Report on Image Classification for Dogs and Cats using CNN

SID-19105042,19105052,19105108

1. INTRODUCTION

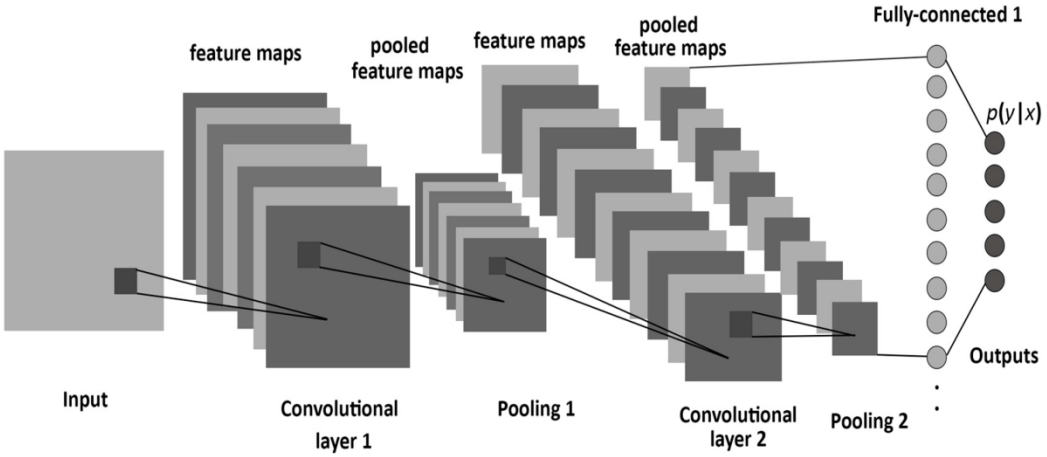
Our basic task is to create an algorithm to classify whether an image contains a dog or a cat. The Dogs and Cats classification is trying to solve the CAPTCHA challenge, which relies on the problem of distinguishing images of dogs and cats. The input for this task is images of dogs or cats from training dataset, while the output is the classification accuracy on test dataset. Our training set contains 23,000 images. In our project we are going to build a convolutional neural network to solve the problem and achieve higher performance and better results.

2. Tools & Language Used –

Jupyter Notebook ,Python, Computer Vision Library - OpenCV ,Deep Learning Libraries - TensorFlow - Keras with TensowFlow backend – Systems , Pickle, Numpy

3. Convolutional Neural Network

The name “convolutional neural network” indicates that the network employs a mathematical (convolution) operation. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers. A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a multiplication or other dot product. The activation function is commonly a RELU layer, and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution. The final convolution, in turn, often involves back propagation in order to more accurately weight the end product.



4. Dataset and Data Augmentation

The dataset in keras is divided into folders for each class. The dataset is divided into training and testing set. The training set and the test set compose of 2 folders one for cat images and

other for dog images. There are 23000 images of each cat and dog for training and 1150 image of each cat and dog for testing. The images are of varing shape and sizes, but in order to train a CNN the images should be of same size. Data Augmentation is being carried out by using the ImageDataGenerator module provided by Keras. Using it the images are resized to 100 x 100. Also, for training of a convolutional neural network large set of images are needed. So, data augmentation is applied on the existing set of images to increase the dataset size. Various data augmentation technique such as rescaling, shear range, zoom range are being used to do the same.

5. Model Architecture

Classifier is the name given to the Sequential model. The model's first layer is a Conv2D layer. Since, it is the first layer of the model, input shape of the images that are going to be supplied to the model is being mentioned. Next layer is a batch normalization layer. Then one activation layer corresponding to the conv2d layer. Further there is another set of conv2d, batch normalization and activation layer with different number of kernels in the conv2d layer. After that a max Pooling layer is there and then a dropout layer is there. The same set of layers is again repeated with different number of kernel's and dropout rate. The convolution layers end with this set. Next are the fully connected layer. The Sequential model API is used to build model. The sequential API allows you to create models layer-by-layer. The 'add()' function to add layers to our model. The model needs to know what input shape it should expect. For this reason, only the first layer in a Sequential model needs to receive information about its input shape. Dropout layer consists in randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting. Activation layer are used to apply the activation function to the output of that layer. The purpose of the activation function is to introduce non-linearity into the output of a neuron. Relu and sigmoid activation are used in the model. Batch Normalization is used for improving the speed, performance, and stability of artificial neural networks. Batch normalization is a method we can use to normalize the inputs of each layer and achieve faster convergence. Maximum pooling, or max pooling, is a pooling operation that calculates the maximum, or largest, value in each patch of each feature map. The results are down sampled or pooled feature maps that highlight the most present feature in the patch. In Conv2D layer, a kernel, convolution matrix, or mask is a small matrix. It is used for blurring, sharpening, embossing, edge detection, and more. This is accomplished by doing a convolution between a kernel and an image. This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. Then comes the fully connected layers. It contains only 2 layers. First one is the global average pooling layer to minimize overfitting by reducing the total number of parameters in the model. Second layer and the final layer is the Dense layer with sigmoid activation. Global Average Pooling 2D layer is used to minimize overfitting by reducing the total number of parameters in the model. GAP layers perform a more extreme type of dimensionality reduction, where a tensor with dimensions h×w×d is reduced in size to have dimensions 1×1×d. GAP layers reduce each h×w feature map to a single number by simply taking the average of all hw values. Dense layer implements the operation: output = activation(dot(input + kernel) + bias, activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use_bias is True).

6. Model Evaluation & Code

The model trained for 5 epochs .The training accuracy kept on increasing but the validation accuracy started to decrease which might be due to overfitting.

Data Importing

```
import numpy as np
import cv2
import os
import random
import matplotlib.pyplot as plt
import pickle
DIRECTORY = r'C:\Users\Aayush Rana\Downloads\dogscats\dogscats\train'
CATEGORIES = ['cats','dogs']
IMG_SIZE = 100
#if image size is less then the computation will be faster
#but the identification whether it is a cat or dog becomes difficult with
naked eye

data = []

for category in CATEGORIES:
    #below line joins the directory with the categories mentioned
    folder = os.path.join(DIRECTORY , category)
    #fetch the index of cat or dog for the neural network to identify if it is
a cat or dog
    label = CATEGORIES.index(category)
    for img in os.listdir(folder):
        #complete image path
        img_path = os.path.join(folder ,img)
        #read the image into the array
        img_arr = cv2.imread(img_path)
        #resize the different images having various sizes
        img_arr = cv2.resize(img_arr,(IMG_SIZE,IMG_SIZE))
        data.append([img_arr,label])

#shuffle the data
random.shuffle(data)
X = []
Y = []
for features,labels in data:
    X.append(features)
    Y.append(labels)
#convert X and Y into arrays
X = np.array(X)
Y = np.array(Y)
#save the variables X and Y in our Computer
pickle.dump(X,open('X.pkl','wb'))
```

```
pickle.dump(Y,open('Y.pkl','wb'))
```

Training

```
import pickle
import tensorflow as tf
# import random
# import matplotlib.pyplot as plt
import numpy as np
import cv2
import keras
X = pickle.load(open('X.pkl','rb'))
Y = pickle.load(open('Y.pkl','rb'))
#now if we output X we get pixel values present in the image
#if we output y we get 0 or 1 , 0->cat and 1->dog

X = X.reshape(-1,100,100,3)
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D ,Flatten ,Dense
model = Sequential()

model.add(Conv2D(64, (3,3) , activation = 'relu'))
model.add(MaxPooling2D((2,2)))

model.add(Conv2D(64, (3,3) , activation = 'relu'))
model.add(MaxPooling2D((2,2)))

model.add(Flatten())

#from the first index till everything
model.add(Dense(128,input_shape = X.shape[1: ], activation = 'relu'))

model.add(Dense(2,activation = 'softmax'))
model.compile(optimizer = 'adam' ,loss =
'sparse_categorical_crossentropy',metrics=['accuracy'])
#validation is a part of the dataset which will be kept for testing
#validation = 0.1 means 10 percent
#out of 23000 images 10 percent images will be taken for validating the model
and not training the model
model.fit(X,Y,epochs = 5 ,validation_split =0.1)
model.evaluate(X,Y)
```

Prediction

```
import pickle
import tensorflow as tf
import numpy as np
import cv2
import keras
model = tf.keras.models.Sequential()
CATEGORIES = ['cat', 'dog']
def image(path):
    img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
    new_arr = cv2.resize(img, (60, 60))
    new_arr = np.array(new_arr)
    new_arr = new_arr.reshape(-1, 60, 60, 1)
    return new_arr

model = keras.models.load_model('3x3x64-catvsdog.model')
import urllib.request
from PIL import Image
URL = r'C:\Users\Aayush Rana\Downloads\dogscats\dogscats\train\dogs\dog.1.jpg'
img = Image.open(URL)
img.show()

prediction = model.predict([image(URL)])
print("Our model says it is a :",CATEGORIES[prediction.argmax()])
```

7. CONCLUSION

In this paper we build a deep convolutional neural network for image classification (cat and dog images). Despite of using only a subset of the images an accuracy of 90.10% was obtained.