CNN classifier for dog breed identification

Carlos Alberto Cuentas Bernal (ccuentas@academia.usbbog.edu.co),
John Edisson Guzmán Cáceres (jeguzman@academia.usbbog.edu.co)

Universidad san buenaventura

***Abstract*—The following work proposes the construction of a classification algorithm for different dog breeds based on images downloaded from flickr using the python programming language and convolutional neural networks (CNN), as well as applying the keras and tensorflow libraries.**

I. Introduction

The convolutional neural networks or better known as CNN are a type of neural network trained under supervisión, which processes multiple layers for the identification of characteristics, patterns, etc. Of images th us mimicking the visual cortex if the human eye for image processing.

Currently CNNs have been widely adopted in the are of artificial visión, so that through these neural networks for the processing of images from the flickr API of different dog breeds it is desired to classify them depending on their characteristics.

II. Methods

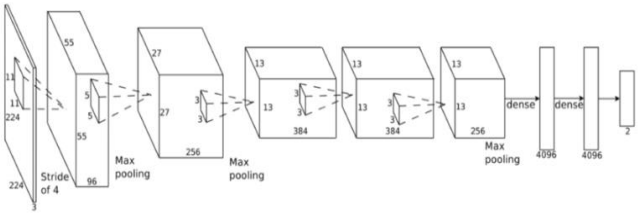
Next, the process carried out for the construction of the classifier is explained.

1. The Flickr API configuration is done by downloading the Flickrapi package and from there importing the packages for downloading and configuring the directories in which the public and secret keys provided by Flickr are to be entered. We must ensure that the input parameters are of the correct type or value, otherwise indicate to the user that he has made a mistake, also determine the direction from which the images should be downloaded.
2. Connect with the Flickr API and créate subfolders by breed of dogs in which the images will be inserted at the time of classification. We enter the Flickr library and look for the images through keywords and image size.
3. Obtain the specific address to download the image, also créate the file names of the images. Later we apply the increase of image data.
4. Import the keras library to perform the transformation to the images within are rotation, width change, rescale, cut, focus, horizontal flippling and there is also tha possibility of combining all the aforementioned transformations. Here are some examples:

5. Create a specific subfolder structure for image storage and create the generators.
6. Build the deep learning model which is composed of several layers for the convolutional neural network among which are ReLu layer, Pooling, Dropout, fully connected layer and sigmoid activation.
7. Define the convolutional neural network structure by importing the keras libraries to add the layers previously made, then compile the network structure and establish loss parameters, generators and data augmentation. Finally adjust the training data and evaluate with the validation data of the convolutional neural network.

Structure of CNNs

For the structure of the convoluted neural network, we sought to use Alexnet where you want to group several times and put dense layers connected, likewise the grouping matrices are decreasing in size.



A Convolutional Neural Network (CNN) architecture has three main parts:

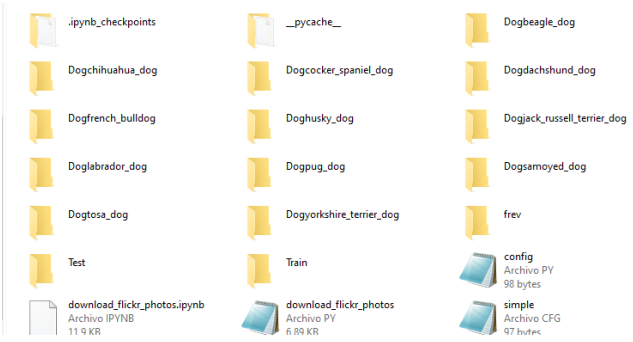
A convolutional layer that extracts features from a source image. Convolution helps with blurring, sharpening, edge detection, noise reduction, or other operations that can help the machine to learn specific characteristics of an image.

A pooling layer that reduces the image dimensionality without losing important features or patterns.

A fully connected layer also known as the dense layer, in which the results of the convolutional layers are fed through one or more neural layers to generate a prediction.

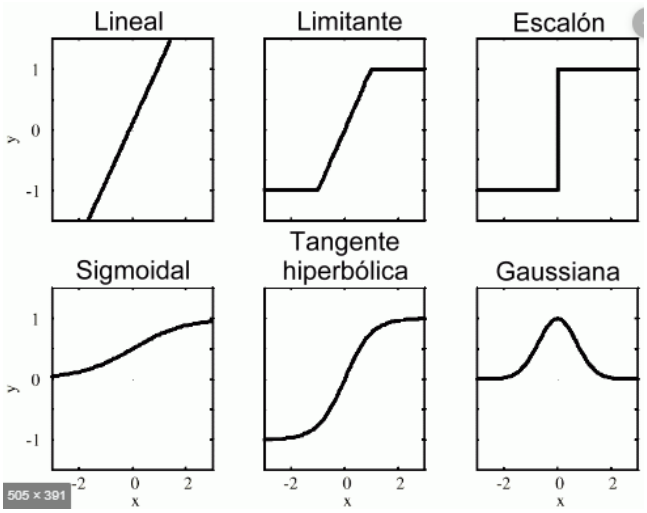
Dataset folders

In the download folders 12 folders were generated where in each one the 750 images of each race were downloaded, which from each folder 70% were extracted for training.



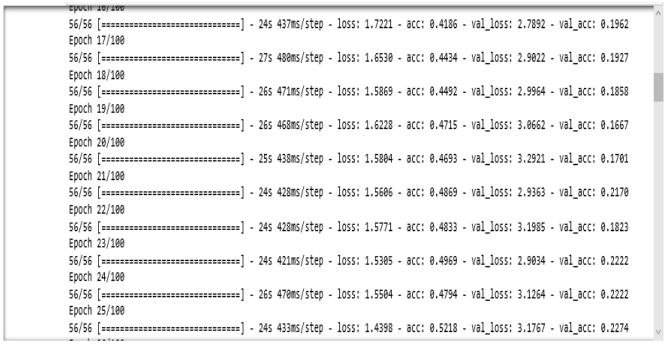
Activation functions

Within the convolution we handle the sigmoid activation function however to categorically transform the images it is recommended to use a different function, in this case a step type



III. Results

After running 100 times, a training accuracy of 22.22% and validation accuracy of 49.69% were obtained. Therefore, it can be concluded that the neural network still has enough over-adjustments, and the accuracy varies a lot, this is due to the small set of validation simples apart from being quite far from the desired accuracy.



IV. Conclusions

- With the help of convolutional neural networks, more robust and simpler image processing can be achieved unlike conventional algorithms
- By using the functions of keras with the convoluted images, we can generate others which help us avoid overfilling and therefore have greater precisión in the convoluted filter
- Depending on the order and type of layers and filters that are available, a better result can be genrated when training convoluted neural networks.

References

- https://medium.com/datos-y-ciencia/construye-tu primer-clasificador-de-deep-learning-con-tensorflow-ejemplo-de-razas-de-perros-ed218bb4df89
- https://www.aprendemachinelearning.com/clasificacio n-de-imagenes-en-python/
- https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
- https://towardsdatascience.com/introducing-convolutional-neural-networks-in-deep-learning-400f9c3ad5e9
- https://www.edureka.co/blog/convolutional-neural-network/

Code 1 CNN

```
1) # Copyright 2014-2017 Bert Carremans
2) # Author: Bert Carremans <bertcarremans.be>
3) #
4) # License: BSD 3 clause


[1] import tensorflow as tf
[2] from tensorflow.keras.models import Sequential
[3] from tensorflow.keras.layers import Conv2D, MaxPooling2D
[4] from tensorflow.keras.layers import Activation, Flatten, Dense, Dropout, MaxPool2D
[5] from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
[6] import os
[7] import matplotlib.pyplot as plt
[8]
import time

[9] IMG_SIZE = 75
[10] NB_CHANNELS = 3
[11] BATCH_SIZE = 32
[12] NB_TRAIN_IMG = 1800
[13] NB_VALID_IMG = 600
os.getcwd()

[14] train_datagen = ImageDataGenerator(
[15]     rotation_range = 40,
[16]     width_shift_range = 0.2,
[17]     height_shift_range = 0.2,
[18]     rescale = 1./255,
[19]     shear_range = 0.2,
[20]     zoom_range = 0.2,
[21]     horizontal_flip = True)
[22]
[23] validation_datagen = ImageDataGenerator(rescale = 1./255)
[24]
[25] train_generator = train_datagen.flow_from_directory(
[26]     '../flickr/Train',
[27]     target_size=(IMG_SIZE,IMG_SIZE),
[28]     class_mode='categorical',
[29]     batch_size = BATCH_SIZE)
[30]
[31] validation_generator = validation_datagen.flow_from_directory(
[32]     '../flickr/Test',
[33]     target_size=(IMG_SIZE,IMG_SIZE),
[34]     class_mode='categorical',
[35]     batch_size = BATCH_SIZE)
[36]
[37] img = load_img('../flickr/Dogtosa_dog/19.jpg')  # this is a PIL image
[38] x = img_to_array(img)  # this is a Numpy array with shape (3, 150, 150)
[39] x = x.reshape((1,) + x.shape)  # this is a Numpy array with shape (1, 3, 150, 150)
[40]
[41] # the .flow() command below generates batches of randomly transformed images
[42] # and saves the results to the `preview/` directory
[43] i = 0
[44] for i, batch in enumerate(train_datagen.flow(x, batch_size=1,
[45]                           save_to_dir='../flickr/Dogtosa_dog', save_format='jpeg')):
[46]     i += 1
[47]     if i > 20:
        break  # otherwise the generator would loop indefinitely

[48] cnn = Sequential()
[49] cnn.add(Conv2D(filters=32,
[50]                 kernel_size=(2,2),
[51]                 strides=(1,1),
[52]                 padding='same',
[53]                 input_shape=(IMG_SIZE,IMG_SIZE,NB_CHANNELS),
[54]                 data_format='channels_last'))
[55] cnn.add(Activation('relu'))
[56] cnn.add(MaxPooling2D(pool_size=(2,2),
[57]                       strides=2))
[58] cnn.add(Dropout(0.4))
[59] cnn.add(Conv2D(filters=64,
[60]                 kernel_size=(2,2),
[61]                 strides=(1,1),
[62]                 padding='valid'))
[63] cnn.add(MaxPooling2D(pool_size=(2,2),
[64]                       strides=2))
[65] cnn.add(Flatten())
[66] cnn.add(Activation('step'))
[67] cnn.add(Dense(64))
[68] cnn.add(Activation('relu'))
[69] cnn.add(Dropout(0.4))
[70] cnn.add(Activation('relu'))
[71] cnn.add(Dense(12))
[72] cnn.add(Activation('step'))
[73]
[74] cnn.compile(loss = 'categorical_crossentropy',
[75]               optimizer = 'rmsprop', metrics = ['accuracy'])
cnn.summary()

[76] start = time.time()
[77] cnn.fit_generator(
[78]     train_generator,
[79]     steps_per_epoch=NB_TRAIN_IMG//BATCH_SIZE,
[80]     epochs=100,
[81]     validation_data=validation_generator,
[82]     validation_steps=NB_VALID_IMG//BATCH_SIZE)
[83] end = time.time()
[84] print('Processing time:',(end - start)/60)
cnn.save_weights('cnn.h5')
```

Code 2 Flickrapi

```
# Copyright 2014-2017 Bert Carremans
# Author: Bert Carremans <bertcarremans.be>
#
# License: BSD 3 clause


from flickrapi import FlickrAPI  # https://pypi.python.org/pypi/flickrapi
import urllib
import os
import config
from random import randint
import time
import numpy as np
import os
import re
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import keras
from keras.utils import to_categorical
from keras.models import Sequential,Input,Model
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.layers.normalization import BatchNormalization
from keras.layers.advanced_activations import LeakyReLU

def download_flickr_photos(keywords, size='original', max_nb_img=-1):
    """
    Downloads images based on keyword search on the Flickr website

    Parameters
    ----------
    keywords : string, list of strings
        Keyword to search for or a list of keywords should be given.
    size : one of the following strings 'thumbnail', 'square', 'medium', default: 'original'.
        Size of the image to download. In this function we only provide
        four options. More options are explained at
        http://librdf.org/flickcurl/api/flickcurl-searching-search-extras.html
    max_nb_img : int, default: -1
        Maximum number of images per keyword to download. If given a value of -1, all images
        will be downloaded

    Returns
    ------
    Images found based on the keyword are saved in a separate subfolder.

    Notes
    -----
    This function uses the Python package flickrapi and its walk method.
    FlickrAPI.walk has same parameters as FlickrAPI.search
    http://www.flickr.com/services/api/flickr.photos.search.html

    To use the Flickr API a set of API keys needs to be created on
    https://www.flickr.com/services/api/misc.api_keys.html
    """
    if not (isinstance(keywords, str) or isinstance(keywords, list)):
        raise AttributeError('keywords must be a string or a list of strings')

    if not (size in ['thumbnail', 'square', 'medium', 'original']):
        raise AttributeError('size must be "thumbnail", "square", "medium" or "original"')

    if not (max_nb_img == -1 or (max_nb_img > 0 and isinstance(max_nb_img, int))):
        raise AttributeError('max_nb_img must be an integer greater than zero or equal to -1')



    flickr = FlickrAPI(config.API_KEY, config.API_SECRET)

    if isinstance(keywords, str):
        keywords_list = []
        keywords_list.append(keywords)
    else:
        keywords_list = keywords

    if size == 'thumbnail':
        size_url = 'url_t'
    elif size == 'square':
        size_url = 'url_q'
    elif size == 'medium':
        size_url = 'url_c'
    elif size == 'original':
        size_url = 'url_o'

    for keyword in keywords_list:
        count = 0

        #print('Downloading images for', keyword)

        results_folder = config.IMG_FOLDER + keyword.replace(" ", "_") + "/"
        if not os.path.exists(results_folder):
            os.makedirs(results_folder)

        photos = flickr.walk(
                    text=keyword,
                    extras=size_url,
                    license='1,2,4,5',
                    per_page=50)

        urls = []
        for photo in photos:
            t = randint(1, 3)
            time.sleep(t)
            count += 1
            if max_nb_img != -1:
                if count > max_nb_img:
                    print('Reached maximum number of images to download')
                    break
            try:
                url=photo.get(size_url)
                urls.append(url)

                urllib.request.urlretrieve(url,  results_folder + str(count) +".jpg")
                print('Downloading image #' + str(count) + ' from url ' + url)
            except Exception as e:
                print(e, 'Download failure')

    print("Total images downloaded:", str(count - 1))

Dog = ['yorkshire terrier dog','pug dog','cocker spaniel dog','jack russell terrier dog','beagle dog','husky dog','samoyed dog','labrador dog','tosa dog','dachshund dog','french bulldog','chihuahua dog']
#['yorkshire terrier dog','pug dog','cocker spaniel dog','jack russell terrier dog','beagle dog','husky dog','samoyed dog','labrador dog','tosa dog','dachshund dog','french bulldog','chihuahua dog']

for Dog in Dog:
  download_flickr_photos(Dog, size='medium', max_nb_img=750)
```