Optimizing Underwater Image Quality with Deep Reinforcement Learning

Marc Reyes
AI 322 Reinforcement Learning Mini Project

Background

- Underwater imaging is critical for various applications such as marine biology, underwater archaeology, and pipeline inspection.
    - New species are being discovered every year.
- Challenges include poor visibility, color distortion, and low contrast due to the water medium.
- Key Issues
    - Degradation Factors
        - Light scattering and absorption cause significant degradation in image quality.
        - Environmental conditions like turbidity and particulate matter further complicate imaging.
    - Existing Solutions
        - Traditional image enhancement techniques often fail to address all underwater image quality issues comprehensively.
        - Recent approaches using convolutional neural networks (CNNs) show promise but have limitations in adaptability to varying underwater conditions.

Objective

Develop a Deep Q-Network (DQN)-based model to enhance underwater image quality by addressing brightness, contrast, and color saturation.

Related Works

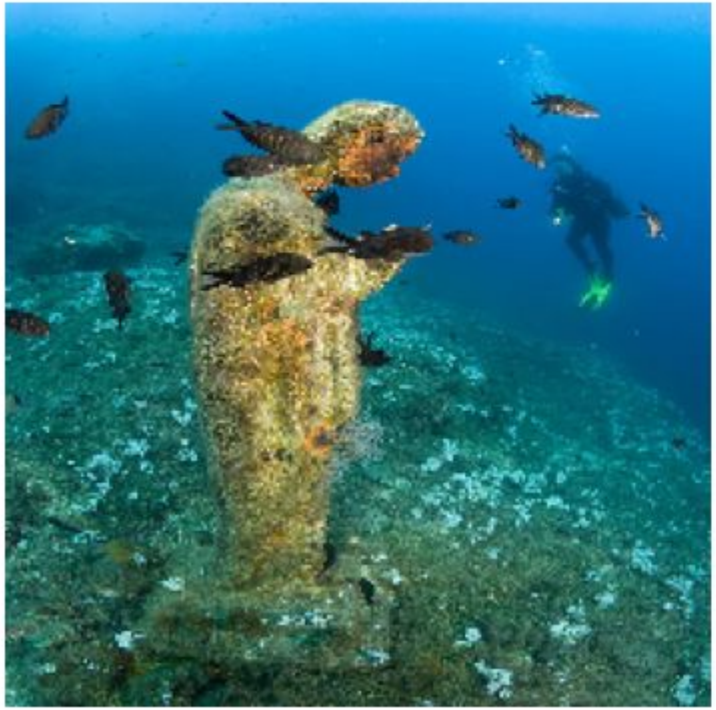
- PixelRL: Fully Convolutional Network with Reinforcement Learning for Image Processing
  - This study introduces PixelRL, applying RL to pixel-wise rewards for tasks like color enhancement and super-resolution.
- U-shape Transformer for Underwater Image Enhancement
  - This paper presents a U-shape Transformer model designed to enhance underwater images by addressing issues like color distortion and low visibility.

Datasets

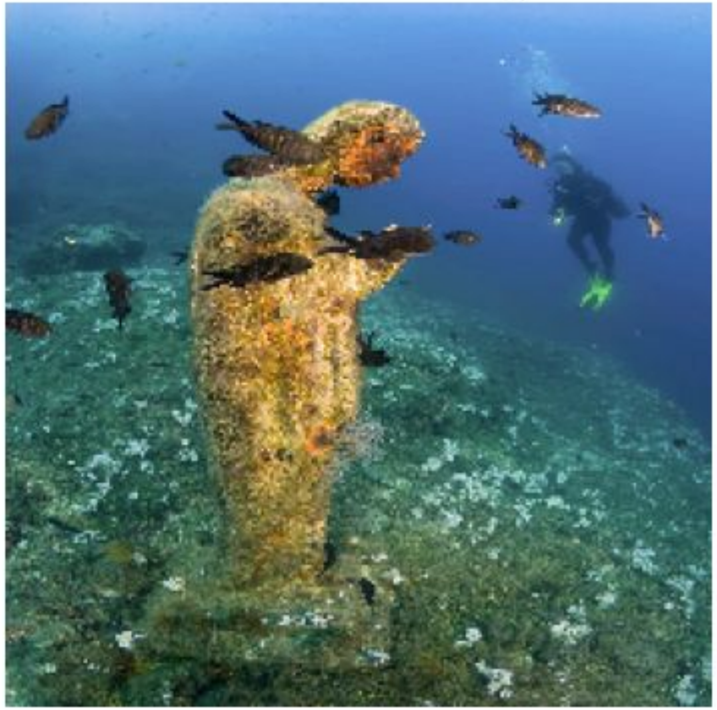
- Large-scale underwater image (LSUI)
  - Dataset with 5004 image pairs. The dataset features diverse underwater scenes with various lighting conditions, water types, and target categories, offering high-quality reference images.
- Underwater Image Enhancement Benchmark (UIEB)
  - Ccontains 890 raw images with corresponding high-quality reference images.
  - It also includes 60 images without a corresponding reference, defined as challenge images.

Datasets

Raw image

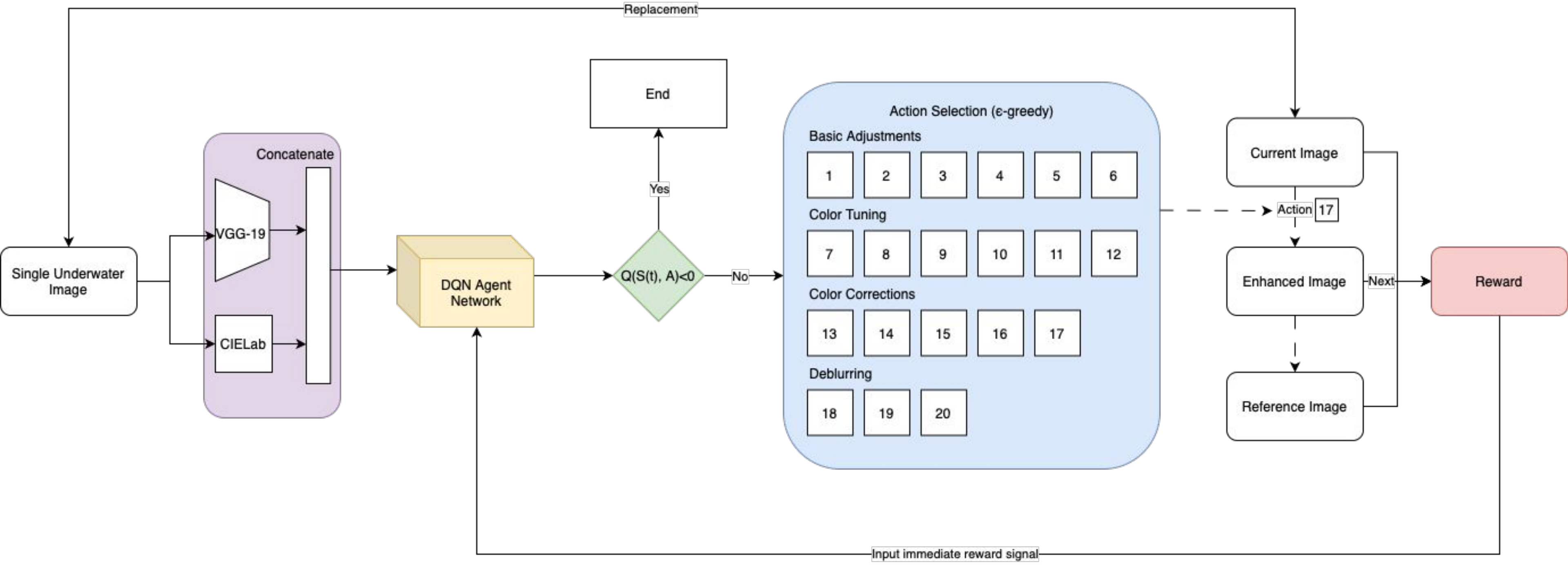


White balanced image



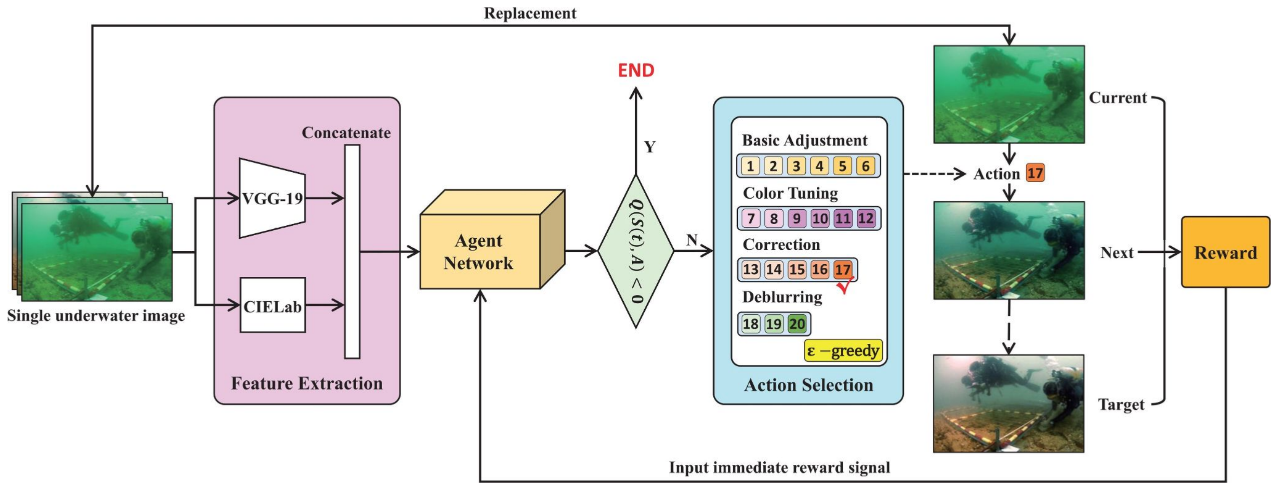
General Framework

- Feature extraction. Extracts features from an image. The extracted features will constitute the states.
- Deep Q-Network. Takes a state as input and returns a $Q(s,a)$ value for each action.
- Action selection. Chooses an action using an ϵ-greedy policy.
- Reward calculation



General Framework

- Feature extraction
  - Extracts features from an image. The extracted features will constitute the states.
- Deep Q-Network
  - Takes a state as input and returns a $Q(s,a)$ value for each action.
- Action selection
  - Chooses an action using an ϵ-greedy policy.
- Reward calculation



General Framework

The model solves a Markov Decision Process (MDP) that consists of the interaction at each step t between an agent and an environment. The MDP is composed of four components:

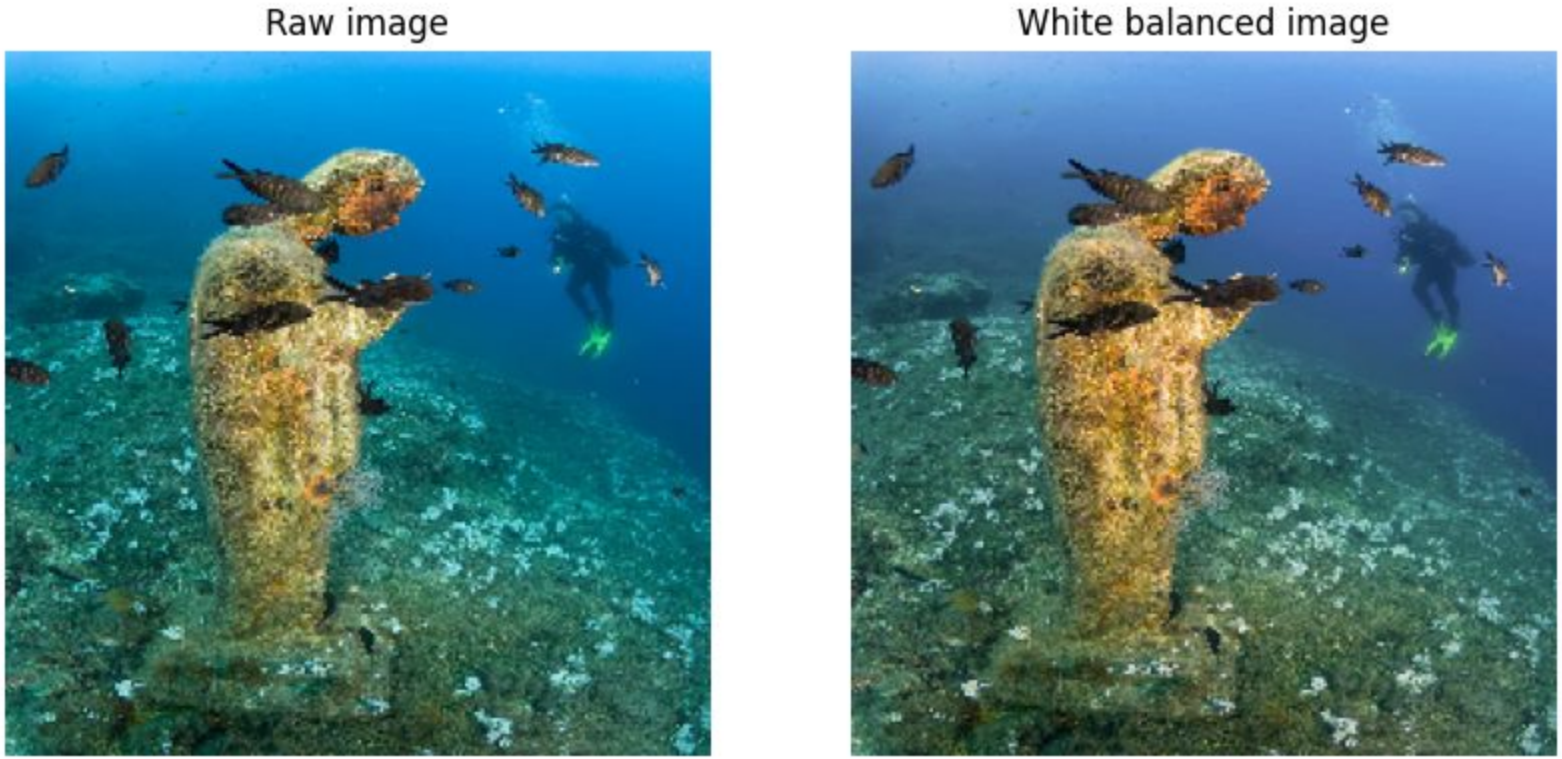
- **a state $s_t \in S$** which consists of the set of information that the agent perceives from the environment: in this case, the state is made up of features extracted from the environment’s images.
- **an action $a_t \in A$** which consists of any behavior that the agent can perform in the environment: in this case, it involves image enhancement operations.
- **a reward $r_t \in R$** which is feedback that the agent receives from the environment after performing an action: in this case, the reward defines how much the action produces a result closer to the ground truth image compared to the starting image.
- **a policy Ω_θ** which is the strategy that the agent applies to decide what its next action will be based on the current state it is in: in this case, an ϵ-greedy policy.

Actions

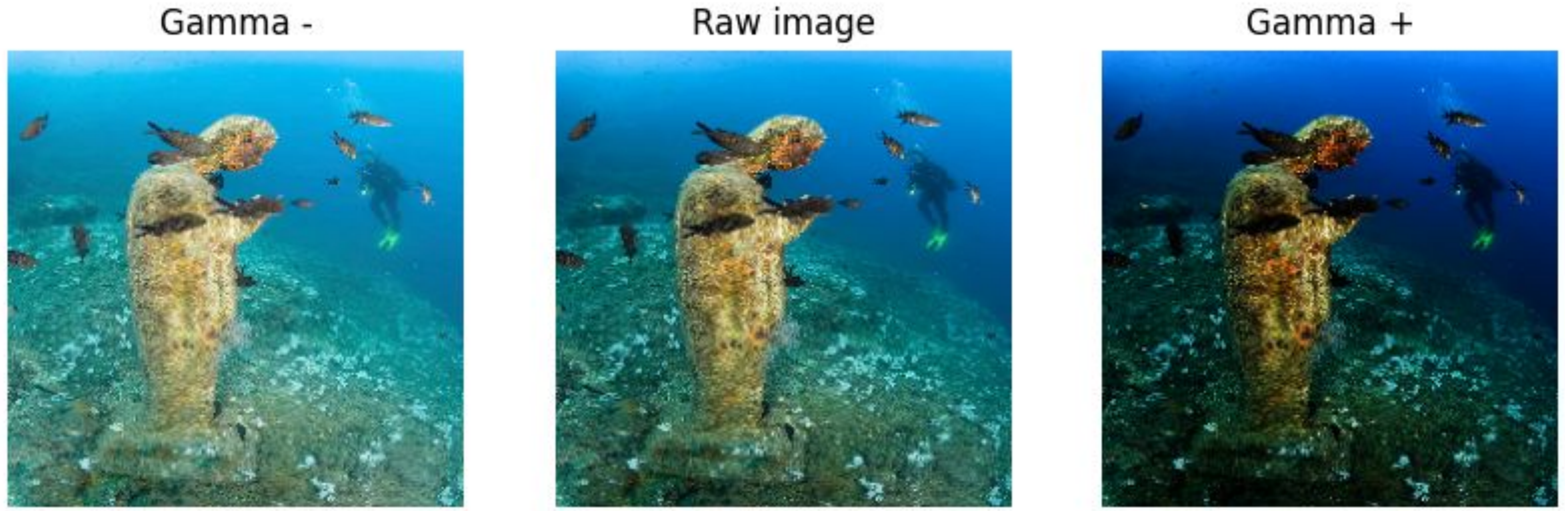
As actions, image enhancement techniques are used. Below are the four categories of techniques, each followed by a detailed list of the specific techniques belonging to that category:

- Basic Adjustments
    - Brightness +5%
    - Brightness -5%
    - Contrast +5%
    - Contrast -5%
    - Color Saturation +5%
    - Color Saturation -5%
- Color Adjustments
    - Red +5%
    - Red -5%
    - Green +5%
    - Green -5%
    - Blue +5%
    - Blue -5%
- Corrections
    - Gamma Transformation +
    - Gamma Transformation -
    - HE (Histogram Equalization)
    - CLAHE (Contrast Limited Adaptive Histogram Equalization)
    - White Balance
- Deblurring
    - Sharpen
    - Emboss
    - DCP (Dark Channel Prior)

White balance correction

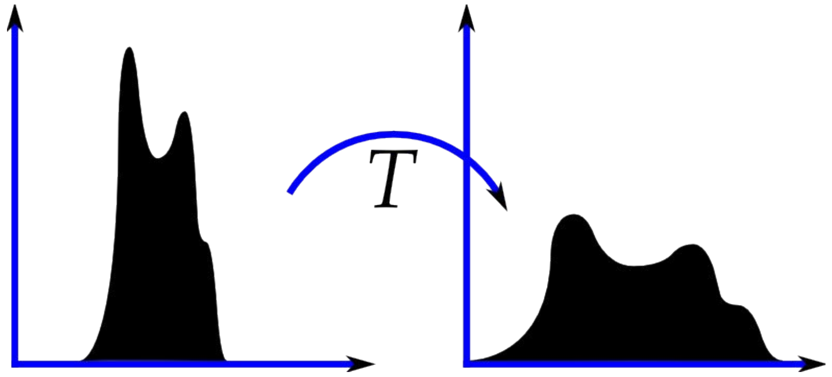


Gamma Correction

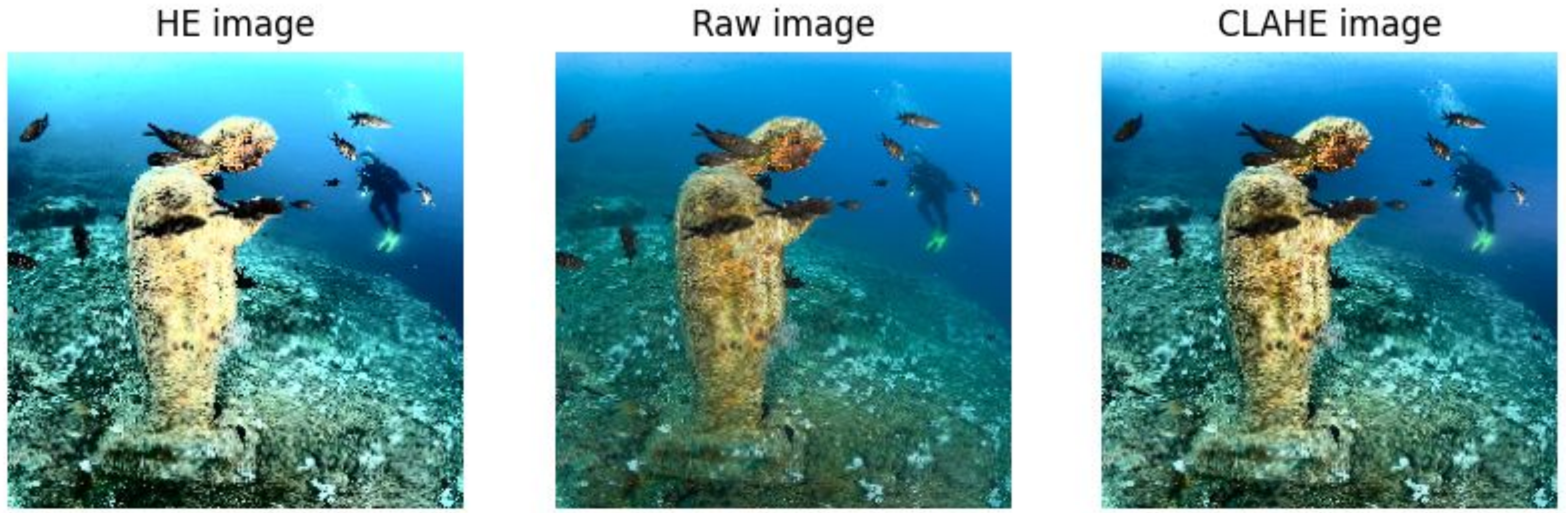


HE and CLAHE

- Histogram Equalization (HE)
  - Enhances image contrast by redistributing intensity values.
- Contrast Limited Adaptive Histogram Equalization (CLAHE)
  - Limits contrast amplification to reduce noise amplification.

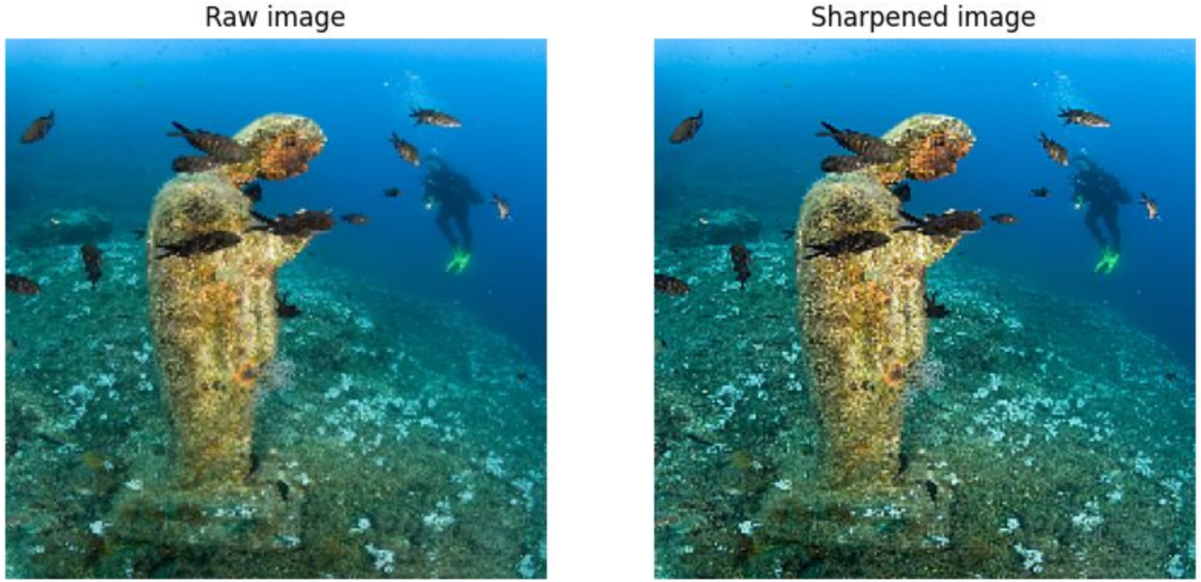


HE and CLAHE



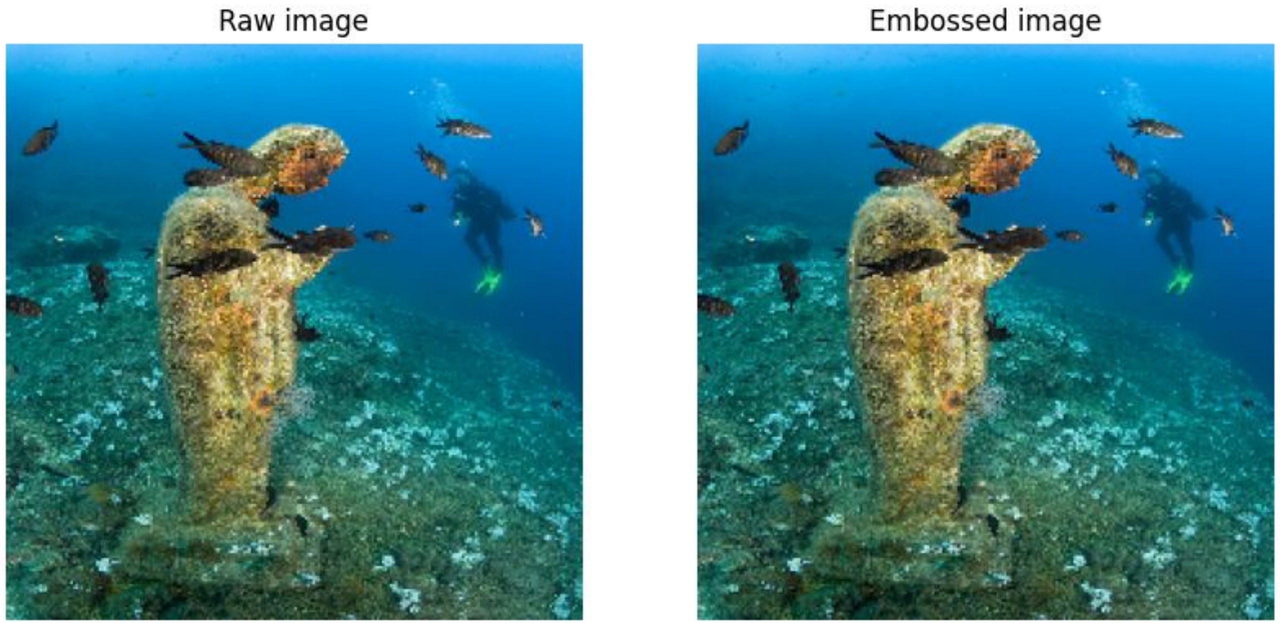
Sharpen

The sharpen filter makes the transition from one region of the image to another more obvious



Emboss

Emboss applies a filter to the image to create a 3D relief cast of it.

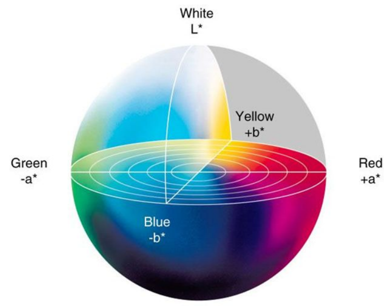


Feature Extraction

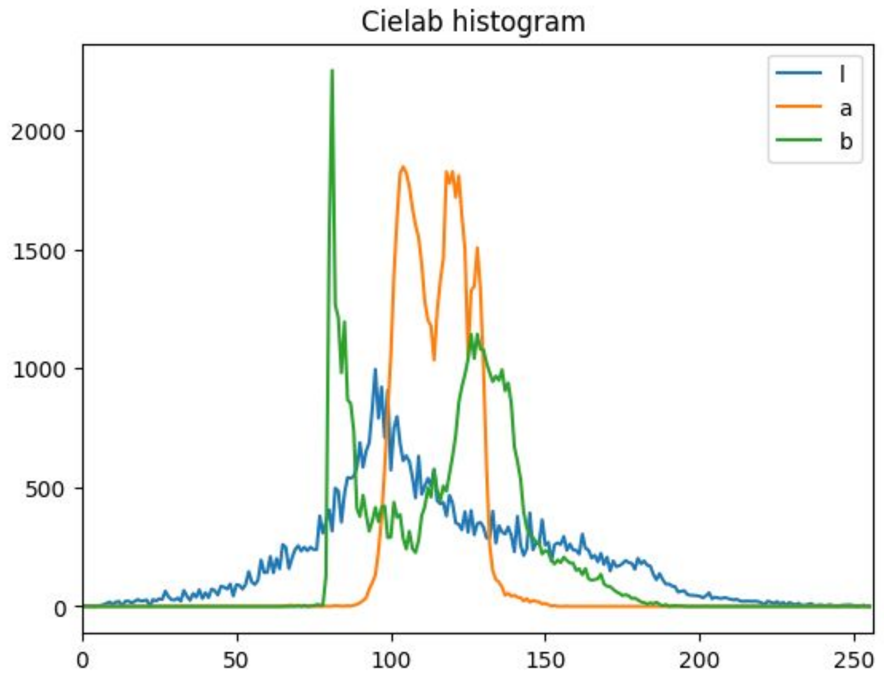
- The concatenation of the Color Features and the Perceptual Features represents the state that the agent takes as input.
    - Color feature: Global features that describe the properties of the image surface.
        - CIELAB histogram is used. Each axis of the CIELAB space is divided into 20 intervals, and for each interval, the number of pixels it contains is calculated. This results in a 202020 histogram with the values of the Color Features (which in total amount to 8000).
    - Perceptual feature: Context-dependent features, inspired by human vision.
        - Extracted through a pre-trained VGG-NET 19. They are obtained by taking the output from one of the fully connected layers, resulting in 4096 features.
        - Lacking a labeled dataset, a model pretrained on the ImageNet dataset was chosen to extract features. Models trained on ImageNet have learned to extract features from a wide variety of images, and should perform well even in the application domain of underwater images.
- 8000 + 4096 = 12096 features

CIELAB

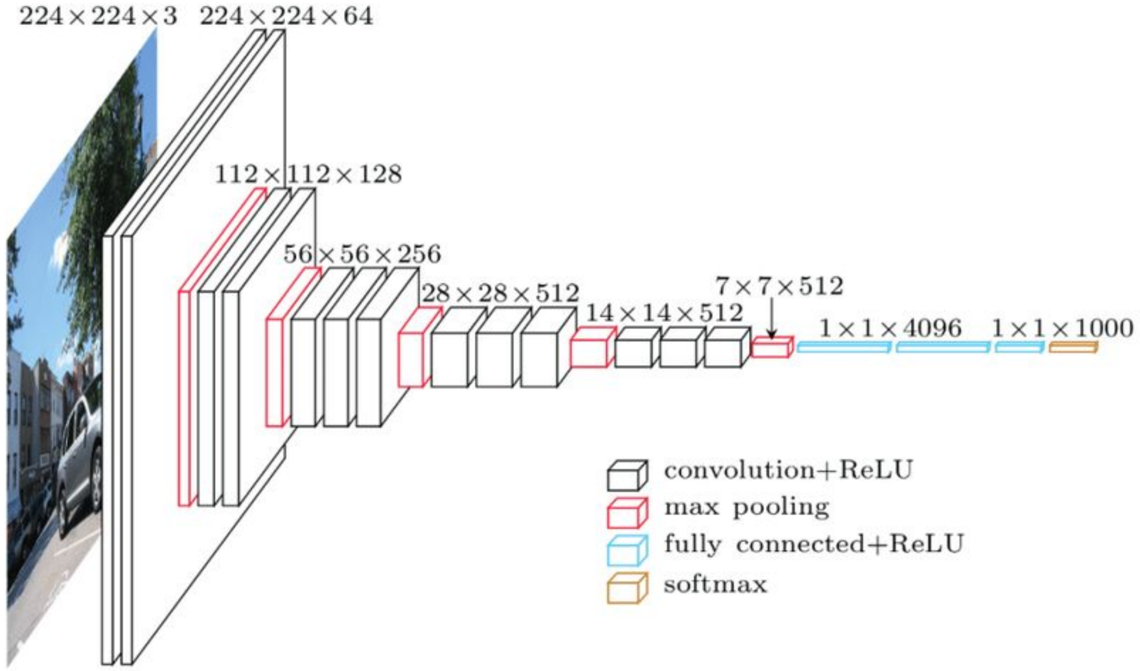
3D color space that enables accurate measurement and comparison of all perceivable colors using three color values



- L (Lightness)
  - Represents the lightness of the color, ranging from 0 (black) to 100 (white).
- A
  - Represents the position between green and red. Negative values indicate green, while positive values indicate red.
- B
  - Represents the position between blue and yellow. Negative values indicate blue, while positive values indicate yellow.

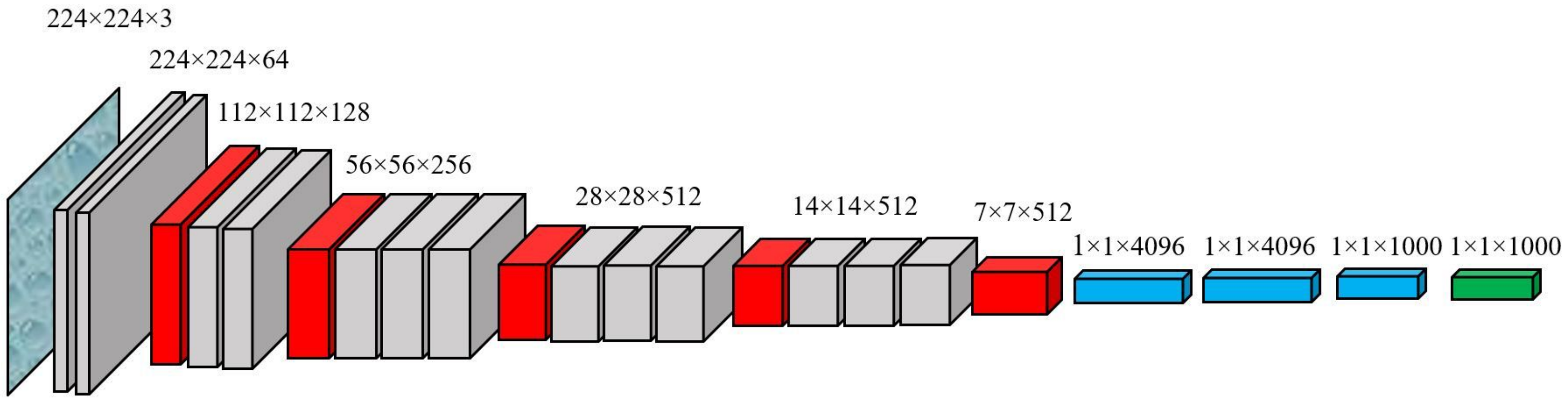


VGG-NET 19


224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax

VGG-NET 19 Feature Extraction

Taking a random image from the dataset, features are extracted by taking the output from one of the fully connected layers. This results in 4096 perceptual features, corresponding to the number of output neurons of the extraction layer.



Rewards

The reward measures the improvement of the raw image using two distinct losses: pixelwise loss and perceptual loss.

- Pixelwise Loss (MAE): Measures pixel-by-pixel color differences between the ground truth and improved image.
  - Reward: Difference in pixelwise losses between the improved and raw images.
- Perceptual Loss (MSE): Measures the distance between feature vectors (from VGG-19) of ground truth and enhanced image.
  - Reward: Difference in perceptual losses between the improved and raw images.
- Final Reward: Combines pixelwise and perceptual rewards using balancing coefficient α and β.
  - Formula: Final reward = α(pixelwise reward) + β(perceptual reward)

Rewards

The reward measures the improvement of the raw image using two distinct losses: pixelwise loss and perceptual loss.

Pixelwise Loss (Mean Absolute Error)

- Measures pixel-by-pixel color difference between ground truth and improved image.
- Formula: $L_c = \frac{1}{WH} \sum_{w,h} |Y_{w,h} - G(X(t))_{w,h}|$
- **Pixelwise Reward**: Difference in pixelwise losses between improved and raw images.
  - Formula: $R_c = -[L_c(t) - L_c(t-1)]$

Perceptual Loss (Mean Squared Error)

- Measures distance between feature vectors (from VGG-19) of ground truth and improved image.
- Formula: $L_p(t) = \frac{1}{F} \sum_{w,h} (\Phi_{VGG}(Y)_{w,h} - \Phi_{VGG}(G(X(t)))_{w,h})^2$
- **Perception Reward**: Difference in perceptual losses between improved and raw images.
  - Formula: $R_p(t) = -[L_p(t) - L_p(t-1)]$

Total Reward

- Combines pixelwise and perceptual rewards using balancing coefficients α and β.
  - Formula: $R = \alpha R_c + \beta R_p$
- Recommended values: $\alpha = 1$, $\beta = 0.05$

Environment

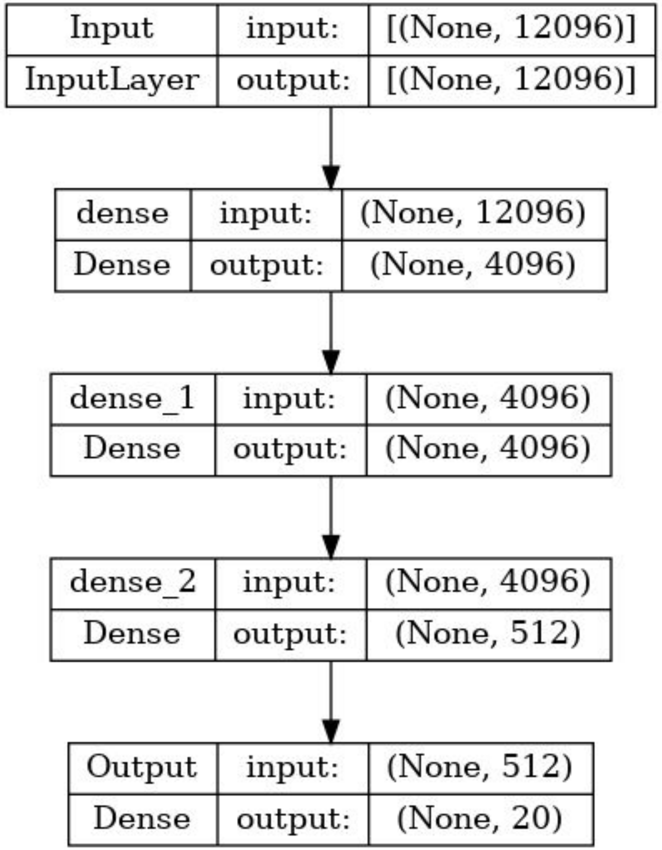
- The environment consists of underwater images, including raw/enhanced images and their reference (ground truth) images.
- Implemented on OpenAI Gym, focusing on two main functions:
  - step: Applies an action, returning the new state and reward.
  - reset: Resets the environment to its initial state for a new episode.
- Key Features
  - Tracks image features to reduce training time by avoiding repeated feature extraction.
  - Extracts new features only after applying an action.
- Functions of the Environment Class
  - step: Performs an action, returns new state (features of new image) and reward.
  - action: Index of the action to be performed.
  - reset: Sets a new raw image and ground truth, returns initial state.
  - raw_path and ref_path: Enable deterministic reset, useful for validation and testing.
  - render: Displays the raw, enhanced, and reference images.

Agent

- The DQN is implemented as an agent.
- The DQN is composed of four fully connected layers: the input is represented by the current state (the image features) and it returns the Q-value for each possible action that the agent could take. There will be as many output neurons as there are actions.

Agent

- Parameters
  - Optimization algorithm: ADAM
  - Loss function: MSE
- The weights of the target model are periodically replaced with those of the evaluation model.



Input | input: | [(None, 12096)]
InputLayer | output: | [(None, 12096)]
dense | input: | (None, 12096)
Dense | output: | (None, 4096)
dense_1 | input: | (None, 4096)
Dense | output: | (None, 4096)
dense_2 | input: | (None, 4096)
Dense | output: | (None, 512)
Output | input: | (None, 512)
Dense | output: | (None, 20)

Experience Replay

- Manages correlation problems by preventing the network from forgetting infrequently seen transitions.
- **Training Process**:
  - Save transitions $(s_t, a_t, r_t, s_{t+1})(s_t, a_t, r_t, s_{t+1})$ from the evaluation DQN into memory.
  - Randomly sample transitions from memory for the target DQN to calculate target values.

Memory Class Functions

- __init__: Initializes memory by constructing lists.
- store: Adds a transition to memory.
  - **Parameters**: state, action, reward, next_state, done.
  - done indicates if the transition leads to a final state.
- sample: Randomly samples a number of transitions from memory.
  - **Parameter**: batch_size - number of transitions to sample.
- reset: Deletes all transitions saved in memory.

Training

- Given a raw image, its features are extracted and these correspond to the initial state $\mathbf{s}_0$
- Using the evaluation DQN, the Q-value for the current state relative to each action is calculated
- The action with the highest Q-value is applied (ϵ-greedy), resulting in an “enhanced” image
- If the Q-values obtained for the “enhanced” image are all negative (or the maximum number of steps allowed per episode is reached), the episode ends; otherwise, the iteration continues starting from point 1.

Validation

Each training session is followed by validation on a separate dataset section. Performances are recorded and compared using three metrics: MSE, PSNR, and SSIM.

1. **Mean Squared Error (MSE)**:
   - Measures the average squared difference between pixels of two images.
   - Lower MSE indicates higher similarity.
   - Formula:

$$MSE = \frac{1}{WH} \sum_{w,h} [I_1(w,h) - I_2(w,h)]^2$$

   - Calculated using `mean_squared_error` from Skimage.

2. **Peak Signal-to-Noise Ratio (PSNR)**:
   - Indicates the quality of a reconstructed image compared to the original.
   - Higher PSNR indicates better quality.
   - Formula:

$$PSNR = 10 \log_{10} \left(\frac{R^2}{MSE} \right)$$

   - Calculated using `peak_signal_noise_ratio` from Skimage.

3. **Structural Similarity (SSIM)**:
   - Evaluates similarity between two images using absolute error.
   - Higher SSIM indicates higher similarity.
   - Calculated using `structural_similarity` from Skimage.

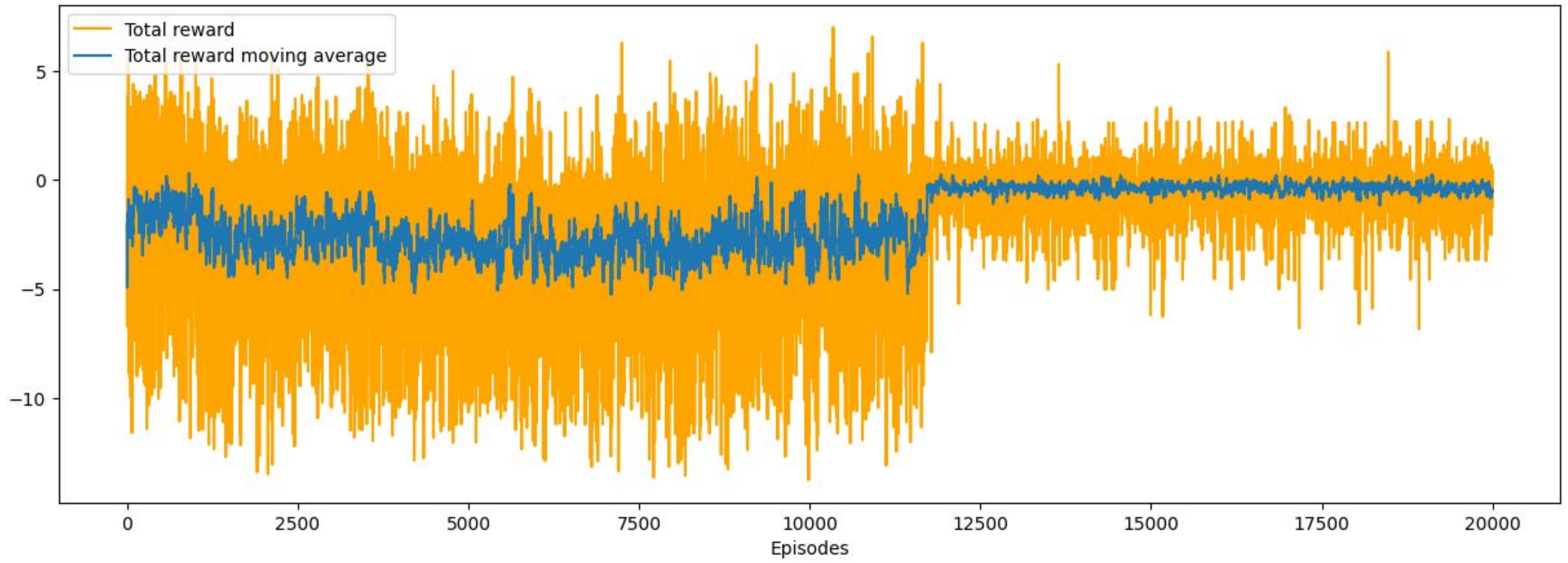
Results

The following are the parameters used for training

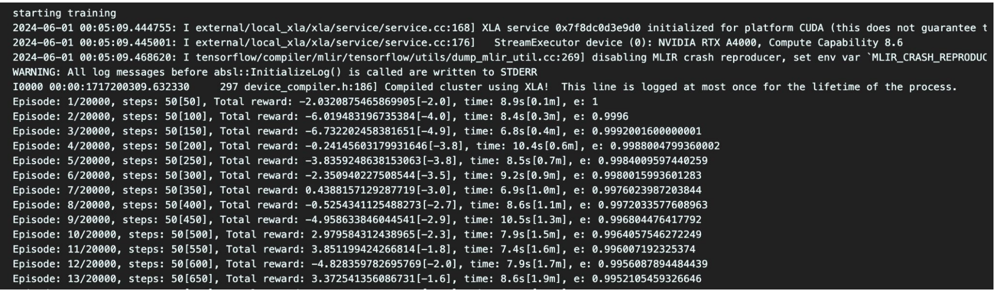
Parameter	Value
learning rate	1e-5
minimum learning rate	1e-8
lr decay step	0.96
discount rate	0.99
batch size	32
initial epsilon	1
minimum epsilon	0.1
epsilon decay step	0.99
episodes	10,000
max steps per episode	50
alpha	1
beta	0.05

Episode Reward

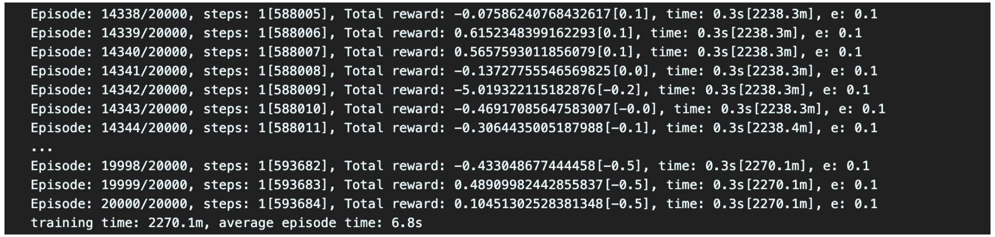
The following graph shows the reward trend during training with the above-listed hyperparameters. An increasing trend indicates that the agent is learning to effectively improve raw images.



Episode Reward



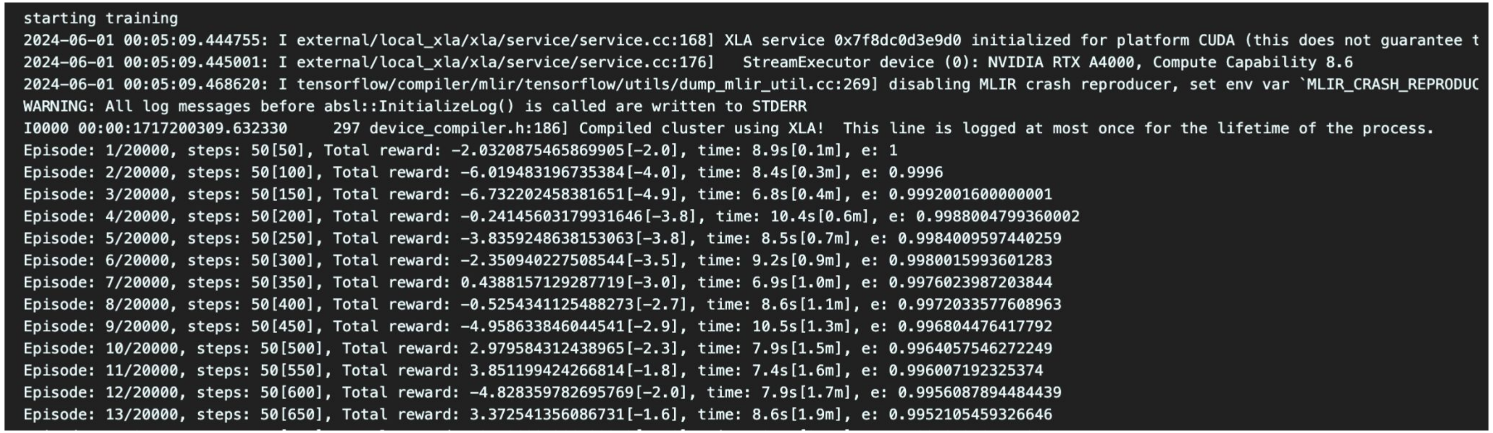
```
starting training
2024-06-01 00:05:09.444755: I external/local_xla/xla/service/service.cc:168] XLA service 0x7f8dc0d3e9d0 initialized for platform CUDA (this does not guarantee t
2024-06-01 00:05:09.445001: I external/local_xla/xla/service/service.cc:176]   StreamExecutor device (0): NVIDIA RTX A4000, Compute Capability 8.6
2024-06-01 00:05:09.468620: I tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:269] disabling MLIR crash reproducer, set env var `MLIR_CRASH_REPRODUC
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1717200309.632330     297 device_compiler.h:186] Compiled cluster using XLA!  This line is logged at most once for the lifetime of the process.
Episode: 1/20000, steps: 50[50], Total reward: -2.0320875465869905[-2.0], time: 8.9s[0.1m], e: 1
Episode: 2/20000, steps: 50[100], Total reward: -6.019483196735384[-4.0], time: 8.4s[0.3m], e: 0.9996
Episode: 3/20000, steps: 50[150], Total reward: -6.732202458381651[-4.9], time: 6.8s[0.4m], e: 0.9992001600000001
Episode: 4/20000, steps: 50[200], Total reward: -0.24145603179931646[-3.8], time: 10.4s[0.6m], e: 0.9988004799360002
Episode: 5/20000, steps: 50[250], Total reward: -3.8359248638153063[-3.8], time: 8.5s[0.7m], e: 0.9984009597440259
Episode: 6/20000, steps: 50[300], Total reward: -2.350940227508544[-3.5], time: 9.2s[0.9m], e: 0.9980015993601283
Episode: 7/20000, steps: 50[350], Total reward: 0.4388157129287719[-3.0], time: 6.9s[1.0m], e: 0.9976023987203844
Episode: 8/20000, steps: 50[400], Total reward: -0.5254341125488273[-2.7], time: 8.6s[1.1m], e: 0.9972033577608963
Episode: 9/20000, steps: 50[450], Total reward: -4.958633846044541[-2.9], time: 10.5s[1.3m], e: 0.996804476417792
Episode: 10/20000, steps: 50[500], Total reward: 2.979584312438965[-2.3], time: 7.9s[1.5m], e: 0.9964057546272249
Episode: 11/20000, steps: 50[550], Total reward: 3.851199424266814[-1.8], time: 7.4s[1.6m], e: 0.996007192325374
Episode: 12/20000, steps: 50[600], Total reward: -4.828359782695769[-2.0], time: 7.9s[1.7m], e: 0.9956087894484439
Episode: 13/20000, steps: 50[650], Total reward: 3.372541356086731[-1.6], time: 8.6s[1.9m], e: 0.9952105459326646
```



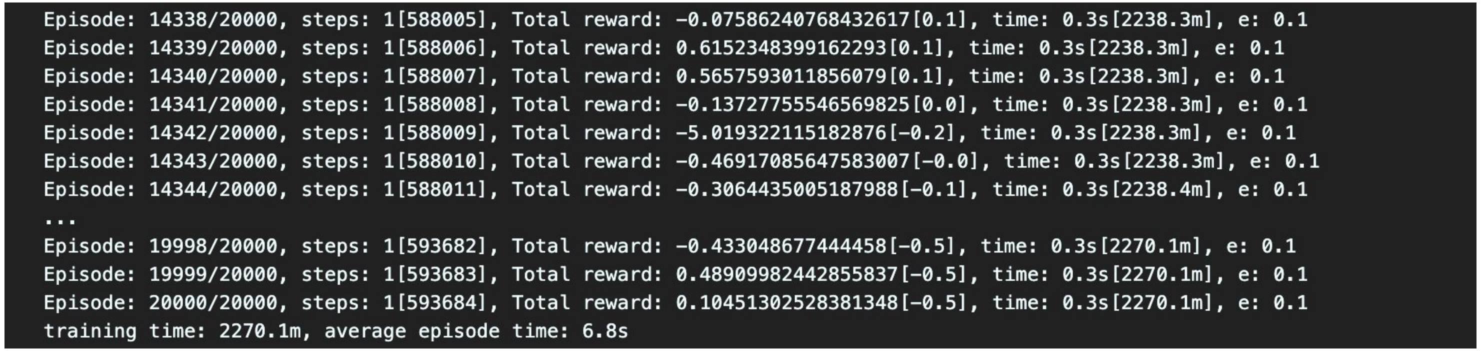
```
  Episode: 14338/20000, steps: 1[588005], Total reward: -0.07586240768432617[0.1], time: 0.3s[2238.3m], e: 0.1
  Episode: 14339/20000, steps: 1[588006], Total reward: 0.6152348399162293[0.1], time: 0.3s[2238.3m], e: 0.1
  Episode: 14340/20000, steps: 1[588007], Total reward: 0.5657593011856079[0.1], time: 0.3s[2238.3m], e: 0.1
  Episode: 14341/20000, steps: 1[588008], Total reward: -0.13727755546569825[0.0], time: 0.3s[2238.3m], e: 0.1
  Episode: 14342/20000, steps: 1[588009], Total reward: -5.019322115182876[-0.2], time: 0.3s[2238.3m], e: 0.1
  Episode: 14343/20000, steps: 1[588010], Total reward: -0.46917085647583007[-0.0], time: 0.3s[2238.3m], e: 0.1
  Episode: 14344/20000, steps: 1[588011], Total reward: -0.3064435005187988[-0.1], time: 0.3s[2238.4m], e: 0.1
  ...
  Episode: 19998/20000, steps: 1[593682], Total reward: -0.433048677444458[-0.5], time: 0.3s[2270.1m], e: 0.1
  Episode: 19999/20000, steps: 1[593683], Total reward: 0.48909982442855837[-0.5], time: 0.3s[2270.1m], e: 0.1
  Episode: 20000/20000, steps: 1[593684], Total reward: 0.10451302528381348[-0.5], time: 0.3s[2270.1m], e: 0.1
  training time: 2270.1m, average episode time: 6.8s
```

Episode	Steps	Episode Reward
1	50	-2.03209
1000	50000	-2.13461
2000	100000	0.278477
3000	150000	-2.60664
4000	200000	-0.675033
10000	500000	-0.785109
11000	550000	0.523002
12000	585665	-0.098416
13000	586667	-1.68988
14000	587667	0.158719
16000	589671	-0.0532499
17000	590678	-0.0574592
18000	591682	-0.717265
19000	592682	-0.461128
20000	593684	0.104513

Total Reward

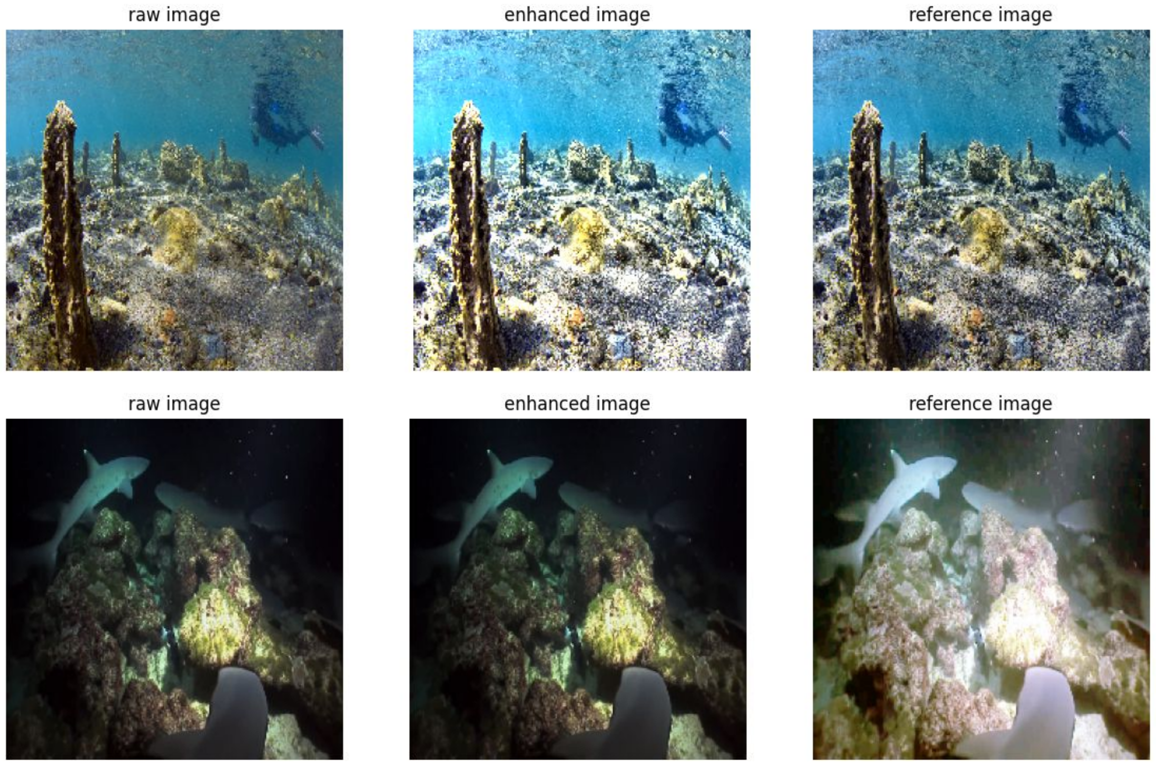


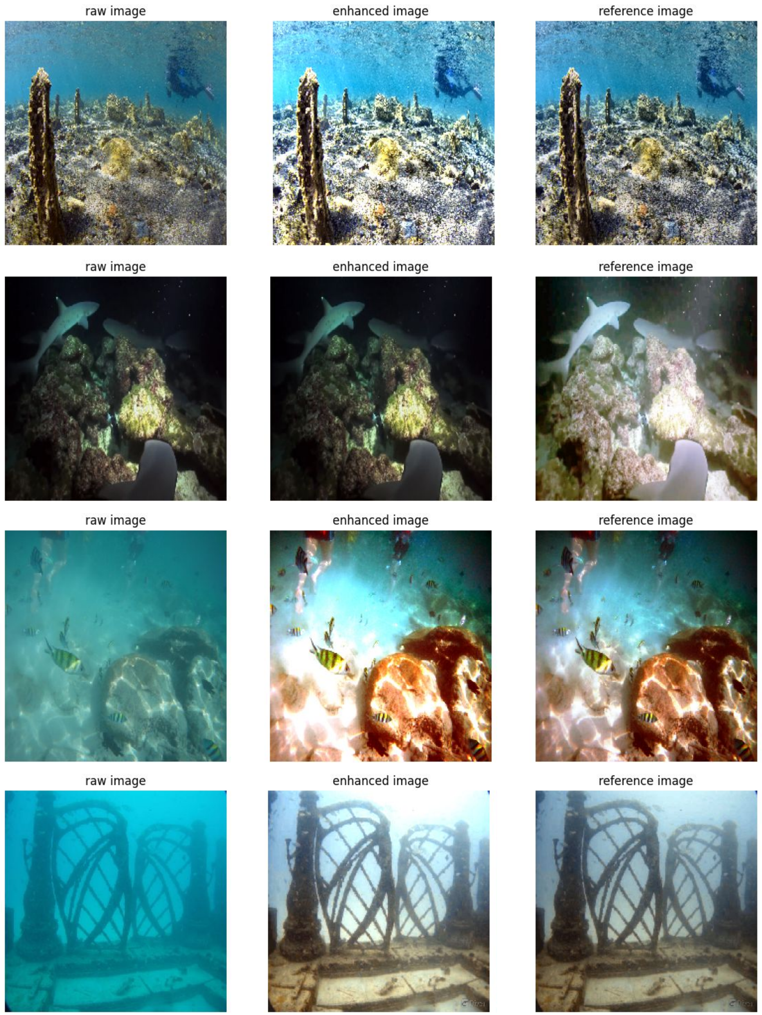
```
starting training
2024-06-01 00:05:09.444755: I external/local_xla/xla/service/service.cc:168] XLA service 0x7f8dc0d3e9d0 initialized for platform CUDA (this does not guarantee t
2024-06-01 00:05:09.445001: I external/local_xla/xla/service/service.cc:176]   StreamExecutor device (0): NVIDIA RTX A4000, Compute Capability 8.6
2024-06-01 00:05:09.468620: I tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:269] disabling MLIR crash reproducer, set env var `MLIR_CRASH_REPRODUC
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1717200309.632330     297 device_compiler.h:186] Compiled cluster using XLA!  This line is logged at most once for the lifetime of the process.
Episode: 1/20000, steps: 50[50], Total reward: -2.0320875465869905[-2.0], time: 8.9s[0.1m], e: 1
Episode: 2/20000, steps: 50[100], Total reward: -6.019483196735384[-4.0], time: 8.4s[0.3m], e: 0.9996
Episode: 3/20000, steps: 50[150], Total reward: -6.732202458381651[-4.9], time: 6.8s[0.4m], e: 0.9992001600000001
Episode: 4/20000, steps: 50[200], Total reward: -0.24145603179931646[-3.8], time: 10.4s[0.6m], e: 0.9988004799360002
Episode: 5/20000, steps: 50[250], Total reward: -3.8359248638153063[-3.8], time: 8.5s[0.7m], e: 0.9984009597440259
Episode: 6/20000, steps: 50[300], Total reward: -2.350940227508544[-3.5], time: 9.2s[0.9m], e: 0.9980015993601283
Episode: 7/20000, steps: 50[350], Total reward: 0.4388157129287719[-3.0], time: 6.9s[1.0m], e: 0.9976023987203844
Episode: 8/20000, steps: 50[400], Total reward: -0.5254341125488273[-2.7], time: 8.6s[1.1m], e: 0.9972033577608963
Episode: 9/20000, steps: 50[450], Total reward: -4.958633846044541[-2.9], time: 10.5s[1.3m], e: 0.996804476417792
Episode: 10/20000, steps: 50[500], Total reward: 2.979584312438965[-2.3], time: 7.9s[1.5m], e: 0.9964057546272249
Episode: 11/20000, steps: 50[550], Total reward: 3.851199424266814[-1.8], time: 7.4s[1.6m], e: 0.996007192325374
Episode: 12/20000, steps: 50[600], Total reward: -4.828359782695769[-2.0], time: 7.9s[1.7m], e: 0.9956087894484439
Episode: 13/20000, steps: 50[650], Total reward: 3.372541356086731[-1.6], time: 8.6s[1.9m], e: 0.9952105459326646
```



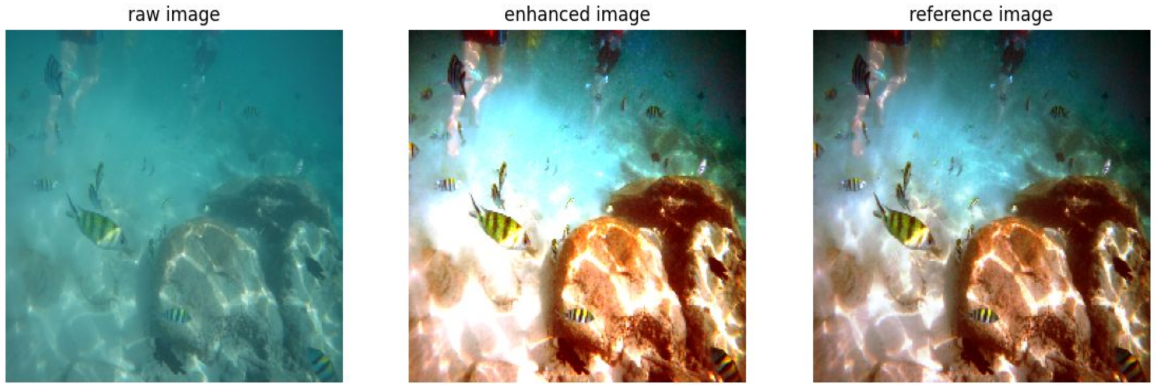
```
Episode: 14338/20000, steps: 1[588005], Total reward: -0.07586240768432617[0.1], time: 0.3s[2238.3m], e: 0.1
Episode: 14339/20000, steps: 1[588006], Total reward: 0.6152348399162293[0.1], time: 0.3s[2238.3m], e: 0.1
Episode: 14340/20000, steps: 1[588007], Total reward: 0.5657593011856079[0.1], time: 0.3s[2238.3m], e: 0.1
Episode: 14341/20000, steps: 1[588008], Total reward: -0.13727755546569825[0.0], time: 0.3s[2238.3m], e: 0.1
Episode: 14342/20000, steps: 1[588009], Total reward: -5.019322115182876[-0.2], time: 0.3s[2238.3m], e: 0.1
Episode: 14343/20000, steps: 1[588010], Total reward: -0.46917085647583007[-0.0], time: 0.3s[2238.3m], e: 0.1
Episode: 14344/20000, steps: 1[588011], Total reward: -0.3064435005187988[-0.1], time: 0.3s[2238.4m], e: 0.1
...
Episode: 19998/20000, steps: 1[593682], Total reward: -0.433048677444458[-0.5], time: 0.3s[2270.1m], e: 0.1
Episode: 19999/20000, steps: 1[593683], Total reward: 0.48909982442855837[-0.5], time: 0.3s[2270.1m], e: 0.1
Episode: 20000/20000, steps: 1[593684], Total reward: 0.10451302528381348[-0.5], time: 0.3s[2270.1m], e: 0.1
training time: 2270.1m, average episode time: 6.8s
```

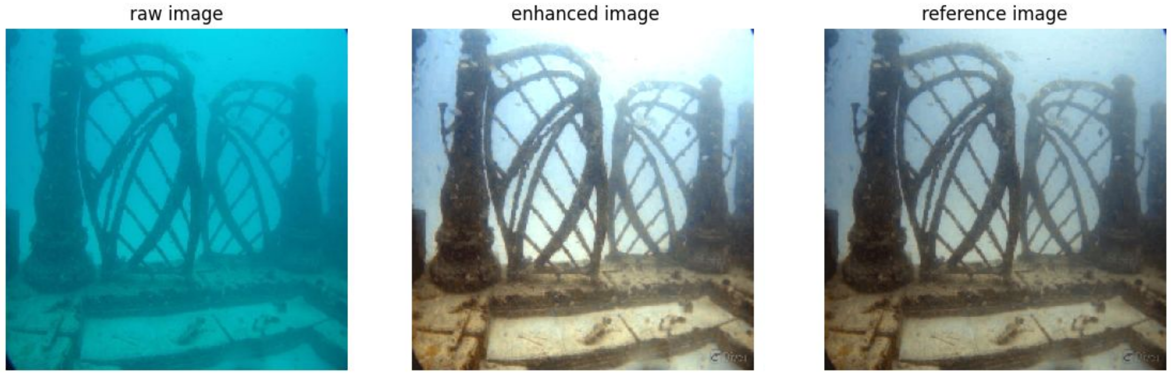
Qualitative Analysis




raw image
enhanced image
reference image
raw image
enhanced image
reference image
raw image
enhanced image
reference image
raw image
enhanced image
reference image

Qualitative Analysis





Quantitative Analysis

- Quantitative analysis is conducted using quantifiable values for evaluation. Specifically, the metrics used for the analysis are MSE, PSNR, and SSIM.

Metric	Value
Average MSE	1858.0198193370575
Average PSNR	16.992879298748356
Average SSNR	0.7424274301354952

Conclusion and Next Steps

- The model has demonstrated significant success in enhancing the quality of underwater images.
  - The rewards during training exhibited an increasing trend, which indicates that the agent is effectively learning to improve raw images over time.
- Next Steps
  - Comparison with existing solutions
    - Benchmark the model's performance against other state-of-the-art image enhancement techniques to identify its relative strengths and weaknesses.
  - Parameter exploration
    - Investigate the impact of different hyperparameters on the model's performance to optimize its effectiveness further.
  - Testing on diverse image types
    - Expand the model's testing framework to include various image types beyond underwater images to evaluate its generalization capabilities and potential for broader application.

Thank you

Marc Reyes
AI 322 Reinforcement Learning Mini Project