Sea-Pix-GAN: Underwater Image Enhancement Using Adversarial Neural Networks - Reproduction

Henrik Brinch van Meenen (60706698), Hans Dahle Kvadsheim (6082955), Yuanfu Pan (5933749), and Thijs Penning (5266297)

Delft University of Technology

{*hkvadsheim, hbvmeenen, y.pan-19, t.j.penning*}*@student.tudelft.nl*

April 14, 2024

GitHub Repository

1 Work distribution

To portray the work distribution load within the team for this project, we constructed the table below. We divided the project into 8 sub-tasks, assigned the main contributors to each sub-task, and evaluated the 'effort' amount in the right-most column.

Table 1: Workload Distribution and Contribution Value

Task	Responsible	Contribution Value
Image Processing and storage	Thijs	10
Generator Model	Henrik - Hans	10 - 5
Discriminator Model	Henrik - Thijs	10 - 5
Training Loop	Henrik - Yuanfu	10 - 10
Metric Calculation	Thijs - Hans	5 - 15
Bug Fixing and code review	Henrik - Hans - Thijs - Yuanfu	5 - 5 - 5 - 5
Environment setup + Training Model	Henrik - Thijs - Yuanfu	5 - 10 - 5
Blog Writing	Henrik - Hans - Thijs - Yuanfu	3 - 5 - 4 - 3

2 Introduction

2.1 About the Paper

The paper "Sea-Pix-GAN: Underwater image enhancement using adversarial neural networks" by Dhiraj Chaurasia and Prateek Chhikara (2024) introduces a Generative Adversarial Network

(GAN)-based model for enhancing underwater images. The primary contribution of this paper is the development of Sea-Pix-GAN, which addresses the challenges of underwater image processing, such as color degradation, low contrast, and texture preservation, by redefining the problem as an image-to-image translation task. Their introduced model is capable of learning to enhance underwater images by tailoring the objective and loss functions to achieve color, content, and style transfer.

2.2 Previous contributions

The Sea-Pix-GAN model builds upon previous work in underwater image enhancement and GAN-based image processing techniques. Prior to this study, various approaches have been utilized to improve underwater image quality, including hardware advancements, image processing techniques like Histogram Equalization (HE), Contrast Limited Adaptive Histogram Equalization (CLAHE), and deep learning models such as LANet and Deep WaveNet. These methods have focused on contrast and color restoration, yet often struggle with color accuracy and detail preservation in underwater imaging.

"U-Shape Transformer for Underwater Image Enhancement" [1] proposes a transformer-based model for enhancing underwater images by addressing the issue of poor imaging quality due to underwater impurities. "Meta underwater camera: A smart protocol for underwater image enhancement"[2] introduces a protocol comprising a comprehensive cascade of seven image enhancement techniques, aiming to improve underwater image visibility. "Underwater Image Enhancement Method via Multi-Interval Subhistogram Perspective Equalization"[3] focuses on a histogram-based method for enhancing underwater images by addressing quality degradation issues. "Domain Adaptation for Underwater Image Enhancement"[4] introduces a domain adaptation method for enhancing underwater images, focusing on inter- and intra-domain concepts. The latest and possibly the most powerful technique so far was the Dual branch Transformer-CNN parametric filtering network for underwater image enhancement, addressed by "Dual branch Transformer-CNN parametric filtering network for underwater image enhancement"[5]. This work suggests a DTCPF network, which has a gradient prediction block, attention prediction block, and smoothing prediction, which are executed in loops to achieve loss function coverage. DTCPF results in desirable smoothness and pixel color accuracy.

3 Methodology

3.1 Data Collection and pre-processing

We gathered all our training and testing data from the University of Minnesota's EUVP dataset [1]. We used their paired dataset, containing 11.435 pairs of underwater pictures. We split the pairs in the ratio of 4:1:1 for training, validation, and testing. Each of the pictures has been resized to a 256x256 frame. We then added random jitter to images by mirroring both vertically and horizontally and randomly cropping a 256x256 frame. All tensors were normalized to a $[-1, 1]$ range.

[1] https://irvlab.cs.umn.edu/resources/euvp-dataset

3.2 The Network

The Sea-Pix-GAN framework is centered around the Generative Adversarial Network (GAN) structure, which comprises two main components: the Generator (G) and the Discriminator (D), engaged in a concurrent training mechanism within an adversarial setup. This framework essentially forms a minimax game where the Generator attempts to create enhanced underwater images from an input noise vector, denoted as $G : z \rightarrow y$, aiming to mimic the real underwater image distribution closely. The Discriminator, D, evaluates whether the images are genuine underwater images or fabrications by the Generator. Incorporating conditional GANs (cGAN), inspired by pix2pix and U-net architectures, facilitates the Generator in learning the transformation $G : x, z \rightarrow y$, enhancing its capability to generate "real" appearing synthetic images. In our reproduction we implemented the noise vector by adding dropout to the first three decoder layers. We used a dropout probability of 0.5 The cGAN's objective function is formalized as:

$$\min_G \max_D L_{GAN_C}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z)))] \tag{1}$$

This adversarial relationship forces the Discriminator to accurately differentiate between real and generated images, thereby guiding the Generator to produce indistinguishable enhancements. To tailor the model for underwater image enhancement, integrating the L1 loss function addresses the aspect of aligning generated images closely with real-world visuals. This L1 loss is expressed mathematically as:

$$L_{L1}(G) = \mathbb{E}_{x,y,z}[|y - G(x, z)|] \tag{2}$$

Therefore, the ultimate objective function for Sea-Pix-GAN is formulated as:

$$G^* = \min_G \max_D \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z)))] + \lambda(\mathbb{E}_{x,y,z}[|y - G(x, z)|]) \tag{3}$$

where λ is a parameter controlling the balance between the adversarial loss and the L1 loss. This refined objective underscores the model's dedication to not only generating visually compelling underwater images but also ensuring these enhancements are practically aligned with their authentic counterparts. In the paper and our recreation, λ is set to 100.

3.3 The Generator

In replicating the Sea-Pix-GAN's approach to underwater image enhancement, we adopted its generative model, emphasizing a consistent pixel structure through an encoder-decoder framework with skip connections. The architecture features convolutional layers, batch normalization, and ReLU activations. We copied the hyperparameters from the original paper in all layers. However, to make the final output match the desired image format, we finish off with a *Tanh* layer to have the final range be between $(-1, 1)$.

$$\text{Tanh}(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \tag{4}$$

This process is not mentioned in the original paper, though the authors must perform some sort of output normalization to use the generated images in the discriminator.

3.4 The Discriminator

In our replication of Sea-Pix-GAN, the Discriminator employs a convolutional Markovian PatchGAN design, due to its high-frequency detail discernment, addressing the blur issues common with using Euclidean losses alone. The model evaluates authenticity at the patch level differentiating between "real" and generated image sections. It processes inputs through a series of 2D convolutional layers, mirroring the encoder's downsampling technique, to decide on the realism of 70 × 70 patches in the input images. We used the same hyperparameters as the original paper for all layers. We needed to make the final output of the Discriminator to represent a probability function. Therefore we finished off the model with a *Sigmoid* function

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

to limit the output to be in the range $(0, 1)$. This process is not mentioned in the original paper but is required to use the binary cross entropy loss function in PyTorch to stay consistent with their reported training procedure.

3.5 Training

In our reproduction of the Sea-Pix-GAN model's training process, we follow the same training procedure as in the paper. The Generator's training (see figure 1) hinges on the binary cross-entropy loss with a tensor of ones, supplemented by an L1 loss to enhance low-frequency detail accuracy, utilizing a lambda (λ) value of 100 as the trade-off factor. The Discriminator's training (see figure 2) evaluates both generated and ground truth images to compute its loss, derived from the binary cross-entropy loss of the generated image against a tensor of zeros and the real image against a tensor of ones, summing these for the final loss. Preprocessing involved resizing images to 256x256 and augmenting them with random jitter and mirroring, followed by normalization within the $[-1, 1]$ range. Both the models' weights were updated at every batch using the Adam Optimizer with *learning rate* $2e - 4$, $\beta_1 = 0.5$, and $\beta_2 = 0.999$ over a batch size of 64.

To train our models, we made use of the Kaggle platform [2] and a privately owned GPU. (TODO reference). When using Kaggle, we uploaded the paired dataset, along with our recreated models. The platform allows access to cloud NVidia K80 GPUs for model training. A similar process was done on the privately owned RTX3060 we had available, with all files stored locally instead. We trained our evaluated models for 150 epochs, aligning as closely to the reported procedure of the original paper as possible. Some GAN papers describe alternating optimization steps for the generator/discriminator, however, this was not described in our paper, therefore, we assume both models were trained using every batch.

3.6 Evaluation

To evaluate the performance of our recreated Sea-Pix-GAN model, we employed the same three quantitative metrics as described in the original research: Peak Signal-to-Noise Ratio (PSNR),

[2] https://www.kaggle.com/

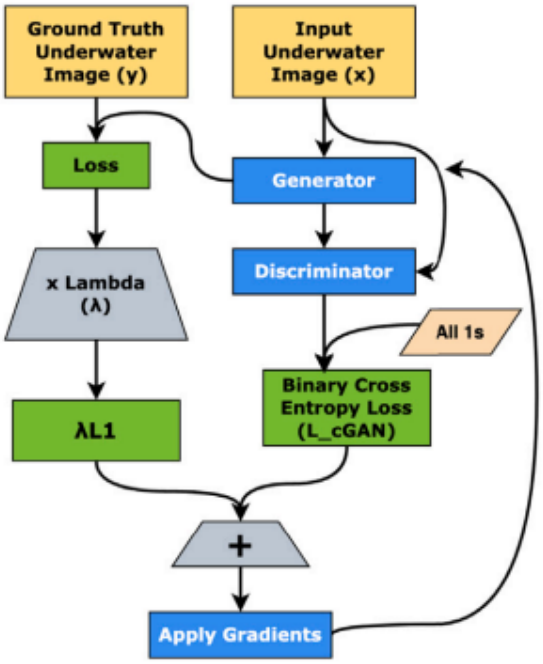

Figure 1: Training process of the Generator.

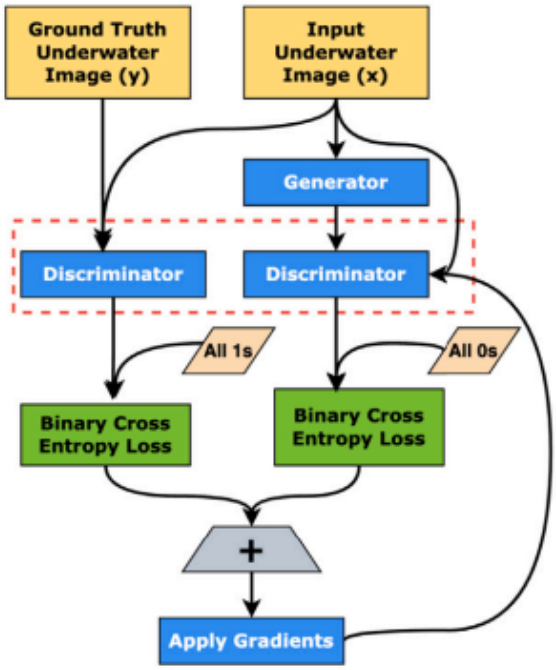

Figure 2: Training process of the Discriminator.

Structural Similarity Index (SSIM), and Human Visual-System-Inspired Underwater Image Quality Measures (UIQM). The paper did not specify if a standard implementation of these was used, so we re-implemented them to the best of our ability.

PSNR evaluates the quality of reconstruction of noisy images by comparing pixel values of the input and reference images, where a higher PSNR value indicates better quality. For computing the ratio we used the paper-provided equation:

$$PSNR(x, y) = 10 \log_{10} \left(\frac{MAX^2}{MSE(x, y)} \right) dB, \tag{6}$$

Here $MAX = 1$ and MSE is the mean squared error between pixel values of the input x and reference images y.

SSIM measures the similarity between two images in terms of luminance, contrast, and structure, providing a more comprehensive reflection of perceived image quality. We computed the SSIM by the equation:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \tag{7}$$

Here μ_x and μ_y are the mean values of x and y, σ_{xy} is the covariance of x and y, and σ_x and σ_y are the variances of x and y. The constants c_1 and c_2 are included to stabilize the division with a weak denominator.

UIQM evaluates underwater image quality based on colorfulness, sharpness, and contrast without necessitating a reference image, offering insights into the subjective aspects of image quality

that are particularly relevant to underwater imaging. For computing the UIQM we relied on the paper "Human-Visual-System-Inspired Underwater Image Quality Measures" by Panetta, K. et al. The UIQM is calculated using three primary components: the Underwater Image Colorfulness Measure (UICM), the Underwater Image Sharpness Measure (UISM), and the Underwater Image Contrast Measure (UIConM), each contributing to the final UIQM value as follows:

$$UIQM = c_1 \times UICM + c_2 \times UISM + c_3 \times UIConM, \tag{8}$$

The constants $c_1 = 0.0282$, $c_2 = 0.2953$, and $c_3 = 3.5753$ are empirically determined weights for each component measure. The UICM evaluates color vividness, UISM assesses image sharpness, and UIConM quantifies image contrast, combining to form a comprehensive quality assessment.

The UICM component is computed as:

$$UICM = c_1 \times (\sqrt{\mu_{RG}^2 + \mu_{YB}^2}) + c_2 \times (\sqrt{\sigma_{RG}^2 + \sigma_{YB}^2}), \tag{9}$$

Here μ_{RG} and μ_{YB} are the mean values of the RG and YB color channels, respectively, and σ_{RG} and σ_{YB} represent their standard deviations. Constants c_1 and c_2 are specifically chosen for UICM to reflect the colorfulness aspect of underwater imagery.

UISM employs edge detection techniques to measure image sharpness:

$$UISM = \lambda_R \times EME_R + \lambda_G \times EME_G + \lambda_B \times EME_B, \tag{10}$$

Here EME_R, EME_G, and EME_B are the Enhancement Measure Estimations for the red, green, and blue channels, respectively, weighted by their perceptual importance λ_R, λ_G, and λ_B.

The Enhancement Measure Estimations are computed as:

$$EME = \frac{2}{k_1 \times k_2} \sum_{l=1}^{k_1} \sum_{k=1}^{k_2} \log \left(\frac{I_{\text{max},k,l}}{I_{\text{min},k,l}} \right), \tag{11}$$

Here the image is divided into blocks of size $k_1 \times k_2$. $I_{\text{max},k,l}$ and $I_{\text{min},k,l}$ denote the maximum and minimum intensity values within each block, respectively. The Sea-Pix-Gan paper does not specify which block size is used in their calculations. We simply used a block size of picture width X height.

The UIConM measure leverages logarithmic transformations to assess the contrast and is defined as:

$$UIConM = \text{logAMEE}(I), \tag{12}$$

$$\text{logAMEE} = \frac{1}{k_1 \times k_2} \bigotimes \sum_{l=1}^{k_1} \sum_{k=1}^{k_2} \frac{I_{\text{max},k,l} \theta I_{\text{min},k,l}}{I_{\text{max},k,l} \oplus I_{\text{min},k,l}} \log \left(\frac{I_{\text{max},k,l} \theta I_{\text{min},k,l}}{I_{\text{max},k,l} \oplus I_{\text{min},k,l}} \right), \tag{13}$$

The image is again divided into $k_1 \times k_2$ blocks of size WxH, and $\oplus$, $\otimes$, and θ are the PLIP (Perceptual LIP) operations, providing nonlinear representations that align with human visual perceptions. The PLIP operations enhance the contrast in areas with low luminance.

Similar to the Sea-Pix-GAN paper, we computed these metrics on a randomly selected subset of 500 paired images.

4 Results

In Figure 3 we provide a qualitative comparison of our reproduced results using the same images used in Figure 10 of the original paper. From the figure, we can see that our results closely match the Sea-Pix-GAN and ground truth images usually being somewhere in between the two. The exact colors vary slightly which is likely due to random factors in the training such as the data distribution. With the different colors some of our images are closer to the ground truth while some are farther off which shows that the models have a similar performance.

In our images there are two main problems, first, it has difficulty in transforming seemingly blue patches to their correct color. This can be seen in the mouth of the turtle, however, the original Sea-Pix-GAN model seems to have the exact same issue here where these areas become gray. Secondly, some images have green pixels on them, we do not know why this occurs but we assume that it is related to why our loss statistics looked different from the paper. All in all, our reproduction can generate similar quality images as Sea-Pix-GAN with the only problem being those green dots.

In this section we present the summary statistics for the different metrics. The first table includes the values reported in the original paper, including the UIQM mean for the ground truth images. This was included in the text, but not in any table, and so the standard deviation of the ground truth UIQM values is not available as far as we can tell. The two tables can not be compared strictly one-to-one, given a lack of specific dataset partition information and (possibly) divergent parameters within the UIQM metric calculation, but they do indicate that the models behave similarly.

Metric	Mean (μ)	Std ($\pm\sigma$)
PSNR	23.30	± 1.68
SSIM	0.79	± 0.09
UIQM (GAN images)	2.84	± 0.2
UIQM (Ground Truth)	2.89	Not specified

Table 2: Metric statistics reported by original paper

Metric	Mean (μ)	Std ($\pm\sigma$)
PSNR	22.96	± 3.21
SSIM	0.93	± 0.07
UIQM (GAN images)	4.87	± 0.10
UIQM (Ground Truth)	4.92	± 0.16

Table 3: Metric statistics for our model reproduction

The PSNR and SSIM metrics are within the same order of magnitude, but our values are somewhat higher. The differences might be due to the specific image selection in the test set partition we

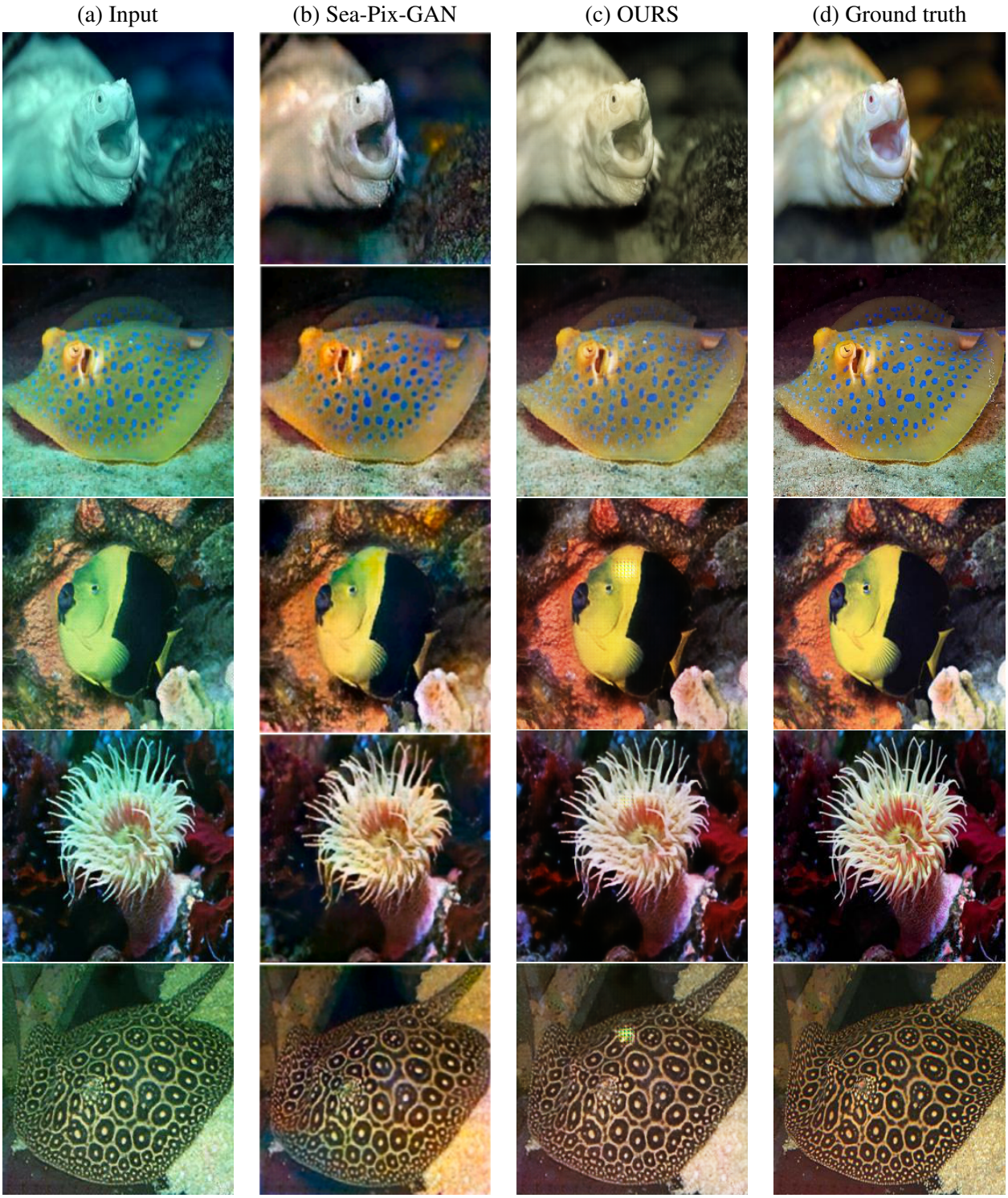

Figure 3: Qualitative comparison between Sea-Pix-GAN and our reproduction using the five test images used in the original paper.

used [3] and plain luck with our training run. The UIQM values are quite a bit higher in our reproduction, which might very well be due to our specific UIQM implementation. The UIQM metric is orders of magnitude more complex than PSNR and SSIM and contains several adjustable parameters within its different sub-metrics, ranging from block size fidelity parameters to adjustable behavior within the PLIP operators. These are left unspecified by the original paper, and so it is difficult to verify the accuracy of our UIQM reproduction. We made (what are hopefully) reasonable assumptions, but have no way to verify how the authors of the Sea-pix-GAN paper set them. However, the paper did luckily specify the mean UIQM value they got on the Ground Truth images. And if we look at the relative difference between the ground truth and the generated images in our reproduction, we do see the same overall result. That is: the mean UIQM of images generated by the model is close to, but a little bit lower than, the mean UIQM of the ground truth images. As mentioned the ground truth UIQM standard deviation is not included, so it is impossible to compare the spread of UIQM in our results to the original paper.

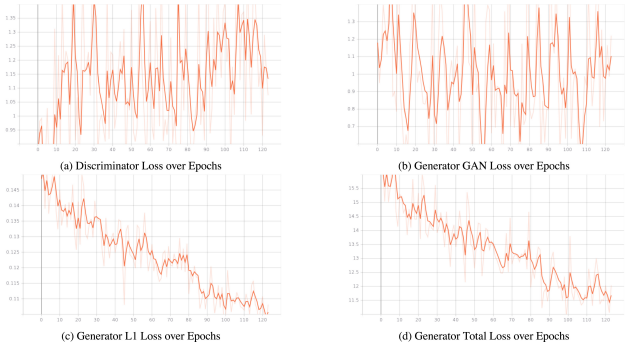

Figure 4: Model training loss curves from the original paper

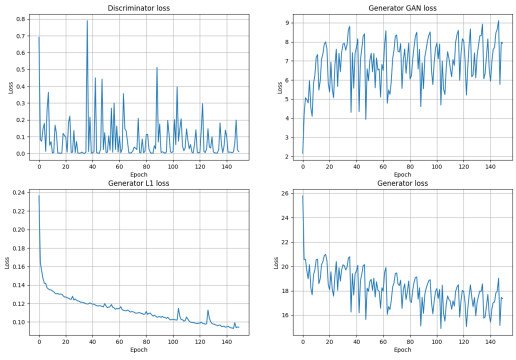

Figure 5: Model training loss curves in our reproduction

We also had some trouble reproducing the loss curves the authors described. The trends we found in our curves don't line up with the explanation written in the paper as can be seen when

[3] Although we cannot say with absolute certainty that our dataset partitions are not identical in terms of image selection, it is *extremely* unlikely

comparing Figure 4 and Figure 5. Most notable is our L1 loss curve which has a sort of exponential drop-off in the beginning, compared to the linear decline in the original paper. We also see an upward trend in the generator GAN loss and our discriminator loss stays close to zero, with sharp spiking losses that grow smaller over time, instead of trending upward. Despite these discrepancies, the resulting images from our model do seem to line up quite well, as previously discussed.

5 Discussion

The paper includes a lot of important detail, but maybe not enough to verify their exact results to a high degree of precision. Most of the basic architectures, along with the dataset and the critical hyper-parameters are readily available within the paper or are well-referenced. Although the training procedure is presented in a somewhat (over-)complicated mathematical fashion to begin with, the figures make it clearer what is going on later in the paper. Getting a basic approximation of the model up and running is not a great challenge as long as one has prior experience with PyTorch [4]. Getting a basic working approach up and going requires just a few disorganized students and (in total) the better part of a work week. To match the paper's quality metrics and decipher the finer details of the paper requires hours of frustration and extensive searches within the paper's references, with no guarantee of results.

Some choices (especially regarding output normalization in both models and how they handle cGAN noise input) are left quite unclear and require consulting other sources and individual judgement. To resolve the output normalization issue we had to perform architecture modifications by adding output-restricting activation functions at the output layers, which are not reported in the paper. Without these, the loss functions specified in the paper would not work, at least not the way they are implemented in PyTorch. On the evaluation side, we have the issue of UIQM. This metric has a host of parameters that the paper left out, making it hard to verify that the metrics are computed equivalently. The metrics are also somewhat complex to implement (especially UIQM), which further increases the risk that our values are not computed equivalently. If the authors used a standard implementation of the included metrics, it would have been very helpful if references were included to minimize necessary work and to (more importantly) improve reproduction accuracy.

Another important omission is the lack of dataset partition information. This effectively makes it impossible to perform one-to-one statistical comparisons between the reproduced model and the model created by the authors. Since the model is stochastically initialized and optimized there will always be some variation between training runs but with access to the specific dataset partition information [5] it would be possible to conclude whether the differences are statistically significant or not by testing our model on the exact same images. Without it, it becomes much harder to say anything quantitatively with certainty since the test set underlying the metrics is almost certainly partitioned differently than that of the authors.

Qualitatively it is very much possible to verify that Sea-Pix-GAN is a viable approach to underwater image enhancement with a host of benefits, but a lack of fine details about the evaluation procedure makes it difficult to verify the specifics of the quantitative performance metrics. Repro-

[4] https://pytorch.org/

[5] That is: which images from the dataset are in which partition. Most critically: which images are in the training set and which images are in the test set

ducing the expected loss curves has also proven to be most challenging, and the provided hyperparameters do not lead to the loss curve trends described by the authors. This might be a result of our output normalization modifications, but this is pure conjecture as we do not have access to a point of direct comparison with the original model, so a comparative test is not possible.

However, overall the results are qualitatively and quantitatively similar. The main conclusions of the paper stand, but with the information provided we are not able to reproduce the quantitative observations to any large degree of confidence. This applies especially to the expected loss curves and the UIQM results.

References

[1] L. Peng, C. Zhu, and L. Bian. U-shape transformer for underwater image enhancement. *IEEE Transactions on Image Processing*, 32:3066–3079, 2021.

[2] H. Wang, S. Sun, and P. Ren. Meta underwater camera: A smart protocol for underwater image enhancement. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2023.

[3] J. Zhou, L. Pang, D. Zhang, and W. Zhang. Underwater image enhancement method via multi-interval subhistogram perspective equalization. *IEEE Journal of Oceanic Engineering*, 48:474–488, 2023.

[4] Z. Wang, L. Shen, M. Xu, M. Yu, K. Wang, and Y. Lin. Domain adaptation for underwater image enhancement. *IEEE Transactions on Image Processing*, 32:1442–1457, 2021.

[5] B. Chang, J. Li, L. Ren, and Z. Chen. Dual branch transformer-cnn parametric filtering network for underwater image enhancement. *Journal of Visual Communication and Image Representation*, 106:1–15, 2024.