De-noise-GAN: De-noising Images to Improve RoboCup Soccer Ball Detection

Daniel Speck(✉), Pablo Barros, and Stefan Wermter

Department of Informatics, University of Hamburg,
Vogt-Koelln-Strasse 30, 22527 Hamburg, Germany
{2speck,barros,wermter}@informatik.uni-hamburg.de

Abstract. A moving robot or moving camera causes motion blur in the robot's vision and distorts recorded images. We show that motion blur, differing lighting, and other distortions heavily affect the object localization performance of deep learning architectures for RoboCup Humanoid Soccer scenes. The paper proposes deep conditional generative models to apply visual noise filtering. Instead of generating new samples for a specific domain our model is constrained by reconstructing RoboCup soccer images. The conditional DCGAN (deep convolutional generative adversarial network) works semi-supervised. Thus there is no need for labeled training data. We show that object localization architectures significantly drop in accuracy when supplied with noisy input data and that our proposed model can significantly increase the accuracy again.

Keywords: TensorFlow · Neural networks · DCGAN · GAN De-noising · RoboCup · Robotics

1 Introduction

With an increasing number of devices that are able to record visual data, the available information grows exponentially. Although this growing amount of data covers a huge potential for various fields, it is not very useful without any labels that categorize this information. In the last couple of years, much attention was spent for discriminative models that solve complex classification tasks, especially in deep learning [4,5,10]. However, there is a recent motivation for a more active development of unsupervised models, since labeling data, like marking object positions with bounding boxes, is an expansive task in both resources and time. In this paper we evaluate GANs (generative adversarial networks) for image de-noising.

In RoboCup Humanoid Soccer the movement of the robot itself and also its camera is a severe problem for the robot's vision. Object localization architectures heavily drop in accuracy during such actions, rendering it hard to make use of the camera input. While this problem could be fixed by enforcing the robot to

© Springer Nature Switzerland AG 2018

stop all actions and just stand still this "hot-fix" is not applicable in RoboCup, since a game of soccer is highly dynamic and robots should continuously move to get the ball, go to the enemy team's goal and get an edge over the enemy team by constantly repositioning on the playfield.

Image De-raining Using a Conditional Generative Adversarial Network by Zhang et al. [11] proposes an architecture for image de-raining. The generator (G) is a composition of convolutional and transposed convolutional layers (sometimes also called *de-convolutional* layers). The network is trained on rainy images, while the discriminator (D) is conditioned with clear images to give G feedback that allows to learn how to de-rain images. Hence, the input can consist of unlabeled, clear images. The only augmentation needed is applying artificial rain to the ground truth, i.e. clear images, in order to produce the conditional input. Another similar approach is the generation of "super-resolution" images out of low-resolution samples with GANs [6].

Our hypothesis is that a deep convolutional generative adversarial network (DCGAN) is able to learn the specific characteristics of RoboCup Humanoid Soccer domain for de-noising real-world images. Due to the fact that real-world scenes out of this domain are highly complex, e.g. they cover different playfields, lighting conditions, presence of audience, different robots and referees, it is difficult to handcraft filter kernels that are able to de-noise high levels of motion blurring. Hence, the model has to learn domain-specific features to be able to reconstruct them in its output. This can be achieved by combining typical de-noising filter kernels and memorizing domain specific features.

Our model, which we call "De-Noise-GAN", is a conditional DCGAN where the generator (G) de-noises artificially noised input and discriminator (D) classifies input images into two different classes"generated" and "real" to supply the adversarial loss to train G. Despite judging the direct output of G, i.e. deciding if it produces reasonable, realistically looking output, we use an evaluation metric: we have a large test dataset for ball localization and use the baseline results of our ball localization model to compare them to the accuracy of (1) artificially noised input and (2) de-noised input generated by G.

2 De-noising Generative Adversarial Network

2.1 Generative Models

Originally proposed in 2014 by Goodfellow et al. [3] *generative adversarial networks* (GANs) recently became a suitable solution for many unsupervised-learning tasks. The idea is to have two networks that train each other instead of one big network for more complex unsupervised learning problems. The discriminator sub-network D tries to categorize input into two classes: *generated* and *real*, where *generated* is an insufficient solution for the problem's domain and should be rejected since it could be distinguished from *real* samples. Thus, *real* is an appropriate solution that fits to the ground truth of the domain. D is trained with real data (unlabeled training data) and generated data by the Generator sub-network G. Hence, G is given the feedback of D in order to try to generate output that is “as good as possible” for a certain task. Therefore D and G efficiently train each other on unlabeled data by playing a two-player min-max game: D tries to minimize its error on distinguishing samples into *real* and *generated* classes and G tries to maximize D’s error by generating output that is close to samples of the *real* class, so that D falsely classifies G’s generated samples as *real.*

2.2 DCGANs

Radford et al. proposed generative adversarial networks in combination with up-to-date deep learning approaches to build deep convolutional generative adversarial networks (DCGANs) [9]. These models were introduced to move on from tasks like MNIST digit generation to more complex environments. DCGANs are capable of learning certain conditions and specific representations of a domain. For example, when DCGANs are trained with faces and different representations, they can remove or add sunglasses to faces, change the gender of a face and so forth [9].

In comparison to GANs, the convolutional layers in DCGANs are able to learn specific filter kernels in order to alter or generate specific features spatially, while the MLPs used in traditional GANs consist of fully-connected layers, which perform worse at complex, spatial filtering. Basically, it is the descriptive ability of the (de-)convolutional layers that enables DCGANs to go for more complex domains compared to vanilla GANs. It is very similar to standard computer vision tasks, where (deep) CNNs outperform MLPs and other traditional approaches, like at the ImageNet challenge for example [5].

2.3 De-noise-GAN

We propose a conditional DCGAN, called De-Noise-GAN, that is trained with RoboCup Humanoid Soccer samples and artificial noise to have Generator G learn how to detect and remove noise like motion blur and occlusions caused by the robot’s walking or camera movements.

G’s architecture is illustrated in Fig. 1. The first two layers are 1×1 convolutions to let the network learn its own color representations for our RoboCup domain. Feeding raw RGB images instead of pre-processing or other color spaces like HSV showed interesting results in Mishkin et al.’s work where they evaluated different techniques for comparable network types and discovered that trained color transformation layers could improve results [7]. HSV and other pre-processing steps also covered worse results for our case. The next step is downsampling the dimensionality of the input through two convolutional and max-pooling layers. Instead of pooling and upsampling (nearest-neighbor upscaling), we also evaluated using (de-)convolutions with a stride of 2, but experienced “checkerboard artifacts” in the output [1,8]. Dahl et al. also dealt with this kind of problems in their paper on pixel recursive super-resolution [2]. The pooling and upsampling layers introduce more smoothness for G’s output. Using just the upsampling part of the network as output often showed incomplete features

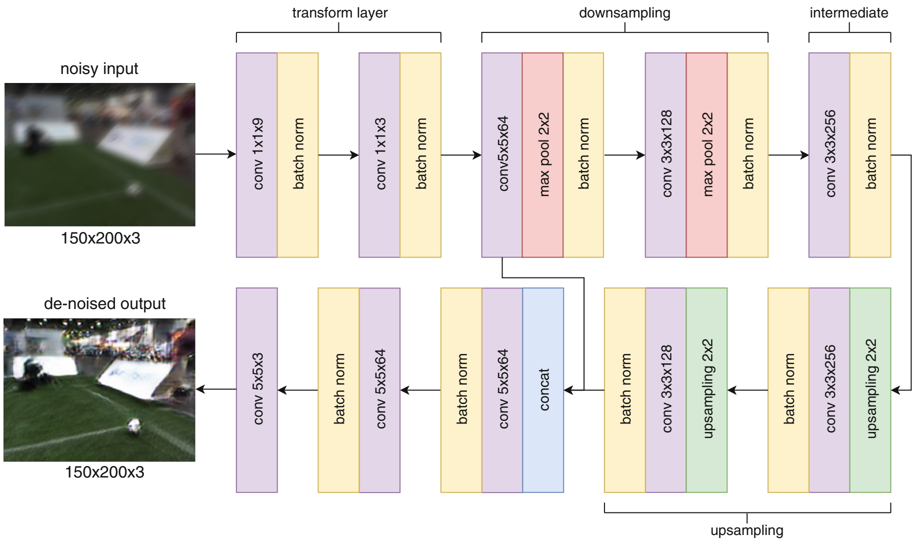

Fig. 1. Generator G of our proposed model. The input image is distorted by random Gaussian noise and translations (to add motion blur effect). Output is a de-noised RGB image. This Figure shows actual training input and output.

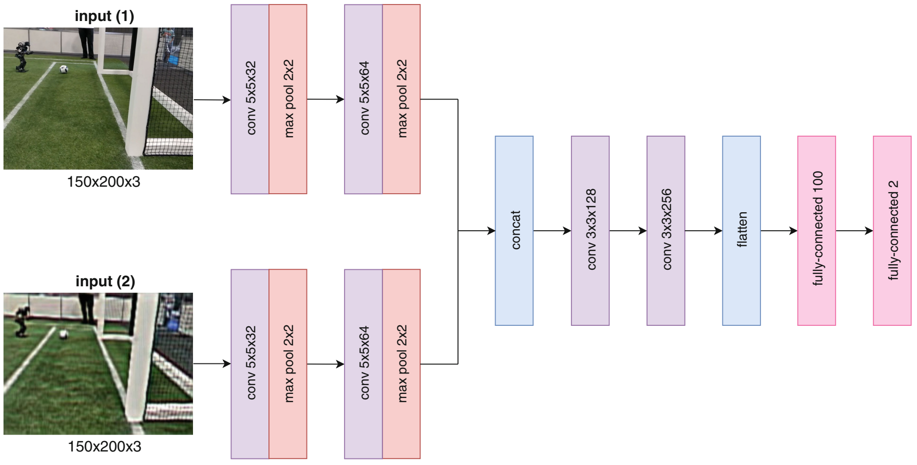

Fig. 2. Discriminator D of our proposed model. The input consists of two images: *input (1)*, which receives clear images, i.e. the *ground truth*, and *input (2)*, which is the conditional input for training. The output layer's two neurons model the probability for "real" and "generated" classes. Training labels are $[0, 1]$ for generated samples and $[1, 0]$ for real samples. This Figure shows an actual training iteration for G, hence *input (1)* is a clear, real training image and *input (2)* the de-noised reconstruction of the noised input for G, which covered a low level of noise for this sample so that the reconstructed output is of high quality.

for bigger objects, so we added two more convolutional layers before the actual output. Additionally, we added a skip layer combining the output of the upsampling part and the first convolutional layer before the downsampling part, which led to smoother output.

The Discriminator's (D) architecture is shown in Fig. 1. It has two inputs: *input (1)* always gets fed with the ground truth, i.e. clear images without noise and *input (2)* is the conditional input. The conditional input either receives clear images (training D on *real* labels) or images de-noised by G. In the case of de-noised input D is trained with *generated* labels and G with *real* labels. These two images are subsampled separately by two independent convolutional layers, the resulting feature maps are then concatenated and fed to the subsequent convolutional layers. Additionally, D has a fully-connected layer at the output level to classify its input into the two classes *generated* and *real*. A graphical illustration of the Discriminator's architecture can be seen in Fig. 2.

3 Experimental Results

3.1 Dataset and Acquisition

Our training dataset consists of 66,623 images and was recorded at three different locations, our old lab, RoboCup 2016, Leipzig, Germany and RoboCup 2017, Nagoya, Japan. The test dataset is composed of 2,177 images from two different locations, RoboCup 2017 and German Open footage. The RoboCup 2017 test images are taken from other games and other playfields than the training images from the same location. The German Open images are *only* included in the test dataset. Therefore this location's features are completely unknown to the network. During the training, we apply random Gaussian noise and image translations to the input image in order to add noise and motion blur effects.

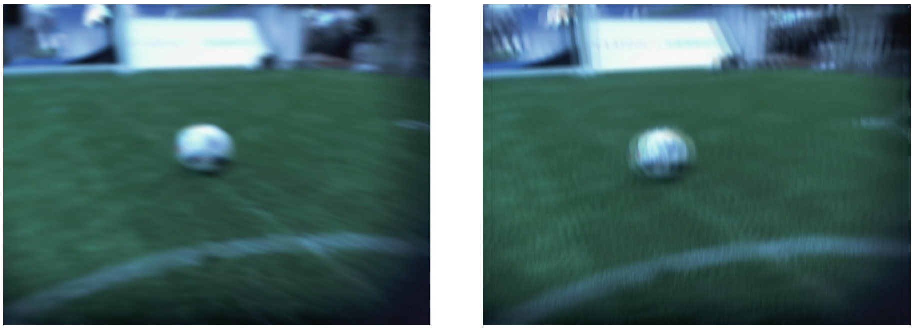

Fig. 3. Left image: **real noise**, right image: **artificial noise**. Both images are taken from a sequence of images at RoboCup 2016, Leipzig, Germany. The left image with real noise was recorded during a sequence with camera movement of the robot's camera, while the right image is the last image of the sequence, where the camera stands still again, and was artificially blurred afterwards.

The image translations (to add the motion effect) and the Gaussian noise kernel are drawn from separated random numbers. This pre-processing to artificially noise the image is done with OpenCV and NumPy. We blend together translated copies of the original image and apply gaussian noise kernels via convolutions using OpenCV. Depending on the random numbers the result of this process ranges from slightly translated, slightly noised images to heavily translated, heavily noised images in order to simulate a broad variety of motion blur effects. A comparison between real and artificial noise of a medium level (non-moving robot, but moving camera) can be seen in Fig. 3. Since a moving robot would not produce *any* clear images, we can not directly compare the high-level artificial noise to the motion blur of a walking robot due to the lack of clear images to apply artificial noise on. However, high-level artificial noise also looks similar to high real-world motion blur caused by walking robots.

3.2 De-noising

Reconstructing noised images in the RoboCup Humanoid Soccer domain is a very complex scenario. The generator has to reconstruct RBG color images of shape 200×150 with a vast amount of variations: different balls, robots, humans, play fields, and so on. Moreover changes in lighting and contrast cause strong differences for various sceneries. Nonetheless, the current architecture shows promising de-noising results. In Fig. 4 four highly noised test images were selected to display the reconstruction abilities of our network. All samples cover a ball and the first and last image also a robot. G learns about common objects in the RoboCup Humanoid soccer domain. Thus small balls, for example, are mostly reconstructed as mostly white spheres. A similar behavior applies for advertis-

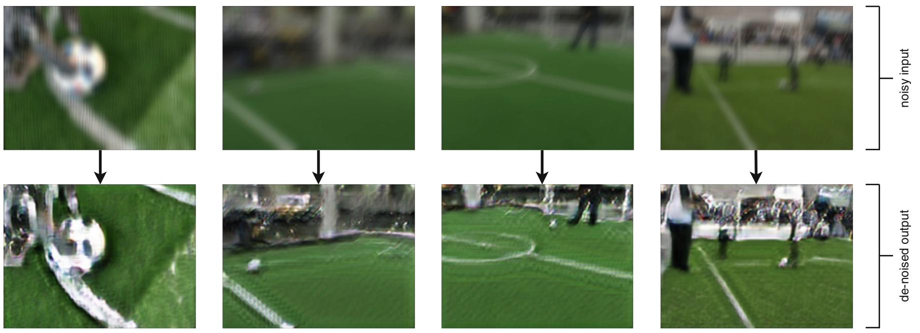

Fig. 4. Upper row displays highly **noised test images** and **lower row** the corresponding **de-noised output**. The first image was recorded in the Hamburg Bit-Bots Lab, the next two from a WF Wolves test game at RoboCup German Open, and the last one at RoboCup 2017, Nagoya, Japan. The first two locations are completely unknown to the network since they are only included in test, but not in training data. The last image from RoboCup 2017, Nagoya, Japan, covers actual game footage from a game that is not included in the training data, but other games from RoboCup 2017 are.

ing boards or the audience's clothes: if the noise in the image is too strong, the generator not only reconstructs the original features but often also mixes in common logos, shapes, and other objects that often appear in the training set. We had a significant improvement in accuracy when we moved from our old training dataset (less than 20,000 images) to our current one (66,000 images).

Figure 5 shows an interesting effect of false positives for our DCGAN. Since the DCGAN cannot reconstruct a high-level of detail without memorizing the typical scenery of our domain it sometimes comes to wrong object reconstructions when the DCGAN is fed an image with a very high-level of noise. In the case of Fig. 5 the original image only showed a penalty spot on the playfield, but the network mistakenly classifies this as a ball and therefore reconstructs it accordingly in the output.

3.3 Ball Localization

For our RoboCup Humanoid Soccer robots, we currently use a Fully-Convolutional Neural Network (FCNN) for ball localization in raw camera input. The FCNN maps the raw RGB input onto a heatmap with the same dimensionality as the input, effectively creating a voting for each pixel to be considered "ball" or"no ball". Out of the heatmap, we calculate clusters and find each cluster's center for post-processing the ball's actual location with respect to the robot (camera angle, ...). To allow for easy comparisons, in addition to Jaccard-index (Intersection of Unions) as well as precision and recall, we measure the FCNN by a "radius accuracy", i.e. comparing the center of each cluster with the ground truth in the original image. This is easily comparable for other teams who do not use bounding boxes or heatmaps, but only absolute coordinates for example. Table 1 shows that our FCNN scores accuracies around 90% on our test images, which consist of over 2,000 images from RoboCup 2017, Nagoya, Japan and German Open. The Nagoya pictures are again only covering games that are not included in the training set and German Open data is included in test data *only*. When applying artificial noise to our test data the accuracies drop to less than a third of the original accuracies. After de-noising the noised input with De-Noise-GAN the accuracies increase again to nearly 80%.

Table 1. Results for FCNN ball localization.

Accuracy type	FCNN clear images	FCNN noisy input	FCNN de-noised input
Radius 3	89.8%	22.4%	73.7%
Radius 5	91.5%	28.3%	77.5%
Radius 10	94.3%	32.4%	78.5%

4 Discussion

The results of De-Noise-GAN for de-noising domain specific real-world scenes are promising. As expected, the accuracies of object detection frameworks like our ball localization architecture heavily drop when fed with noisy input. In our case, the accuracies dropped to less than a third (see Table 1), but when fed with our de-noised images from our DCGAN the accuracies more than doubled again, peaking to nearly 80%. In our first models there was still a lot of artifacts, due to the comparably high resolution of our output and the features that need to be reconstructed there, making it easy for D to discriminate, but hard for G to precisely de-noise images. Also, over-sharpening the output happened often. The FCNN's localization accuracy was lower than 70% with these models. Two steps improved the results up to the proposed results: we introduced a skip connection and an addition to the loss function. The skip connection caused smoother output since it keeps some of the more noisy, higher-level features of the early convolutions. This alone increased the accuracies by nearly 5%. The second step was an alteration for the loss function. Originally G was only trained in an adversarial fashion, but we had quite interesting results for (variational) Autoencoders. While all of our Autoencoders did not come close to the level of detail of DCGANs, their output covered considerably fewer artifacts and oversharpening. The output, as expected, was smoother. For our current model, we tried to combine the DCGAN's high-level of detail with the Autoencoder's "smoothness". In addition to the min-max game D and G play during training, we simply added a mean squared error (MSE) to G's error function between G's output and the ground truth (clear image). This approach increased the accuracies by more than 5%, leading to smoother output and fewer artifacts than before. However, there are still artifacts in some images and also false positives like shown in Fig. 5 due to the high reconstruction capability of the GAN (we scale down the MSE to focus on the adversarial loss during learning and only add the MSE to decrease

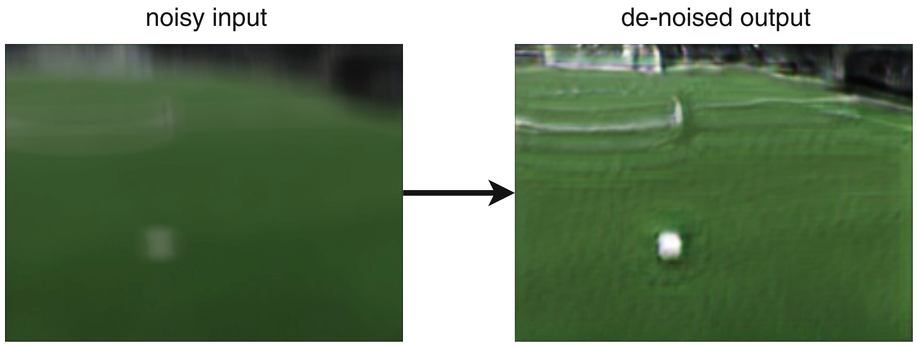

Fig. 5. The **input image (left)** shows a high-level of artificial noise. The **de-noised output (right)** clearly shows a ball, reconstructed by the DCGAN. However, in the input image this object was originally a penalty spot on the playfield, *not* a ball.

oversharpening and artifacts). This behavior suggests that for complex scenes the Generator always also learns which kind of objects appear in the scenery, effectively memorizing the given domain. Given this knowledge G de-noises by a mix of real de-noising and memorizing. This memorizing effect could be observed best for balls and lights in our domain since many training images show a ball and lights on the ceiling. Despite the discussed object mis-classifications G sometimes still has mosaic-like patterns in the output if the scenes get too complex, e.g. when high noise is present in combination with many persons in the background. Especially for the audience's clothes G then usually reconstructs some mix of commonly represented logos, typical clothes, . . . of the training set, which looks unrealistic. This might be a problem with our learning procedure or the size of the training dataset: the 66,000 images were only recorded at three different locations: our old Lab, RoboCup 2016 in Leipzig, Germany and RoboCup 2017 in Nagoya, Japan.

5 Conclusion

Our proposed architecture, De-Noise-GAN, showed reasonable and promising results for complex tasks such as de-noising RoboCup Humanoid Soccer images. If supplied with enough training samples, the generator can learn the characteristic features of different objects and reconstruct them in noisy images quite well, including real-world noise. Although the de-noising quality of unknown objects is rather poor, i.e. images look unrealistic to humans, they also show that the generator is not only memorizing different objects but also learning the characteristics of noise in images to generalize de-noising filters. However, the quality of de-noising is significantly better, if the generator has seen the objects in the scene during training, like the ball for example. This *may* suggest that this process is somehow comparable to the reconstruction in biology: e.g. a human most likely recognizes objects behind a window even in heavy rain if the object is well known to the observer, while it is significantly harder to recognize unknown objects.

6 Future Work

We plan to continue our work by evaluating new architectures. The mix-in of MSE to the loss functions, for example, shows interesting results. This suggests that mixing GANs with Autoencoders is doable and can actually decrease the downsides of each of the single approaches. Besides that, a deeper look into the reconstruction abilities of complex scenes could be interesting to understand the importance of well-known domains for the de-noising process. Moreover, this could be compared to similar tasks in neuroscience, which might suggest new alterations for the current architecture to increase the model's robustness.

Acknowledgements. We are grateful to the NVIDIA corporation for supporting our research through the NVIDIA GPU Grant Program (https://developer.nvidia.com/academic_gpu_seeding). We used the donated NVIDIA Titan X (Pascal) to train our models. The work was made in collaboration with the TRR 169 "Crossmodal Learning", funded by the DFG.

References

1. Aitken, A., Ledig, C., Theis, L., Caballero, J., Wang, Z., Shi, W.: Checkerboard artifact free sub-pixel convolution: a note on sub-pixel convolution, resize convolution and convolution resize, July 2017. http://arxiv.org/abs/1707.02937
2. Dahl, R., Norouzi, M., Shlens, J.: Pixel recursive super resolution. arXiv preprint arXiv:1702.00783 (2017)
3. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, vol. 27, pp. 2672–2680 (2014). http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf
4. Karpathy, A., Leung, T.: Large-scale video classification with convolutional neural networks. In: Proceedings of 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1725–1732 (2014). https://doi.org/10.1109/CVPR.2014.223
5. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances In Neural Information Processing Systems, pp. 1–9 (2012)
6. Ledig, C., et al.: Photo-realistic single image super-resolution using a generative adversarial network, September 2016. http://arxiv.org/abs/1609.04802
7. Mishkin, D., Sergievskiy, N., Matas, J.: Systematic evaluation of CNN advances on the ImageNet, June 2016. http://arxiv.org/abs/1606.02228
8. Odena, A., Dumoulin, V., Olah, C.: Deconvolution and checkerboard artifacts. Drill **1**(10), 1–14 (2016). https://doi.org/10.23915/distill.00003
9. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, pp. 1–16 (2016)
10. Szegedy, C., Reed, S., Sermanet, P., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions, pp. 1–12 (2014)
11. Zhang, H., Sindagi, V., Patel, V.M.: Image de-raining using a conditional generative adversarial network. arXiv preprint arXiv:1701.05957, pp. 1–13 (2017)