**Water Marks removal using pix2pix**

Gili Katabi

Lior Schneider

Ayelet Tal

Contents

[Motivation 3](#_Toc526614021)

[Background 4](#_Toc526614022)

[Method 6](#_Toc526614023)

[Results 8](#_Toc526614024)

[Git 12](#_Toc526614025)

# Motivation

These days, which characterized by loads of fast access information in general and more specifically by many images, it is a very common mechanism to use watermark logo as a mean to mark and protect the image owners’ copyrights.

The reason that this mechanism is so popular is that it enables the consumers to enjoy the image while it ensures that there will be no unauthorized or unlicensed use of it.

As always, where there are mechanisms to protect, there will be someone who tries to break it.

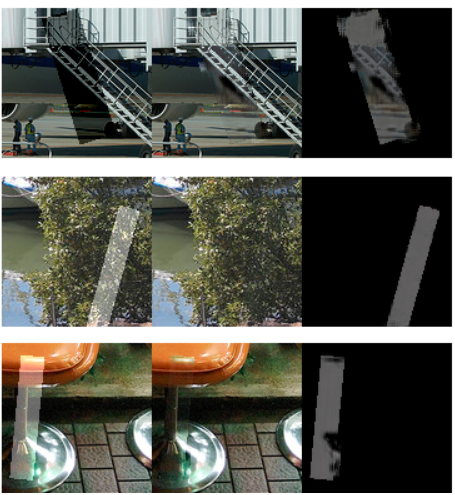
Our project is focusing on try to use a well-known algorithm (pix2pix) to break the watermark mechanism.

This project is not barring a great scientific progression, but it is a great mean to learn the basics of ML and GAN.

# Background

Watermark removal is not a very popular research area. Having that, we did find some methods to use neural networks to remove watermark from images.

<https://github.com/marcbelmont/cnn-watermark-removal> which uses a pre-given rectangle shaped mask in the size of the watermark, than it uses dilated convolution and output the image.



1. Input image b. fake image c. watermark extraction

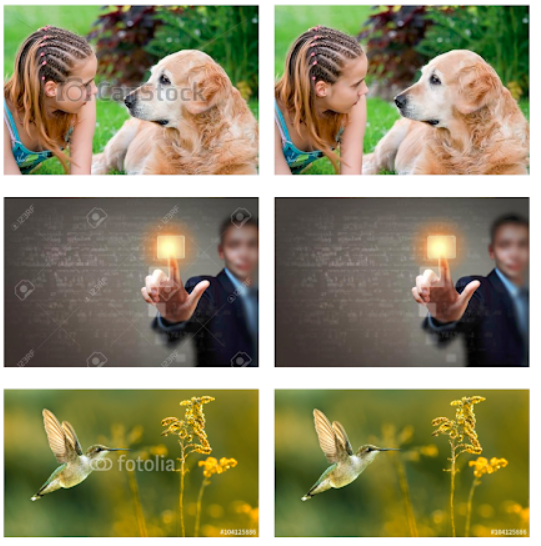
Its results are not that good, and it is very limited due to how this network was implemented.

<https://ai.googleblog.com/2017/08/making-visible-watermarks-more-effective.html>

Google took it another level and characterized the structure of many watermarks.

To do so, they took datasets with the same watermark and used median image + multi-image matting method to estimate how the watermark should be.

By doing so, they could assess which exact watermark (or a very similar one) is being used, then they could detect its pattern and characterization (shadows and color gradient) and remove it.



1. Watermarked image b. faked image

They get extremely good results.

Even though google found a great way to “hack” the watermark system, our main goal is to use this problem to learn and use the basics of machine learning area and therefore we tried to use a different solution.

# Method

We have examined a few known projects that might work for our goal of a watermark removal:

* Attentive Generative Adversarial Network for Raindrop Removal from a Single Image
* Single Image Reflection Removal Using Deep Encoder-Decoder Network
* Pix2pix algorithm

All the above-mentioned researches are transferring images from domain A to domain B while trying to characterize the differences between these domains.

Due to our very little experience working with GAN, we decided to work with the most known and documented algorithm – pix2pix.

So, most of our work would be to try and use this algorithm to get a good results of watermark removal.

First, we created the data set. The data set we use originated in the pix2pix dataset where we created the same images with watermark. During the project we changed the places of the watermarks (fixed place \ random places), and we changed the size of the dataset as describes below.

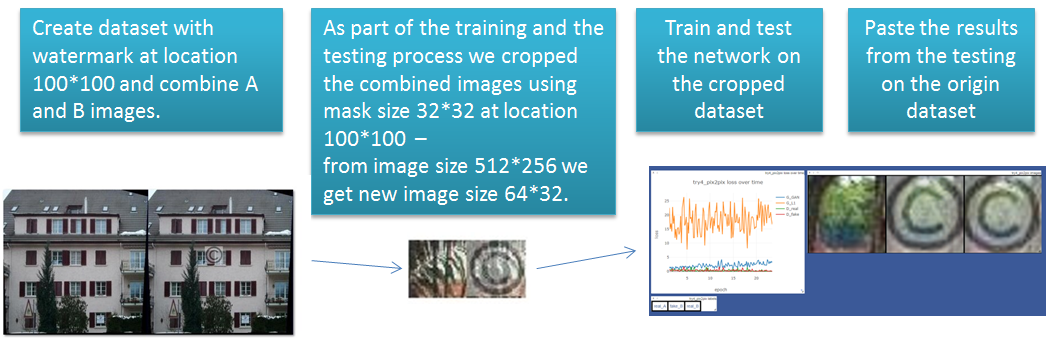
at the beginning we want to get familiar with running the pix2pix process and we try to find out how big the dataset should be. We start training the network on 100 images. This run had poor results, so we enlarge the dataset to 1000 and then to 3000 images. This time we got much better results.

Than we tried to see if this method is affected by randomization of the watermark, we noticed that when we trained the network on many images, the random location of the watermark did not affect the results that we got.

And last, we tried to use a mask which will tell the network exactly where the watermark is and will work on that specific area. Unfortunately, this stage produced very bad results. Probably because the network did not have the full picture but only the relevant cropped area of the image, it was very hard for it to “guess” the missing information.

Eventually, due to server connectivity issues we couldn’t manage to complete the code and paste the “fake” results back in the original image to get a full view for the results.

Explanation and a drawing of what the architecture looks like :



# Results

As a first stage, while training the network on 60 images out of dataset in size 100, we got unsatisfying results:

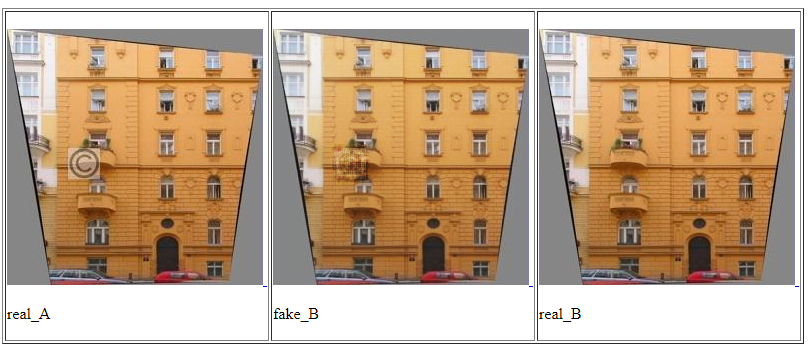
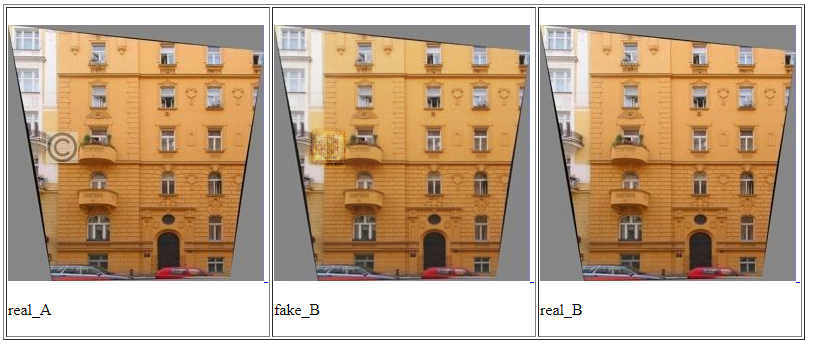
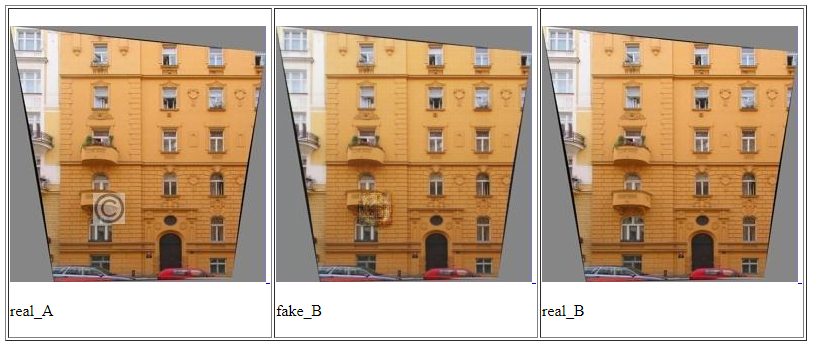
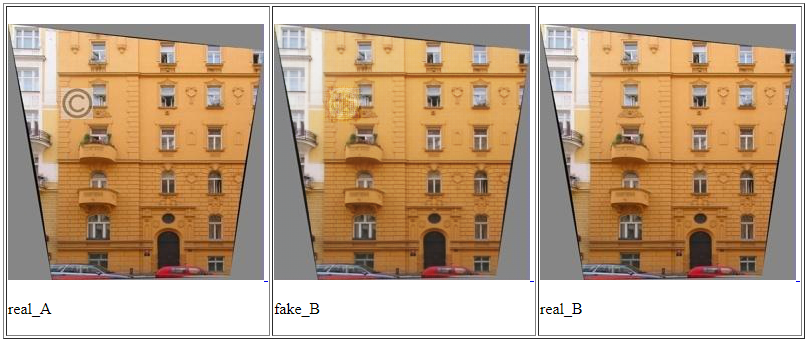


We can see that the watermarks are in random places but the network for some reason tries to fix the same incorrect place.

After that we training the network on 1800 images out of dataset in size 3000, random locations of the watermark and got much better results:

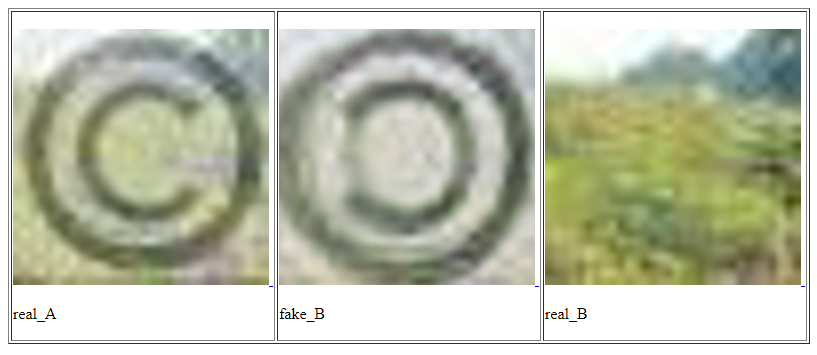


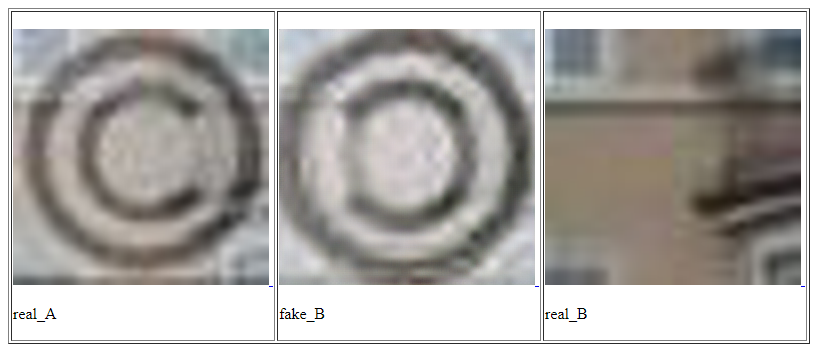
Thistime, as mentioned above we place the watermarks in random places:



For a final check, we have tested it with a mask for the specific location of the watermark.

As mentioned above, the results were not good, probably because the model didn’t had enough data to use for the convolution. We got the following results for example:





The entire images with the network results:

# Git

The git attached contains:

* pix2pix code with our changes in files :
  + test.py
  + train.py
  + util.py
* Our script for creating the dataset with the watermarks.

We changed the script according to what we needed (amount \ random) so attached the latest version we used.

* Our results:
  + First run on small dataset.
  + Second run on big random dataset.
  + Third run with mask.