A few minor changes and the project will be complete.
Hope you found the material interesting and enjoyable.
Great job and good luck going forward.

Files Submitted

The submission includes all required, complete notebook files.

Step 1: Detect Humans

The submission returns the percentage of the first 100 images in the dog and human face datasets that include a detected, human face.

Step 2: Detect Dogs

Use a pre-trained VGG16 Net to find the predicted class for a given image. Use this to complete a `dog_detector` function below that returns True if a dog is detected in an image (and False if not).

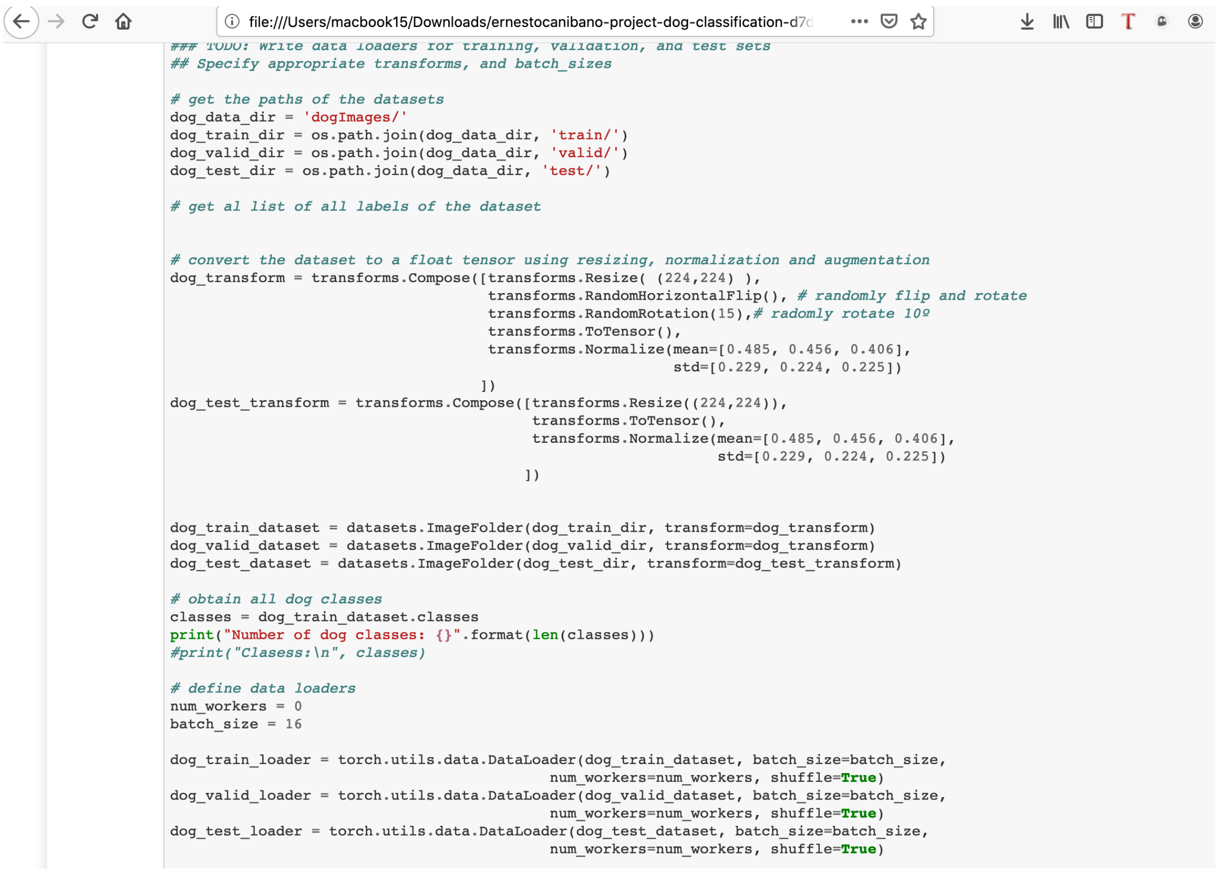
The submission returns the percentage of the first 100 images in the dog and human face datasets that include a detected dog.

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

Write three separate data loaders for the training, validation, and test datasets of dog images. These images should be pre-processed to be of the correct size.

Please, set the validation transform to the one without data augmentation as that is only appropriate for training.
Perhaps it was a typo/oversight.

Also, best to set Shuffle=False for the validation and test data loaders.
This may or may affect results in this particular case.


file:///Users/macbook15/Downloads/ernestocanibano-project-dog-classification-d7c

```
### TODO: Write data loaders for training, validation, and test sets
## Specify appropriate transforms, and batch_sizes

# get the paths of the datasets
dog_data_dir = 'dogImages/'
dog_train_dir = os.path.join(dog_data_dir, 'train/')
dog_valid_dir = os.path.join(dog_data_dir, 'valid/')
dog_test_dir = os.path.join(dog_data_dir, 'test/')

# get al list of all labels of the dataset


# convert the dataset to a float tensor using resizing, normalization and augmentation
dog_transform = transforms.Compose([transforms.Resize( (224,224) ),
                                   transforms.RandomHorizontalFlip(), # randomly flip and rotate
                                   transforms.RandomRotation(15),# radomly rotate 10º
                                   transforms.ToTensor(),
                                   transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                                        std=[0.229, 0.224, 0.225])
                                  ])
dog_test_transform = transforms.Compose([transforms.Resize((224,224)),
                                        transforms.ToTensor(),
                                        transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                                             std=[0.229, 0.224, 0.225])
                                       ])


dog_train_dataset = datasets.ImageFolder(dog_train_dir, transform=dog_transform)
dog_valid_dataset = datasets.ImageFolder(dog_valid_dir, transform=dog_transform)
dog_test_dataset = datasets.ImageFolder(dog_test_dir, transform=dog_test_transform)

# obtain all dog classes
classes = dog_train_dataset.classes
print("Number of dog classes: {}".format(len(classes)))
#print("Classess:\n", classes)

# define data loaders
num_workers = 0
batch_size = 16

dog_train_loader = torch.utils.data.DataLoader(dog_train_dataset, batch_size=batch_size,
                                          num_workers=num_workers, shuffle=True)
dog_valid_loader = torch.utils.data.DataLoader(dog_valid_dataset, batch_size=batch_size,
                                          num_workers=num_workers, shuffle=True)
dog_test_loader = torch.utils.data.DataLoader(dog_test_dataset, batch_size=batch_size,
                                          num_workers=num_workers, shuffle=True)
```

Answer describes how the images were pre-processed and/or augmented.

The submission specifies a CNN architecture.

Great you experimented with dropout. Not sure if you tested different levels but, usually, it performas best at a high rate perhaps 40-50%.

When you next have the opportunity, if you like, have a look at batch normalization for improving training speed, as well as other aspects.

Also, ELU activation is gaining in popularity vs Relu though they are similar.

https://pytorch.org/docs/master/nn.html#batchnorm1d
https://pytorch.org/docs/stable/nn.html#non-linear-activations-weighted-sum-nonlinearity

Answer describes the reasoning behind the selection of layer types.

Choose appropriate loss and optimization functions for this classification task. Train the model for a number of epochs and save the "best" result.

The trained model attains at least 10% accuracy on the test set.

Excellent % for this section.

Step 4: Create a CNN Using Transfer Learning

The submission specifies a model architecture that uses part of a pre-trained model.

The submission details why the chosen architecture is suitable for this classification task.

Train your model for a number of epochs and save the result wth the lowest validation loss.

Accuracy on the test set is 60% or greater.

VGG19 does the job.

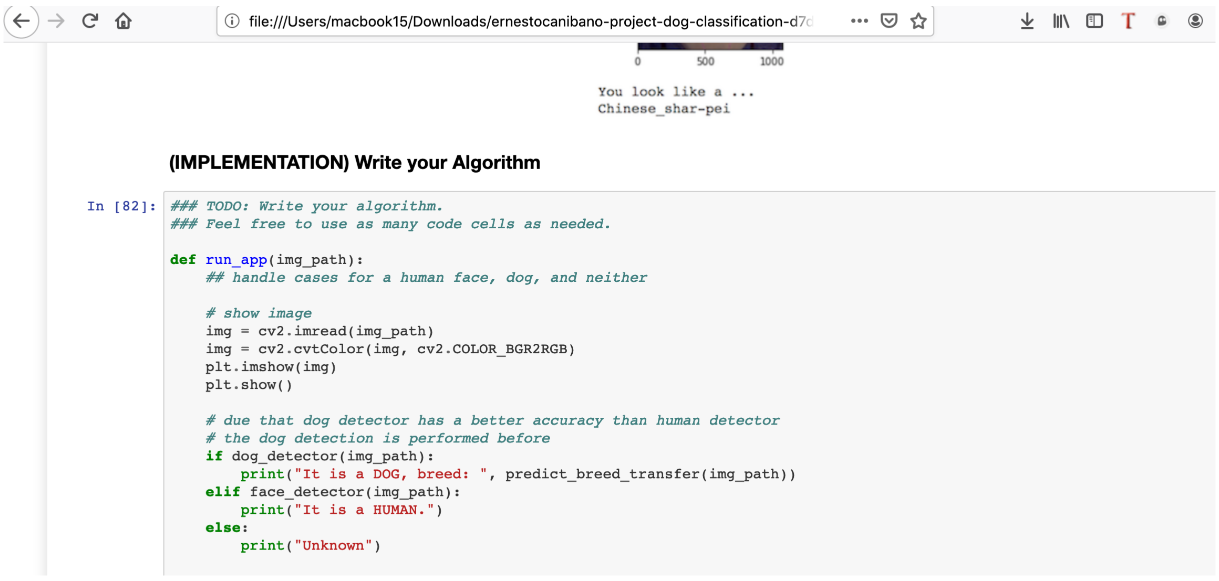
The submission includes a function that takes a file path to an image as input and returns the dog breed that is predicted by the CNN.

Step 5: Write Your Algorithm

The submission uses the CNN from the previous step 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.

Correct logical ordering. Nice to have the display code written only once before the logic clauses.

Please, add the predicted breed in the case of human/face detection as per the project rubric as that is currently only shown for the dog case.



file:///Users/macbook15/Downloads/ernestocanibano-project-dog-classification-d7c

```
You look like a ...
Chinese_shar-pei
```

(IMPLEMENTATION) Write your Algorithm

```
In [82]: ### TODO: Write your algorithm.
         ### Feel free to use as many code cells as needed.

         def run_app(img_path):
             ## handle cases for a human face, dog, and neither

             # show image
             img = cv2.imread(img_path)
             img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
             plt.imshow(img)
             plt.show()

             # due that dog detector has a better accuracy than human detector
             # the dog detection is performed before
             if dog_detector(img_path):
                 print("It is a DOG, breed: ", predict_breed_transfer(img_path))
             elif face_detector(img_path):
                 print("It is a HUMAN.")
             else:
                 print("Unknown")
```

Step 6: Test Your Algorithm

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

Nice looping through images and calling run_app() only once.

Please, do not test on training set images from the project. It is crucial to always use previously unseen data for testing any model/algorithm.
Any from the web or your own should be fine.
https://machinelearningmastery.com/data-leakage-machine-learning/

Submission provides at least three possible points of improvement for the classification algorithm.

Here is a resource on improving general neural networks, if you like:
https://machinelearningmastery.com/improve-deep-learning-performance/

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

Rate this review