CENTRE BORELLI

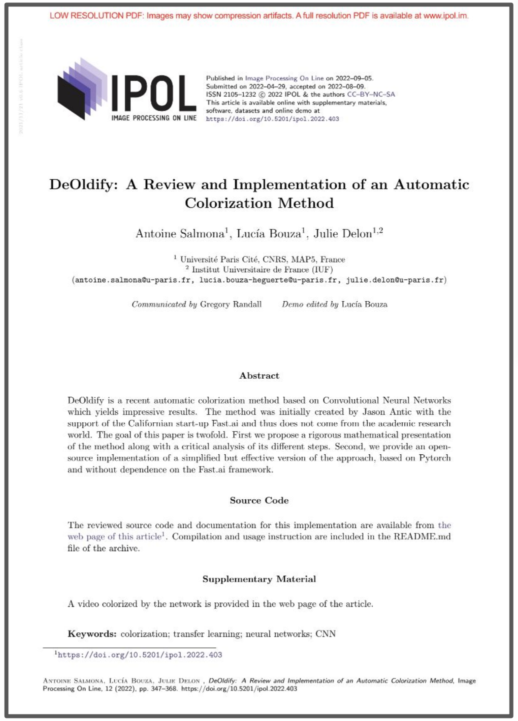
IPOL Journal - Image Processing On Line
www.ipol.im

IPOL MLBriefs 3
31/05 — 01/06

Day 1

Publication to IPOL/MLBriefs
How, why?

What is an IPOL publication?



LOW RESOLUTION PDF: Images may show compression artifacts. A full resolution PDF is available at www.ipol.im.

Published in Image Processing On Line on 2022–09–05.
Submitted on 2022–04–29, accepted on 2022–08–09.
ISSN 2105–1232 © 2022 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials, software, datasets and online demo at
https://doi.org/10.5201/ipol.2022.403

DeOldify: A Review and Implementation of an Automatic Colorization Method

Antoine Salmona[1], Lucía Bouza[1], Julie Delon[1,2]

[1] Université Paris Cité, CNRS, MAP5, France
[2] Institut Universitaire de France (IUF)
(antoine.salmona@u-paris.fr, lucia.bouza-heguerte@u-paris.fr, julie.delon@u-paris.fr)

Communicated by Gregory Randall *Demo edited by* Lucía Bouza

Abstract

DeOldify is a recent automatic colorization method based on Convolutional Neural Networks which yields impressive results. The method was initially created by Jason Antic with the support of the Californian start-up Fast.ai and thus does not come from the academic research world. The goal of this paper is twofold. First we propose a rigorous mathematical presentation of the method along with a critical analysis of its different steps. Second, we provide an open-source implementation of a simplified but effective version of the approach, based on Pytorch and without dependence on the Fast.ai framework.

Source Code

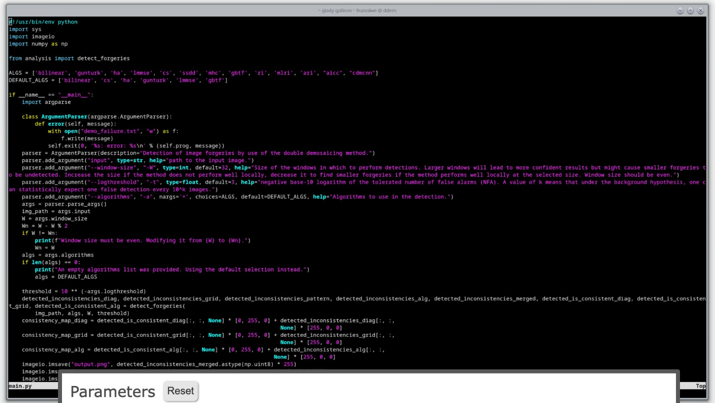
The reviewed source code and documentation for this implementation are available from the web page of this article[1]. Compilation and usage instruction are included in the README.md file of the archive.

Supplementary Material

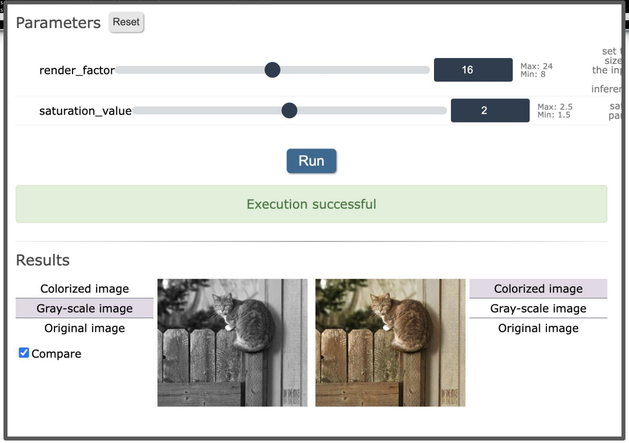
A video colorized by the network is provided in the web page of the article.

Keywords: colorization; transfer learning; neural networks; CNN

[1] https://doi.org/10.5201/ipol.2022.403



```
#!/usr/bin/env python
import sys
import imageio
import numpy as np

from analysis import detect_forgeries

ALGS = ['bilinear', 'gunturk', 'ha', 'lmmse', 'cs', 'ssdd', 'mhc', 'gbtf', 'ri', 'mlri', 'ari', "aicc", "cdmcnn"]
DEFAULT_ALGS = ['bilinear', 'cs', 'ha', 'gunturk', 'lmmse', 'gbtf']

if __name__ == "__main__":
    import argparse

    class ArgumentParser(argparse.ArgumentParser):
        def error(self, message):
            with open("demo_failure.txt", "w") as f:
                f.write(message)
            self.exit(0, '%s: error: %s\n' % (self.prog, message))
    parser = ArgumentParser(description="Detection of image forgeries by use of the double demosaicing method.")
    parser.add_argument("input", type=str, help="path to the input image.")
    parser.add_argument("--window-size", "-W", type=int, default=32, help="Size of the windows in which to perform detections. Larger windows will lead to more confident results but might cause smaller forgeries t
o be undetected. Increase the size if the method does not perform well locally, decrease it to find smaller forgeries if the method performs well locally at the selected size. Window size should be even.")
    parser.add_argument("--logthreshold", "-t", type=float, default=3, help="negative base-10 logarithm of the tolerated number of false alarms (NFA). A value of k means that under the background hypothesis, one c
an statistically expect one false detection every 10^k images.")
    parser.add_argument("--algorithms", "-a", nargs='+', choices=ALGS, default=DEFAULT_ALGS, help="Algorithms to use in the detection.")
    args = parser.parse_args()
    img_path = args.input
    W = args.window_size
    Wn = W - W % 2
    if W != Wn:
        print(f"Window size must be even. Modifying it from {W} to {Wn}.")
        Wn = W
    algs = args.algorithms
    if len(algs) == 0:
        print("An empty algorithms list was provided. Using the default selection instead.")
        algs = DEFAULT_ALGS

    threshold = 10 ** (-args.logthreshold)
    detected_inconsistencies_diag, detected_inconsistencies_grid, detected_inconsistencies_pattern, detected_inconsistencies_alg, detected_inconsistencies_merged, detected_is_consistent_diag, detected_is_consisten
t_grid, detected_is_consistent_alg = detect_forgeries(
        img_path, algs, W, threshold)
    consistency_map_diag = detected_is_consistent_diag[:, :, None] * [0, 255, 0] + detected_inconsistencies_diag[:, :,
                                                                                    None] * [255, 0, 0]
    consistency_map_grid = detected_is_consistent_grid[:, :, None] * [0, 255, 0] + detected_inconsistencies_grid[:, :,
                                                                                    None] * [255, 0, 0]
    consistency_map_alg = detected_is_consistent_alg[:, :, None] * [0, 255, 0] + detected_inconsistencies_alg[:, :,
                                                                                  None] * [255, 0, 0]
    imageio.imsave("output.png", detected_inconsistencies_merged.astype(np.uint8) * 255)
    imageio.ims
    imageio.ims
main.py                                                                                                   Top
```

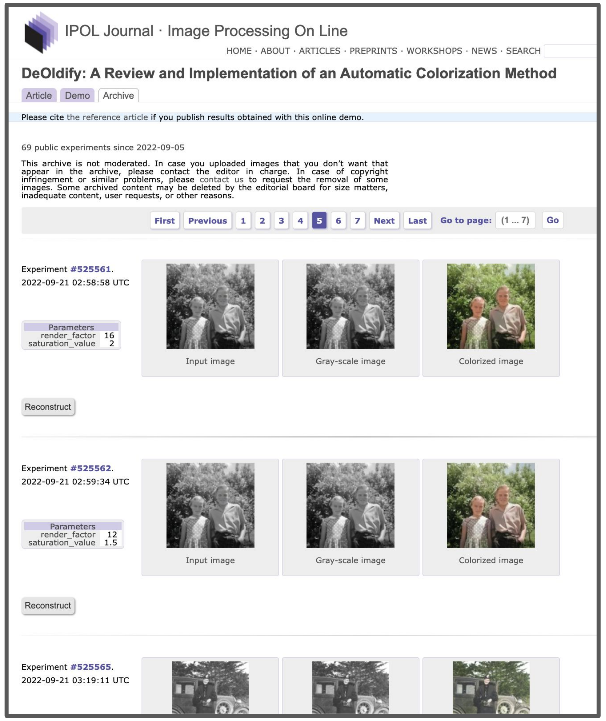


Parameters

render_factor	16	Max: 24 Min: 8
saturation_value	2	Max: 2.5 Min: 1.5

Execution successful

Results



IPOL Journal · Image Processing On Line

HOME · ABOUT · ARTICLES · PREPRINTS · WORKSHOPS · NEWS · SEARCH

DeOldify: A Review and Implementation of an Automatic Colorization Method

Please cite the reference article if you publish results obtained with this online demo.

69 public experiments since 2022-09-05

This archive is not moderated. In case you uploaded images that you don't want that appear in the archive, please contact the editor in charge. In case of copyright infringement or similar problems, please contact us to request the removal of some images. Some archived content may be deleted by the editorial board for size matters, inadequate content, user requests, or other reasons.

Experiment #525561.
2022-09-21 02:58:58 UTC

Parameters	
render_factor	16
saturation_value	2

Experiment #525562.
2022-09-21 02:59:34 UTC

Parameters	
render_factor	12
saturation_value	1.5

Experiment #525565.
2022-09-21 03:19:11 UTC

What is an IPOL publication?

- The code to a method, written or not by you
- An online demo using this code, so people can try it with their own data
- An article to accompany the method
  - Not a traditional conference/journal article that describes a novel method and establish it best in some way, with mainly quantitative experiments on benchmark datasets
  - A more or less extensive description of the method (depending on the track)
  - Mainly qualitative experiments to highlight the strengths and shortcomings of the method and the impact of the different parameters, if any
  - The method does not have to beat the SOTA to be published: honest review
  - The method does not have to be new and unpublished
- An archive that populates with everyone's experiments

IPOL or MLBriefs tracks

MLBriefs track	**IPOL track**
Typically short paper (about 5 single-column pages)	More extensive paper
Relatively brief description of the method (Mainly qualitative) experiments to highlight the strengths and flaws of the method and the impact of the different parameters.	The method is fully described in pseudo-code, which corresponds to the code (same functions). If any learning, it is fully described, ideally retrained by the authors. Experiments remain, they can be more expansive.
Same method as the original.	Improvements on the method can be proposed
The code can be in any language.	Python, C, C++, Matlab/octave (possible exceptions)
The article and demo are peer-reviewed. The code is **not** peer-reviewed.	The article, demo and code are peer-reviewed. **Guarantee of reproducibility:** The code corresponds exactly to the pseudocode.
Ideal on methods that are not yours: easy to prepare without delving into the code	Ideal for your own works or methods you studied and understand very well: the pseudo-code is then easy to write

Why should I submit to IPOL/MLBriefs?

- For your own methods:
  - The online demo and article increase the visibility of your method
  - Easily show your methods to partners and clients, they can try it on their own data
  - IPOL articles act as a proof that your code does exactly what the article says it does
- For other methods:
  - You have to study and experiment with state-of-the-art methods for your own research
  - ⇒ IPOL/MLBriefs articles reward you with a publication for work you have to do anyways

In both cases, IPOL and MLBriefs reward and enable sharing work you have to do anyways but generally goes unpublished

Preparations: Understand the article and code

- ❏ Is the code under an open-source license? If not, are you at least allowed to use it online?
  - ❏ IPOL track: you need to be able to redistribute the code, normally under an open-source license. This is not a strict requirement in the MLBriefs track (but is still preferred)
  - ❏ If no licence are provided: ask the authors for authorization
- ❏ Understand how to run the code, how it is compiled (if relevant)
- ❏ Prepare the following information:
  - ❏ Language, package requirements and versions
  - ❏ What are the inputs?
  - ❏ Which parameters do you want to expose to the user? Which possible/reasonable values? Expose them to the command-line call of the code

Timetable

TODAY

- Morning
  - A short introduction to IPOL/MLBriefs
  - Authors present their intended works
  - Tutorial: How to create an IPOL demo?
    - Authors who already know how to create a demo can start working in the workshop rooms
    - Instructions are the same as for MLBriefs 2 (GitHub, control panel, nextcloud, ...)
- Lunch: 13:00, 1E29
- Afternoon
  - Authors work on their demo in the workshop rooms
  - Coffee break at 16:00, delivered directly to the working spaces

Tomorrow

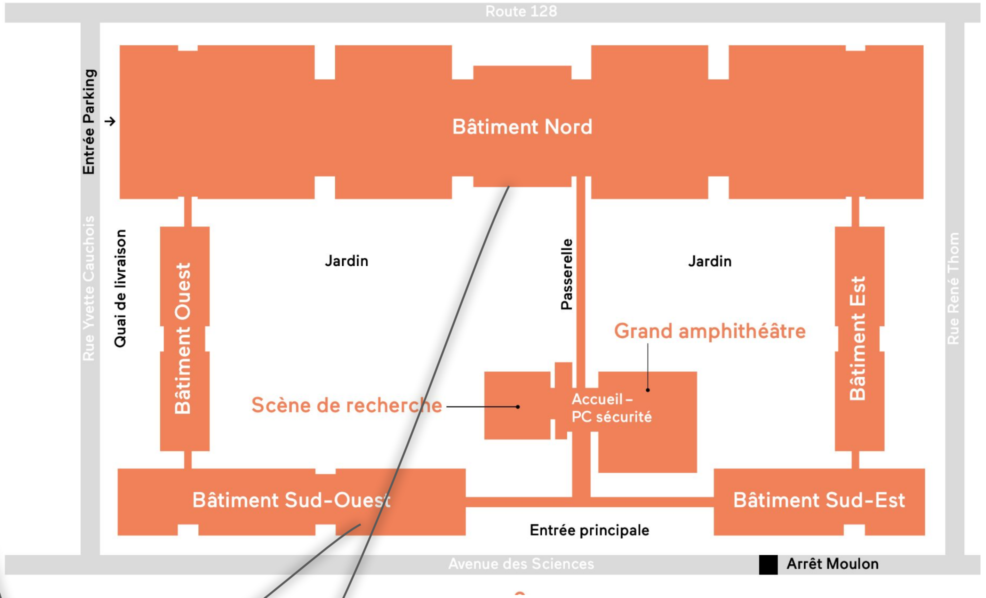
- Morning: Presentation by invited speakers
  - Pascal Monasse
  - François Role
  - Pablo Musé
  - Mercedes Marzoa Tanco
- Lunch: 12:00, 1E29
- Afternoon
  - Authors work on their demo in the workshop rooms
  - Coffee break at 15:30, delivered directly to the working spaces

Thursday, 1st June

- Progress report by each author/group
  - Quick reminder of the initial objective
  - Is the demo working yet?
  - Difficulties encountered. Have solutions been found? If so, which?
  - Feedbacks on the workshop and on IPOL
  - About 3 minutes each
- Lunch: 13:00, 1E29
- Afternoon
  - Authors work on their demo in the workshop rooms
  - Coffee break at 15:30, delivered directly to the working spaces

Where?

- Morning plenary meetings:
  - Today, Thursday: 1B26 (here)
  - Tomorrow: 1B36 (next door)
- Lunches: 1E29 (in front of her
- Afternoon in the workshop rooms: 2U42, 2U47, 2S41, 2S48 (2nd floor)



Join the workshop slack!

https://join.slack.com/t/mlbriefsworkshop/shared_invite/zt-1w3jo4e8l-zSBR3OXyKfL4YvtGHjzDtA