Improving CAPTCHAs with deep learning technique

Paolo Didier Alfano

Mine Altunay, Jeny Teheran, Scientific Computing Division

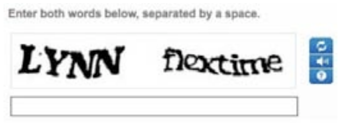
28 August 2018

CAPTCHA

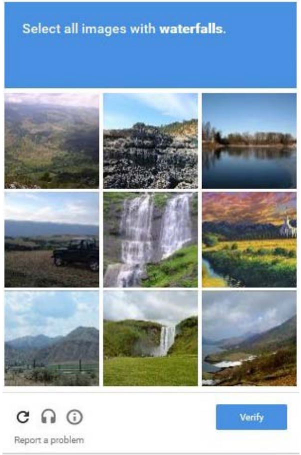
Completely **A**utomated **P**ublic **T**uring-test-to-tell **C**omputers and **H**umans **A**part

- Challenge-response test used in computer science to determine whether or not the user is a human
- Old CAPTCHAs consisted in typing one or more letters from a distorted image

- Google's reCAPTCHA: simple for humans, hard for bots.
- User must solve either an audio or visual challenge


Enter both words below, separated by a space.

CAPTCHA


Select all images with **waterfalls**.

reCAPTCHA

Main purpose

- Improve the security of CAPTCHAs by reducing their predictability trough machine learning techniques.

- reCAPTCHA’s weaknesses:
  - Hint: information we can use to reduce the domain size of the problem
  - Small amount of challenge categories

Breaking CAPTCHAs

- Different approches:
  - Cheap human labor
  - Exploiting bugs
  - Machine learning technique

- We decided to attack the reCAPTCHA system with one of the most popular and effective tool in machine vision context nowadays:

Convolutional Neural Network (CNN)

Deep learning in a nutshell

- *Deep learning*: machine learning algorithms that use a cascade of multiple layers of processing units for feature extraction. Each successive layer uses the output from the previous layer as input.

- What the algorithm learns? A set of parameters called **weights** used to recognize the feature

- How do it “learn” this weights? By updating them in order to **minimize** the **error** they commit

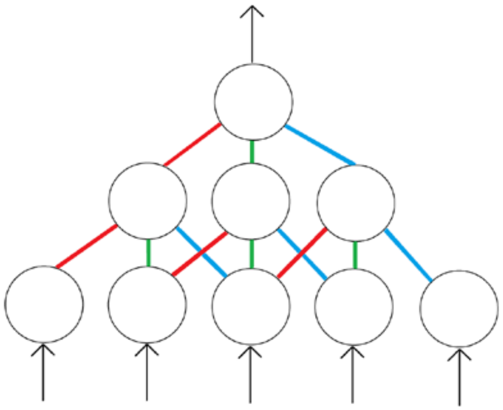
Convolutional Neural Network(CNN): overview

What are and why CNN?

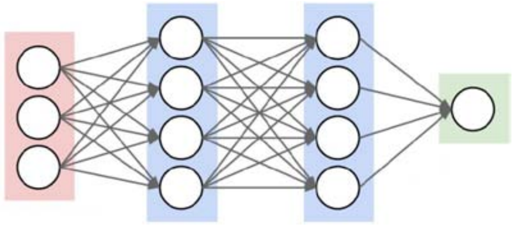
- CNN are sparse connected network: output at level n used as input to a little number of nodes at level n+1

- Different from fully connected net with too many parameters

High reduction of parameters number!

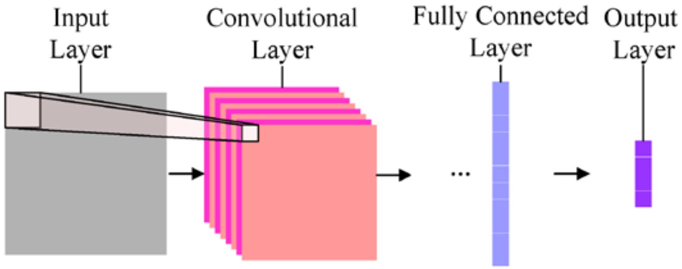


Convolutional



Fully connected

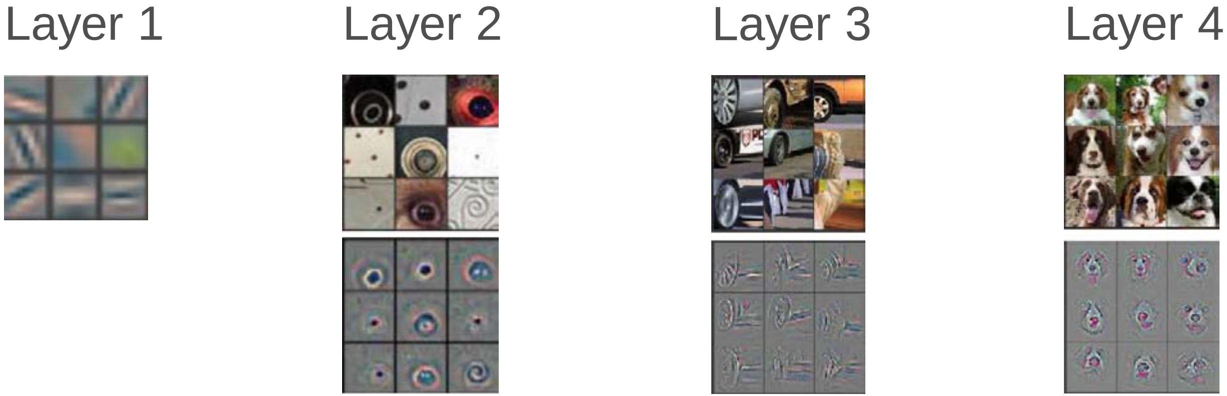
Convolutional Neural Network(CNN): structure



Essentially two parts:

- Convolutional part: many layers, recognize features
- Fully connected part: one layer that puts all together

The network recognize more complex feature in subsequent layers



Adaptive convolution

Convolutional filter (kernel): a matrix filled with weights. It's defined by:

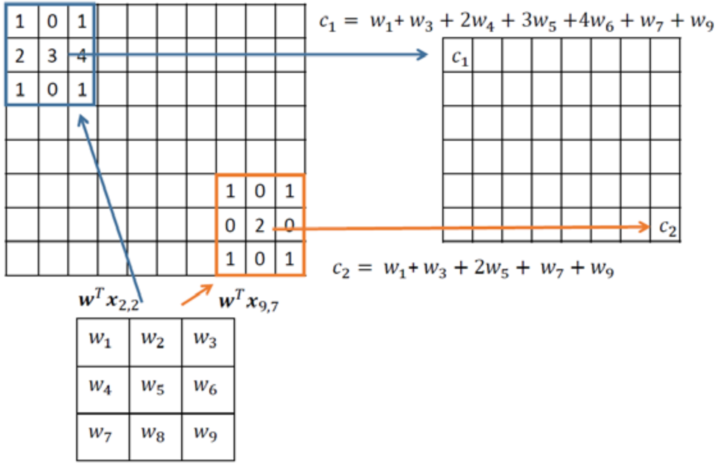
Width N $N \in \mathbb{N}, N \leq I_w$

Height M $M \in \mathbb{N}, M \leq I_h$

Stride S $S \in \mathbb{N}$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Convolve (shift) over the input: every time we shift the kernel, we obtain a new value in the output



$c_1 = w_1 + w_3 + 2w_4 + 3w_5 + 4w_6 + w_7 + w_9$

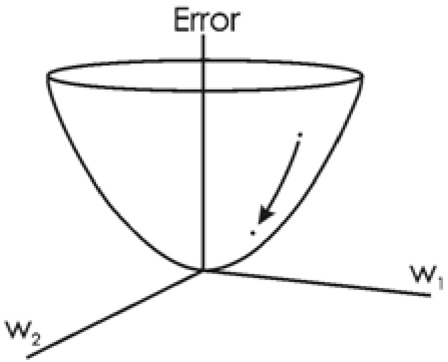
$c_2 = w_1 + w_3 + 2w_5 + w_7 + w_9$

At the end we obtain an *output*, the most probable according to the net

How the training works

Change the weights to minimize the error function:

$$E_{total} = \sum \frac{1}{2}(trueOutput - ourOutput)^2$$

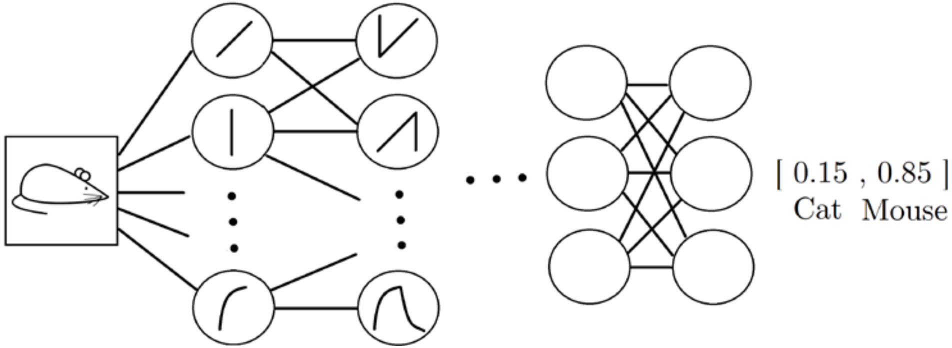


Update weights trough Gradient descent technique

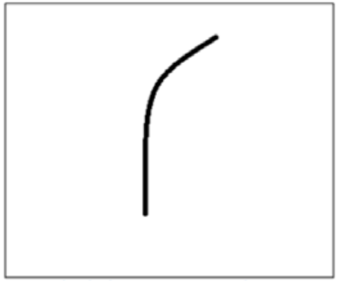
$$w' = w - \eta \frac{dE_{total}}{dW}$$

Does it works?

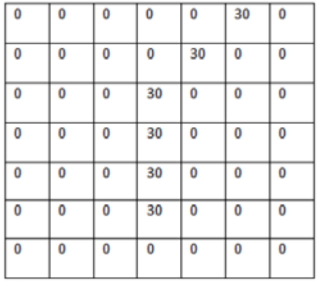
Suppose after the training we obtain the CNN shown below



Let’s examine one of the filters learned:



Visual filter



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 represented filter

Does it works?

Input very different from the filter:

Convolution value:

C=**0**

Very low!

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

*

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

Input similar to the filter:

Convolution value:

C =(50x30) + (50x30) + (50x30) +
+ (20x30) + (50x30) = **6600**

Very High!

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

*

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

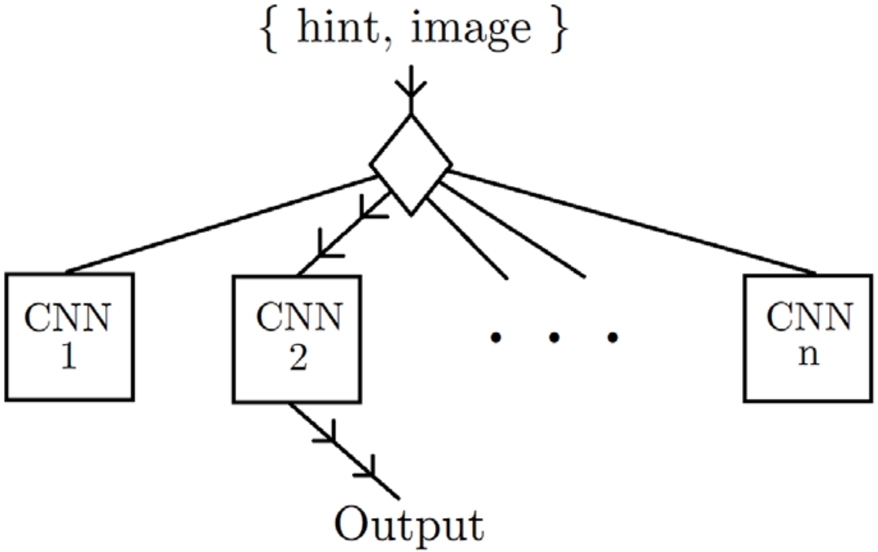
Breaking reCAPTCHA: general architecture

Train ***n*** different CNNs, one for every category (Roads, Crosswalks..)

Input: the hint, the image

Pass image to the proper CNN using the hint

Output: probability that image contains the entity the hint refers to

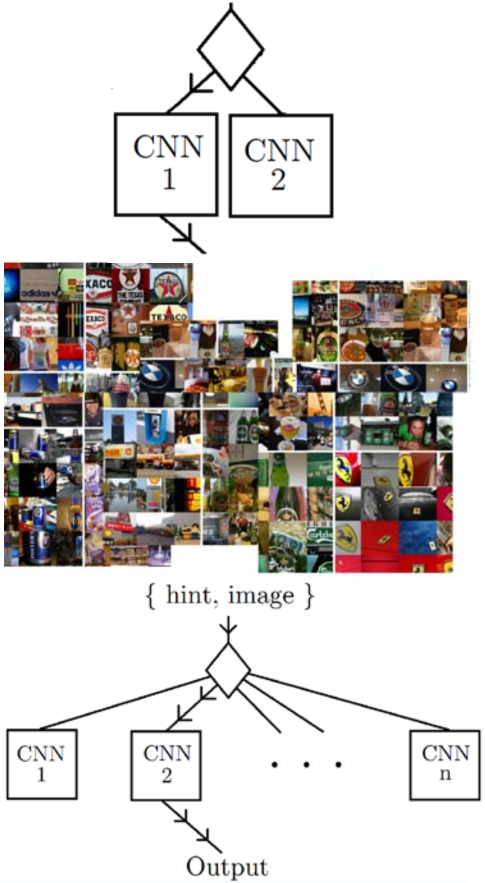


Next weeks

1. Implement prototype of the architecture
2. Test over a reduced ReCaptcha challenge

3. Find dataset for every category

4. Implement the full architecture
5. Test over ReCaptcha challenge
6. Make recommendations to improve the CAPTCHA mechanism



References

- A Beginner's Guide To Understanding ConvolutionalNeural Networks,
  AditDeshpande

- Convolutional Neural Networks,
  Davide Bacciu

- Visualizing and Understanding ConvolutionalNetworks,
  Matthew D. Zeilerand Rob Fergus

- Deep learning, chapter 11,
  Ian Goodfellowand YoshuaBengioand Aaron Courville