‹ Return to "Deep Learning" in the classroom

Dog Breed Classifier

REVIEW | HISTORY

Meets Specifications

Hi,
Good work! You passed the dog breed classifier project.
Some additional AI resources:
My post about School of AI NDs
9 Deep Learning papers
MIT AGI: Building machines that see, learn, and think like people
YOLO Object Detection
⭐New Deep Reinforcement Learning Nanodegree⭐
The AlphaGo Movie

Keep up learning!
Sincerely
Leticia

Files Submitted

✓ **The submission includes all required, complete notebook files.**

You submitted all the required 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.**

You applied the face detector over dog and human images

```
Percentage from human_files_short with detected human fac
e: 98.00%
Percentage from dog_files_short with detected human face:
17.00%
```

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).**

You created a Dog Detector using a VGG16 model

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

You applied the dog_detector over dog and human images

```
Percentage from human_files_short with detected dog: 0.00
%
Percentage from dog_files_short with detected dog: 100.00
%
```

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.**

You used three different transformations for the training, test, and validation sets.

```
data_transforms = {'train': transforms.Compose([transform
s.RandomResizedCrop(224),

transforms.RandomHorizontalFlip(),

transforms.ToTensor(),

standard_normalization]),
                   'val': transforms.Compose([transforms.
Resize(256),

transforms.CenterCrop(224),

transforms.ToTensor(),

standard_normalization]),
                   'test': transforms.Compose([transforms
.Resize(size=(224,224)),

transforms.ToTensor(),

standard_normalization])
                  }
```

Great work using Data Augmentation for the training set.

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

You answered the question 3. Good idea using 224x224 images size.

✓ **The submission specifies a CNN architecture.**

You specified a CNN for dog breed detection.
You used Dropout layers for preventing overfitting. Great!
Consider trying to use BatchNormalization layers to normalize the input between layers.
https://discuss.pytorch.org/t/example-on-how-to-use-batch-norm/216

✓ **Answer describes the reasoning behind the selection of layer types.**

You answered the question 4.
Very detailed answer. Excellent work!

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

You used CrossEntropyLoss loss and RMSProp optimizer.
Consider trying Adam optimizer.

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

Test Accuracy: 18% (154/836) great

Step 4: Create a CNN Using Transfer Learning

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

You used resnet50 pre-trained model
More information: http://torch.ch/blog/2016/02/04/resnets.html

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

You answered the question 5.
This answer could include more information. The idea is to explain how transfer learning works.
Remember that we are using resnet50 and redefining the final layers.
More information: http://ruder.io/transfer-learning/

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

You trained the model for 33 epochs.

✓ **Accuracy on the test set is 60% or greater.**

Test Accuracy: 78% (653/836)
great

✓ **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.**

You wrote predict_breed_transfer function for meeting specifications.

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.**

You wrote the required algorithm

Step 6: Test Your Algorithm

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

You tested the algorithm over more than 6 images including two human and two dog images.

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

You answered the question 6. Good enough but this idea:

> incrase the overall performance of the model

Could be replaced for train for more epochs for example.
More ideas: https://machinelearningmastery.com/improve-deep-learning-performance/

DOWNLOAD PROJECT

RETURN TO PATH