“[ENPM 673] Project-5”

Perception for Autonomous Robots

PROJECT REPORT

Nevil Patel-116897068

Shesh Mali-116831055

Akshay kurhade-116914529

Cat & Dog Classifier using CNN

Table of Contents

Abstract

The Dogs vs. Cats dataset is a standard computer vision dataset that involves classifying photos as either containing a dog or cat. In this project, we have discovered how to develop a convolutional neural network to classify photos of dogs and cats. In order to implement that we have gone through steps like to load and prepare photos of dogs and cats, develop CNN for classification and to improve model function. The dataset provided 25,000 labelled photos: 12,500 photos of dogs and cats each. Predictions were then made on a test dataset of 12,500 unlabelled photographs. Report shows the results obtained and network taken to improve the Accuracy of the model.

Implementation Flow

Pre-Processing

We've got the data, but we can't exactly just stuff raw images right through our convolutional neural network. First, we need all of the images to be the same size, and then we also will probably want to just grayscale them. Also, the labels of "cat" and "dog" are not useful, we want them to be one-hot arrays using our function label_img(). we will create train data of 50*50px using our function create_train_data(). Later we will feed those standardized data to our CNN.

Training Data

Here we train our data using CNN and based on the trained model we will run the Trained model on the test data to classify images of Dogs and Cats.

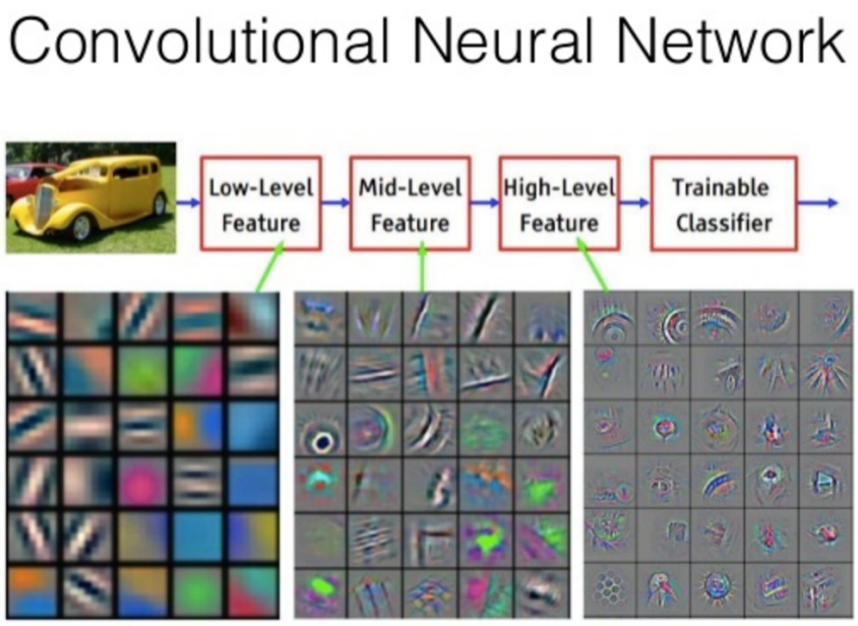
Convolutional Neural Network

Input

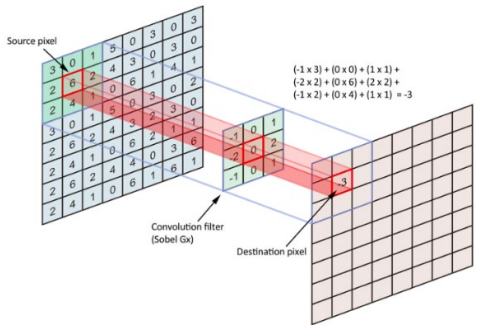
A Matrix of pixel values in the shape of [W, H, C]. Let's assume that our input is [32x32x3].

Convolution

The ultimate purpose of this layer is to receive a **feature map**. Usually, we start with low number of filters for low-level feature detection. The deeper we go into the CNN, the more filters (usually they are also smaller) we use to detect high-level features.



Feature detection is based on 'scanning' the input with the filter of a given size and applying matrix computations in order to derive a feature map.



Let's take a look at the following 3x3 filter with stride 1.

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

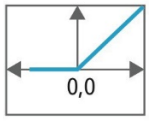
Assuming that we use 12 filters, we will end up with a [32x32x12] output.

Activation

Without going into further details, we will use ReLU activation function that returns 0 for every negative value in the input image while it returns the same value for every positive value.

15	20	-10	35
18	-110	25	100
20	-15	25	-10
101	75	18	23

Transfer Function



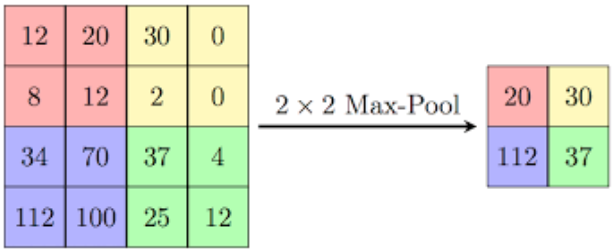
15	20	0	35
18	0	25	100
20	0	25	0
101	75	18	23

Shape remains unchanged, it's still [32x32x12]

Pooling

The goal of this layer is to provide spatial variance, which simply means that the system will be capable of recognizing an object as an object even when its appearance varies in some way.

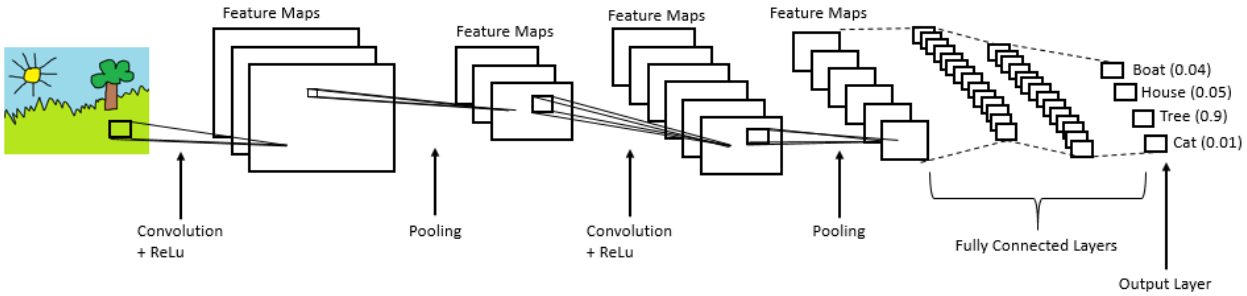
Pooling layer will perform a downsampling operation along the spatial dimensions (width, height), resulting in output such as [16x16x12] for pooling_size=(2, 2).



Fully Connected

In a fully connected layer, we flatten the output of the last convolution layer and connect every node of the current layer with the other node of the next layer. Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks and work in a similar way.

The last layer of our CNN will compute the class probability scores, resulting in a volume of size [1x1xNUMBER_OF_CLASSES].



<u>Our Model</u>

Input Image: 50*50*1 px

Convolution Connected Layer: 5 (32,64,128,64,32)

Activation Function: Relu

Optimizer: Adam

Fully Connected Layer: 2

Learning Rate: 1e-3

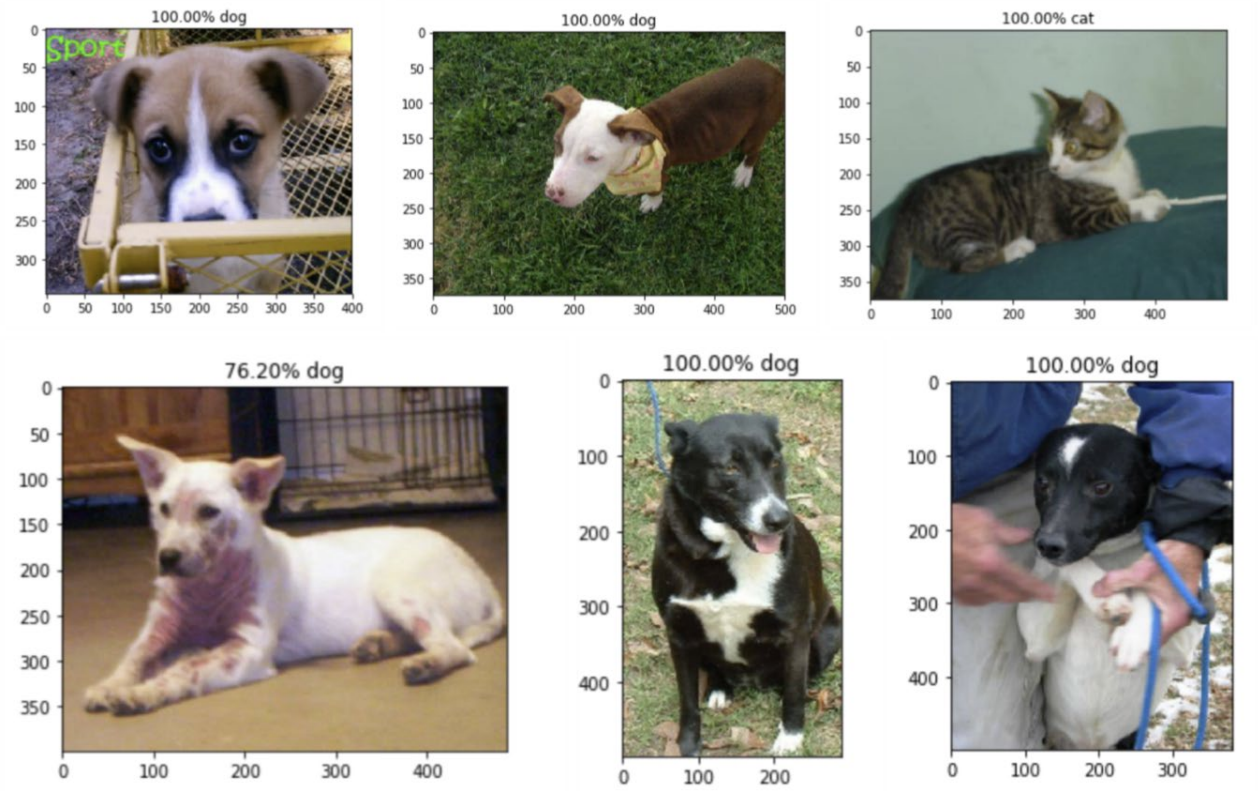
Loss: categorical cross entropy

Epoch: 10

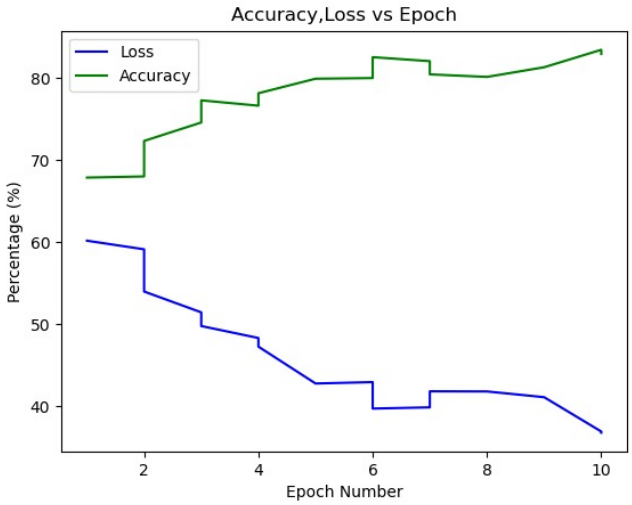
Batch: 500

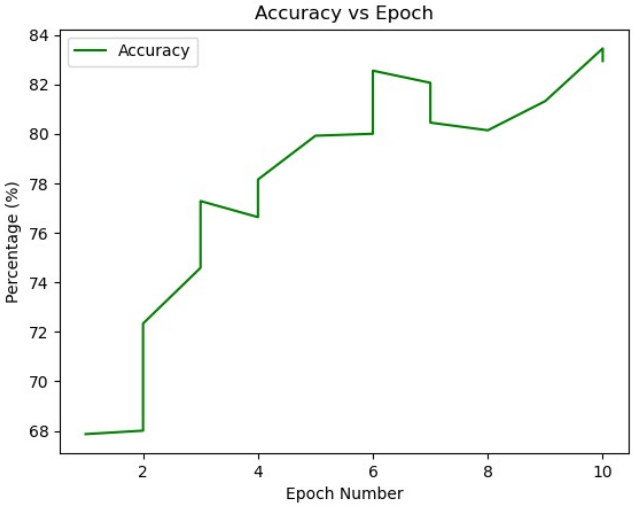
Testing Data:

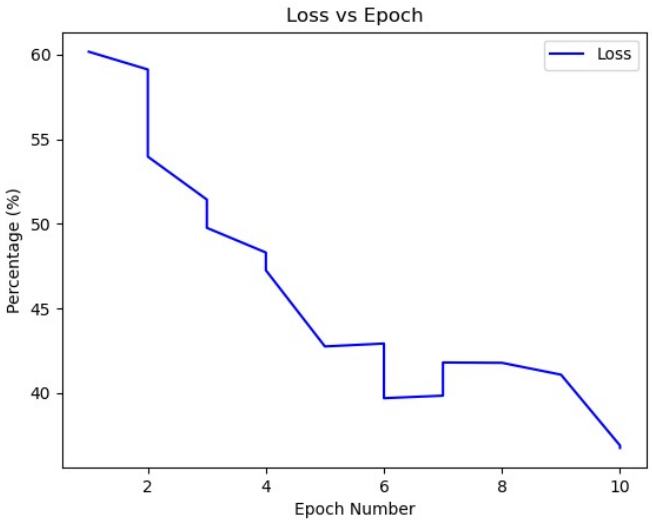
Here the testing dataset is loaded and classified with our trained model. It is done by our function process_test_data(). The function takes the raw image standardize them by resizing and shuffling the test data. Lastly we feed each image to our trained model and store the result to our csv file. As we can see below, our model learned to successfully identify both dogs and cats. Its predictions are very confident in the majority of the cases and they are lower only for anything that's unusual like pose, fur color etc.



Output Results







In overall, our image classification system proved to be **very successful (84% accuracy**) in the task of dogs vs cats classification.

References:

- https://towardsdatascience.com/image-classifier-cats-vs-dogs-with-convolutional-neural-networks-cnns-and-google-colabs-4e9af21ae7a8
- https://pythonprogramming.net/convolutional-neural-network-kats-vs-dogs-machine-learning-tutorial/