Using Machine Learning to Process Filters and Mimic Instant Camera Effect

Deirdre Chong, John Farhad Hanifzai, Hassan Adam, Jorge Garcia, and Jorge Ramón Fonseca Cacho

Abstract

In this paper we use Machine Learning to convert images taken with an iPhone camera and visually alter them to appear as if taken with a Leica Sofort Instant Camera, more commonly known as the Polaroid look. While such image filters already exist and are highly effective, they function using ad-hoc techniques. Our goal is to achieve similar results by having a model learn what the Polaroid look is on its own and how many image pairs are required to train it. We found that using linear regression we need, on average, 800 images before the model began displaying good consistent results while using Pix2Pix (Isola et al., Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1125–1134, 2017) (Conditional Adversarial Networks) and CycleGAN (Goodfellow et al., Generative adversarial nets. In Ghahramani Z, Welling M, Cortes C, Lawrence ND, Weinberger KQ (eds) Advances in neural information processing systems, vol 27, pp 2672–2680. Curran Associates, Inc., Red Hook, NY, 2014 [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf) (Generative Adversarial Networks) only required 500 images.

Keywords

Machine learning · Polaroid · Computer vision · Deep learning · Image processing

D. Chong · J. F. Hanifzai · H. Adam · J. Garcia
J. R. Fonseca Cacho (✉)
Department of Computer Science, University of Nevada, Las Vegas, Las Vegas, NV, USA
e-mail: Jorge.FonsecaCacho@unlv.edu

57.1 Introduction

Machine Learning has seen a massive demand in computer applications over the past few years. From self-driving cars to predicting the future, Machine Learning makes it possible for computers to solve problems more quickly and efficiently than humans can [1]. Our research aims to apply Machine Learning in image processing. The potential of Machine Learning in image processing is limitless [2]. Machine Learning can be implemented in image processing in various ways, including photo enhancement [3]. Image-to-image translation has been implemented in intriguing ways, such as converting a stock photo to a canvas, mimicking Van Gogh and Monet [4]. This research aims to apply Machine Learning to develop a model that will convert a standard unprocessed photo into a photo that visually looks as if taken with an instant camera, Leica Sofort. This camera has a unique look that many people know as the 'Polaroid' look (Fig. 57.3). There are pre-existing image filters that replicate instant camera images, but none accurately capture all the differences and subtleties [5]. Image filters made with Machine Learning can replace outdated image processing approaches and lead endless image processing possibilities with the right training data set [6].

57.2 Background

Usually, image-to-image translation utilize pixel-to-pixel models to predict the corresponding output image. Translation in this sense is transposing from a representation or a format to another, this could either be an RGB image, edge maps, label maps, etc. These image translations can be implemented with many different types of algorithms, one is linear regression. This research uses linear regression models

S. Latifi (ed.), *ITNG 2021 18th International Conference on Information Technology-New Generations*, Advances in Intelligent Systems and Computing 1346,
https://doi.org/10.1007/978-3-030-70416-2_57

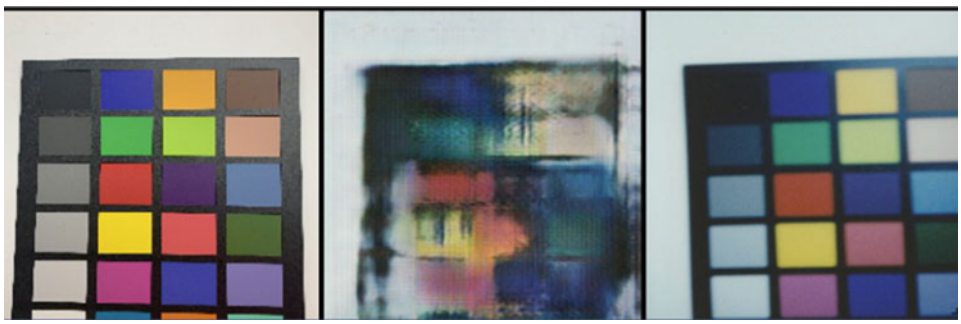

Fig. 57.1 Pix2Pix Real A—Generated B—Real B [4]

to parse each of the RGB values per pixel of the paired image data sets, as per the following,

$$R_{ij} = w_1 old R_{ij} + w_2 old G_{ij} + w_3 old B_{ij} + w_0$$
$$G_{ij} = w_1 old R_{ij} + w_2 old G_{ij} + w_3 old B_{ij} + w_0$$
$$B_{ij} = w_1 old R_{ij} + w_2 old G_{ij} + w_3 old B_{ij} + w_0$$

Given a pixel at coordinate (i,j), we independently compute three regressions for each color. Each color has three weights that represent the old RGB value of the original pixel and the bias. The three weights of each of model are independent of the weights of the other models. The reason RGB values are utilized to compute each color separately is to try to find a relation between the other colors and how each individual color is affected.

Previous image-to-image translation research has shown that it is possible to predict pixel structure using convolutional neural networks (CNNs) [7]. These solutions work by having an image with a missing region, processing it through the convolutional neural network and regress the missing pixel values. These solutions are trained in a completely unsupervised manner. A similar image-to-image translation solution can be seen using conditional adversarial networks as seen on the paper. Conditional adversarial networks efficiently learn the mapping and learn a loss function to train the model [4].

Pix2Pix aims to solve the same problem of translating an input image into its corresponding output. We explore the use of generative adversarial networks (GANs) for these tasks. GANs are suited in the conditional setting because they can automatically learn a loss function to classify if an image is real and at the same time, train a generative model to minimize this loss. Figure 57.1 shows the early stages of training the Pix2Pix generative adversarial network.

It is worth noting that the results that will eventually come from processing images though our several models, have to be reviewed by a human to interpret and compare results. Even though this is inherently objective, as there is no algorithmic way to tell if the image transformations are effective, we established a criteria to determine if an image passed the transformation or is rejected. For this criteria, we consider attributes like:

- Image maintains its colors after transformations
- Image is still discernible without major distortions
- Image still resembles original without loss of detail
- Image still maintains most of the original colors
- Image kept similar proportions

57.3 Methods

Machine Learning requires enough data to teach the computer how to convert the given input into the desired output. The dataset is split into two parts: one that contains normal images and ones that contain our desired output. The more examples we have, the better it can detect subtle unexplainable patterns.

For this research we devised a very specific workflow. The first step was getting a dataset of images that would eventually go through the regression model as a first attempt at image transformation. The regression model would then take the R, G, or B values from the whole dataset. We would then need a new script to extract the RBG values from these images in the dataset, and lastly, the linear regression script.

As mentioned before, the main step for creating a dataset was downloading 6000 images. We requested downloads from the Bing search engine's API with the help of libraries like OpenCV, to test if the images were valid or corrupted, otherwise they would have to be deleted as a first step into normalization. Having created a big enough batch of images to process, we then had a new script to resize all images to 300×300 pixels, and we edited the images by introducing contrast and color changes and altered them into different styles using image processing tools like ImageMagik and OpenCV.

```
#Dataset config snippet

# train and test split for dataset
if argv.split is not None and
          argv.split < 1:

    split_ratio = argv.split
else:
    split_ratio = .75

img_count = 0
```

```
for photo in img_A:
    if photo.lower().endswith(
              ('.png', '.jpg', '.jpeg')):

        img_count += 1

img_split = math.ceil(img_count
        * (split_ratio))


...


        exit()
```

This step was necessary to reduce film cost by estimating the minimum number of images required for the Machine Learning algorithm to produce useful outputs. After we found the optimal number of images, we collected Polaroid images and digital images, then matched the pixels using Adobe Photoshop.

Afterwards, we read the RGB values of each photo and save them at a main NumPy array with each image image array inside it.

```
#RGB Numpy Array creation code snippet

image = np.array([])
r_ds = np.array([])
g_ds = np.array([])
b_ds = np.array([])


currentDirectory = os.getcwd()


for filename in os.listdir("./"):
    if filename.endswith(".jpg"):
        directory = ''
        fileDir = cv2.imread(os.path.join
                    (directory, filename))
        #print('Working on file:', fileDir)
        img = cv2.cvtColor
          (fileDir, cv2.COLOR_BGR2RGB)/255

        r_val = img[:, :, 0]
        g_val = img[:, :, 1]
        b_val = img[:, :, 2]


        r_ds = np.append(r_ds, r_val)
        g_ds = np.append(g_ds, g_val)
        b_ds = np.append(b_ds, b_val)

        index += 1
        if index == 1000:
            break;
        continue


np.save('r.npy', r_ds)
np.save('g.npy', g_ds)
np.save('b.npy', b_ds)
```

Finally, we used our polaroid dataset on machine learning models like Linear Regression, GANs (Generative Adversarial Networks) such as Cycle-GANs, and Pix2Pix GANs. Generative Adversarial Networks, more broadly the adversarial nets frameworks were chosen because they generative models sidestep the difficulties they usually convey, like the difficulty to approximate untraceable probabilistic computations that come from estimations. This framework is interesting because it places a model against an adversary model that learns to determine whether a sample if from the first model or from dataset. This competition between the two models usually produces improvements in their results [8].

57.4 Test Results

As our initial test, we applied linear regression to predict images on our test datasets. The paired images are resized to the same resolution and RGB values are extracted from the image as mentioned previously. These values are then fitted through our linear regression script. Figure 57.2 shows the sample images and the result using linear regression for 800 training images.

```
# Linear regression R G B

print('loading ....')
x1 = np.load('./npy/r.npy')
x2 = np.load('./npy/g.npy')
x3 = np.load('./npy/b.npy')

y1 = np.load('./npy/r1.npy')
y2 = np.load('./npy/g1.npy')
y3 = np.load('./npy/b1.npy')


print('dataframe ....')
data = pd.DataFrame
        (np.c_[x1,x2,x3,y1,y2,y3])

print("Data: \n",data)
#print('reshaping ....')

X = data[[0,1,2]]
print("X: \n", X)

y = data[[3]]
print("y: \n", y)

print('splitting ....')
X_train, X_test,y_train,y_test = t
        rain_test_split(X,y,test_size=.
                3,random_state=1)

print('fitting ....')
simplemodel = LinearRegression()
simplemodel.fit(X_train, y_train)
```

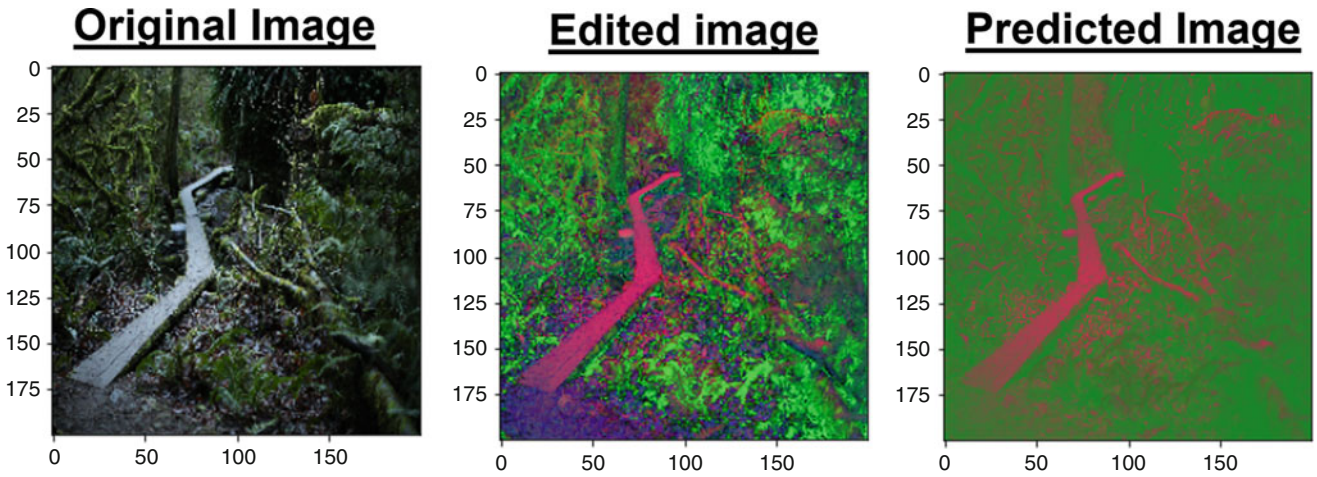

Fig. 57.2 Side-by-side of a sample image that was then modified and fed to our Linear Regression model and the resulting image

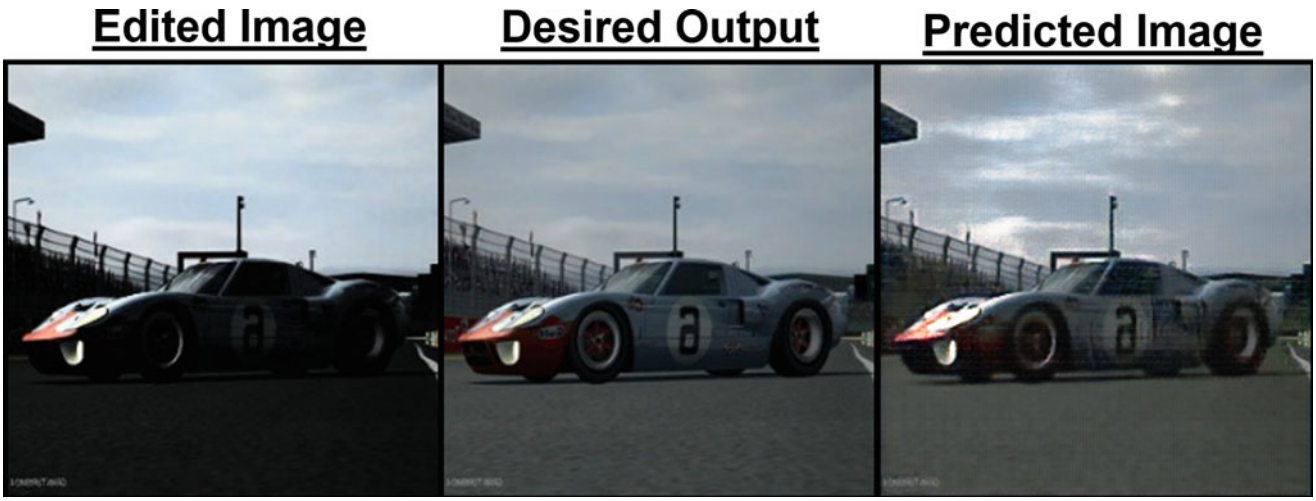

Fig. 57.3 Pix2Pix on contrast test dataset [4]

The next step on our test is to use Pix2Pix GAN on our test dataset. The contrast is edited with ImageMagik and used as the original input image. The model does a good job recovering back details that were lost when the contrast was heightened. Figure 57.3 shows the results of the model after 50 epochs trained with 500 paired images.

57.5 Results

The trial run with digitally edited images resulted in successful prediction in Linear Regression with 800 images. Machine Learning models like Pix2Pix, predicted the edited styles with 500 images.

Since Polaroid film was scarce and expensive, we used 150 Instant Camera Images with a corresponding digital image. The images were then fitted through the Linear Regression Model.

Figure 57.4 shows the prediction using linear regression using digital image and polaroid image datasets. The results alter the color to appear duller which slightly mimics the polaroid image as shown in Fig. 57.5.

Then we used GANs on the same dataset and compared the results. We used the Pix2Pix GAN and CycleGAN on the 150 Instant Camera Images. Figure 57.6 shows the image prediction using Pix2Pix GAN. Figure 57.7 shows the image prediction using CycleGAN.

57.6 Findings

Our findings predict accurate results with 500 images on our edited datasets. Our datasets with 150 instant camera images showed promising results with Linear Regression and Generative Adversarial Networks. GANs work better in predicting and finding the right mapping. Linear Regres-

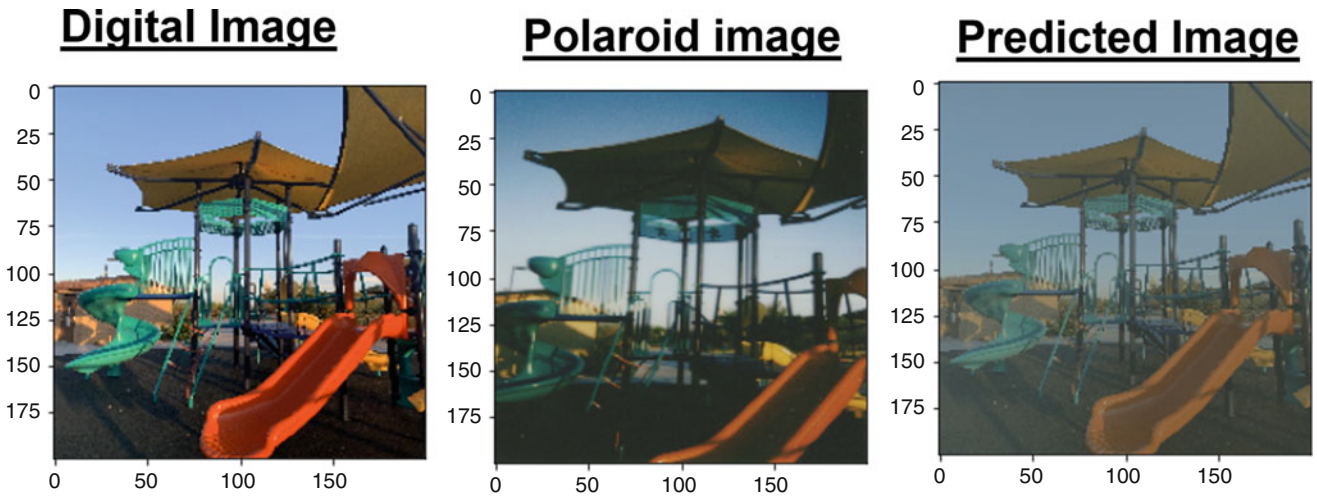

Fig. 57.4 Linear Regression side-by-side on our dataset

Fig. 57.5 Color comparison between original, polaroid, and the predicted data set

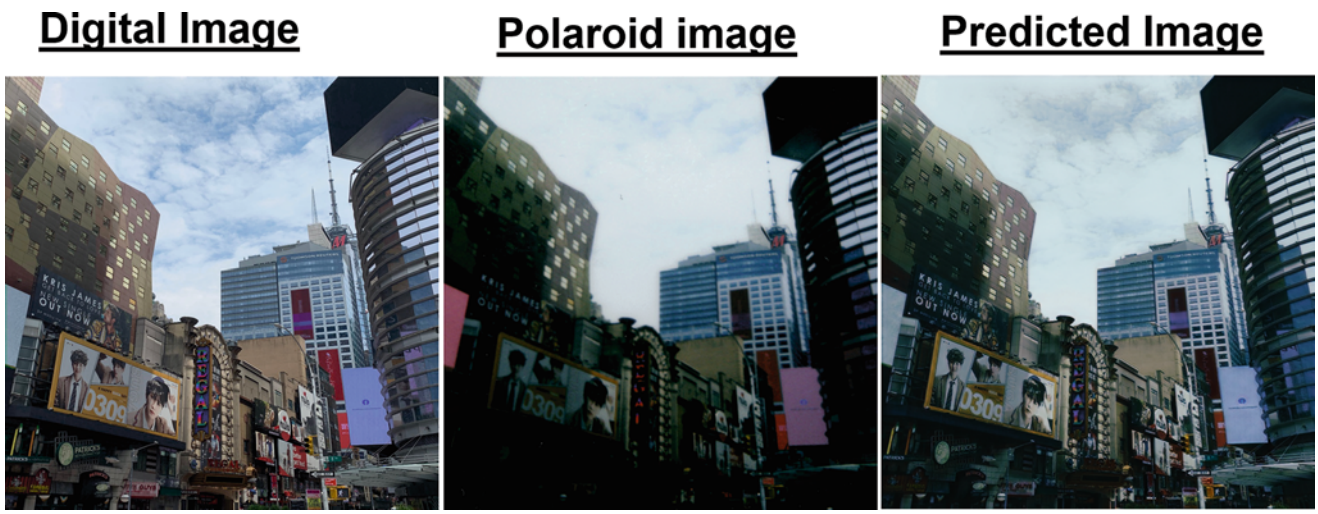

Fig. 57.6 Running Pix2Pix [4] on our dataset

sion is very limited and GANs can pick up more efficient mappings by training a loss function. Our research found that GANs predicts colors with much higher accuracy and at least 500 images are necessary to create accurate results. It is worth noting that an accurate result might be subjective, but we interpret it as one that any untrained person would not be able to distinguish by simply looking at a sample.

57.7 Conclusion

Machine Learning has a lot of potential applications in image processing. Our research found accurate results with GANs like Pix2Pix and CycleGAN. Our research showed that Machine Learning can be implemented to create better image processing solutions. Machine Learning can find new ways to

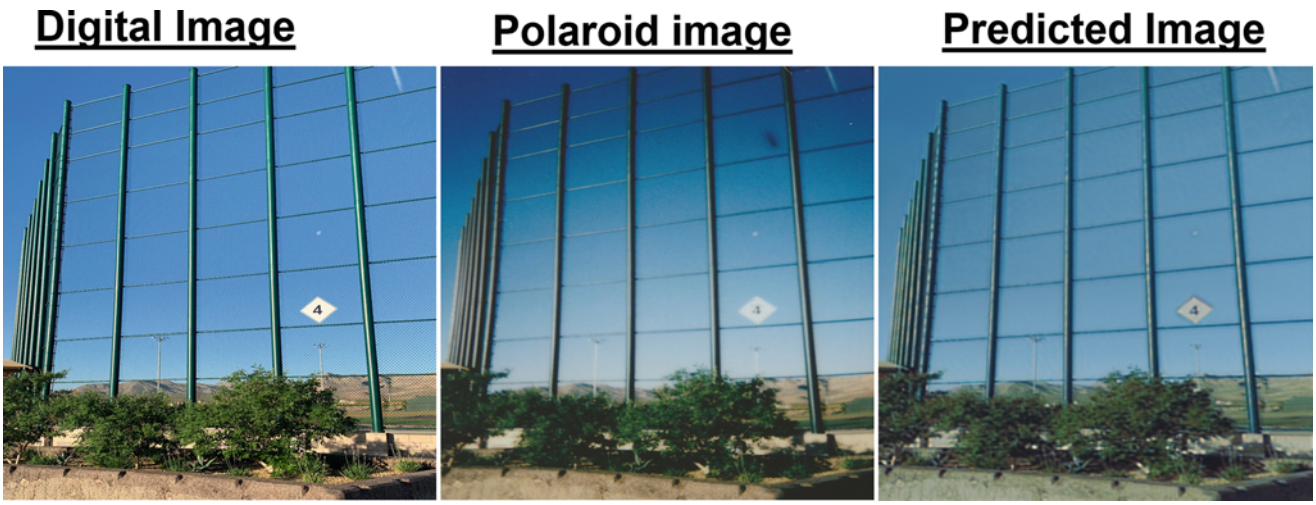

Fig. 57.7 Running CycleGAN [3] on our dataset

mimic subtle differences and create a new approach to image processing [6].

The size of our datasets limited the results of our research. Our research dataset only contained 150 images, due to the difficulty of taking pictures using a Polaroid and scanning them in a high amount. By using more images, we anticipate an increase in the accuracy of the machine learning models. Other machine learning models also have yet to be tested to conclude the best model for this effect. There is a lot of potential using machine learning to create new filters and image effects. We can experiment with our datasets and create colder effects. Other aspects of photos can also increase accuracy, such as brightness, contrast, grain, etc.

Even though our experiment was limited, we can see a promising result. The result of our Linear Regression model showed cold alterations of the colors used while GANs replicated the Polaroid effect very well. In future work, we intend to attempt ensemble learning to combine results of these and other Machine Learning models to improve on these results, along with increasing the size of our dataset. Because reproducible research [9] is important to allow others to review and improve on our results, we are making our source code and data publicly available at UNLV and Github repositories.

Acknowledgments We would like to thank UNLV's Center for Academic Enrichment and Outreach and the Office of Undergraduate Research for funding this research. This material is based upon work supported by the National Science Foundation under Grant No. 1625677. This material was supported by the U.S. Department of Education: Asian American and Native American Pacific Islander-Serving Institutions (AANAPISI) program, Grant No. P031150019.

References

1. D. Howard, D. Dai, Public perceptions of self-driving cars: the case of Berkeley, California, in *Transportation Research Board 93rd Annual Meeting*, vol. 14(4502), 1–16 (2014)
2. J. Leban, Image recognition with machine learning on python, image processing. https://towardsdatascience.com/image-recognition-with-machine-learning-on-python-image-processing-3abe6b158e9a. Accessed 20 October 2020
3. J.-Y. Zhu, T. Park, P. Isola, A.A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, in *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 2223–2232
4. P. Isola, J.-Y. Zhu, T. Zhou, A.A. Efros, Image-to-image translation with conditional adversarial networks, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 1125–1134
5. Easybluecode, Instalab iOS app. https://apps.apple.com/us/app/instalab-instant-films/id1113395996. Accessed 20 October 2020
6. L.J. Spreeuwers, Image filtering with neural networks: applications and performance evaluation (1994)
7. D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, A.A. Efros, Context encoders: feature learning by inpainting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016
8. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in *Advances in Neural Information Processing Systems*, vol. 27, ed. by Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, K.Q. Weinberger (Curran Associates, Inc., Red Hook, NY, 2014), pp. 2672–2680 [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf
9. J.R.F. Cacho, K. Taghva, The state of reproducible research in computer science, in *17th International Conference on Information Technology–New Generations (ITNG 2020)* (Springer, New York, 2020), pp. 519–524