Project: Write an Algorithm for a Dog Breed Identification App

The goals / steps of this project are the following:

In this notebook, it is required to make the first steps towards developing an algorithm that could be used as part of a mobile or web app. At the end of this project, the finished code will accept any user-supplied image as input and do detections:

- If a dog is detected in the image, it will provide an estimate of the dog's breed, therefore it needs two classification: **`dog-vs-non_dog`** , and **`dog breed`** classification.
- If a human is detected, it will provide an estimate of the dog breed that is most resembling, therefore a **`face-detector`** model is needed.

Steps:

1. Datasets import: The data set is downloaded and put under the same directory as dog_app.ipynb is located. dog images are under the **`dog_images`** directory, while human face images is under **`lfw`** directory

With data been loaded, the following steps are to define three detectors mentioned above: `human face detector, dog detector and dog breed detector`. For all those three detector, either specific computer vision algorithms or deep neural networks can be used. If neural network, either creating one from scratch or using transfer learning method.

2. Human face detector definition: in cell 2 and 3 human face detector based on **`Haar feature-based cascade classifiers`**, this model achieved **`98%`** and **`18%`** face detection accuaracy on human face data set and dog data set respectively

3. Dog detector definition: from cell 6 t0 9, a dog detector using transfer learning is defined based on **`VGG16`**. This model achieved around **`93%`** and **`0%`** dog detection accuaracy on dog data set and human face data set respectively. The final model uses **`inception_v3`** instead, which showed better performance as **`95%`** dog detection accuracy

4. Dog breed detector definition, cell 14 defines a dog breed detector from scratch. It has 3 convolution layers and 2 fully connected layers shown below:

```
Net(
  (layer1): Sequential(
    (0): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (layer2): Sequential(
    (0): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (layer3): Sequential(
    (0): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fc1): Linear(in_features=50176, out_features=200, bias=True)
  (fc2): Linear(in_features=200, out_features=133, bias=True)
  (dropout): Dropout(p=0.3)
)
```


This model achieved around `11%` dog breed detection accuaracy on dog data set

5. Final model definition, The final model is based on `inception_v3`**.** The main reasons to choose this model is: a) size is smaller than VGG, and b) better performance on imagenet data set.

In this model, dog-non-dog classification and dog breed predication are combined into the same inception model by adding one extra fully connected layer called **self.fc_dog_breed** at the output with size (4086,133), in parallel with the original final fully connected output layer **self.fc.** The detailed steps are shown below:

- Saved torchvision/models/inception.py into a local directory where dog_app.ipynb is located with name as **inception_dog.py**
- Load pretrained parameters:

The following lines (line 30 to 41) from file **inception_dog.py** shows how to load all pretrained parameters except new **self.fc_dog_breed** layer's

```
    if pretrained:
        if 'transform_input' not in kwargs:
            kwargs['transform_input'] = True
        model = Inception3(**kwargs)
        pretrained_dict = model_zoo.load_url(model_urls['inception_v3_google'])
        model_dict = model.state_dict()
        for name, param in model_dict.items():
            if name not in pretrained_dict:
                continue
            param = pretrained_dict[name].data
            model_dict[name].copy_(param)
        return model
```

The following line (line 72) from **inception_dog.py** shows how extra layer self.fc_dog_breed is added into **init()** function

```
self.fc_dog_breed = nn.Linear(2048, dog_breed_classes)
```

The following lines (Line 135 to 143) in file **inception_dog.py** shows how the new layers is added in parrallel with the original output layer

```
# 1 x 1 x 2048
x = x.view(x.size(0), -1)
x_dog_breed = self.fc_dog_breed(x)
# 2048
x = self.fc(x)
# 1000 (num_classes)
if self.training and self.aux_logits:
    return x_dog_breed, x, aux
return x_dog_breed, x
```

During training, all layers parameters are freezed except added **self.fc_dog_breed.weight** and **self.fc_dog_breed.bias.** The batch normalization layers running_mean and running_var are also freezed in cell 16 shown below:

```
if pretrained:
    for module in model.modules():
        if isinstance(module, torch.nn.modules.BatchNorm1d):
            module.eval()
        if isinstance(module, torch.nn.modules.BatchNorm2d):
            module.eval()
        if isinstance(module, torch.nn.modules.BatchNorm3d):
            module.eval()
```

If not freezing batch normalization layer's running_mean and running_var, these values will be updated when training model for dog-breed classes. These updated running_mean and running_var will reduce model accuracy when doing dog-non-dog classification from 95% to 81%

Final model's dog breed prediction accuracy is up to __85%__ after 5 epoch training, while dog-non-dog classification accuracy stays the same as before training, **95%**

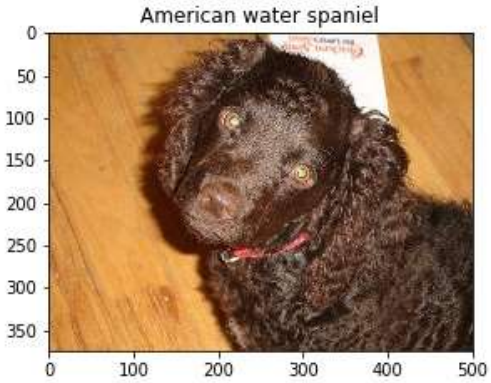
Results:

Performance on dog_images

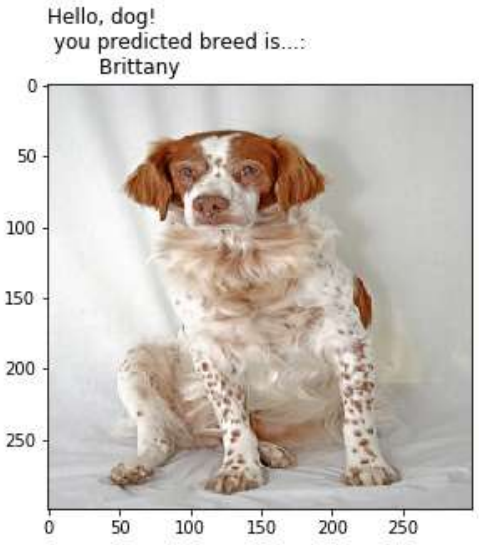
Performance is checked on five images from test dataset, two images got from intenet, and three skatched dog pictures

The following five pictures show correct dog breed predictions on the test data

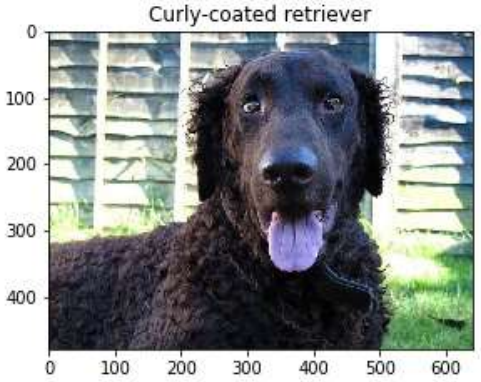
American water spaniel



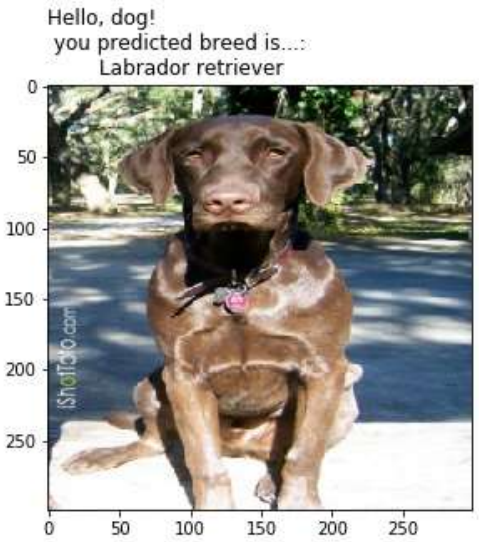
Brittany



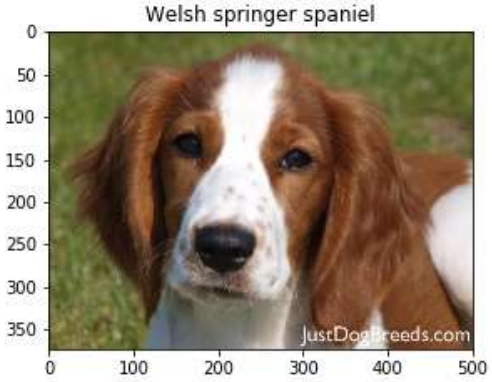
Curly-coated retriever

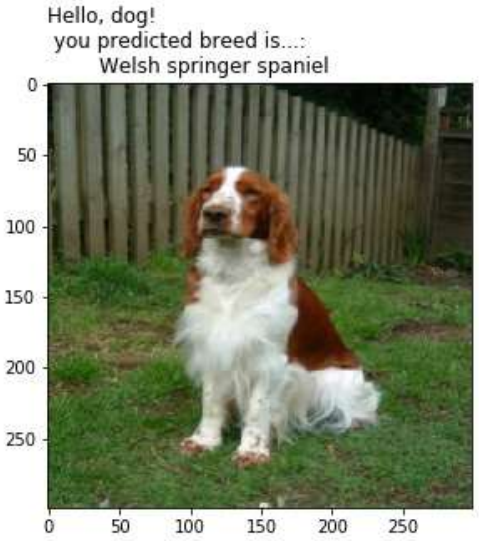


Labrador retriever



Welsh springer spaniel

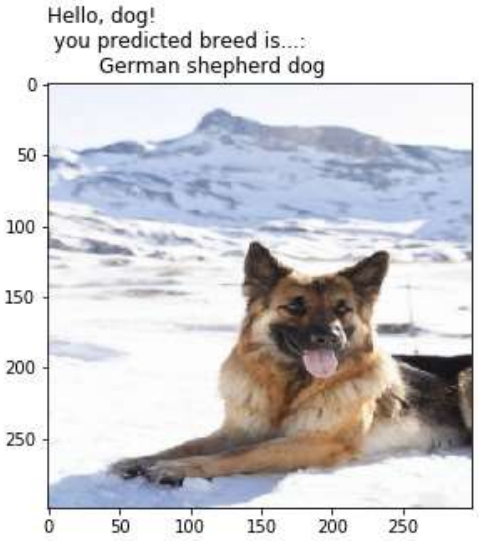




The following two pictures show correct dog breed predictions on pictures got from __internet__

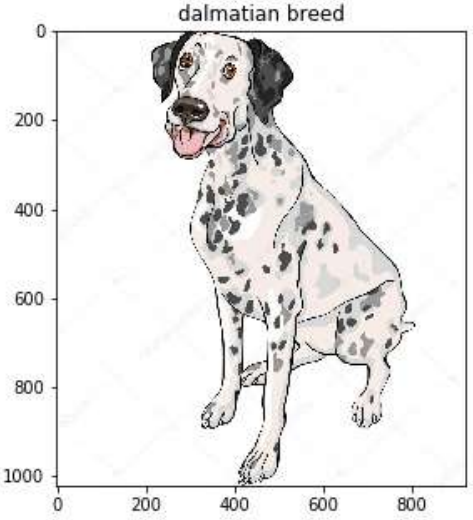
beagle puppy

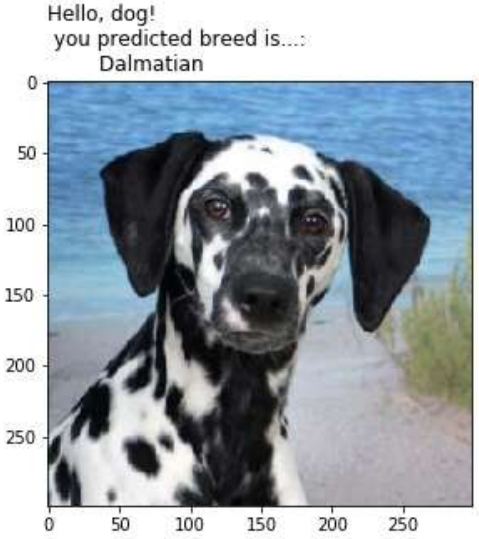
german shepherd



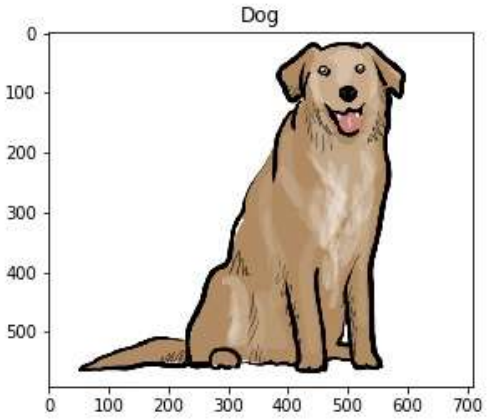
The following three pictures show correct dog breed predictions on __skatched__ dog pictures

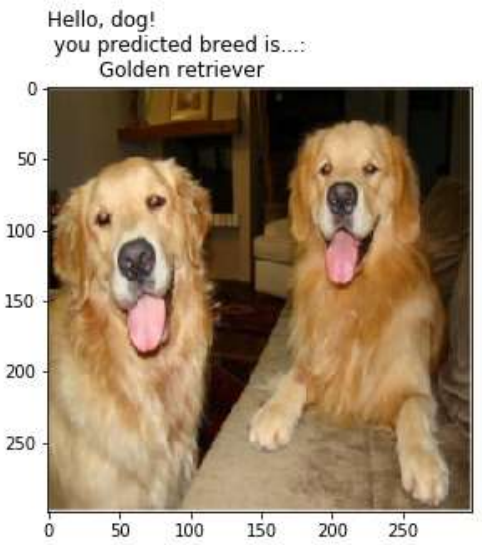
dalmatian breed



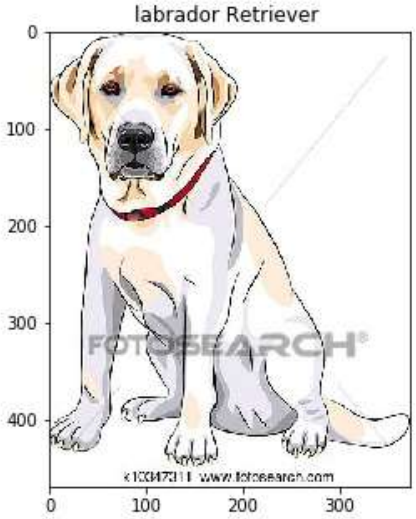


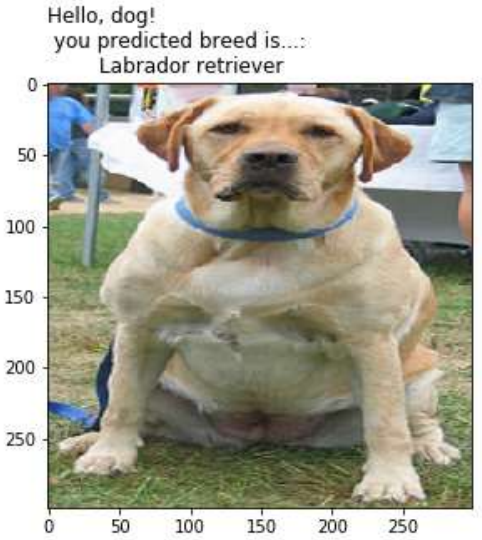
Dog





labrador Retriever



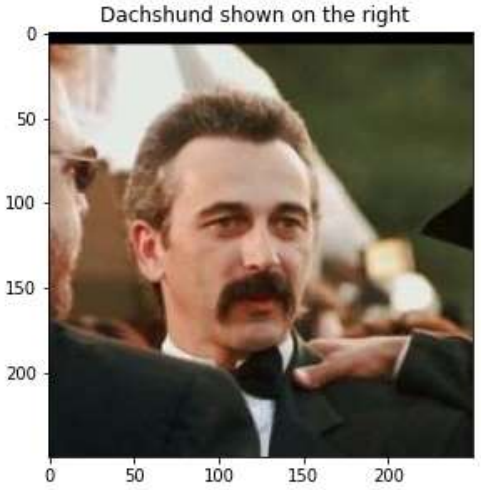


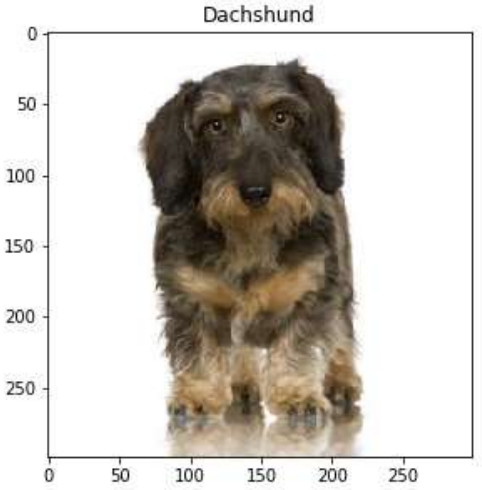
Performance on human_face

Performance is checked on two human face images

Aaron Tippin

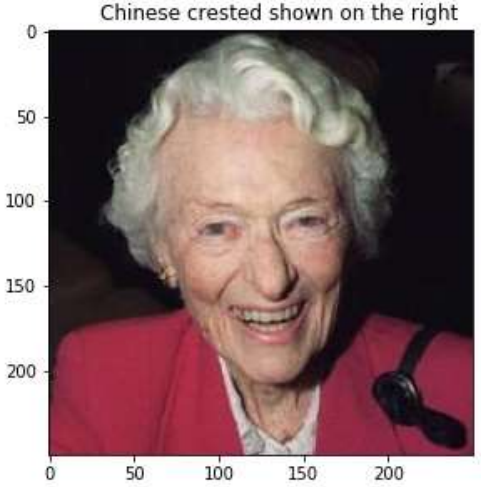
Hello: Aaron Tippin!
You look like a ...

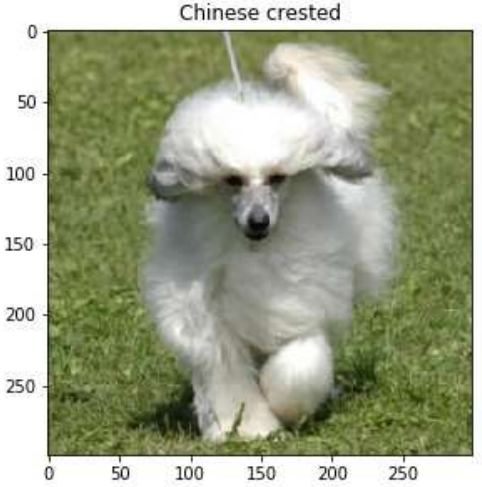




Aileen Riggin Soule

Hello: Aileen Riggin Soule!
You look like a ...





Performance on non human_face or dog images

Performance is checked on five images: 1 car, two cats and two wolves

car