Classification of Cat vs. Dog images using CNN

Sanjeev Kumar Singh

Date: 13/11/2016

Objective :

We have given binary classification problem of images. Given images is composed of Cat and Dog. We have to build a model using Convolutional Neural Network to classify them.

Step#01 : Dataset

Size of number training sample given was **25000**. The data is given in form of matrix with size **(25000,12888)**, where each row contains one image.

12288=64*64*3;

Training data has been reshaped in order to get the images straight. trainX and trainY has be obtained from training data and pixel values are converted from between 0 to 1. Training labels were converted into categorical variable for training purpose. Similar steps would be also performed on the testX (converting pixels between 0 to 1) and testResults (after prediction converting into 1 dimensional vector). The number of samples for testing was **12500.**

Step#02 : Library Installation

For building CNN model I have used **'keras**' library which is a high level python library. It uses 'tensorflow' by default as to train the model in its backend but I have used '**theano**'.

Step#03 : Model Architecture

I have started training with **2 convolutional** layer but accuracy was not improved by 60% despite of tuning various parameters. In my final architecture. I have finally got best validation accuracy with **3 convolutional** layers in the architecture. First convolution layer has **32 filters** of size **[5,5]**, **second convolutional** layer has **64 filters** of size **[5,5]** consequently **third laye**r has **128 filters** of size **[3,3].** As it is obvious from architecture I have used bi-pyramid approach because as number of filters increases filter sizes decreases. **Max Pooling [2x2]** and **dropout 0.25** was added only after second and third convolutional layer only.

After convolutional layer **two fully connected layer** were added in the architecture with activation function **ReLu** and **dropout 0.5**. After this **softmax laye**r was added for **classification task.**

The parameters of architecture was as follows :

Activation Function : ReLu
Optimizer : SGD
Loss : categorical_crossentropy
metrics='accuracy'
decay=1e-6

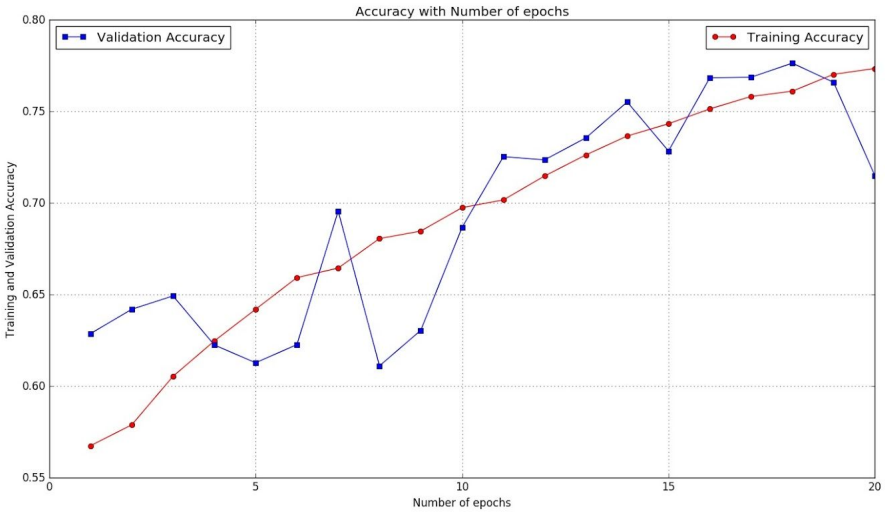
momentum=0.9

Step#04 : Results and Post Processing

I have trained the model using above architecture. I have trained the network for 20 epochs. In my laptop it took approximately 30 minutes to train 1 epoch making total time for training approximately **600 minutes**. The results generated by code was not in the format required for submission. Due to this, post processing and results has been done in matlab and labels of test data has been saved as Results.txt file which is also added in this folder.

[i]. The following graph is for variation **Training and validation accuracy** with number of epoch

.



[ii]. The following graph is for variation **Training and validation loss** with number of epoch