Dog Breed Classifier

REVIEW

HISTORY

Meets Specifications

Congratulations!

You have successfully passed the project!

Files Submitted

The submission includes all required, complete notebook files.

The IPython notebook is rightly submitted.

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.

HaarCascade classifier is used correctly to test the human face existence on human and dog images.

```
98% of human images had face detections
17% of dog images had face detections
```

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 VGG16 model is loaded perfectly. You have coded the transforms rightly.

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

```
100%|██████████| 100/100 [00:07<00:00, 14.24it/s]
Dogs detected in dog files: 100%
Dogs detected in human files: 0%
```

The model is tested perfectly on human and dog images.

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.

The dataloaders for training, validation and testing are precisely implemented. It's great that you have used the `Normalize` transforms. ✅

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

Image augmentation is done correctly. This helps to introduce some randomness and improve the model's learning ability.

The submission specifies a CNN architecture.

Good Job!

The combination of convolution layers with batch normalization, dropout layers and pooling layers are precisely done. I encourage you to read about leaky relu activation function too.

Answer describes the reasoning behind the selection of layer types.

The answers clearly explain the steps taken to code the network.

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

```
import torch.optim as optim

### TODO: select loss function
criterion_scratch = nn.CrossEntropyLoss()

### TODO: select optimizer
optimizer_scratch = optim.SGD(model_scratch.parameters(), lr=0.002, momentum=
0.9)
```

The CrossEntropyLoss function and SGD optimizer are used correctly.

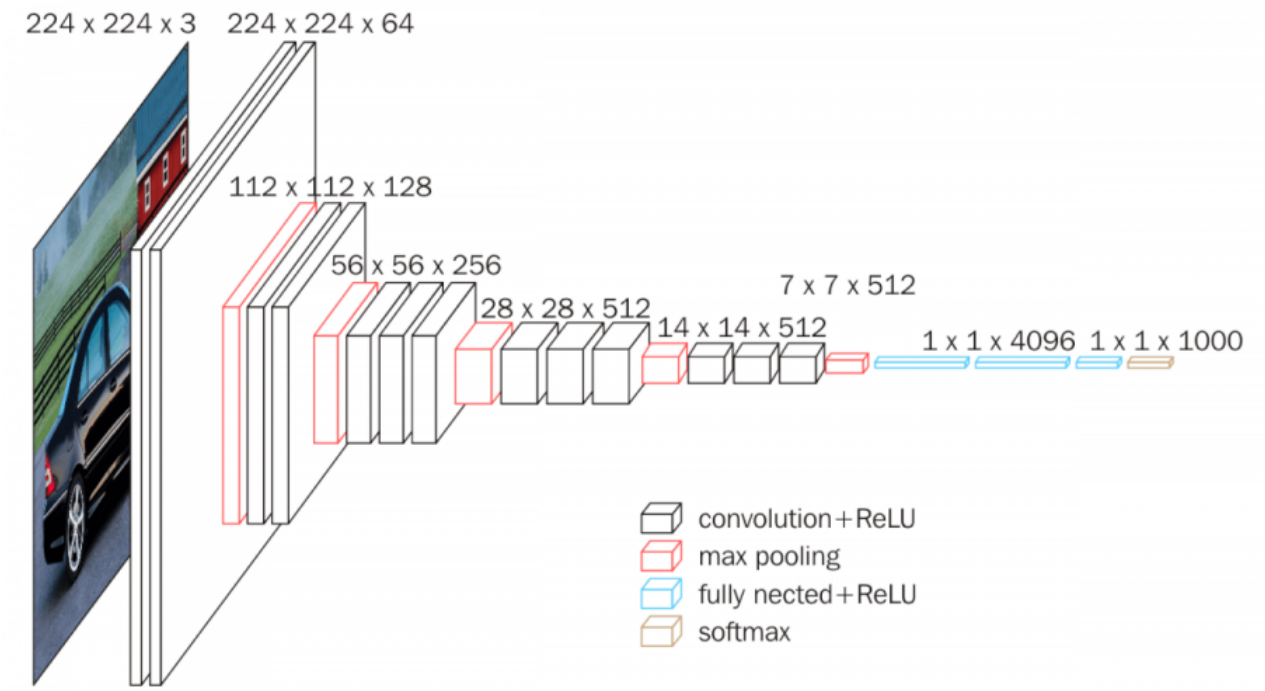
The trained model attains at least 10% accuracy on the test set.

Amazing!

The model is getting trained perfectly. It's giving an accuracy of around 19%. ✌🏾

Step 4: Create a CNN Using Transfer Learning

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



The VGG model is precisely loaded. You have replaced the final fully-connected layer with a new Fully connected layer.

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

It would be better if you add some comparison between the VGG16, ResNet50 and other models.

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

Training the model for 10 epochs is acceptable. Although I encourage you to try increasing it and experiment with a convergence value of epochs. There's a tradeoff of using a small as well as a large number of epochs. With each epoch, the model gets the opportunity to improve.

Accuracy on the test set is 60% or greater.

```
Test Accuracy: 73% (614/836)
```

The accuracy of model is 74%. ⭐

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.

The `predict_breed_transfer` function takes the image as input and predicts the breed.

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.

Step 6: Test Your Algorithm

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

Fantastic!

The model has been tested on a diverse set of images. It's classifying the images correctly. 👏🏼

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

I highly recommend you to try implementing all the possible options. You can discuss these pointers over Udacity's study hub too.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this project