Cats vs. Dogs Classification Report

1. Definition

Project Overview

The project is to write an algorithm to classify whether images contains either a dog or a cat. This is easy for humans, dogs, and cats. However, the computer will find it a bit more difficult. Such a challenge is often called a CAPTCHA[1] (Completely Automated Public Turing test to tell Computers and Humans Apart) or HIP (Human Interactive Proof). HIPs are used for many purposes, such as to reduce email and blog spam and prevent brute-force attacks on web site passwords.

This project will focus on algorithm approaches, from classic Convolutional Neural Network (CNN) to state-of-the-art models (e.g. VGGNet, ResNet, Inception, Xception). I will compare the accuracy and performance of all of these algorithms.

There are many public image datasets online for this challenge. Here, I will use the dataset from Kaggle’s Dogs vs. Cats Redux: Kernels Edition competition[2]. It has two folders: train and test. The train folder contains 25,000 images of dogs and cats. Each image in this folder has the label as part of the filename. The test folder contains 12,500 images, named according to a numeric id. For each image in the test set, I should predict a probability that the image is a dog (1 = dog, 0 = cat).

Project Statement

The “Cats vs. Dogs” competition is a supervised binary classification problem.There are two types of images: cats and dogs. The goal is to extract proper features and build an effective model to classify each image contains either cat or dog.

By exploring the dataset, basic information about the dataset can be obtained. The size of each image could be different and have to be resized as each model has a standard for input image size (e.g. 224\*224 for VGGNet and 299\*299 for Inception). After that, the full dataset (under train folder) should be split into training set and validation set. When the models are being trained, the performance is also evaluated on the testing set. By checking the loss and accuracy on training and validation sets, I can know if the model built is correct and if the model is under- or over-fitting. Finally, I will run prediction on test set and upload the result to Kaggle to see what position I can occupy.

Mertics

Usually we use ROC curve (receiver operating characteristic curve) and PR curve (Precision-Recall curve) to evaluate models for binary classification problem. However, in order to comply with Kaggle’s rule, I will use log loss instead.

where

n is the number of images in the test set

is the predicted probability of the image being a dog

is 1 if the image is a dog, 0 if cat

is the natural (base e) logarithm

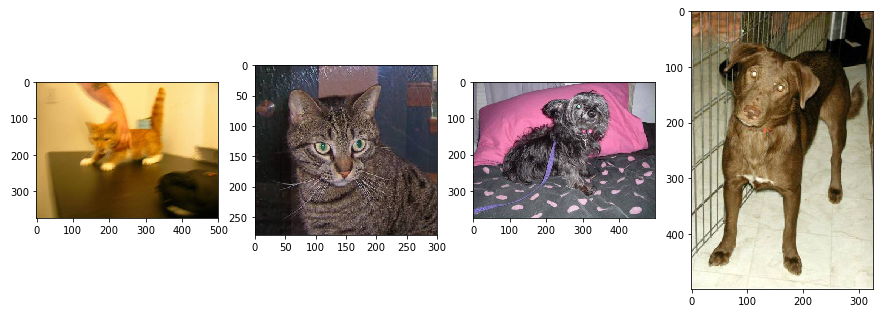
A smaller log loss is better.

1. Analysis

Data Exploration

The dataset is downloaded from <https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/data>. There are two folders: train and test. The train folder has 25,000 images with names like dog.1.jpg, dog.2.jpg, cat.1.jpg, etc. The test folder has 12,500 images with names like 1.jpg, 2.jpg, 3.jpg, etc.

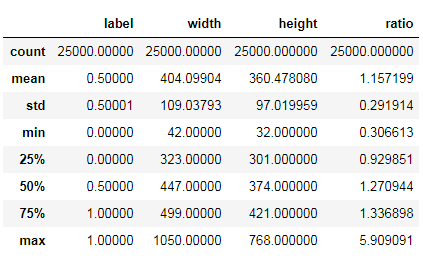
Here are some sample images with associated width and height.



Here are a few things we can notice directly by looking at above images:

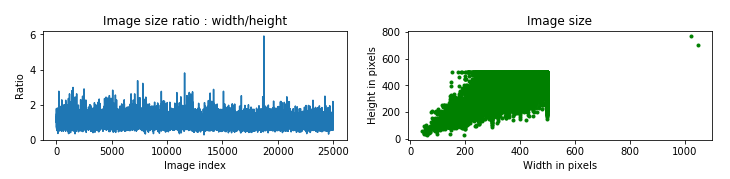
1. Images are generally well centered on the animal.
2. Images have different sizes.
3. The quality of some images are poor (e.g. 1st image).
4. Background images seem to vary a lot and be independent from the categories (the human hand in 1st image and pink pillow in 3rd image).

The following table shows the summary of image info: width, height and ratio (width / height).You can neglect “label” as it is a categorical value. The table clearly shows the sizes of images are varied.



Exploratory Visualization

We have noticed the dataset contains different sizes of images. The following plots show the distribution of each image’s width, height and ratio (width / height).



The average ratio is 1.157.

We can see two outliers in the “Image size” graph which have far too high height and width compared to the rest of the data points. On the “ratio” graph, we can also notice an outlier, which has a ratio of 6 whereas most other images have a ratio below 2.

Since most of models (e.g. VGGNet, Inception) require fixed image size, these graphs can help us determine if we need to resize or crop our images before using them as input of the models.

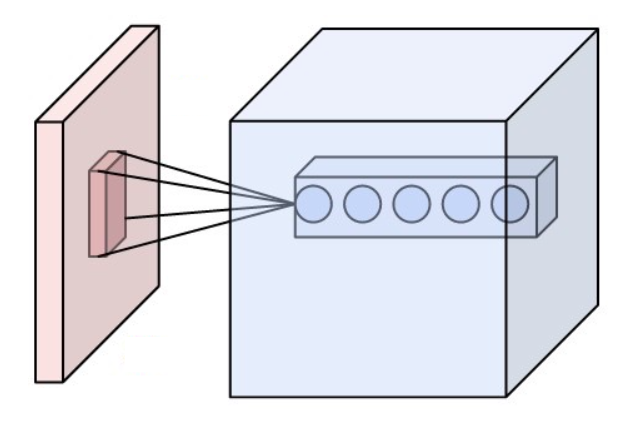
Algorithm and Techniques

For classification problem, the common method we will use is logistic regression or more generally, softmax regression. However, they have limited efficiency when the problem it tries to solve is using data with a high variability (many different inputs but have the same label), which is the case here: two images of cats can be extremely different. Images represent animals in different positions and background make it even harder for this method.

Recently a popular tool used for image classification is Convolutional Neural Network (CNN). It is very useful for images where important features (like a portion of a cat ear) can be anywhere, which is the case here as animals are in different position and sometimes more than one per image.

A CNN consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typical consist of convolutional layers, pooling layers, fully connected layer, ReLu layers and sometimes dropout layers.

The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input.



*Neurons of a convolutional layer (blue), connected to their receptive field (red)*

Another important concept of CNNs is pooling, which is a form of non-linear down-sampling. There are several non-linear functions to implement pooling among which max pooling is the most common. It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum. The intuition is that the exact location of a feature is less important than its rough location relative to other features. The pooling layer serves to progressively reduce the spatial size of the representation, to reduce the number of parameters and amount of computation in the network, and hence to also control overfitting. It is common to periodically insert a pooling layer between successive convolutional layers in a CNN architecture. The pooling operation provides another form of translation invariance.



Max pooling with a 2\*2 filter and stride = 2

ReLU is the abbreviation of Rectified Linear Units. This layer applies the non-saturating activation function . It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer.

Other functions are also used to increase nonlinearity, for example the saturating hyperbolic tangent , , and the sigmoid function . ReLU is often preferred to other functions, because it trains the neural network several times faster without a significant penalty to generalisation accuracy.

Finally, after several convolutional and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular neural networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

Sometimes there is a dropout layer. Because a fully connected layer occupies most of the parameters, it is prone to overfitting. One method to reduce overfitting is dropout. At each training stage, individual nodes are either "dropped out" of the net with probability or kept with probability , so that a reduced network is left; incoming and outgoing edges to a dropped-out node are also removed. Only the reduced network is trained on the data in that stage. The removed nodes are then reinserted into the network with their original weights.

Since the image is labeled (let’s say it is a cat) and we know the influence of each neuron’s weight, we automatically tweak these weights to make our model closer to a 100% confidence it is a cat. This adjustment is called “backpropagation”. To do this, we compute a loss function and find its lowest value with an optimizer to find the best combination of these weights.

References

[1] https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition