

[◀ Return to "Deep Learning" in the classroom](#)[DISCUSS ON STUDENT HUB](#)

Dog Breed Classifier

REVIEW

CODE REVIEW

HISTORY


Meets Specifications

Brilliant Udacity Learner,

Congratulations! 🎉

You've successfully passed all the specification in this project submission. I must admit that the structure of this project implementation is impressive!

You should be proud of the work done as you seem to have a good understanding towards building a Convolution Neural Network and also you are familiar with using PyTorch.

It was my pleasure reviewing this wonderful project. Please continue with this same spirit of hard work in the projects ahead. 

Extra Material

- [Convolutional Neural Network With PyTorch](#)
- [How To Improve Deep Learning Performance](#)
- [Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates](#)

Files Submitted

The submission includes all required, complete notebook files.

Great Work 👍

Your submission consists of everything that is required.

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.

Good Job ✨

The Harr-Cascade Filters used in this code give `96% Humans as Humans` and `18% Dogs as Humans`

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 implementation is just the way it was expected to be. Great Job here 👍
Simplifying the code in python, instead of

```
if result >= 151 and result <= 268:
    return True
else:
    return False
```

We can use,

```
return 151 <= result <= 268
```

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

The pre-trained VGG16 model gives `95% Dogs as Dogs` and `0% Humans as Dogs`

Nice job. This is nearly the same as the expected result. ✨

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 defined are great. 👍

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

The answer correctly describes about the Data Pre-Processing. AWESOME 🌟

The submission specifies a CNN architecture.

Great Work 👍

The model created from scratch was small and shallow, so this wouldn't have took much time to train. But we can certainly improve this model by making it dense to capture more features.

Answer describes the reasoning behind the selection of layer types.

The answer briefly explains about the architecture of the model in a very curated manner. Nice and Clean. Good Work here.

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

Excellent ⭐

`CrossEntropyLoss` and `Stochastic Gradient Descent`. Just the way it was required 👍
It is seen that in this case SGD often works better than Adam

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

Amazing 🙌

Awesome. 24% test accuracy is superb.

Step 4: Create a CNN Using Transfer Learning

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

Great, so you used the pre-trained VGG19 for this classification. Its nice that you have also printed the model for understanding the outputs well. Well Done.

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

The answer provided was sufficient enough as a reason to the approach. Good

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

Good Work. The model was trained properly on `20 epochs` . But it is seen that this pre-trained model can also perform well if you train the Fully Connected layers for some 10 epochs or so

Accuracy on the test set is 60% or greater.

Well done with the outputs.

Test Loss: 0.510275

Test Accuracy: 83% (701/836)

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.

Awesome 🍷

The `predict_breed_transfer` method is defined correctly.

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.

Good job implementing the application for predicting dog breed from the images. The usage of previously defined functions is done very well. Nice and Clean Code. Good Job 👍

Step 6: Test Your Algorithm

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

The submission tests at least 6 Images for verification of the model 👍

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

You have included the points which can be used to improve the model.

 [DOWNLOAD PROJECT](#)

[RETURN TO PATH](#)

Rate this review