Requires Changes

3 specifications require changes

Nice submission. Models are constructed and trained well so results pass the thresholds and no need to go through that time consuming process again.
Code is generally clean and concise.
Please, make sure to correct the data loaders.
Also, to answer Question 6 in full.
Should be quick fixes and the project will be complete.
Couple of resources for more on the subject if interested:
Ensembles
GANS
Hope you found the material interesting and enjoyable.
Great job and good luck going forward.

Files

Submitted

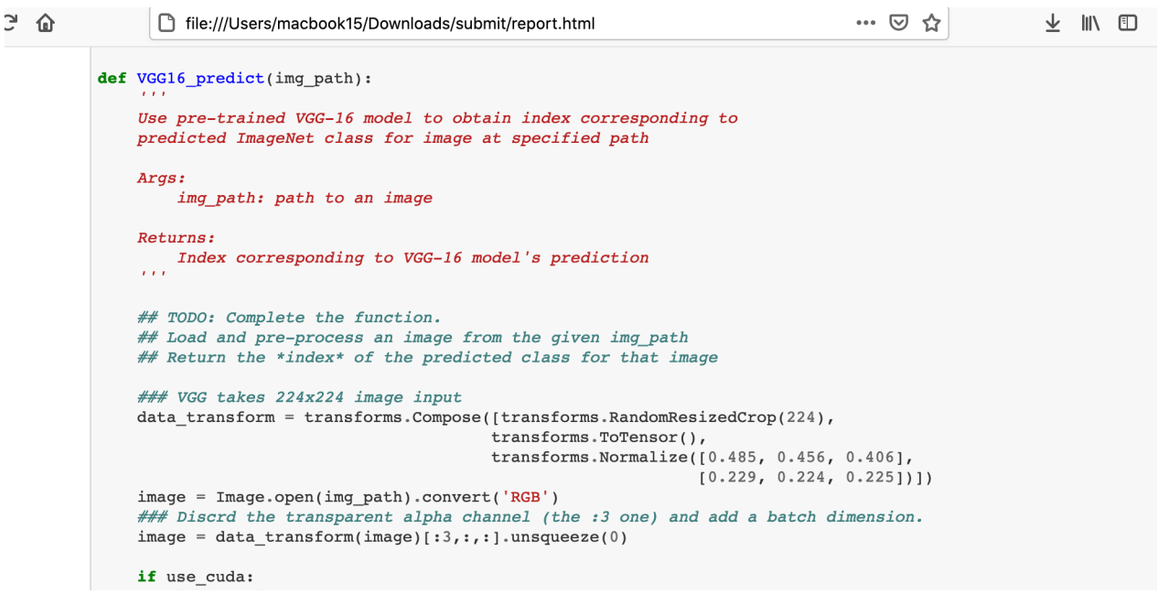
The submission includes all required, complete notebook files.
Required files contained in submission.

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).
Please, remove any random aspects from the transform as data augmentation is only appropriate for model training. This may or may not affect results in this section.




```
def VGG16_predict(img_path):
    '''
    Use pre-trained VGG-16 model to obtain index corresponding to
    predicted ImageNet class for image at specified path

    Args:
        img_path: path to an image

    Returns:
        Index corresponding to VGG-16 model's prediction
    '''

    ## TODO: Complete the function.
    ## Load and pre-process an image from the given img_path
    ## Return the *index* of the predicted class for that image

    ### VGG takes 224x224 image input
    data_transform = transforms.Compose([transforms.RandomResizedCrop(224),
                                         transforms.ToTensor(),
                                         transforms.Normalize([0.485, 0.456, 0.406],
                                                              [0.229, 0.224, 0.225])])
    image = Image.open(img_path).convert('RGB')
    ### Discrd the transparent alpha channel (the :3 one) and add a batch dimension.
    image = data_transform(image)[:3,:,:].unsqueeze(0)

    if use_cuda:
```

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.

Best to set Shuffle=False for the test and validation data loaders(similar reasoning to data augmentation).

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 performs best at a rate as high as 50%.

Batch normalizaiton is commented out. Did you try it? If so and it turned out worse, OK, though that would be unusual. If not, try it in a future CNN if that comes along. It is, typically, the simplest change to improve results.

Also, you could apply it to each layer and , especially, each convolutional layer.

Answer describes the reasoning behind the selection of layer types.

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

ELU

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

If you work on a similar problem in the future, might be worth testing an optimizer with variable learning rate such as the popular Adam.

Optimizers

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

Lots of training yielded a very nice result 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.

Please, make sure corrected data loaders(as per earlier section) are copied here.

Quick one liner for reapplying same data loaders with a different variable name for this specific model:

```
loaders_transfer = loaders_scratch.copy()
```

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.

VGG16 does the job with only minimal training.

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.

Step 6: Test Your Algorithm

Technically correct but in this simple case more pythonic to eliminate the '== True'.

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

Output format looks correct.

Because the notebook and directions are not clear and you ran the cell with training data from the project skeleton just making sure it is known that one should never use training data for testing in any production scenario.

No worries, if you were clear on this and your test images afterward are all fine.

Data Leakage

Just a note that with test images in their own folder you could loop through it and call your function once. Less typing and/or copy/paste and somewhat cleaner code.

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

Please, answer the second part of Question 6 with 3 points of improvement.