UDACITY

Return to "Deep Learning" in the classroom

DISCUSS ON STUDENT HUB

# Dog Breed Classifier

REVIEW

HISTORY

## Meets Specifications

## Congratulations!

This is an outstanding submission. You have coded things pretty well. I have added some suggestions to help you further improve the results.

I highly encourage you to keep reading some of the blogposts, research papers. We are in deep learning domain and new research is happening on daily basis, so it's really challenging to keep ourselves updated.

The COVID19 is one of the topics on which researchers have started working. Many labs are trying to automate the testing by Chest Xrays. The CNNs play an important role in it. They are classifying the chest X-ray images and finding patterns in COVID and Non-COVID patients. You can read about this via these links:

https://medium.com/@sheldon.fernandez/covid-net-an-open-source-neural-network-for-covid-19-detection-48b8a55e6d44
https://confengine.com/odsc-india-2020/proposal/14473/detection-of-covid-19-patients-using-deep-convolutional-neural-networks-on-chest-x-ray-images

Keep Learning! Deep Learning!

### Files Submitted

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

All the required set of files are 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.**

```
Human Images Detected with a Face :  99.0
Dog images detected with the face :  7.000000000000001
```

The face detector has been tested on both human and dog images correctly. It's great that you have used other classifiers too.

## 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 VGG model is loaded rightly. You have coded the transforms and other steps correctly.

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

```
Human Images Detected with a Face :  0.0
Dog images detected with the face :  96.0
```

The model tests the performance of VGG on both human and dogs data.

## Step 3: Create a CNN to Classify Dog Breeds (from Scratch)

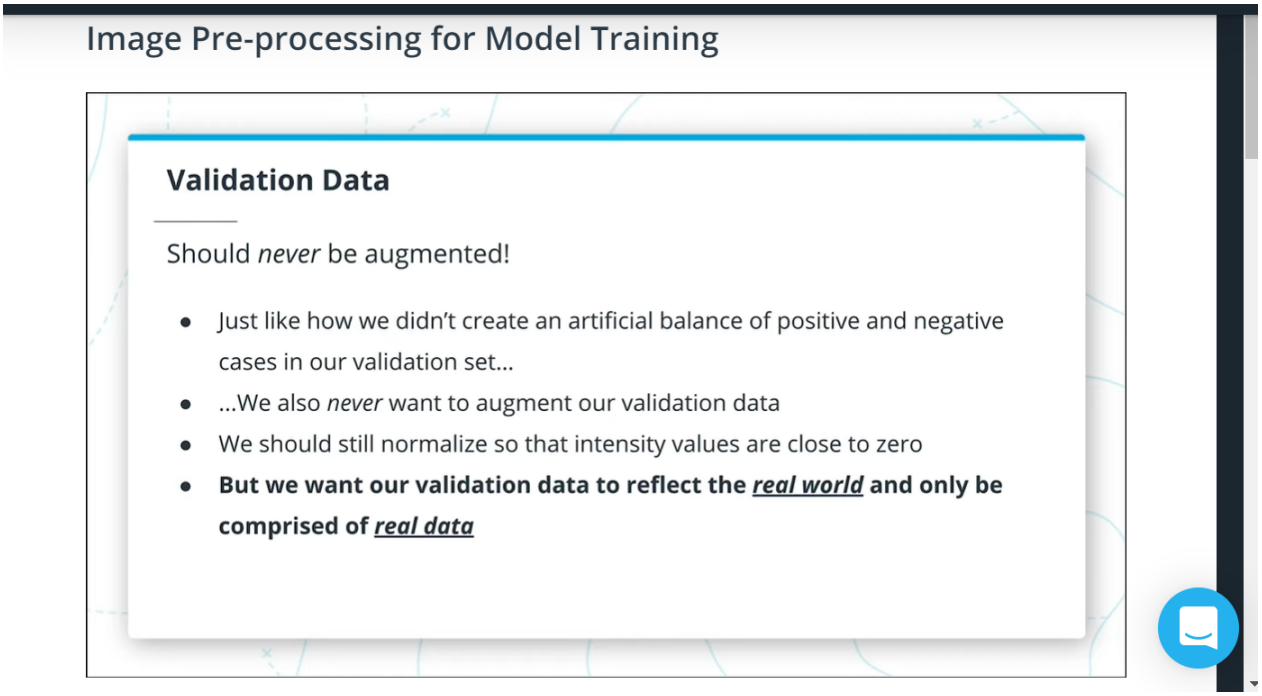
**Write three separate data loaders for the training, validation, and test datasets of dog images. These images should be pre-processed to be of the correct size.**

## Awesome!

The training, testing and validation transforms are well coded. You have correctly used the augmentation techniques with the training transform.

## Required

It's suggested not to use any augmentation technique while inferencing the model. You should not use augmentation techniques with validation and testing transforms.




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

The answer mentions the steps chosen to perform the preprocessing.

**The submission specifies a CNN architecture.**

The CNN is well coded. All the steps are rightly coded:

Convolution layers are used correctly with ReLU and max-pooling layers.
Fully connected layers with dropout are chosen correctly. The dropout layers reduce the chances of overfitting.

## Suggestion

You can further try using the Batch Normalization technique: https://www.learnopencv.com/batch-normalization-in-deep-networks/

While coding the convolution network, we can consider initializing the weights. This helps to improve the performance of model. You can go through this Q&A: https://discuss.pytorch.org/t/how-to-initialize-the-conv-layers-with-xavier-weights-initialization/8419

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

Perfect!

The answer explains the architecture of Conv Net.

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

The loss function and optimizer are used correctly.

### Suggestion

The learning rate as 0.1 seems quite high. I encourage you to use a lower learning rate.

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

```
Test Accuracy: 14% (122/836)
```

The network is perfectly trained. It's able to fetch us an accuracy of 14%.

## Step 4: Create a CNN Using Transfer Learning

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

As suggested above, consider working on the transforms.

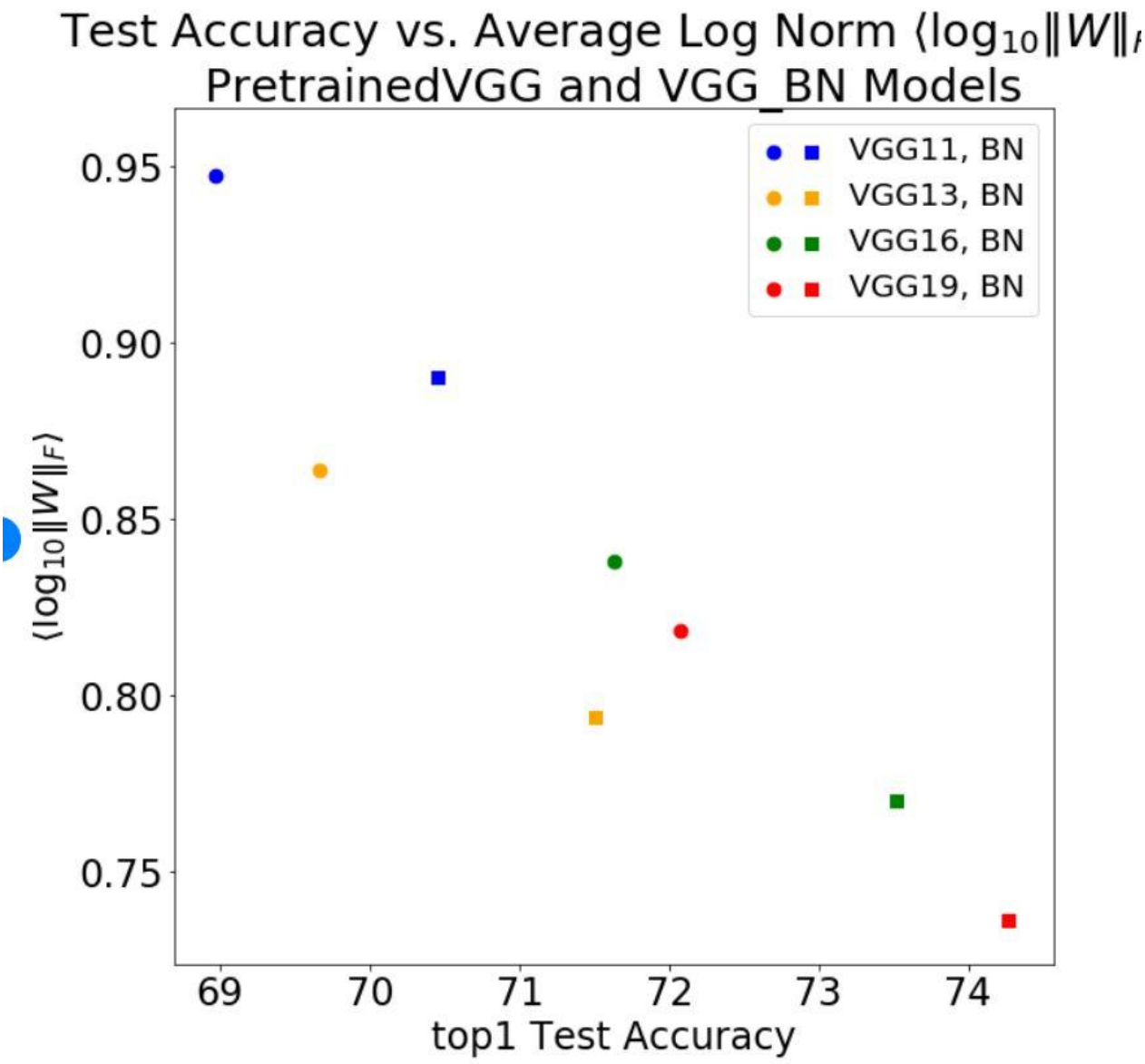
The VGG19 is loaded precisely. You have replaced the final fully connected layer with a new layer.

```
n_input = model_transfer.classifier[6].in_features
last_layer = nn.Linear(n_input ,133)
model_transfer.classifier[6] = last_layer
```

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

## Great!

The answer justifies the reason of using VGG19. You can also consider using VGG with Batch Normalization:




source: https://www.researchgate.net/figure/Pre-trained-VGG-and-VGG-BN-Architectures-and-DNNs-Top-1-Test-Accuracy-versus-average-log_fig3_330638379

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

Training of the model for 35 epochs provides enough opportunity to help it learn.

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

### Perfect!

```
Test Accuracy: 85% (715/836)
```

The network is well trained. Test accuracy as 85% looks 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.**

The predict_breed_transfer function is precisely coded. All the steps are rightly implemented:

- The transforms are coded correctly to process the image before feeding it to the model.
- It predicts the breed name using the predicted index.

### Suggestion

Please consider changing the mode of model to evaluation mode before using it for testing. You can read about it by checking these Q&As: link link2

## 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.**

The run_app() method is perfectly coded. It uses the dog_detector, face_detector and predict_breed_transfer to predict the class of images.

## Step 6: Test Your Algorithm

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

It's great that you have tested with a diverse set of images. You have tried various breed dogs to test the model. Although using the images that are in `dogImages` i.e, our dataset can lead us to get biased results.

As model has already seen those images. This can also lead to data leakage.
So, it's always suggested to test the model on a new set of images to observe it's actual performance. 😄

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

Great observations!

The previous reviewer also suggested some great things. I encourage you to try working on these to further improve the performance.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review