UDACITY

PROJECT

Dog Breed Classifier

A part of the Artificial Intelligence Program

PROJECT REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

Hi there,

It was a pleasure reviewing your project.
You showed passion and dedication for this project. Those are excellent skills that any employee would like the employees to have. Congratulations!!
I've found some detail missed in the algorithm implementation, check the review below.
You passed the *Dog Breed Classifier* project.

Keep up the impressive work you are doing in this ND!
Sincerely,
Leticia
You can reach me for any question on this review: @letyrodri (at Slack)

Files Submitted

The submission includes all required files.

You submitted all the requested files.

Step 1: Detect Humans

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

You applied the dog detector over dog and human images.

The submission opines whether Haar cascades for face detection are an appropriate technique for human detection.

Excellent answer! For additional information check: Modern Face Recognition with Deep Learning

Step 2: Detect Dogs

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

You applied the dog detector over dog and human images.

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

The submission specifies a CNN architecture.

Excellent work you used DropOut layers to prevent overfitting and BatchNormalization layers to speed up training and increase accuracy.
You could use more than one of these layers. BatchNormalization is used to normalize the output data in each layer.

The submission specifies the number of epochs used to train the algorithm.

Impressive. You trained your model for 25 epochs with data augmentation and 20 epochs without data augmentation.

The trained model attains at least 1% accuracy on the test set.

TOTALLY IMPRESSIVE. Usually, a good submitted model is around 7~13%. You made data augmentation work on this project.

> Without data augmentation : Test accuracy: 12.0813%
> With data augmentation : Test accuracy: 26.1962%

It's probable that if you use some more batchnormalization layers and dropout, the accuracy could be increased.

Step 5: Create a CNN to Classify Dog Breeds

The submission downloads the bottleneck features corresponding to one of the Keras pre-trained models (VGG-19, ResNet-50, Inception, or Xception).

You used ResNet50 bottleneck features. Excellent choice.

The submission specifies a model architecture.

You used a simple model for transfer learning. Notice the difference with the previous model. We are going to achieve impressive accuracies re-using others work.

The submission details why the chosen architecture succeeded in the classification task and why earlier attempts were not as successful.

Very good answer. You have provided a very detailed explanation of the different approaches you have tried.

The submission compiles the architecture by specifying the loss function and optimizer.

You used categorical_crossentropy loss and rmsprop optimizer.
It's probably that for inceptionV3, you can get higher accuracies trying some other optimizers.
Even with ResNet. One recommendation is to use Adam
But other could work very well, check : Optimizers in Keras

The submission uses model checkpointing to train the model and saves the model weights with the best validation loss.

You used `save_best_only=True` to save the best model.

The submission loads the model weights that attained the least validation loss.

You loaded the saved weights.

Accuracy on the test set is 60% or greater.

Test accuracy: 82.1770% good work.
Val_loss and loss are still decreasing by epoch 15. Maybe you can train it for more epochs.

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 correctly implemented this requirement in the function `Resnet50_predict_breed`.

Step 6: Write Your Algorithm

The submission uses the CNN from Step 5 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 have implemented an algorithm that meets the specifications:

> if a dog is detected in the image, return the predicted breed.
> if a human is detected in the image, return the resembling dog breed.
> if neither is detected in the image, provide output that indicates an error.

It's in the method `custom_dog_breed_classificator`.

Notice that the output of the algorithm could be improved. When it's not a dog or a human, a special message needs to be displayed indicating that neither a human or a dog was detected.

Step 7: Test Your Algorithm

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

You answered `Question 6`. Excellent answer. Good link explaining complex practices.
You made such a great work but the images you tested on weren't submitted and they are not displayed in the notebook. This rubric point is not complete but taking into account all the extra work you have done, it will meet specifications. My recommendation is to consider include the images in the notebook to have a perfect work in your portfolio.
You can display the images inline in the notebook. You will need to first upload the images to the AWS if you are using it. You can use scp or winscp to do it.
Then, show the images using this code similar to this one:

```
img = cv2.imread(img_path)
cv_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
imgplot = plt.imshow(cv_rgb)
```

In your tested samples, there aren't cases where neither a dog or a person was detected. It could be an issue on the test images or it is just the default in the algorithm return the "Dog" message.

DOWNLOAD PROJECT

RETURN TO PATH

Student FAQ