UDACITY

PROJECT

Dog Breed Classifier

A part of the Deep Learning Nanodegree Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

Awesome! Congratulations on completing this project. Your understanding of CNN concepts is very good. All the best 👍

For more knowledge on CNNs, refer:
More about CNNs
Transfer Learning
CNN Tricks
Creating an Image classifier

Files Submitted

The submission includes all required files.

All files correctly submitted!

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.

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

Correct reasoning!

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.

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

The submission specifies a CNN architecture.

```
  model = Sequential()
  ### TODO: Define your architecture.
 model.add(Conv2D(filters=16, kernel_size=2, padding='same', activation= 'rel
 u',  input_shape=(224, 224, 3))) model.add(MaxPooling2D(pool_size=2))
 model.add(Conv2D(filters=32, kernel_size=2, padding='same', activation= 'rel
 u')) model.add(MaxPooling2D(pool_size=2))
 model.add(Conv2D(filters=64, kernel_size=2, padding='same', activation= 'rel
 u')) model.add(MaxPooling2D(pool_size=2))
 model.add(Dropout(0.4))
 model.add(Flatten()) model.add(Dense(500, activation='relu'))
 model.add(Dropout(0.5)) model.add(Dense(133, activation='softmax'))
```

Good job creating a CNN model from scratch. The given architecture is good, great job giving the detail explanation about the chosen model and the design constraints you considered.

You have correctly used conv layers with max pooling, you can additionally use batch normalisation and dropout to improve your model.

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

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

```
Test accuracy: 9.5694%
```

Looks good!

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).

ResNet is a great choice, do try out other models as well.

The submission specifies a model architecture.

Good job giving brief description on your model architecture. The pre-trained model itself does most of the heavy-lifting, so it is good to have a simple model like this.

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

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

I would suggest you to try out other optimizers like adam, adagrad too: link, link2

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

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

Accuracy on the test set is 60% or greater.

Well done!

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.

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.

Good job implementing the application for predicting dog breed from the images. The usage of previously defined functions is done very well.

Step 7: Test Your Algorithm

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

Great job testing the images and suggesting the improvements.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

Student FAQ