dog_breed

December 15, 2017

In [ ]:
```
%matplotlib inline
import matplotlib.pyplot as plt
```

In [ ]:
```
import numpy as np
import pandas as pd
import os
from keras.preprocessing import image
```

In [ ]:
```
from keras.applications.vgg16 import VGG16, preprocess_input, decode_predictions
from tqdm import tqdm
from keras.models import Model
from keras.layers import Dropout, Flatten, Dense
from keras import optimizers
from keras.callbacks import EarlyStopping, ReduceLROnPlateau
```

In [ ]:
```
data_dir = '/Users/alexeydemyanchuk/Kaggle/DogBreedIdent'
```

In [ ]:
```
# dataframes for training and testing
labels = pd.read_csv('labels.csv')
sample_submission = pd.read_csv('sample_submission.csv')
```

In [ ]:
```
SEED = 1987
INPUT_SIZE = 64
```

In [ ]:
```
# print(len(os.listdir(os.path.join(data_dir, 'train'))), len(labels))
# print(len(os.listdir(os.path.join(data_dir, 'test'))), len(sample_submission))
```

In [ ]:
```
# train/valid split indecise
np.random.seed(seed=SEED)
rnd = np.random.random(len(labels))
train_idx = rnd < 0.8
valid_idx = rnd >= 0.8
```

In [ ]:
```
# from categorical to one hot
selected_breed_list = list(labels.groupby('breed').count().sort_values(by='id', ascendin
labels['target'] = 1
labels['rank'] = labels.groupby('breed').rank()['id']
labels_pivot = labels.pivot('id', 'breed', 'target').reset_index().fillna(0)
```

```
In [ ]: y_train = labels_pivot[selected_breed_list].values
        ytr = y_train[train_idx]
        yv = y_train[valid_idx]

In [ ]: print(ytr.shape, yv.shape)

In [ ]: # helper to read, resize image and convert it to numpy array
        def read_img(img_id, train_or_test, size):
            """Read and resize image.
            # Arguments
                img_id: string
                train_or_test: string 'train' or 'test'.
                size: resize the original image.
            # Returns
                Image as numpy array.
            """
            img = image.load_img(os.path.join(data_dir, train_or_test, '%s.jpg' % img_id), targe
            img = image.img_to_array(img)
            return img

In [ ]: # process all training images
        x_train = np.zeros((len(labels), INPUT_SIZE, INPUT_SIZE, 3), dtype='float32')
        for i, img_id in tqdm(enumerate(labels['id'])):
            img = read_img(img_id, 'train', (INPUT_SIZE, INPUT_SIZE))
            x = preprocess_input(np.expand_dims(img.copy(), axis=0))
            x_train[i] = x
        print('Train Images shape: {} size: {:,}'.format(x_train.shape, x_train.size))

In [ ]: # split training data in training and validation sets
        Xtr = x_train[train_idx]
        Xv = x_train[valid_idx]
        print(Xtr.shape, Xv.shape)

In [ ]: # importing pretrained model not including FC layers for imagenet classification
        model = VGG16(weights="imagenet", include_top=False, input_shape=(INPUT_SIZE, INPUT_SIZE
        model.summary()

In [ ]: len(model.layers)

In [ ]: # froze layers from pretrained
        for layer in model.layers[:19]:
            layer.trainable = False

In [ ]: # connecting custom layers to classify dog breeds
        x = model.output
        x = Flatten()(x)
        x = Dense(1024, activation="relu")(x)
        predictions = Dense(ytr.shape[1], activation="softmax")(x)

In [ ]: model_final = Model(inputs=model.input, outputs=predictions)
```

```
In [ ]: model_final.compile(loss="categorical_crossentropy", optimizer=optimizers.SGD(lr=0.0001,

In [ ]: model_final.summary()

In [ ]: callbacks = [EarlyStopping(), ReduceLROnPlateau()]

In [ ]: model_final.fit(Xtr, ytr, validation_data=(Xv, yv), batch_size=64,
                       epochs=50, callbacks=callbacks,verbose=1)

In [ ]: filepath = os.path.join(data_dir, "vgg16_imgnet_pretrained_50epochs.h5")
        model_final.save_weights(filepath)
```