Deraining water drops with GAN

Introduction

Image restoration has been a long-standing research topic in deep learning, with one of the earliest studies dating back to 1972 by William Hadley Richardson [1]. Since then, numerous approaches have been explored, as extensively explained in "A Survey of Deep Learning Approaches to Image Restoration" [2]. However, most research has focused on common degradation types such as denoising, deblurring, and super-resolution, while deraining has received relatively less attention.

This does not mean deraining has been entirely neglected. In "Towards Unified Deep Image Deraining: A Survey and a New Benchmark" [3], it is shown that researchers have studied various experimental approaches in the field. Notably, most existing methods rely on CNNs, with only a few (around 10) employing Generative Adversarial Networks (GANs). Since 2022, transformers have seemed to gain popularity, seemingly overshadowing GAN-based solutions. We argue, however, that GANs remain a promising architecture for deraining.

Another limitation in existing research is the focus on thin rain streaks in outdoor scenes, overlooking scenarios where water droplets adhere directly to the camera lens. Such raindrops distort images more severely, yet few datasets address this issue. To bridge this gap, we utilize the dataset from "Attentive Generative Adversarial Network for Raindrop Removal from A Single Image" [4], which contains 1,119 paired images with diverse backgrounds and raindrop patterns.

Beyond raindrop removal, our study extends to other real-world scenarios, such as water splashes from swimming pools, faucets, or ocean waves. By tackling these challenges, we aim to enhance image restoration for a broader range of practical applications.

State of the art

We always knew GAN made images from noisy input on the generator, to then feed the discriminator and perfect that random noise to fool the discriminator. However, in our case, we had clean and rainy images, so it didn’t make sense to feed random noise, but it also didn’t make sense to feed only the image.

Doing some research, we found Conditional Generative Adversarial Nets (CGAN) [5]. Generative adversarial nets can be extended to a conditional model if both the generator and discriminator are conditioned on some extra information y. y could be any kind of auxiliary information, such as class labels or data from other modalities. We can perform the

conditioning by feeding y into the both the discriminator and generator as additional input layer.

In the generator, the prior input noise $p_z(z)$, and y are combined in joint hidden representation, and the adversarial training framework allows for considerable flexibility in how this hidden representation is composed. 1

In the discriminator, x and y are presented as inputs and to a discriminative function (embodied again by a MLP in this case).

The objective function of a two-player minimax game would be as:

$$min_G max_D V(D,G) = E_{x \sim p_{data}(x)}[logD(x|y)] + E_{z \sim p_z(z)}[log(1 - D(G(z|y)))]$$

Then we search for specifically image CGAN and found the Pix2Pix [6]. For the generator architecture, they used UNet [7], which builds upon the encoder-decoder structure with the addition of skip connections; it consists of a contracting path (encoder) that progressively downsamples spatial dimensions while increasing feature channels, followed by an expansive path (decoder) that upsamples back to the original resolution. For the discriminator, they use a convolutional “PatchGAN” classifier, which only penalizes structure at the scale of image patches. We extracted the code directly from their GitHub repository [8].

Methodology

- **Data Analysis**

The dataset we used, as mentioned above, consists of around 1,119 pairs of images taken in different locations. Each pair has one image with the lens clean, used as a ground truth, and the other has been taken with water splashed on the lens, so it has water drops on it, used as the “noisy” image. All the images are 480 by 720 pixels.

We chose this dataset since most of them only worked with the long water drops the dry lens takes while raining, while this one focuses more on what the lens sees after it stops raining, but the lens has been corrupted by the water drops.

There are three folders already predefined: the train with 860 pairs (≈ 77%), the test A with 57 pairs (≈ 5%), and the test B: 202 pairs (≈ 18%). The dataset is small, but it will be enough to do some basic training and obtain adequate result if exploited properly.

- **Models and Optimization**

Our first model consists of a generator and a discriminator, both built on a modular block design. The generator follows an encoder-decoder structure: the encoder comprises four blocks, each with a 3×3 convolution, batch normalization, ReLU activation, and optional 2×2 average pooling (applied every two blocks), followed by an adaptive average pooling layer and a linear projection to a latent space. The decoder mirrors this structure, beginning with a linear layer to expand the latent representation, followed by four blocks of batch normalization, ReLU, 3×3 convolution, and 2×2 upsampling (applied every two blocks). The discriminator uses only the encoder component, sharing its architecture with the generator's encoder but terminating in a linear layer for binary classification. Both networks leverage repeated downsampling/upsampling to progressively compress and reconstruct features, with batch normalization stabilizing training.

We adopt Binary Cross-Entropy (BCE) loss since it is the standard choice for GAN discriminators, as it explicitly measures the probability of real vs. fake samples through a binary classification task, promoting clear gradient signals for both the generator and discriminator. It handles the discrete nature of the discriminator's decision boundary more effectively, penalizing confident misclassifications more aggressively. This property helps mitigate early saturation issues, where the generator or discriminator dominates prematurely.

Our model uses a 128-dimensional latent space, chosen to balance expressiveness and computational efficiency, as lower dimensions risk mode collapse while higher dimensions increase training time without guaranteed gains. Batch size 8 was selected due to hardware constraints, though we compensate with batch normalization to stabilize gradient estimates The network's base width of 32 channels scales exponentially with down/upsampling, ensuring sufficient capacity for feature extraction while avoiding overparameterization given our small batch size. We set the learning rate to 0.001 for both generator and discriminator, a standard value for the Adaptive Moment Optimizer (Adam) optimizer in GANs, promoting stable updates without oscillatory behavior. Training runs for 20 epochs, to experiment how much it improves in such a little epoch that may take a long time to process. We use Adam (β_1=0.9, β_2=0.999) for its adaptive momentum, which helps navigate saddle points common in adversarial optimization.

- **<u>Pix2Pix</u>**

The second model, Pix2Pix, is a type of Generative Adversarial Network oriented to image-to-image translation. In this scenario, this model was trained to clean and reconstruct images, removing raindrops that obstructed the view.

Similar to the previous model, our generator follows an encoder-decoder architecture with skip connections. This type of architecture is known as a U-Net, due to the skip connections it has, allowing the model to preserve small but important details of each image. The network was made of the following: six downsampling blocks. Each of these blocks was created by a 4x4 convolutional layer, batch normalization, and a LeakyReLU activation function; on the other hand, we have five upsampling blocks. Each of these blocks is conformed by transposed convolutions, ReLU activation function, and concatenation of the encoder outputs.

For this project, a PatchGAN discriminator was used; it receives two inputs, the real image and the generated image. Unlike a traditional discriminator, PatchGAN generates a grid that highlights how realistic each part of the image looks, it analyzes each patch of the image individually while comparing it to the actual image. The structure of the PatchGAN is the following: first, it's made up of a series of convolutional layers, which reduce the quality of the actual image while preserving the most important details. These layers downsample constantly the concatenated input, allowing the network to focus exclusively on the abstract features that make up the image. Each convolutional block is made up of a convolutional layer, batch normalization, and a Leaky ReLU activation function. The final layer outputs a features map that shows how realistic each part of the image looks.

In order to train the model properly, two loss functions were used: BCEWithLogitsLoss, which measures how well the generator fools the discriminator; it wants the images to be as realistic as possible. On the other hand, we have L1 loss, which measures the distance between the generated photo and the real image, it makes sure that the generated photo, besides looking realistic, looks similar to the original image.

Now, in order to optimize the training of the model, different strategies were used. To start with, both models were trained using Adam, an optimizer that combines momentum with RMSdrop, with a learning rate of 0.0002 and betas of 0.5 and 0.999 (these betas allow the GAN to stay stable while training). Throughout the whole training, the loss per epoch is saved to calculate the Peak Signal to Noise Ratio (PSNR) as a metric to evaluate how accurate was the reconstruction of the image.

Finally, the hyperparameters selected were the following. First, the image size was 256 x 256, this allows the model to capture the most significant details of the picture while being able to reconstruct it without making excessive use of hardware. Second, a batch size of 8 was selected due to the limited capacity of the GPU being used. Finally, 100 epochs were considered an optimal number of repetitions, it avoided overfitting, allowed us to get an

accurate PSNR graph and, in some cases, was able to reach a convergence between generator and discriminator.

Experiment 1: Baseline GAN Architecture

In the initial experiment, we implemented a basic GAN model as described in the Models section. This architecture consists of a simple convolutional Generator and a CNN-based Discriminator. The goal of this baseline was to assess the GAN's raw ability to learn the mapping from input images with raindrops to clean target images without sophisticated loss functions or architectural tweaks.

The training setup included:

- Learning rate: 0.0002 for both Generator and Discriminator
- Loss function: Binary Cross Entropy (BCE)
- 200 training epochs
- Batch size: 16

This setup was chosen to establish a baseline performance against which improvements in future experiments could be measured.

Results 1

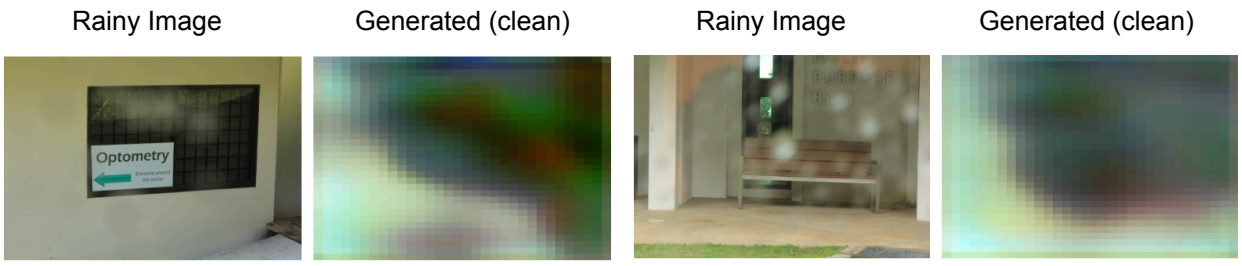

Fig.1 Image results Experiment 1

The generated images (Fig.1) were extremely blurry and lacked detail. While the model learned some shadow information (i.e., detection of light/dark regions), it failed to reconstruct coherent or realistic clean images. This was visually confirmed through image samples and reflected in the loss curves.

- **Generator loss** fluctuates heavily between 0 and ~15 (Fig. 2), suggesting unstable training.

- **Discriminator loss** quickly dropped to near zero, indicating it rapidly overpowered the Generator (Fig. 2).

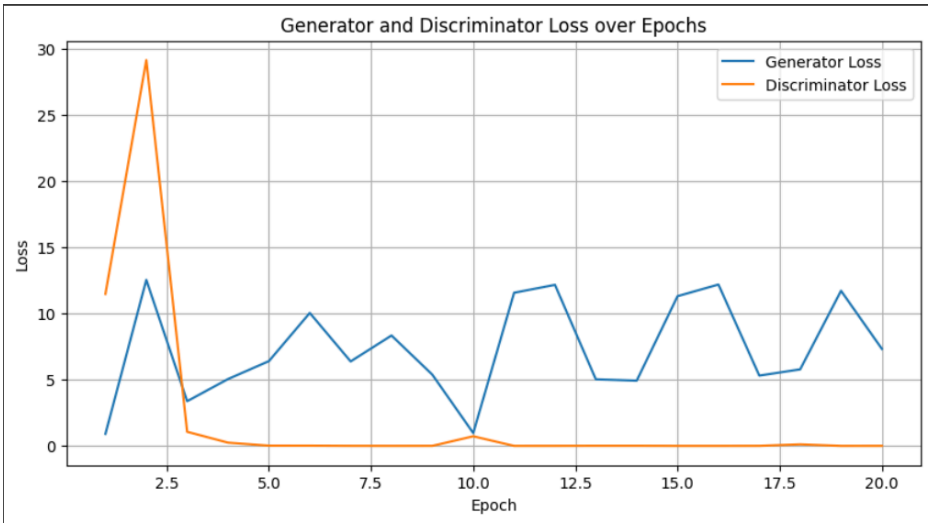

Fig.2 Generator & Discriminator losses for Ex. 1

Interpretation: This result clearly indicates mode collapse and generator underperformance. The model struggled to converge towards meaningful outputs. The imbalance between the Discriminator and Generator was likely a major cause.

Experiment 2: Pix2Pix Conditional GAN

For the second experiment, we employed the Pix2Pix architecture, which leverages a conditional GAN combined with an L1 loss for improved image-to-image translation performance. This model includes a U-Net generator and a PatchGAN discriminator.

Training details:

- Generator: U-Net
- Discriminator: 70×70 PatchGAN
- Loss: BCE + λ·L1 (λ = 100)
- Batch size: 16, 200 epochs
- Learning rate: 0.0002 for both networks

This setup aimed to encourage pixel-wise reconstruction accuracy while still leveraging adversarial learning for realism.

Results 2

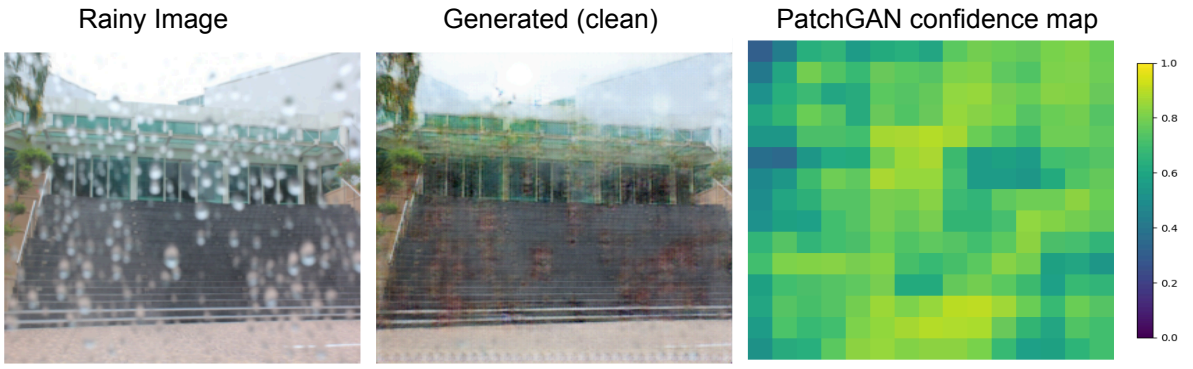

Fig.3 Image results and PatchGAN confidence map for Ex.2

Image quality significantly improved over the baseline. While raindrops were still only partially removed, the generated images exhibited clearer structure, better textures, and fewer artifacts (Fig. 3). The addition of Pix2Pix led to more visually convincing restorations, especially in background regions and object contours.

PatchGAN confidence maps further confirmed the improvement in local realism. These maps reflect the discriminator's per-patch confidence that generated outputs resemble real images (Fig. 3). In contrast to the baseline, the Pix2Pix confidence maps displayed broader, brighter regions, indicating higher realism and more consistent textures. This supports the claim that PatchGAN helps enforce localized visual fidelity across the image.

Generator loss steadily decreased and stabilized around 7, indicating effective learning progress (Fig. 4).

Discriminator and L1 losses remained low throughout training, suggesting rapid convergence of the discriminator and consistent pixel-level guidance from the L1 term (Fig. 4).

PSNR (Peak Signal-to-Noise Ratio) fluctuated but showed an overall upward trend, peaking around 24 dB in later epochs. Early training exhibited volatility with dips near 8-10 dB, but performance stabilized as training progressed (Fig. 5).

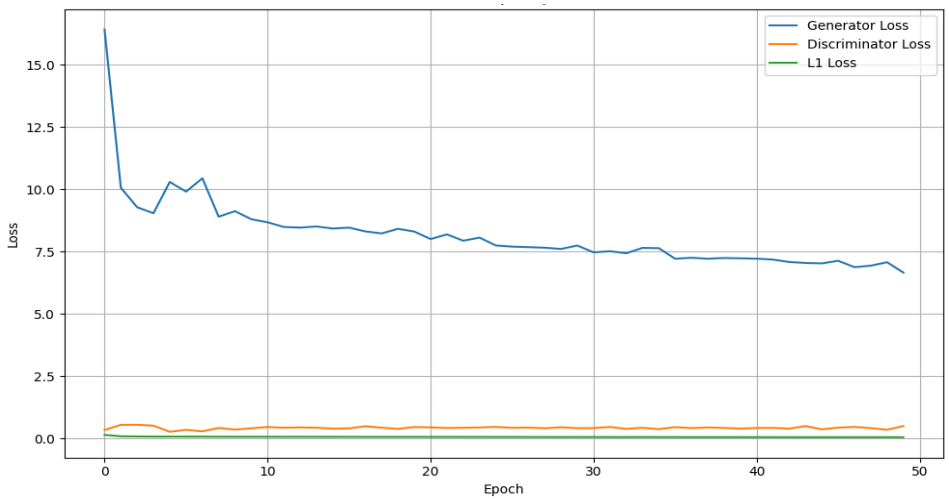

Fig.4 Generator, Discriminator and L1 losses for Ex.2

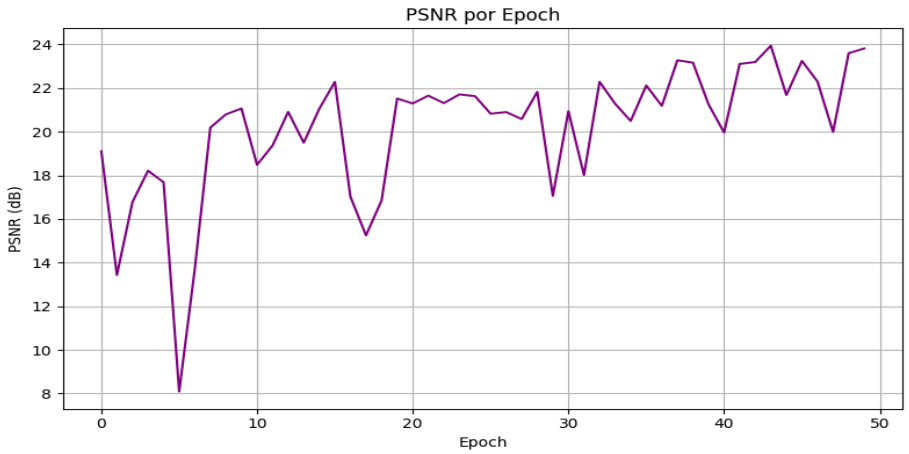

Fig.5 Generator PSNR for Ex.2

Interpretation:

The use of Pix2Pix and the L1 loss encouraged sharper reconstructions and better structure preservation. The improvement in confidence maps shows that the discriminator effectively guided the generator toward more photorealistic outputs. However, PSNR instability and partial raindrop removal suggest room for further architectural or training enhancements.

Experiment 3: Pix2Pix + Training Stabilization Techniques

In this final experiment, we augmented the Pix2Pix setup with multiple stabilization techniques to mitigate the Discriminator overpowering the Generator:

- **Learning rates**: 0.0005 (Generator), 0.0002 (Discriminator)
- **Label smoothing**: Real = 0.9, Fake = 0.1
- **Label noise**: 5–10% label flips during training

These changes aimed to rebalance the training dynamics and make it harder for the Discriminator to dominate, giving the Generator a better learning signal.

Results 3

Fig.6 Image results for Ex.3

The modified setup led to better visual and quantitative results:

- Images appeared sharper, and raindrop removal improved, especially in regions with moderate background-contrast (Fig.6).

- **Generator loss** decreased and stabilized around 5 (Fig. 7).

- **Discriminator accuracy** fluctuated between 70–80%, indicating a balanced adversarial game (Fig. 8).

- **PSNR** increased quickly in early training and stabilized between 20–30 dB (Fig. 8), showing a significant improvement in consistency compared to previous runs.

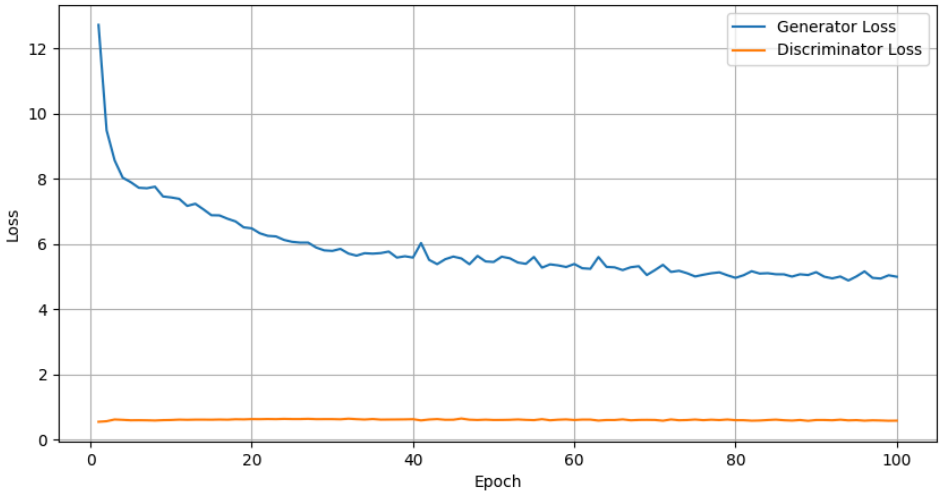

Fig.7 Discriminator and Generator losses for Ex.3

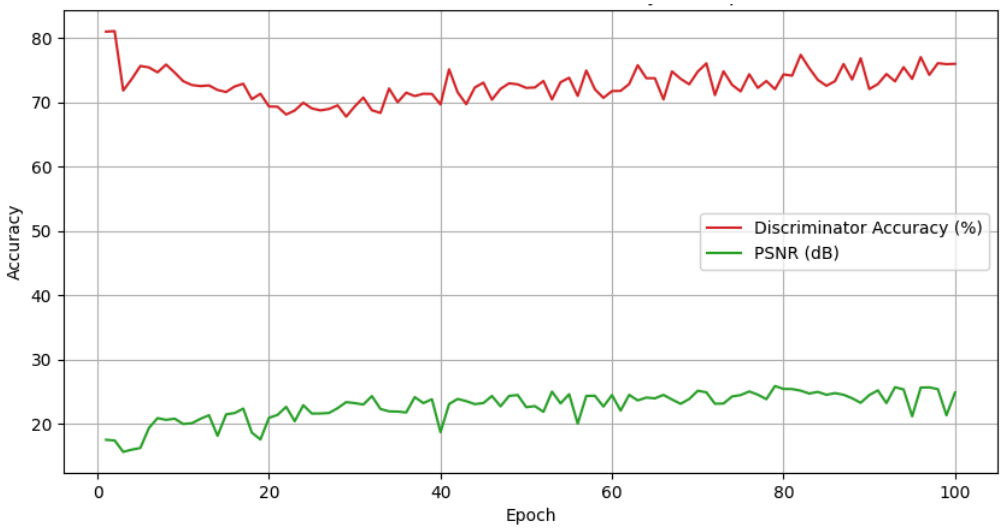

Fig.8 Discriminator Accuracy and PSNR for Ex.3

Interpretation: The regularization strategies successfully addressed the generator–discriminator imbalance. Label smoothing, noisy labels, and asymmetric learning rates helped maintain adversarial pressure while avoiding mode collapse. The PSNR values now consistently remain above 20 dB, which is typically regarded as an acceptable threshold for good perceptual quality. However, the model still struggles in scenes with strong contrast around raindrops, indicating a limitation in generalization under difficult lighting or texture conditions.

Discussion and future work

The first thing we observe is how much it struggles to remove the water drops from those images that have more. For that, data augmentation would be good. The data set we use we could grab test B and fuse it with the training, we could grab normal images and add the water drops ourselves with a generator such as “ROLE: Raindrop on Lens Effect” [9] or search for similar datasets.

Another observation is the fact that it struggled a bit with contrasts. For that, we could add the contrastive loss. Contrastive loss is a key loss function used in contrastive learning. Its primary objective is to maximize the similarity between positive pairs (instances from the same sample) while minimizing the similarity between negative pairs (instances from different samples) in the learned embedding space [10].

Conclusions

- The use of GANs is considered to be efficient and optimal for raindrop removal.
- U-Net architecture seems to be a success, preserves most important details of the images.
- The combination of BCEWithLogitsLoss and L1loss allows the image to be realistic, as well as similar to the dataset.
- Both models struggled to recreate highly distorted images, an improvement opportunity can still be explored.
- For the Pix2Pix model, PSNR metric is an accurate metric to show how realistic the images look.
- Data Augmentation would be a great implementation for training, would allow the dataset to be expanded.

References

[1] William Hadley Richardson, "Bayesian-Based Iterative Method of Image Restoration*," J. Opt. Soc. Am. 62, 55-59 (1972)

[2] Su, J., Xu, B., & Yin, H. (2022). A survey of deep learning approaches to image restoration. Neurocomputing, 487, 46–65. https://doi.org/10.1016/j.neucom.2022.02.046

[3] Chen, X., Pan, J., Dong, J., & Tang, J. (2023, October 5). Towards unified Deep Image Deraining: a survey and a new benchmark. arXiv.org. https://arxiv.org/abs/2310.03535

[4] Qian, R., Tan, R. T., Yang, W., Su, J., & Liu, J. (2017, November 28). Attentive Generative Adversarial Network for Raindrop Removal from a Single Image. arXiv.org. https://arxiv.org/abs/1711.10098

[5] Mirza, M., & Osindero, S. (2014, November 6). Conditional generative adversarial Nets. arXiv.org. https://arxiv.org/abs/1411.1784

[6] Isola, P., Zhu, J., Zhou, T., & Efros, A. A. (2016, November 21). Image-to-Image Translation with Conditional Adversarial Networks. arXiv.org. https://arxiv.org/abs/1611.07004

[7] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In MICCAI, 2015.

[8] Pix2Pix model GitHub repository

[9] ricky40403. ROLE: Raindrop on lens effect. (2019). GitHub. https://github.com/ricky40403/ROLE

[10] P, P. (2024, October 7). What is Contrastive Learning? A guide. Roboflow Blog. https://blog.roboflow.com/contrastive-learning-machine-learning/