Delft University of Technology

Deep Learning
Department of Computer Science

Replication - Sea-Pix-GAN: Underwater image enhancement using adversarial neural network

Authors:
Jerez, E. (4841115, E.E.Jerez@student.tudelft.nl)
Ning, C. (5914558, C.Ning-1@student.tudelft.nl)
Li, P. (5952433, P.Li-15@student.tudelft.nl)
Ren, Q. (5977657, Q.Ren-4@student.tudelft.nl)

April 14, 2024

1 Introduction

Underwater image fidelity is a challenging topic, even within the context of modern technology, as underwater conditions affect images in complex and seldom consistent ways. Two prime reasons for this are the absorption spectrum of water and underwater visibility. (Pure) water absorbs shorter wavelengths of radiation in the visible light spectrum much quicker than it does longer wavelengths, as shown in Figure 1, meaning that the intensity of red light decreases the fastest with depth, followed by green light, eventually leaving mostly blue light and at further depths none at all. This means underwater images are tinted differently at different depths. To make matters even more complicated, the presence of dissolved minerals and other impurities in the water can further tint the water different hues, and the presence of sediments suspended in the water result in different visibility ranges, often resulting in close-by objects being seen more clearly than far-away objects, with the latter sometimes being partially occluded.

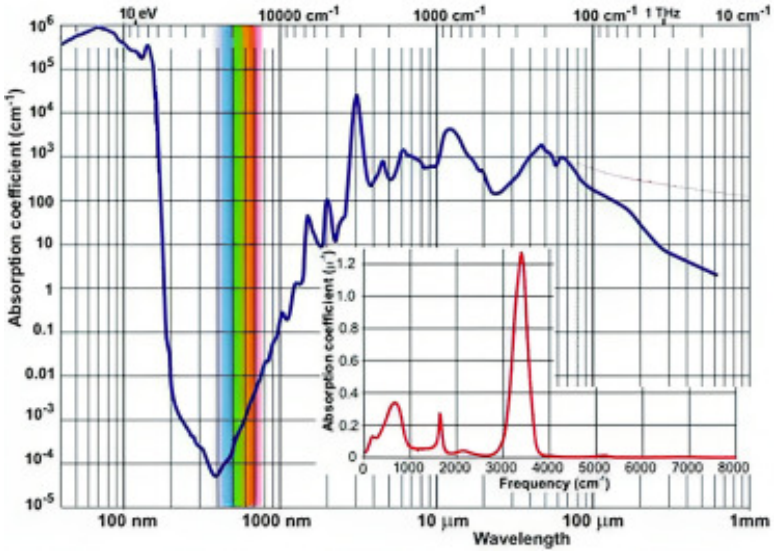


Figure 1: Absorption spectrum of water [6]

Many attempts have been made to create a neural network-based solution that can compete with state-of-the-art professional editing of underwater images. In the paper being replicated presently, a Generative Adversarial Network (GAN) architecture is utilized in order to obtain potentially better results than benchmarked in previous methods. A GAN is an architecture that utilizes two separate neural networks "competing" with each other: a generator that attempts to generate images as close as possible to the desired result, and a discriminator that attempts to distinguish generated or "fake" images from real or "ground truth" images as well as possible. By evaluating the output of the generator network with the discriminator network, and then utilizing the output of the discriminator to further train the generator, coupled with a carefully tuned loss function, both the generator and the discriminator are adversarially trained to become highly effective at generating the desired results and discerning these desired results from undersirable ones. [6]

2 Architecture of the GAN

2.1 Loss function

$$G^* = \min_G \max_D E_{x,y}[\log \mathrm{D}(x,y)] \quad + \quad E_{x,z}[\log(1 - \mathrm{D}(x, \mathrm{G}(x,z)))] + \lambda \left(E_{x,y,z}[|y - \mathrm{G}(x,z)|]\right) \tag{1}$$

In the loss function shown in Equation 1 [6], D(x,y) is the probability of the discriminator where x is a "raw" (non-generated) input image and y is the "ground truth" image. It penalizes the discriminator when it fails to output a high probability for "ground truth" images. The second term of the loss function quantifies the logarithmic expectation of 1 minus the probability of the discriminator with the "raw" image and the image generated by the generator as inputs (with z being noise). It penalizes the generator when the discriminator classifies the images it generates as "fake". [6]

In order to obtain an image that actually looks like the "ground truth" images, a third Euclidean loss term is necessary, with an arbitrary parameter λ multiplied by the expectation of the Euclidean norm between the "ground truth" image and the generated one, essentially rewarding the generator for creating an image that looks exactly like the "ground truth".

The loss function is minimized by the generator and maximized by the discriminator. If a heavy imbalance exists, one of them will overpower the other and the GAN will perform suboptimally.

2.2 Generator

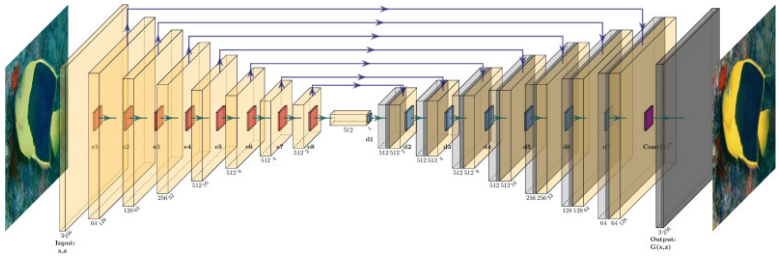


Figure 2: Architecture of the generator [6]

The generator network consists of an enconder and a decoder. The encoder takes as input a 256x256x3 RGB image and consists of 7 convolutional layers that apply a 4x4 filter and a stride of 2, and further perform Batch Normalization and utilize a leaky ReLU activation function. The implemented downsampling is in the following order, respectively with the input and then the output of layers 1 through 8: 256x256x3, 128x128x64, 64x64x128, 32x32x256, 16x16x512, 8x8x512, 4x4x512 and finally 1x1x512 (a single dimension feature map with 512 features)[6].

The 1x1x512 output of the enconder is then fed into the decoder, consisting of 7 (de)convolutional layers with a filter of 4x4 and a stride of 2, followed by Batch Normalization and a standard ReLU activation function, with layers 1 through 3 having 50% dropout. The upsampling in the decoder is in the following order: 1x1x512, 2x2x1024, 4x4x1024, 8x8x1024, 16x16x1024, 32x32x512, 64x64x256, 128x128x128. A deconvolution is then used to map the tensor output by the last layer back to the desired 256x256x3 RGB image. Note that there are also skip connections between each layer i+1 of the encoder and 7-i of the decoder[6].

2.3 Discriminator

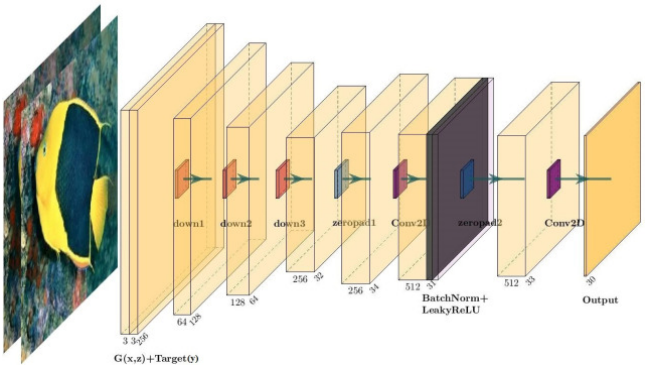


Figure 3: Architecture of the discriminator [6]

In order to prevent blurry images resulting from relying only on Euclidean losses, the discriminator evaluates the cross-entropy of patches instead of the entire image pixel-by-pixel. The discriminator is given two images as input: the image generated by the generator and the target "ground truth" image, with dimensions 256x256x3 (RGB). The downsampling layers are exactly the same as with the generator, with 4x4 filters and stride of 2, followed by Batch Normalization and a Leaky ReLU activation function. They are followed by convolution layers with filter of 4x4 and stride of 1, and then the two inputs are concatenated into a tensor of dimension

256x256x6. In total, seven layers are used to transform the outpu to a 30x30x1 tensor. Each block of this tensor classifies a 70x70 patch of the output of the generator[6].

2.4 Implementation

The network in this replication was mainly inspired by the implementation of FUnie-GAN[5] and UGAN [4]. Both references are recent GAN-based underwater image enhancement technique that build upon the same dataset (EUVP Dataset) of Sea-Pix-GAN. We forked from xahidbuffon's Github repository and added our own PyTorch network and training scripts. We reused some of the components from xahidbuffon's implementation such as the data pipeline and evaluation scripts. A pull request was submitted but still under review at the moment this report is published.

3 Training

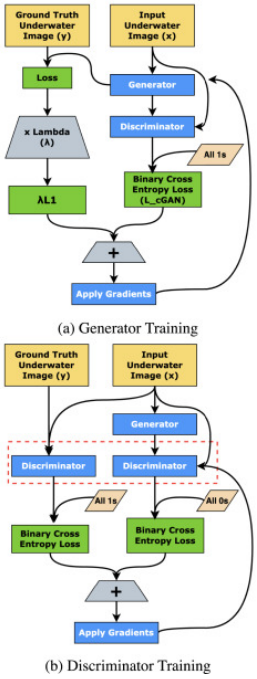


Figure 4: Diagram of training algorithm [6]

For training, the generator loss is summarized as the cross-entropy of the generated image and a tensor of ones, plus parameter λ times the mean absolute (Euclidean) error. Parameter λ is initially set to 100.

The discriminator has two loss functions, a "generated loss" and a "real loss". The first is simply summarized as the cross-entropy of the generated image and a tensor of zeros, and the second the cross-entropy of the "ground truth" image and a tensor of ones.

Preprocessing to resize the images to 256x256 pixels (using nearest neighbor) as well as normalization of the image tensor to the range [-1,1] is necessary before using the images as input. Note that for this replication jitter was not added to the input images.

The training was performed on Kaggle, which provided an Nvidia P100 card for processing (note that the original paper's authors used Google Collab instead, though this is unlikely to result in any discrepancies). The training log is available here.

4 Dataset

Same as the original paper, the dataset we used was the Enhancing Underwater Visual Perception (EUVP) Dataset, of which only the paired images were used for training and evaluation[6]. We used the Dataset on Kaggle.

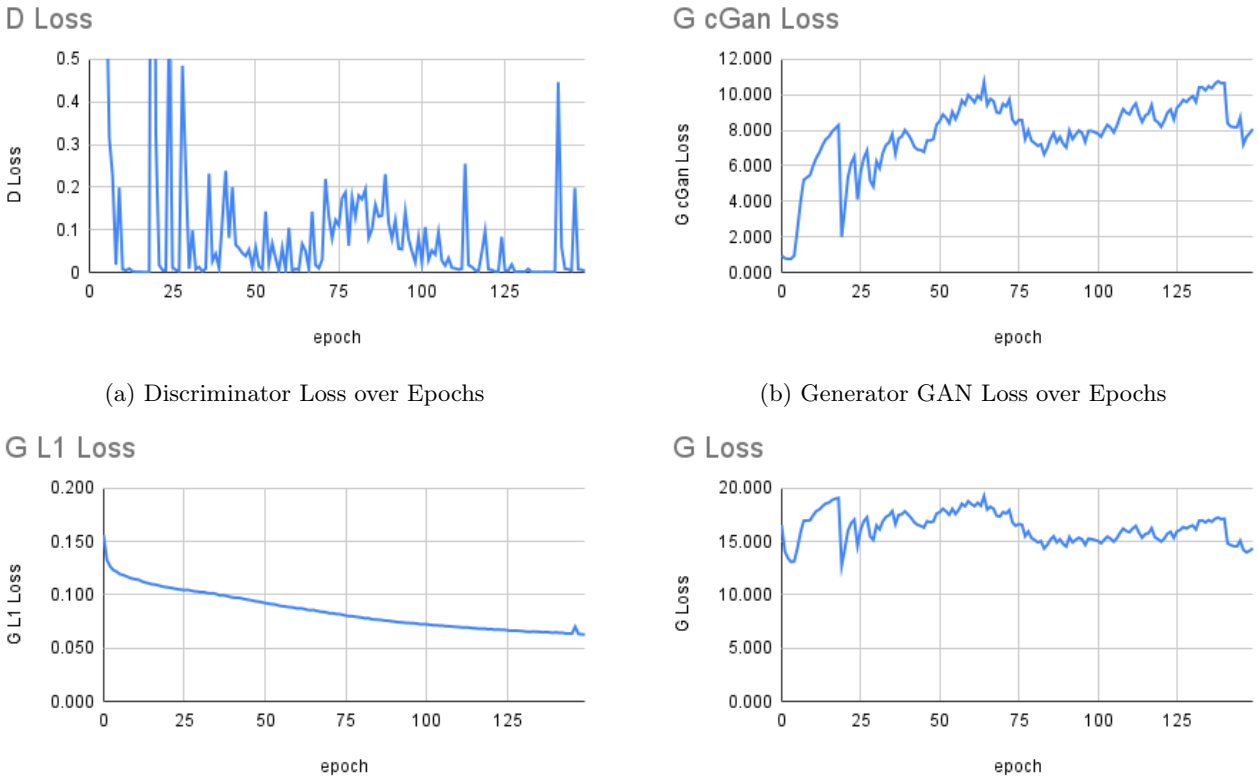
Note that the EUVP Dataset is not really a set of real corrected underwater images and their original raw counterparts, as it is logistically complicated if not impossible to obtain the same image underwater in both favorable and poor conditions. Instead, the authors of the dataset took a number of corrected images and then manually distorted them in different ways that approximately recreate the way that underwater images are initially distorted.

5 Results

Similar to [6], we first plot the loss curve to track our training progress to understand how the model learns. Then, we assess how well our model performs, both by visually inspecting its results and by measuring its performance with numbers.

5.1 The Training Progress

As an adversarial neural network, the loss curves show fluctuations, which is shown in the Figure 5 below:



(a) Discriminator Loss over Epochs

(b) Generator GAN Loss over Epochs

(c) Generator L1 Loss over Epochs

(d) Generator Loss over Epochs

Figure 5: Loss curves for Generator and Discriminator, x-axis represents epochs and y-axis represents the loss.

Despite generally mirroring the trends in the loss curve of the original paper, our result reveals a different dynamic in the distribution of losses. While the paper suggests that the generator loss is primarily driven by the L1 loss, our results indicate that the conditional GAN (cGAN) loss takes precedence, dominating the generator loss. Additionally, our observations suggest that the discriminator consistently outperforms the generator by a significant margin, indicative of a more pronounced imbalance in their performance.

5.2 Qualitative analysis

Using the identical images selected from the test dataset as those in the original paper, we utilize our Sea-Pix-GAN model to generate enhanced images. As shown in Figure 6, the layout comprises three columns showcasing

the input images, images produced by our trained Sea-Pix-GAN model, and the ground truth images, akin to Fig.10 in the original paper.

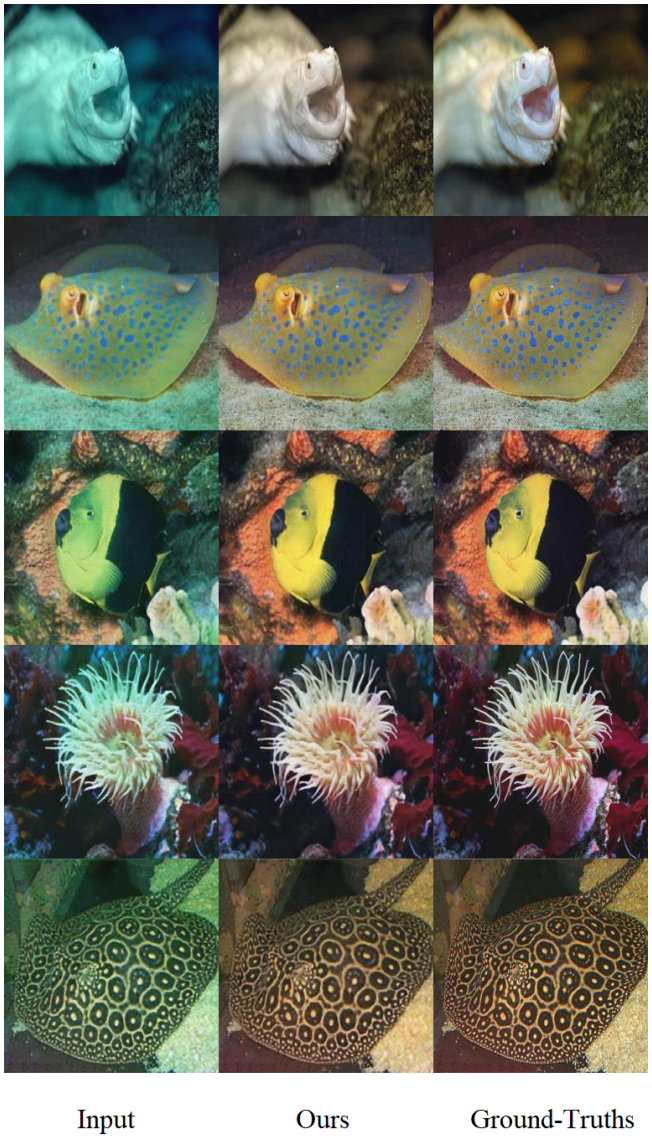


Figure 6: Enhanced images generated by our Sea-Pix-Gan model

It can be observed that the results generated by our model are comparable to the ground truth images, which also shows the effectiveness of our model. It is also notable that our Sea-Pix-GAN model consistently achieves significant enhancements across diverse scenes.

5.3 Quantitative analysis

For quantitative analysis of our model's performance, we employ Peak signal-to-noise ratio (PSNR)[2], Structural Similarity Index (SSIM)[2, 1], and Human-Visual-System-Inspired Underwater Image Quality Measures (UIQM)[3] as metrics. PSNR measures image quality by comparing original signal peak to introduced noise. Usually, Higher PSNR values indicate higher image quality, representing a smaller difference between the original and reconstructed images. SSIM assesses image similarity based on luminance, contrast, and structure. Higher SSIM values correspond to higher perceived similarity between the images. UIQM, tailored for underwater images, evaluates color, contrast, and sharpness, reflecting perceived image quality. Higher UIQM scores indicate better color rendition, contrast, and sharpness, leading to higher perceived image quality. Same as the original paper, we also limit our evaluation to 500 randomly selected paired images from the test dataset. The mean values and standard deviation for the metrics are presented in the table below:

Model	PSNR $\mu \pm \sigma$	SSIM $\mu \pm \sigma$	UIQM $\mu \pm \sigma$
Sea-Pix-Gan	24.56 ± 3.26	0.82 ± 0.09	3.10 ± 0.36

Interestingly, our model demonstrates slightly better performance than the original paper across all metrics, with higher mean values. However, the larger standard deviations observed in PSNR and SSIM suggest that our model may be less stable compared to the original.

6 Analysis and Discrepancies

In comparing our results with those presented in the paper, several discrepancies emerged, potentially attributable to variations in network configurations, hyperparameters, dataset selection, and testing procedures.

Notably, while the paper does not specify the bias settings in convolution layers, we opt to set all biases to false in our model.

Additionally, the paper lacks explicit mention of the beta_2 hyperparameter in the Adam optimizer, for which we choose to use PyTorch default value 0.999.

Another deviation from the paper's methodology pertains to the selection of the loss function. While the paper specifies the use of binary cross-entropy, which could potentially correspond to torch.nn.BCE, we choose torch.nn.BCEWithLogitsLoss in implementation to resolve the inputs that may not necessarily fall within the range of 0 to 1.

Moreover, although we utilized the EUVP dataset from Kaggle, we cannot confirm whether it precisely matches the dataset used in the paper.

Another possible explanation for the differences in results could be the variance in the composition of the training set. While the paper outlines specific subsets and their respective sizes for training, validation, and testing, our implementation may have split the dataset differently. This discrepancy in the training data composition could also impact the model's learning process and subsequent performance.

7 Appendix

7.1 Work Division

Person	Work
Charlotte Ning	• Set up project structure • Implement generator network & generator's training scripts • Run training sessions • Loss analysis
Peijie Li	• Implement discriminator network & discriminator's training scripts • Implement and run test sessions • Results and analysis section of the report
Qinxin Ren	• Set up Kaggle for training • Implement and run test sessions
Eduardo Jerez	• Implement loss functions • Set up initial integrated training algorithm • Write Introduction, Architecture, Training and Dataset sections of report

7.2 Source code

- Github repo: https://github.com/w4a2y4/sea-pix-GAN-reproduction/tree/sea-pix-gan
- PR to Funie-GAN (easier to tell which parts are specific to Sea-Pix-GAN): https://github.com/xahidbuffon/FUnIE-GAN/pull/5/files

References

[1] Alan C Bovik. *Handbook of image and video processing*. Academic press, 2010.

[2] Alain Hore and Djemel Ziou. "Image quality metrics: PSNR vs. SSIM". In: *2010 20th international conference on pattern recognition*. IEEE. 2010, pp. 2366–2369.

[3] Karen Panetta, Chen Gao, and Sos Agaian. "Human-visual-system-inspired underwater image quality measures". In: *IEEE Journal of Oceanic Engineering* 41.3 (2015), pp. 541–551.

[4] Cameron Fabbri, Md Jahidul Islam, and Junaed Sattar. "Enhancing Underwater Imagery Using Generative Adversarial Networks". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 7159–7165. DOI: 10.1109/ICRA.2018.8460552.

[5] J. Islam, Y. Xia, and J. Sattar. "Fast underwater image enhancement for improved visual perception". In: *IEEE Robotics and Automation Letters* 5 (2 2020), pp. 3227–3234. DOI: 10.1109/lra.2020.2974710.

[6] Dhiraj Chaurasia and Prateek Chhikara. "Sea-Pix-GAN: Underwater image enhancement using adversarial neural network". In: *Journal of Visual Communication and Image Representation* 98 (2024), p. 104021. ISSN: 1047-3203. DOI: https://doi.org/10.1016/j.jvcir.2023.104021. URL: https://www.sciencedirect.com/science/article/pii/S1047320323002717.