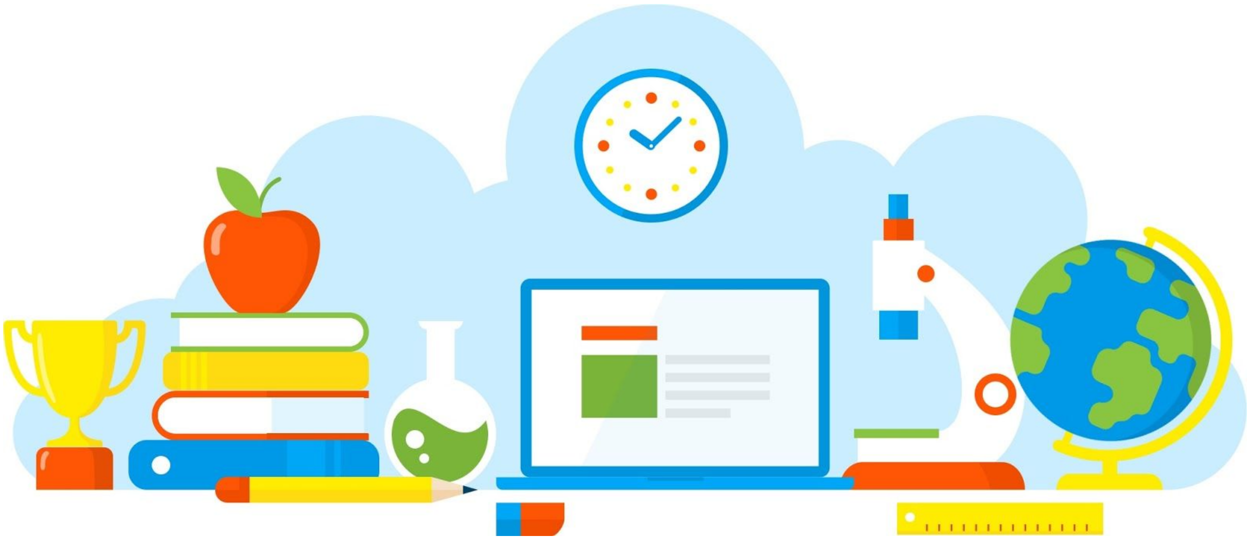


Machine Learning Engineer Nanodegree

Capstone Project

Build a Dogs Vs Cats Classifier

Ntepp Jean Marc

09/04/2017

I. Definition

Computer Vision has become ubiquitous in our society, with applications in search, image understanding, apps, mapping, medicine, drones, and self-driving cars. Core to many of these applications are visual recognition tasks such as image classification, localization and detection. Recent developments in neural network (aka "deep learning") approaches have greatly advanced the performance of these state-of-the-art visual recognition systems. From https://deeplearning4j.org/neuralnet-overview we can read a succinct definition of Deep Learning: Deep learning is the name we use for "stacked neural networks"; that is, networks composed of several layers.

The goal of this project is to build a cat/dog image classifier using Deep Learning algorithm called Convolutional Neural network. In other word, we will create an algorithm to distinguish dogs from cats. In this Capstone, we will be resolving a project from kaggle competition and using a dataset from Kaggle.

Problem Statement

Our goal is to build a machine learning algorithm capable of detecting the correct animal (cat or dog) in new unseen images. So we are trying to solve a classification problem.

1. In this project, I will write an algorithm to classify whether images contain either a dog or a cat.
2. Our algorithm has images as input and return whether image contains dogs or cats.
3. Personally I will also try to locate the cat or dog in the image. This is not required by Kaggle competition.

In this project we will apply Convolutional Neural Network(CNN) to predict whether images contains dogs or cats. CNN is used here because CNN are widely used for images classification task.

Metrics

We have used accuracy metric. Accuracy measures the ratio of correct predictions to the total number of cases evaluated.

$$A(M) = \frac{TN + TP}{TN + FP + FN + TP}$$

where

- TN is the number of true negative cases

- FP is the number of false positive cases

- FN is the number of false negative cases

- TP is the number of true positive cases

II. Analysis

Data Exploration

The dataset is divided into training and testing. They can be found in https://www.kaggle.com/c/dogs-vs-cats/data. The training archive contains 25,000 labeled images of dogs(12500) and cats(12500). We will train our algorithm on these files and predict the labels (1 = dog, 0 = cat). The sizes of the images are different, the ranges of dimensions is between 1023 X 768 and 60 X 39. Some examples of two differents format of inputs images are presented below.

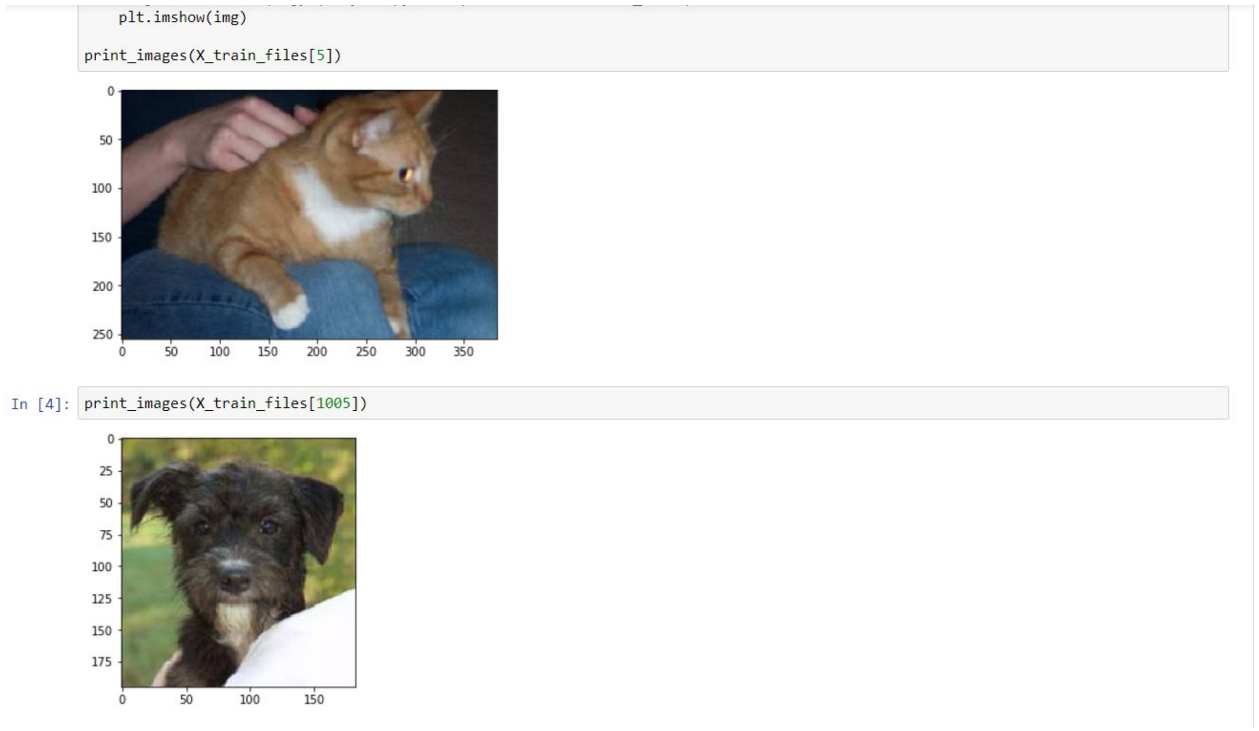
The testing archive contains 12500 images of cats and dogs unlabeled images that we will use for testing our Convolutional Neural Network model. Note here that the labels are contained in the name of the images.

Image1: Dimension: 286 X 270

Image2: 469 X 600

Exploratory Visualization

I plotted the new images below from the training data to present the data. We can observe here that the first image has a size of 250X350 and second has a size of 200X175. This will imply in the preprocessing step to resize the training image.



```
    plt.imshow(img)

print_images(X_train_files[5])
```

```
In [4]: print_images(X_train_files[1005])
```

Image 3:

Algorithms and Techniques

We have created a Convolutional Neural Network that classifies dogs vs cats. We have build our architecture using Keras with TensorFlow backend. Our model architecture is defined below:

```
_________________________________________________________________
Layer (type)              Output Shape          Param #
=================================================================
conv2d_21 (Conv2D)          (None, 224, 224, 16)      208
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 112, 112, 16)      0
_________________________________________________________________
conv2d_22 (Conv2D)          (None, 112, 112, 32)      2080
_________________________________________________________________
```

```
max_pooling2d_22 (MaxPooling (None, 56, 56, 32)      0
_________________________________________________________________
conv2d_23 (Conv2D)          (None, 56, 56, 64)      8256
_________________________________________________________________
max_pooling2d_23 (MaxPooling (None, 28, 28, 64)      0
_________________________________________________________________
conv2d_24 (Conv2D)          (None, 28, 28, 128)     32896
_________________________________________________________________
max_pooling2d_24 (MaxPooling (None, 14, 14, 128)     0
_________________________________________________________________
conv2d_25 (Conv2D)          (None, 14, 14, 256)     131328
_________________________________________________________________
max_pooling2d_25 (MaxPooling (None, 7, 7, 256)       0
_________________________________________________________________
flatten_5 (Flatten)         (None, 12544)           0
_________________________________________________________________
dense_9 (Dense)             (None, 250)             3136250
_________________________________________________________________
dropout_5 (Dropout)         (None, 250)             0
_________________________________________________________________
dense_10 (Dense)            (None, 2)               502
=================================================================
```

Figure 1: Model architecture 1

Conv2D: This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. We added **relu** as non linear activation function in each the Convolutional layer.

MaxPooling2D: The objective of max pooling layer is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned.

Flatten layer: The Flatten layer is a utility layer that flattens an input of shape a * b * c * d to a simple vector output of shape a * (a*c*d).

Dense: Dense layer is a Fully connected layers, which perform classification on the features extracted by the convolutional layers and downsampled by the pooling layers. In a dense layer, every node in the layer is connected to every node in the preceding layer.

Dropout layer was introduced as a regularization technique for reducing overfitting.

Benchmark

We will compare our solution with the current best solution that use Transfer Learning model named VGG-19 (https://github.com/mrgloom/kaggle-dogs-vs-cats-solution)
We will see if our CNN model build from scratch is more powerful that the VGG-19.

This Benchmark solve the same problem from https://www.kaggle.com/c/dogs-vs-cats kaggle competition.

Methodology

Data Preprocessing

We have used 2 pre-processing techniques.

First, as mentioned in the Data Exploration section we have observed that the dataset was not have the same dimension. We have converted all the training and validation images with shape (224, 224, 3).

Second, we have rescaled the images by dividing every pixel in every image by 255.

Implementation

1. Import Datasets

We used python glob library to import the dogs and cats dataset images. We populate a few variables through the use of the load_files function from the scikit-learn library and use numpy arrays to contain file paths to images. We created a function print_images to print images in the notebook with OpenCv. We printed two images to observe if our datasets imported correctly.

2. **Data Pre-processing**

We will discuss this in the data preprocessing section. However it is important to note that we used keras.preprocessing image to preprocess the images

3. **Build the model**

I have used the accuracy metric as explained in the Metric section, for optimizer I have used rmsprop with the default parameters as suggested by keras. It is recommended to leave the parameters of this optimizer at their default values (except the learning rate, which can be freely tuned). The model run with 5 epochs.
The model architecture implemented is defined in Figure 1.

Refinement

https://keras.io/optimizers/
https://stackoverflow.com/questions/42081257/keras-binary-crossentropy-vs-categorical-crossentropy-performance