Image Colorization Using Generative Adversarial Networks

Kamyar Nazeri, Eric Ng, and Mehran Ebrahimi(✉)

Faculty of Science, University of Ontario Institute of Technology,
2000 Simcoe Street North, Oshawa, ON L1H 7K4, Canada
{kamyar.nazeri,eric.ng,mehran.ebrahimi}@uoit.ca
http://www.ImagingLab.ca/

Abstract. Over the last decade, the process of automatic image colorization has been of significant interest for several application areas including restoration of aged or degraded images. This problem is highly ill-posed due to the large degrees of freedom during the assignment of color information. Many of the recent developments in automatic colorization involve images that contain a common theme or require highly processed data such as semantic maps as input. In our approach, we attempt to fully generalize the colorization procedure using a conditional Deep Convolutional Generative Adversarial Network (DCGAN). The network is trained over datasets that are publicly available such as CIFAR-10 and Places365. The results between the generative model and traditional deep neural networks are compared.

1 Introduction

The automatic colorization of grayscale images has been an active area of research in machine learning for an extensive period of time. This is due to the large variety of applications such color restoration and image colorization for animations. In this manuscript, we will explore the method of colorization using generative adversarial networks (GANs) proposed by Goodfellow et al. [1]. The network is trained on the datasets CIFAR-10 and Places365 [2] and its results will be compared with those obtained using existing convolutional neural networks (CNN).

Models for the colorization of grayscales began back in the early 2000s. In 2002, Welsh et al. [3] proposed an algorithm that colorized images through texture synthesis. Colorization was done by matching luminance and texture information between an existing color image and the grayscale image to be colorized. However, this proposed algorithm was defined as a forward problem, thus all solutions were deterministic. Levin et al. [4] proposed an alternative formulation to the colorization problem in 2004. This formulation followed an inverse approach, where the cost function was designed by penalizing the difference between each pixel and a weighted average of its neighboring pixels. Both of these proposed methods still required significant user intervention which made the solutions less than ideal.

F. J. Perales and J. Kittler (Eds.): AMDO 2018, LNCS 10945, pp. 85–94, 2018.
https://doi.org/10.1007/978-3-319-94544-6_9

In [5], a colorization method was proposed by comparing colorization differences between those generated by convolutional neural networks and GAN. The models in the study not only learn the mapping from input to output image, but also learn a loss function to train this mapping. Their approach was effective in ill-posed problems such as synthesizing photos from label maps, reconstructing objects from edge maps, and colorizing images. We aim to extend their approach by generalizing the colorization procedure to high resolution images and suggest training strategies that speed up the process and greatly stabilize it.

2 Generative Adversarial Network

In 2014, Goodfellow et al. [1] proposed a new type of generative model: generative adversarial networks (GANs). A GAN is composed of two smaller networks called the generator and discriminator. As the name suggests, the generator's task is to produce results that are indistinguishable from real data. The discriminator's task is to classify whether a sample came from the generator's model distribution or the original data distribution. Both of these subnetworks are trained simultaneously until the generator is able to consistently produce results that the discriminator cannot classify.

The architectures of the generator and discriminator both follow a multilayer perceptron model. Since colorization is a class of image translation problems, the generator and discriminator are both convolutional neural networks (CNNs). The generator is represented by the mapping $G(z; \theta_G)$, where z is a noise variable (uniformly distributed) that acts as the input of the generator. Similarly, the discriminator is represented by the mapping $D(x; \theta_D)$ to produce a scalar between 0 and 1, where x is a color image. The output of the discriminator can be interpreted as the probability of the input originating from the training data. These constructions of G and D enable us to determine the optimization problem for training the generator and discriminator: G is trained to minimize the probability that the discriminator makes a correct prediction in generated data, while D is trained to maximize the probability of assigning the correct label. Mathematically, this can be expressed as

$$\min_{\theta_G} J^{(G)}(\theta_D, \theta_G) = \min_{\theta_G} \mathbb{E}_z \left[\log(1 - D(G(z))) \right], \tag{1}$$

$$\max_{\theta_D} J^{(D)}(\theta_D, \theta_G) = \max_{\theta_D} \left(\mathbb{E}_x \left[\log(D(x)) \right] + \mathbb{E}_z \left[\log(1 - D(G(z))) \right] \right). \tag{2}$$

The above two equations provide the cost functions required to train a GAN. In literature, these two cost functions are often presented as a single minimax game problem with the value function $V(G, D)$:

$$\min_G \max_D V(G, D) = \mathbb{E}_x \left[\log D(x) \right] + \mathbb{E}_z \left[\log(1 - D(G(z))) \right]. \tag{3}$$

In our model, we have decided to use an alternate cost function for the generator. In Eq. 1, the cost function is defined by minimizing the probability of the discriminator being correct. However, this approach presents two issues: (1) If the discriminator performs well during training stages, the generator will have a near-zero gradient during back-propagation. This will tremendously slow down convergence rate because the generator will continue to produce similar results during training. (2) The original cost function is a strictly decreasing function that is unbounded below. This will cause the cost function to diverge to $-\infty$ during the minimization process.

To address the above issues, we have redefined the generator's cost function by maximizing the probability of the discriminator being mistaken, as opposed to minimizing the probability of the discriminator being correct. The new cost function was suggested by Goodfellow at NIPS 2016 Tutorial [6] as a heuristic, non-saturating game, and is presented as:

$$\max_{\theta_G} J^{(G)^*}(\theta_D, \theta_G) = \max_{\theta_G} \mathbb{E}_z \left[\log(D(G(z)))\right], \tag{4}$$

which can also be written as the minimization problem:

$$\min_{\theta_G} -J^{(G)^*}(\theta_D, \theta_G) = \min_{\theta_G} -\mathbb{E}_z \left[\log(D(G(z)))\right]. \tag{5}$$

The comparison between the cost functions in Eqs. 1 and 5 can be visualized in Fig. 1 by the blue and red curves respectively. In addition, the cost function was further modified by using the ℓ^1-norm in the regularization term [5]. This produces an effect where the generator is forced to produce results that are similar to the ground truth images. This will theoretically preserve the structure of the original images and prevent the generator from assigning arbitrary colors to pixels just to "fool" the discriminator. The cost function takes the form

$$\min_{\theta_G} J^{(G)^*}(\theta_D, \theta_G) = \min_{\theta_G} -\mathbb{E}_z \left[\log(D(G(z)))\right] + \lambda \|G(z) - y\|_1 \tag{6}$$

where λ is a regularization parameter and y is the ground truth color labels.

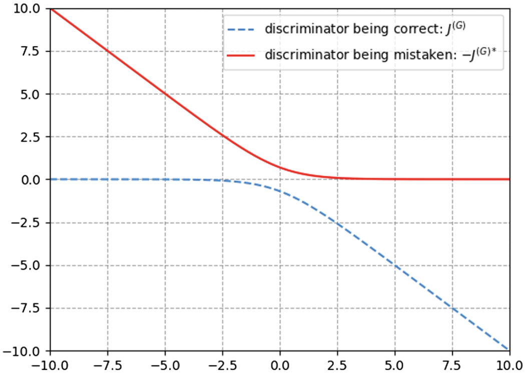

Fig. 1. Comparison of cost functions $J^{(G)}$ (dashed blue) and $-J^{(G)^*}$ (red). (Color figure online)

2.1 Conditional GAN

In a traditional GAN, the input of the generator is randomly generated noise data z. However, this approach is not applicable to the automatic colorization problem because grayscale images serve as the inputs of our problem rather than noise. This problem was addressed by using a variant of GAN called conditional generative adversarial networks [7]. Since no noise is introduced, the input of the generator is treated as zero noise with the grayscale input as a prior, or mathematically speaking, $G(\mathbf{0}_z|x)$. In addition, the input of the discriminator was also modified to accommodate for the conditional network. By introducing these modifications, our final cost functions are as follows:

$$\min_{\theta_G} J^{(G)}(\theta_D, \theta_G) = \min_{\theta_G} -\mathbb{E}_z \left[\log(D(G(\mathbf{0}_z|x)))\right] + \lambda \|G(\mathbf{0}_z|x) - y\|_1 \tag{7}$$

$$\max_{\theta_D} J^{(D)}(\theta_D, \theta_G) = \max_{\theta_D} \left(\mathbb{E}_y \left[\log(D(y|x))\right] + \mathbb{E}_z \left[\log(1 - D(G(\mathbf{0}_z|x)|x))\right]\right) \tag{8}$$

The discriminator gets colored images from both generator and original data along with the grayscale input as the condition and tries to decide which pair contains the true colored image.

3 Method

Image colorization is an image-to-image translation problem that maps a high dimensional input to a high dimensional output. It can be seen as pixel-wise regression problem where structure in the input is highly aligned with structure in the output. That means the network needs not only to generate an output with the same spatial dimension as the input, but also to provide color information to each pixel in the grayscale input image. We provide an entirely convolutional model architecture using a regression loss as our baseline and then extend the idea to adversarial nets.

In this work we utilize the L*a*b* color space for the colorization task. This is because L*a*b* color space contains dedicated channel to depict the brightness of the image and the color information is fully encoded in the remaining two channels. As a result, this prevents any sudden variations in both color and brightness through small perturbations in intensity values that are experienced through RGB.

3.1 Baseline Network

For our baseline model, we follow the “fully convolutional network” [8] model where the fully connected layers are replaced by convolutional layers which include upsampling instead of pooling operators. This idea is based on encoder-decoder networks [9] where input is progressively downsampled using a series of contractive encoding layers, and then the process is reversed using a series of expansive decoding layers to reconstruct the input. Using this method we can train the model end-to-end without consuming large amounts of memory. Note that the subsequent downsampling leads to a much more compact feature learning in the middle layers. This strategy forms a crucial attribute to the network, otherwise the resolution would be limited by GPU memory.

Our baseline model needs to find a direct mapping from the grayscale image space to color image space. However, there is an information bottleneck that prevents flow of the low level information in the network in the encoder-decoder architecture. To fix this problem, features from the contracting path are concatenated with the upsampled output in the expansive path within the network. This also makes the input and output share the locations of prominent edges in grayscale and colored images. This architecture is called U-Net [10], where skip connections are added between layer i and layer n-i.

The architecture of the model is symmetric, with n encoding units and n decoding units. The contracting path consists of 4×4 convolution layers with stride 2 for downsampling, each followed by batch normalization [11] and Leaky-ReLU [12] activation function with the slope of 0.2. The number of channels are doubled after each step. Each unit in the expansive path consists of a 4×4 transposed convolutional layer with stride 2 for upsampling, concatenation with the activation map of the mirroring layer in the contracting path, followed by batch normalization and ReLU activation function. The last layer of the network is a 1×1 convolution which is equivalent to cross-channel parametric pooling layer. We use tanh function for the last layer as proposed by [5]. The number of channels in the output layer is 3 with L*a*b* color space (Fig. 2).

We train the baseline model to minimize the Euclidean distance between predicted and ground truth averaged over all pixels:

$$J(x;\theta) = \frac{1}{3n} \sum_{\ell=1}^{3} \sum_{p=1}^{n} \| h(x;\theta)^{(p,\ell)} - y^{(p,\ell)} \|_2^2 \tag{9}$$

where x is our grayscale input image, y is the corresponding color image, p and ℓ are indices of pixels and color channels respectively, n is the total number of pixels, and h is a function mapping from grayscale to color images.

3.2 Convolutional GAN

For the generator and discriminator models, we followed Deep Convolutional GANs (DCGAN) [13] guidelines and employed convolutional networks in both

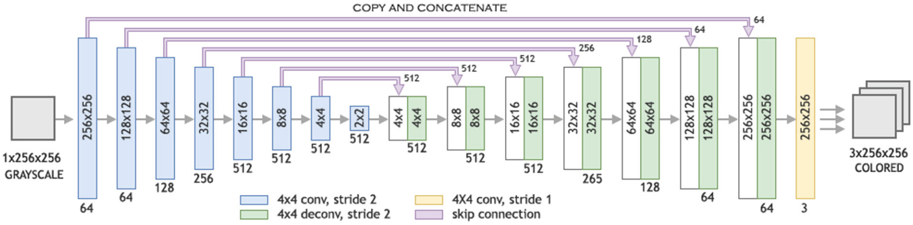

Fig. 2. U-Net architecture (256 × 256 input)

generator and discriminator architectures. The architecture was also modified as a conditional GAN instead of a traditional DCGAN; we also follow guideline in [5] and provide noise only in the form of dropout [14], applied on several layers of our generator. The architecture of generator G is the same as the baseline. For discriminator D, we use similar architecture as the baselines contractive path: a series of 4×4 convolutional layers with stride 2 with the number of channels being doubled after each downsampling. All convolution layers are followed by batch normalization, leaky ReLU activation with slope 0.2. After the last layer, a convolution is applied to map to a 1 dimensional output, followed by a sigmoid function to return a probability value of the input being real or fake. The input of the discriminator is a colored image either coming from the generator or true labels, concatenated with the grayscale image.

3.3 Training Strategies

For training our network, we used Adam [15] optimization and weight initialization as proposed by [16]. We used initial learning rate of 2×10^{-4} for both generator and discriminator and manually decayed the learning rate by a factor of 10 whenever the loss function started to plateau. For the hyper-parameter λ we followed the protocol from [5] and chose $\lambda = 100$, which forces the generator to produce images similar to ground truth.

GANs have been known to be very difficult to train as it requires finding a Nash equilibrium of a non-convex game with continuous, high dimensional parameters [17]. We followed a set of constraints and techniques proposed by [5,13,17,18] to encourage convergence of our convolutional GAN and make it stable to train.

- **Alternative Cost Function**
  This heuristic alternative cost function [6] was selected due to its non-saturating nature; the motivation for this cost function is to ensure that each player has a strong gradient when that player is "losing" the game.
- **One Sided Label Smoothing**
  Deep neural networks normally tend to produce extremely confident outputs when used in classification. It is shown that replacing the 0 and 1 targets for a classifier with smoothed values, like .1 and .9 is an excellent regularizer for convolutional networks [19]. Salimans et al. [17] demonstrated that *one-sided label smoothing* will encourage the discriminator to estimate soft probabilities and reduce the vulnerability of GANs to adversarial examples. In this technique we smooth *only* the positive labels to 0.9, leaving negative labels set to 0.
- **Batch Normalization**
  One of the main difficulties when training GANs is for the generator to collapse to a parameter setting where it always emits the same output [17]. This phenomenon is called *mode-collapse*, also known as **the Helvetica scenario** [6]. When mode-collapse has occurred, the generator learns that a single output is able to consistently trick the discriminator. This is non-ideal as the goal is for the network to learn the distribution of the data rather than the most ideal way

of fooling the discriminator. Batch normalization [11] is proven to be essential to train both networks preventing the generator from collapsing all samples to a single point [13]. Batch-Norm is not applied on the first layer of generator and discriminator and the last layer of the generator as suggested by [5].

- **All Convolutional Net**
  Strided convolutions are used instead of spatial pooling functions. This effectively allows the model to learn its own downsampling/upsampling rather than relying on a fixed downsampling/upsampling method. This idea was proposed in [20] and has shown to improve training performance as the network learns all necessary invariances just with convolutional layers.
- **Reduced Momentum**
  We use Adam optimizer [15] for training both networks. Recent research has shown that using a large momentum term β_1 (0.9 as suggested), could result in oscillation and instability in training. We followed the suggestion in [13] to reduce the momentum term to 0.5.
- **LeakyReLU Activation Function**
  Radford et al. [13] showed that using leaky ReLU [5] activation functions in the discriminator resulted in better performance over using regular ReLUs. We also found that using leaky ReLU in the encoder part of the generator as suggested by [5] works slightly better.

4 Experimental Results

To measure the performance, we have chosen to employ mean absolute error (MAE) and accuracy. MAE is computed by taking the mean of the absolute error of the generated and source images on a pixel level for each color channel. Accuracy is measured by the ratio between the number of pixels that have the same color information as the source and the total number of pixels. Any two pixels are considered to have the same color if their underlying color channels lie within some threshold distance ϵ. This is mathematically represented by

$$acc(x, y) = \frac{1}{n} \sum_{p=1}^{n} \prod_{\ell=1}^{3} \mathbf{1}_{[0,\epsilon_\ell]}(|h(x)^{(p,\ell)} - y^{(p,\ell)}|) \tag{10}$$

where $\mathbf{1}_{[0,\epsilon_\ell]}(\mathbf{x}), \mathbf{x} \in \mathbb{R}$ denotes the indicator function, y is the corresponding color image, h is a function mapping from grayscale to color images, and ϵ_ℓ is a threshold distance used for each color channel. The training results for each model are summarized in Table 1. Some of the preliminary results using the CIFAR-10 (32 × 32) dataset are shown in Appendix A. The images from GAN had a clear visual improvement than those generated by the baseline CNN. The images generated by GAN contained colors that were more vibrant whereas the results from CNN suffered from a light hue. In some cases, the GAN was able to nearly replicate the ground truth. However, one drawback was that the GAN tends to colorize objects in colors that are most frequently seen. For example, many car images were colored red. This is most likely due to the significantly larger number of images with red cars than images with cars of another color.

Table 1. Training results of baseline model and GAN.

Dataset	Network	Batch size	EPOCHs	MAE	Accuracy $\epsilon = 2\%$	Accuracy $\epsilon = 5\%$
CIFAR-10	U-Net	128	200	7.9	13.7	37.2%
CIFAR-10	GAN	128	200	5.1	24.1	65.5%
Places365	GAN	16	20	7.5	18.3	47.3 %

The preliminary results using Places365 (256×256) are shown in Appendix B. We noticed that there were some instances of mis-colorization: regions of images that have high fluctuations are frequently colored green. This is likely caused by the large number of grassland images in the training set, thus the model leans towards green whenever it detects a region with high fluctuations in pixel intensity values. We also noticed that some colorized images experienced a "sepia effect" seen with CIFAR-10 under U-Net. This hue is evident especially with images with clear sky, where the color of the sky includes a strange color gradient between blue and light yellow. We suspect that this was caused by insufficient training, and will correct itself over time.

5 Conclusion and Future Work

In this study, we were able to automatically colorize grayscale images using GAN, to an acceptable visual degree. With the CIFAR-10 dataset, the model was able to consistently produce better looking (qualitatively) images than U-Net. Many of the images generated by U-Net had a brown-ish hue in the results known as the "Sepia effect" across L*a*b* color space. This is due to the L2 loss function that was applied to the baseline CNN, which is known to cause a blurring effect.

We obtained mixed results when colorizing grayscale images using the Places365 dataset. Mis-colorization was a frequent occurrence with images containing high levels of textured details. This leads us to believe that the model has identified these regions as grass since many images in the training set contained leaves or grass in an open field. In addition, this network was not as well-trained as the CIFAR-10 counterpart due to its significant increase in resolution (256×256 versus 32×32) and the size of the dataset (1.8 million versus $50,000$). We expect the results will improve if the network is trained further.

We would also need to seek a better quantitative metric to measure performance. This is because all evaluations of image quality were qualitative in our tests. Thus, having a new or existing quantitative metric such as peak signal-to-noise ratio (PSNR) and root mean square error (RMSE) will enable a much more robust process of quantifying performance.

Source code is publicly available at:
https://github.com/ImagingLab/Colorizing-with-GANs

Acknowledgments. This research was supported in part by an NSERC Discovery Grant for ME. The authors gratefully acknowledge the support of NVIDIA Corporation for donation of GPUs through its Academic Grant Program.

A CIFAR-10 Results

See Fig. 3.

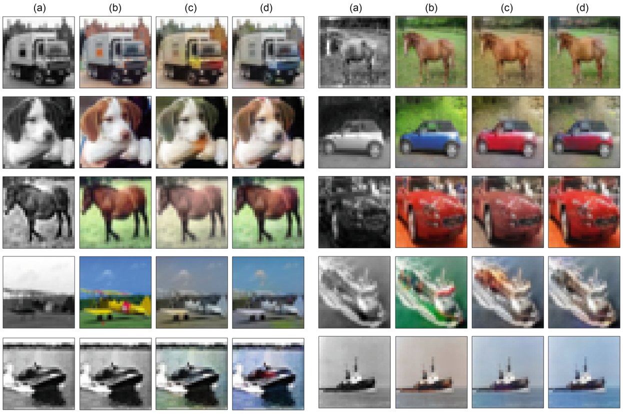

Fig. 3. Colorization results with CIFAR10. (a) Grayscale. (b) Original Image. (c) Colorized with U-Net. (d) Colorized with GAN. (Color figure online)

B Places365 Results

See Fig. 4.

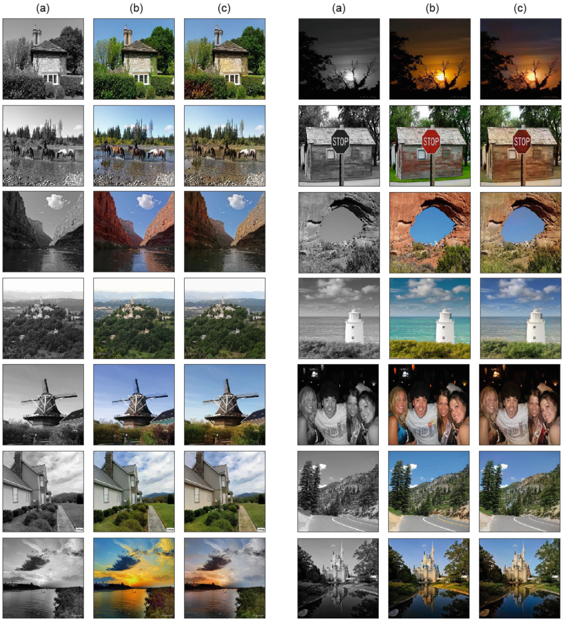

Fig. 4. Colorization results with Places365 (a) Grayscale. (b) Original Image. (c) Colorized with GAN. (Color figure online)

References

1. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2672–2680 (2014)
2. Zhou, B., Khosla, A., Lapedriza, A., Torralba, A., Oliva, A.: Places: an image database for deep scene understanding (2016)
3. Welsh, T., Ashikhmin, M., Mueller, K.: Transferring color to greyscale images. ACM TOG **21**, 277–280 (2002)
4. Levin, A., Lischinski, D., Weiss, Y.: Colorization using optimization. ACM Trans. Graph. (TOG) **23**, 689–694 (2004)
5. Isola, P., Zhu, J.-Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks (2016)
6. Goodfellow, I.: NIPS 2016 tutorial: generative adversarial networks (2016)
7. Mirza, M., Osindero, S.: Conditional generative adversarial nets (2014)
8. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440 (2015)
9. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
10. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
11. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning (2015)
12. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: Proceedings of the ICML, vol. 30 (2013)
13. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks (2015)
14. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
15. Kingma, D., Ba, J.: Adam: a method for stochastic optimization
16. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision (2015)
17. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. In: Advances in Neural Information Processing Systems, pp. 2234–2242 (2016)
18. Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., Bharath, A.A.: Generative adversarial networks: an overview (2017)
19. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2016)
20. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for simplicity: the all convolutional net (2014)