Google Landmark Recognition Challenge

Klim Markelov
Seth Gyamerah
Daniil Fishman
Artem Sergeev
Vitaly Poteshkin

Task

1) To recognize a landmark (if any) on an input image of uncertain size.
2) Usually, the problem is to have the annotated dataset.

Dataset

1) It’s about 1.200.000 pictures covering about 15k landmarks.
2) Each image may contain ≥ 0 landmarks on it.
3) Actual .csv-file:

   [image id], [image url], [landmark id]

Building a dataset

1) ~20 GPS tagged images
2) Online tour websites

1) + 2) → unsupervised clustering model (m.)

Photo sharing websites → (m.) → asking authors of images.

Preparing images

Problems:

1) Images can have different size

2) Images can have different quality

3) Images can have some noize (humans, which hide part of landmark and something like this)

4) Transform images to matrix

Different size of images

Solution:

1) Plot distribution of image`s sizes.

2) Split dataset images to classes based on our plot (If amount of images with small size less then 5%, for example, we will drop this class from our dataset)

3) Scale all images of one class to mean value size

Different quality

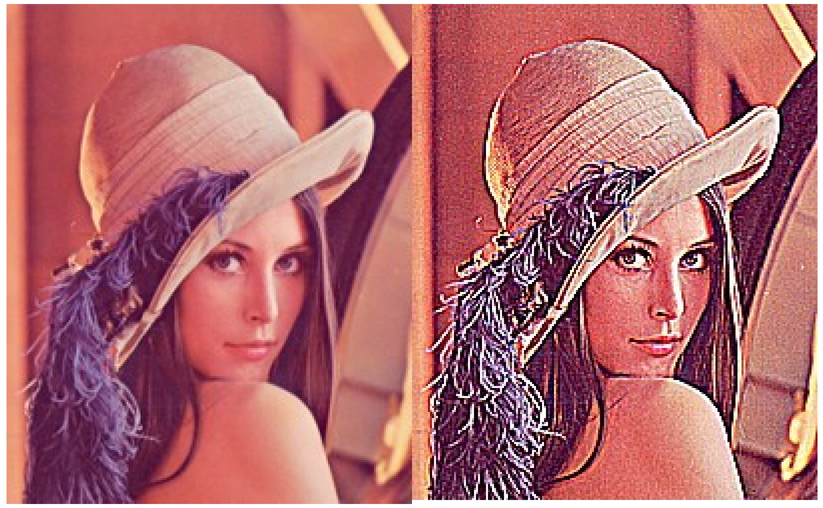
Solution:

Apply «Enhanced clarity filter» to each image

Filter Matrix:

-1	-1	-1
-1	9	-1
-1	-1	-1

Example:



Delete some noize

Now we don`t know how to delete this noize. But we are working on it. First idea was apply another NN which recognize humans on images and after recognizing fill humans by green pixels, for example.

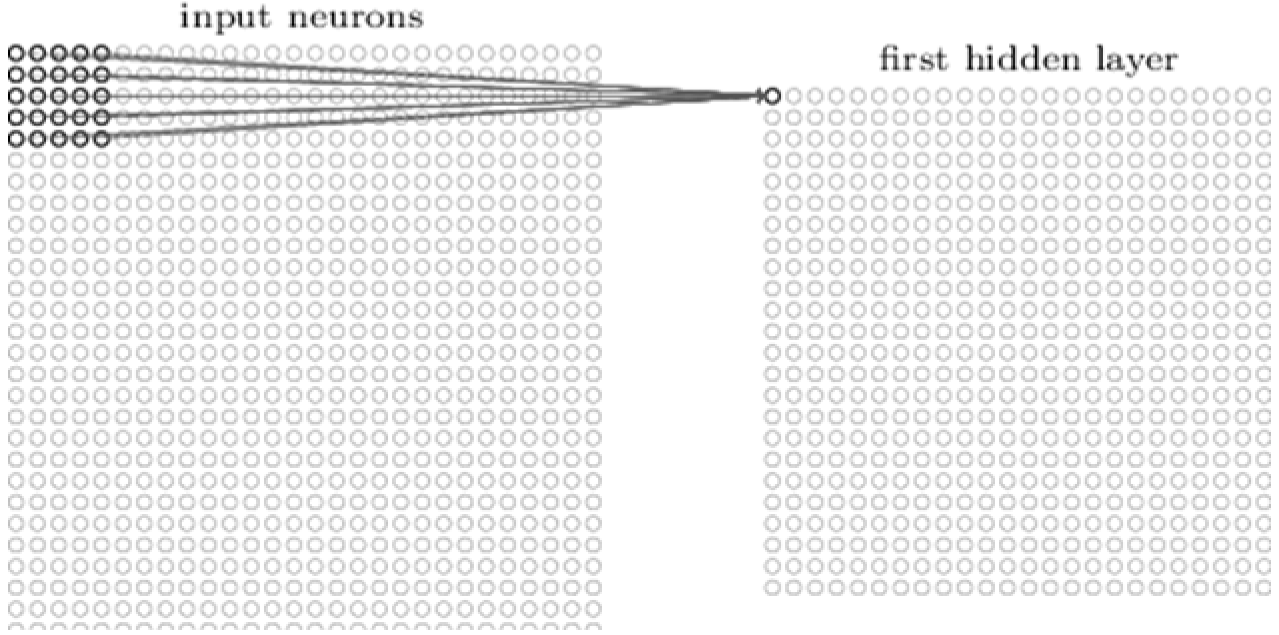
Transform image to matrix

We decided to transform prepared image to matrix of RGB colors. Each pixel will presented by three values:

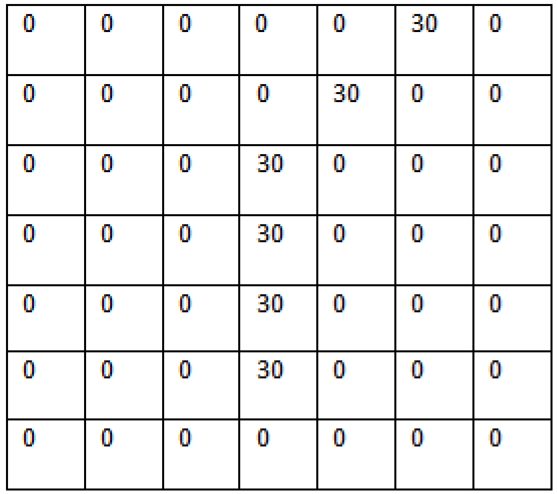
1) Value of red
2) Value of green
3) Value of blue

Convolutional neural networks

A usual neural network with “Convolutional ” layers

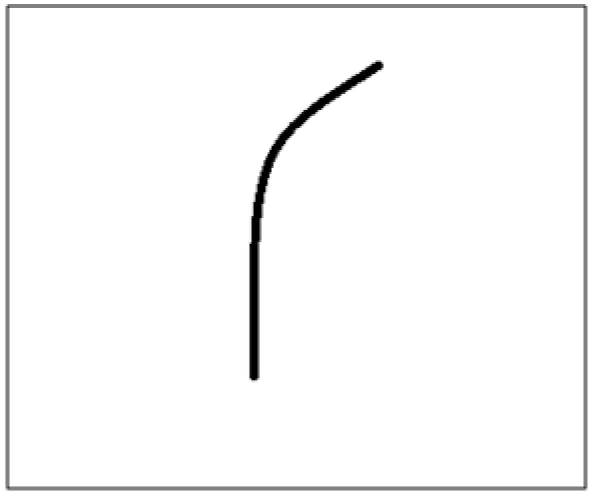


Filters



0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



Visualization of a curve detector filter

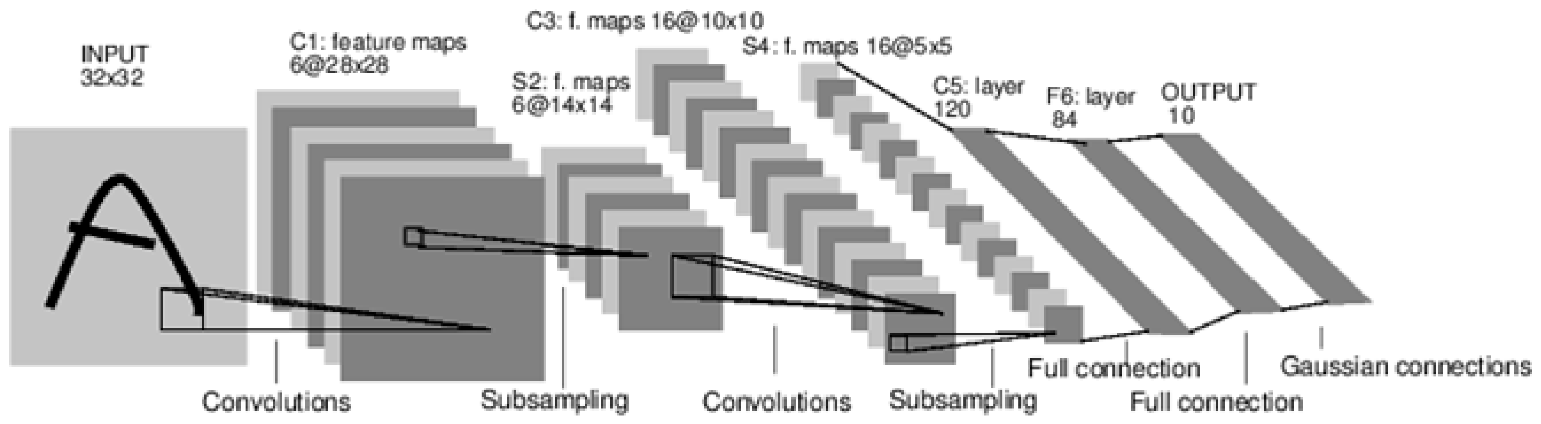
When the picture passes through one convolution layer, the output of the first layer becomes the input value of the 2nd layer. Now it's a bit harder to visualize. When we talked about the first layer we used only the data of the original image. But when we moved to the 2nd layer, the input value for it was one or more property maps - the result of processing the previous layer. Each set of input data describes the positions where certain basic characteristics occur on the source image.

Subsampling and full-connected layer

Now that we can bind high-level properties we could attach a fully connected layer at the end of the network. This layer takes input data and outputs an N-spatial vector, where N is the number of classes from which the program selects the desired one.

For example, if you want a program for recognizing numbers, N will have a value of 10, because the numbers are 10. Each number in this N-spatial vector represents the probability of the class. For example, if the result vector for the digit recognition program is [0 0.1 0.1 0.75 0 0 0 0 0.05], then there is a 10% probability that in the image "1", 10% the probability that the image "2", 75% probability - "3", and 5% probability - "9" (of course, there are other ways of presenting the conclusion).

Full architecture


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections