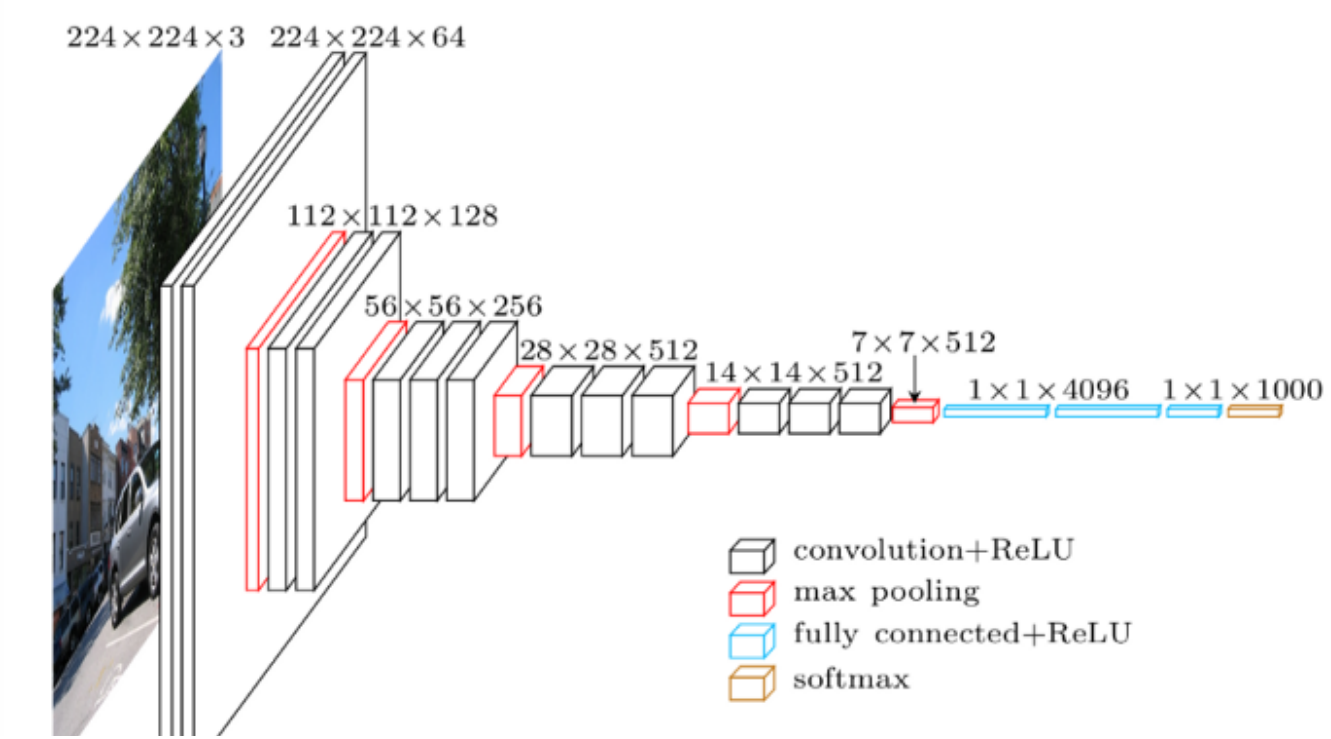
Làm quen với Convolutional Neural Network và bài toán nhận diện ảnh.

Data Analysis- VelaCorp Follow
Nov 8, 2018 · 6 min read



Hình 1: CNN Process (Source: https://www.jeremyjordan.me/convnet-architectures/)

CNN- Mạng Nơ-ron tích chập ra đời nhằm khắc phục các nhược điểm của Deep neural network do các mạng lưới đào tạo ngày càng phức tạp. Chi tiết về CNN bạn đọc có thể xem thêm tại (http://cs231n.github.io/convolutional-networks/).

Một cách tóm tắt đơn giản nhất về CNN, Khi quan sát Process ở hình 1, ta có thể thấy CNN trong bài toán nhận dạng ảnh thực hiện một số bước như sau:

- Bóc tách các ảnh thành các mảng nhỏ, chồng chéo
- Truyền các ảnh nhỏ sau khi được tách vào một neural nhỏ
- Lưu trữ kết quả vào thành mảng
- Giảm mẫu, tìm các đặc trưng lớn nhất để giữ lại
- Đưa kết quả sau khi giảm mẫu vào một mạng neural khác và dự đoán

Các bước trên có thể được lặp lại nhiều lần như trong hình vẽ tùy theo độ phức tạp của bài toán.

Phâ`n lý thuyê´t bạn đọc có thể đọc thêm trong link đính kèm, chúng ta sẽ bă´t tay vào thực hành với bài toán nhận dạng chó và mèo trong các bức ảnh, chúng ta sẽ đi qua các bước với trình tự như sau:

Trước khi bước vào thực hiện, ta câ`n chuẩn bị dữ liệu đâ`u vào. Để có được dữ liệu training cho CNN model, chúng ta câ`n có bộ ảnh chó, mèo đủ lớn để thực hiện training cho model. Bài toán phân loại ảnh này xuâ´t phát từ một cuộc thi trên Kaggle. Chúng ta có thể download tại (https://www.kaggle.com/c/dogs-vs-cats/data). Sau khi download và giải nén dữ liệu, ta có hai thư mục train và test

Hình 2: Bộ ảnh từ tập Train trong bộ dữ liệu

1/ Import các thư viện câ`n thiê´t cho Project

```
#Load các thư viện cần thiết
#Nội dung tham khảo:
https://tiendv.wordpress.com/2016/12/25/convolutional-neural-
networks/
#https://medium.com/@curiousily/tensorflow-for-hackers-part-iii-
convolutional-neural-networks-c077618e590b
#http://tflearn.org/models/dnn/
import os
import cv2 #Thư viện dùng để xử lý ảnh: Load ảnh và resize…
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import tflearn #Thư viện dùng để training model CNN
from random import shuffle
from tqdm import tqdm
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import
```

2/ Sau khi đã import xong các thư viện cần thiết cho Project. Chúng ta sẽ tiếp tục khai báo một số sử dụng trong cả dự án như Learning rate, kích cỡ của ảnh..

```
#Đặt các thông số learning rate,IMG size đường dẫn với data training
và data testing
Learningrate=0.001
TRAIN_DIR='.../train'
TEST_DIR='.../test'
Img_size=50
MODEL_NAME = 'Catanddog_Detect'
```

3/ Sau khi đã import các thư viện cần thiết và khai báo một số thông tin chung cho bài toán. Ta sẽ thực hiện các bước trong Feature Engineering để có được dữ liệu đầu vào cho training model.

Với các bức ảnh trong tập train, chúng ta có thể thấy tên các bức ảnh có dạng như sau: dog.1, cat.2… cấu trúc là dog/cat.number. Với Y là dog hoặc cat, ta cần lấy ra được thông tin này của bức ảnh (Cắt từ tên ảnh) và thực hiện chuyển sang one-hot.

```
def label(image_name): #Image_nme có dạng dog.1,cat.2
    word_label=image_name[-3]
    if word_label =='cat': #Nếu wordlabel sau khi được cắt có giá
#trị là cat thì ta trả ra giá trị [1,0]
        return np.array([1,0])
    elif word_label =='dog':
        return np.array([0,1])
```

Với dữ liệu ảnh chó/ mèo trong tập train, ta cần xử lý một số bước nhằm lấy các ảnh lên, resize ảnh… như sau:

```
#Khai báo hàm xử lý để tạo ra dữ liệu training
def create_train_data():
    training_data=[]
    for img in tqdm(os.listdir(TRAIN_DIR)):
        path=os.path.join(TRAIN_DIR,img)
```

```
        img_data=cv2.imread(path,cv2.IMREAD_GRAYSCALE) #Load ảnh và
convert sang dạng matrix, trước đó chuyển hết ảnh sang màu gray
        img_data=cv2.resize(img_data,(Img_size,Img_size))
        training_data.append([np.array(img_data),create_label(img)])
    shuffle(training_data)
    np.save('train_data.npy', training_data)
    return training_data
```

4/ Tiếp theo chúng ta sẽ thực hiện create_train_data và tách dữ liệu thành hai phần Training data và testing data với Testting data là 80% của bộ dữ liệu

```
#Load data về
train_data=create_train_data()
#Chia tập dữ liệu thành train và test từ bộ dữ liệu train ban đầu.
#, bộ dữ liệu test không có label nên chỉ dùng để test random
from sklearn.model_selection import train_test_split
train,test=train_test_split(train_data,test_size=0.2,
random_state=42)
#Sau khi tách được bộ train, test. Ta lấy ra X và Y tương ứng từ các
tập này
X_train = np.array([i[0] for i in train]).reshape(-1, Img_size,
Img_size, 1)
y_train = [i[1] for i in train]
X_test = np.array([i[0] for i in test]).reshape(-1, Img_size,
Img_size, 1)
y_test = [i[1] for i in test]
```

Sau khi đã có đầy đủ dữ liệu đầu vào để thực hiện training CNN model, chúng ta sẽ đưa vào mô hình để thực hiện training, chi tiết các bước sẽ được note trong comment:

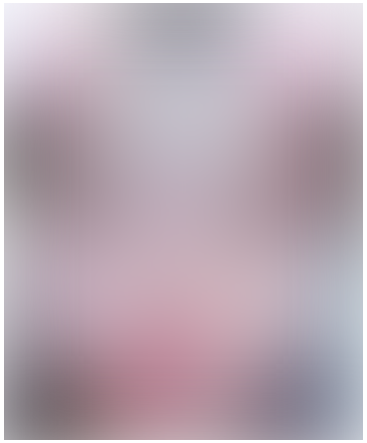
```
tf.reset_default_graph()
convnet = input_data(shape=[None, Img_size, Img_size, 1],
name='input')
convnet = conv_2d(convnet, 32, 5, activation='relu') #Phần quan
#trọng nhất trong mang neural, dùng các cửa sổ trượt
#là các kenel, filter hay feature detector để lấy ra các đặc trưng
#trên mỗi vùng ảnh với kích thước của nó được khai báo như trên
convnet = max_pool_2d(convnet, 5) #Lấy ra giá trị lớn nhất, đặc tính
#nổi trội nhất của vùng dữ liệu để làm giảm kích thước và tăng
#tính đại diện
convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)


convnet = conv_2d(convnet, 128, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)


convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)


convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)


convnet = fully_connected(convnet, 1024, activation='relu') #Sau khi
#các tầng được phân tách và thực hiện conv_2d Và chọn ra max_pool
#sẽ được kết nối lại với nhau
convnet = dropout(convnet, 0.8) #Loại bỏ việc học lẫn nhau giữa các
#neural
convnet = fully_connected(convnet, 2, activation='softmax')
convnet = regression(convnet, optimizer='adam',
```

```
learning_rate=Learningrate, loss='categorical_crossentropy',
name='targets')
model = tflearn.DNN(convnet, tensorboard_dir='log',
tensorboard_verbose=0)
model.fit({'input': X_train}, {'targets': y_train}, n_epoch=10,
          validation_set=({'input': X_test}, {'targets': y_test}),
          snapshot_step=500, show_metric=True, run_id=MODEL_NAME)
#Epoch số vòng thực hiện lại cả quá trình, Snapshot step- Số step
#sau mỗi lần snapshot
```

Với việc training, Nê´u có sự hỗ trợ của GPU việc này sẽ được thực hiện râ´t nhanh do cơ chê´ xử lý song song râ´t nhiê`u các phép toán. Tuy nhiên với chỉ CPU chúng ta đã có thể thực hiện được Project này nhưng thời gian training có thể chiê´m tới vài tiê´ng đô`ng hô`. Sau khi training, ta thu được kê´t quả như sau:

Hình 3: Kết quả

5/ Sau khi training xong model, chúng ta sẽ thực hiện test. Với kê´t quả test từ bộ dữ liệu ban đâ`u, kê´t quả khá tô´t. Tôi sẽ thử với dữ liệu là một bức ảnh được sử dụng công cụ chỉnh ảnh như sau, bức ảnh được lâ´y từ internet:



Hình 4: Ảnh test cho mô hình

Sau khi thực hiện các bước load ảnh, resize và đưa vào dự đoán ta thu được kê´t quả khá thú vị :

Hình 5: Kết quả nhận diện

Kê´t quả: Mô hình đã dự đoán với tỷ lệ hình ảnh trên là ảnh của mèo lên tới 0.96. Rấ´t tuyệt vời!

Bài viê´t có tham khảo kiê´n thức từ một sô´ nguô`n:

https://medium.com/@curiousily/tensorflow-for-hackers-part-iii-convolutional-neural-networks-c077618e590b
http://tflearn.org/models/dnn/

Deep Learning Convolution Neural Net Machine Learning Data Science