Reproduction and Extended Evaluation of Zero-Reference Deep Curve Estimation for Low-Light Image Enhancement

COMP478 - Image Processing

Shuo Feng
Student ID: 40224138
s_feng20@live.concordia.ca

November 28, 2025

I. Introduction

A. Problem Statement

Capturing high-quality images in low-light conditions remains a persistent challenge in digital photography and computer vision. Images acquired in suboptimal lighting suffer from low visibility, suppressed contrast, and significant noise artifacts. These degradations not only compromise the aesthetic quality of personal photography but also impede the performance of high-level vision tasks such as object recognition and face detection.

Existing enhancement solutions typically face two distinct limitations. Traditional histogram-based methods (e.g., HE) often introduce unrealistic color shifts or over-enhancement artifacts. Conversely, modern deep learning approaches (CNNs or GANs) generally yield superior visual results but rely heavily on supervised learning. This dependency requires extensive datasets of paired low/normal-light images, which are computationally expensive to process and difficult to capture in real-world scenarios without synthetic fabrication.

B. Motivation

This project focuses on the "Zero-Reference Deep Curve Estimation" (Zero-DCE) framework [1] because it addresses the data dependency bottleneck. Unlike conventional deep learning models, Zero-DCE eliminates the need for paired reference images. Instead, it formulates enhancement as a curve estimation task guided by non-reference loss functions, which essentially learning a generalized understanding of "good" exposure implicitly.

Furthermore, Zero-DCE diverges from standard "black box" image-to-image translation networks. It predicts a set of pixel-wise light-enhancement curves, analogous to curve adjustment layers in photo editing software. This architecture renders the model computationally lightweight and highly interpretable, making it suitable for real-time inference on resource-constrained platforms.

C. Project Objectives

The primary objective of this report is to reproduce the inference capabilities of the Zero-DCE framework and extend its evaluation to novel datasets and environments. The specific objectives are:

1) **Validation on Original Benchmarks:** Reproduce the original Python inference pipeline and validate performance using the paper's original benchmark datasets [1], specifically **LIME** and **DICM**, to establish a baseline for enhancement quality.
2) **MATLAB Interoperability Implementation:** Construct a hybrid inference pipeline to demonstrate cross-platform interoperability. While preserving the original PyTorch training environment, the trained model weights are exported to ONNX format and imported into a custom MATLAB environment. This implementation removes the strict dependency on CUDA, enabling deployment on non-NVIDIA hardware (such as MacOS) with negligible performance overhead.
3) **Extended Evaluation on Alternative Datasets:** Evaluate the model's generalization capability on datasets that differ from the likely training distribution (curated or synthetic). This includes the LOL training split and a Custom Real-World collection. Assessment uses non-reference metrics (NIQE, PIQE) to quantify whether enhancement introduces artifacts or distortion. Results are expected to reveal limitations and trade-offs in the Zero-DCE framework when applied beyond curated scenarios.

II. Theoretical Framework: Zero-DCE

A. Light-Enhancement Curve (LE-Curve)

The fundamental innovation of the Zero-DCE framework [1] is the reformulation of low-light enhancement as a curve estimation problem. Inspired by photometric adjustment curves, the method employs a differentiable quadratic curve to map low-light input pixels to an enhanced dynamic range. The design adheres to three constraints: differentiability for backpropagation, monotonicity to preserve neighbor contrast, and computational simplicity.

The enhancement function is mathematically formulated as:

$$LE(I(x);\alpha) = I(x) + \alpha I(x)(1 - I(x)) \tag{1}$$

where x denotes the pixel coordinates and α is a trainable parameter regulating the exposure magnitude. To approximate complex, higher-order adjustments, this curve is applied iteratively. The implementation utilizes 8 iterations, where the output of the previous iteration serves as the input for the next. Crucially, α is not a global scalar but a pixel-wise parameter map, allowing the network to apply spatially variant enhancement tailored to local illumination conditions.

B. DCE-Net Architecture

The Deep Curve Estimation Network (DCE-Net) serves as the estimator for the pixel-wise curve parameters. The architecture is explicitly designed for efficiency, comprising a shallow Convolutional Neural Network (CNN) with seven convolutional layers and symmetrical skip connections.

To maintain high-resolution spatial details, the network excludes down-sampling and batch normalization layers. Each layer utilizes 3×3 kernels followed by ReLU activation, with the exception of the final layer. The output layer employs a Tanh activation function to generate 24 parameter maps (corresponding to 8 iterations across 3 RGB channels). This lightweight design ensures rapid inference speeds suitable for real-time applications.

C. Non-Reference Loss Functions

Zero-DCE achieves zero-reference learning through a combination of four differentiable non-reference loss functions that implicitly guide the optimization process.

The **Spatial Consistency Loss** enforces spatial coherence by preserving the gradient differences between the input and enhanced images. The **Exposure Control Loss** mitigates exposure errors by minimizing the distance between local average intensity and a target gray level. To maintain chromatic validity, the **Color Constancy Loss** corrects color deviations based on the Gray-World assumption. Finally, the **Illumination Smoothness Loss** constrains the gradients of the predicted parameter maps, ensuring that the estimated curves vary smoothly across the image to prevent artifact generation.

III. Methodology and MATLAB Implementation

A. Hybrid Inference Strategy

To demonstrate cross-platform interoperability and validate the architectural independence of the Zero-DCE framework, this project implements a hybrid inference pipeline. The training phase remains in the original PyTorch environment to leverage the pre-calculated weights. However, the inference logic is fully migrated to the MATLAB environment. The Open Neural Network Exchange (ONNX) format serves as the bridge between these two ecosystems. This approach verifies that the lightweight nature of DCE-Net is not dependent on specific Python libraries (like CUDA) and can be deployed in standard signal processing environments.

B. ONNX Export and Compatibility Optimization

Migrating modern deep learning models from PyTorch to MATLAB presents specific compatibility challenges regarding Intermediate Representation (IR) versions and operator support. Three distinct optimization steps were required to generate a valid, importable model for the MATLAB Deep Learning Toolbox.

First, standard PyTorch export routines often produce models with cutting-edge IR versions that exceed the support window of MATLAB's importer. To resolve this, the export script was constrained to utilize **Opset 13** and strictly enforce **IR Version 9**. This ensures the graph definitions remain within the compatibility matrix of the current MATLAB release.

Second, by default, PyTorch exports larger models or specific configurations by separating the topology definition (`.onnx`) from the binary weights (`.data`). The MATLAB importer requires a monolithic file structure where topology and weights are encapsulated together. The export routine was modified to force the generation of a single, self-contained `.onnx` binary.

Third, and most critically, the original DCE-Net architecture utilizes a `Split` operator within the network graph to separate the final 24-channel output into 8 distinct iteration maps (3 channels each) during the forward pass. MATLAB's ONNX importer exhibits limited support for this specific implementation of the split operation, leading to graph construction errors. To circumvent this, the computational graph was pruned: the `Split` operator and the subsequent iterative enhancement loop were removed from the ONNX definition. The exported model serves strictly as a parameter estimator, outputting a raw tensor of shape $(N, 24, H, W)$, allowing the iterative application of curves to be reconstructed natively using MATLAB's efficient matrix slicing and element-wise operations.

C. MATLAB Logic Reconstruction

Following the successful export, the raw estimator network is imported into MATLAB as a `DAGNetwork` object. Because the iterative logic was pruned from the graph to resolve the operator incompatibility, the enhancement mechanism was reconstructed natively using MATLAB's matrix operations.

The reconstruction involves a vectorized implementation of the curve estimation formula. The network's 24-channel output is mathematically sliced into 8 distinct parameter maps ($\mathcal{A}_1$ through $\mathcal{A}_8$), corresponding to the 3 RGB channels for each iteration. The iterative enhancement, originally defined in Python as a loop, is implemented in MATLAB as a sequential matrix operation following the equation:

$$\begin{aligned} LE_n(x) = LE_{n-1}(x) \\ +\mathcal{A}_n(x)LE_{n-1}(x)(1 - LE_{n-1}(x)) \end{aligned} \tag{2}$$

This manual reconstruction not only circumvents the importer limitations but also leverages MATLAB's optimized memory management for large matrix manipulations, ensuring efficient processing of high-resolution images.

IV. Alternative Data Sets

A. Dataset Selection Criteria

To rigorously evaluate the generalization capability of the reproduced Zero-DCE model, the experimental design extends beyond the original training data used in the paper. Two alternative datasets were selected to represent distinct testing paradigms: a standardized academic benchmark and an unconstrained real-world collection. The primary selection criterion was realistic,choosing datasets with lighting distributions and noise characteristics that differ from the original training set to verify if the "Zero-Reference" learning strategy truly yields robust, universal features.

B. Description of Selected Datasets

The first alternative dataset is the **LOL (LOw-Light) Dataset** [2]. Originally created for supervised low-light enhancement research, it contains pairs of low-light and normal-light images captured in diverse indoor and outdoor scenes. For this evaluation, the dataset is adapted to a single-image inference protocol: the 485 low-light images from the dataset's training split are utilized as an inference test bed, while the corresponding normal-light ground truth pairs are excluded.

The second dataset is a **Custom Real-World Collection** ("MYOWN"). This set comprises spontaneously acquired images capturing high-contrast night scenes, backlit subjects, and unevenly illuminated indoor environments. Unlike curated academic datasets, these images were not filtered for quality, meaning they contain unpredictable artifacts such as ISO grain, compression blocking, and motion blur.

C. Justification for Selection

The proposal to use these specific datasets is grounded in the necessity of "wild" testing. The **LOL** dataset [2] serves as a bridge; it is a known quantity in the literature with established difficulty, providing a semi-controlled environment to benchmark the MATLAB implementation against known standards. Conversely, the **Custom Real-World Collection** acts as a stress test for unconstrained deployment. By processing raw, unfiltered images from a consumer camera, this dataset exposes whether the model's smoothness constraints (specifically the L_{tv_A} loss) hold up against real-world sensor noise, or if the model creates artifacts when faced with unknown dynamic ranges.

V. Experimental Results

A. Experimental Setup

All inference experiments were conducted using the hybrid pipeline described in Section III. The pre-trained weights were loaded into the MATLAB environment via the optimized ONNX interface. No further post-processing (such as denoising or color grading) was applied to the output to ensure the results solely reflect the raw performance of the estimated curves.

B. Visual Analysis

Visual inspection reveals a mixed performance profile across datasets.While the Zero-DCE framework successfully brightens extremely dark regions, the enhancement introduces systematic artifacts that vary with input characteristics.

On the LOL dataset, the model exhibits aggressive over-enhancement in several scenarios. Bright light sources (windows, lamps) become bloomed and washed out, while the shadow recovery often appears artificially uniform, lacking natural texture gradation. Some images display visible halo-like artifacts at high-contrast boundaries, contrary to the paper's claims of artifact-free enhancement.

On the Custom Real-World Collection, performance degrades less significantly. However, the enhancement process still amplifies visible sensor noise, particularly in previously dark regions which now exhibit obvious grain amplification. Additionally, textureless dark areas display the expected gray cast, indicating the model's reliance on the Gray-World assumption breaks down on non-standard illumination patterns.

C. Quantitative Evaluation

a) *Metric Selection and Interpretation:* Non-reference metrics NIQE [3] and PIQE [4] were employed to assess enhancement quality without ground truth. Critically, both metrics are **inversely correlated to perceptual quality**: lower scores indicate better naturalness (NIQE) and fewer distortions (PIQE). Higher scores indicate deviation from natural image statistics and increased perceptual distortion.

b) *Results Summary: Contrary to the visual inspection expectations*, PIQE scores **increased** across all datasets post-enhancement (Table 1), indicating that enhanced images exhibit greater deviation from natural statistics and higher perceptual distortion compared to raw inputs.

TABLE I
Quantitative metric comparison

Dataset	Metric	Input (Mean)	Enhanced (Mean)
LOL	NIQE	6.03	8.54
LOL	PIQE	18.87	37.75
MYOWN	NIQE	4.73	3.83
MYOWN	PIQE	63.73	67.26

c) *Interpretation and Analysis:* The metric analysis reveals a divergence between naturalness (NIQE) and distortion (PIQE):

1) **Mixed NIQE Performance**: On the LOL dataset, NIQE worsened (increased), confirming visual observations of artificial blooming and washing out. However, on the Custom Real-World dataset, NIQE actually **improved** (decreased from 4.73 to 3.83), suggesting that despite the noise, the model successfully moved the global image statistics closer to a "natural" distribution by correcting the extreme darkness.
2) **Consistent PIQE Degradation**: Unlike NIQE, PIQE scores increased (worsened) across **all** datasets. This indicates that while the image statistics might look more "natural" (better NIQE), the local structural integrity has been compromised by blocking and noise artifacts–a finding that aligns perfectly with the visual observation of grain amplification in the custom dataset. However, the PIQE is already high in the raw MYOWN images (63.73), indicating that the input images are already heavily distorted, and enhancement only exacerbates this.

VI. Discussion and Main Findings

A. Metric-Visual Discrepancy

The quantitative results contradict the qualitative visual assessment. While brightening is achieved, the metrics quantify that enhancement introduces more distortion than exists in the original. This reveals a fundamental trade-off in Zero-DCE when applied to non-standard data: illumination recovery is coupled with artifact generation.

B. Contextual Comparison

Published Zero-DCE results [1] on benchmark datasets (LIME, DICM) report NIQE minimal degradation. The larger degradation observed here on LOL suggests that Zero-DCE's non-reference loss functions (particularly the smoothness constraint) may not be optimized for real-world camera noise and extreme dynamic ranges, confirming the model's sensitivity to training data distribution.

VII. Conclusion

This project reproduced the Zero-DCE framework [1] within a cross-platform MATLAB environment, validating its architectural efficiency and the viability of zero-reference learning for exposure correction. While the model demonstrates a robust ability to recover visibility in extremely dark scenarios without paired training data, the extended evaluation highlights a significant generalization gap. Therefore, practical deployment in "wild" environments would benefit from coupling this lightweight estimator with a dedicated denoising post-processing stage.

VIII. Appendix: MATLAB Implementation

A. Repository and Dependencies

Repository: `https://github.com/vonhyou/comp478-zero-dce-reproduce`

MATLAB Requirements: R2024a or later with Deep Learning Toolbox, Image Processing Toolbox, and ONNX Converter. The repository contains three core scripts (`inference_single.m`, `inference_batch.m`, `apply_zerodce.m`), the ONNX model (`models/ZeroDCE.onnx`), and test datasets in `images/input/`.

B. Core Implementation

a) *Enhancement Function:* The `apply_zerodce.m` function implements the iterative curve estimation formula from Section II:

```
function x = apply_zerodce(x, p_map, iterations)
  for i = 1:iterations
    istart = (i-1)*3 + 1;
    iend = i*3;
    alpha_n = p_map(:, :, istart:iend, :);

    x = x + alpha_n .* (x .^ 2 - x);
  end
end
```

Listing 1. Core enhancement function implementing iterative LE-Curve application

Key Features: Input `x` is a dlarray with 'SSCB' format. The `paramsMap` contains 24 channels (8 iterations × 3 RGB). Vectorized operations leverage MATLAB's matrix engine for efficient processing without explicit pixel loops.

b) *Single Image Pipeline:* The `inference_single.m` script demonstrates the complete workflow:

1) Load ONNX model: `net = importNetworkFromONNX('models/ZeroDCE.onnx')`
2) Preprocess image to dlarray with 'SSCB' format
3) Initialize network for dynamic image sizes: `net = initialize(net, dlInput)`
4) Predict curve parameters: `paramsMap = predict(net, dlInput)`
5) Apply enhancement: `enhanced = apply_zerodce(dlInput, paramsMap, 8)`
6) Calculate metrics: NIQE [3] and PIQE [4]
7) Display side-by-side comparison with quality scores

c) *Batch Processing:* The `inference_batch.m` script processes entire datasets automatically:

```
net = initialize(netBase, dlInput);
t_img = tic;
paramsMap = predict(net, dlInput);
x = apply_zerodce(dlInput, paramsMap, 8);
elapsed = toc(t_img);

niqe_score = niqe(enhancedImg_uint8);
piqe_score = piqe(enhancedImg_uint8);
```

Listing 2. Batch processing loop with timing and quality metrics

Features: Discovers datasets in `images/input/` subdirectories, processes all images, calculates average NIQE/PIQE metrics, profiles inference time, and generates `Report.md` with quantitative results.

C. ONNX Conversion Strategy

The `convert_onnx.py` script resolves three MATLAB compatibility issues:

1) **IR Version:** Constrains export to Opset 13 / IR Version 9
2) **File Structure:** Forces monolithic export (weights embedded)
3) **Split Operator:** Removes incompatible operator by exporting only the 24-channel parameter estimator; iterative logic reconstructed in MATLAB

This approach enables deployment on non-CUDA hardware (MacOS) with native MATLAB operations.

D. Usage Instructions

Quick Start:

```
git clone --recursive \
  https://github.com/vonhyou/comp478-zero-dce-
reproduce.git
```

Listing 3. Repository cloning command with submodules

Open MATLAB and run `inference_batch` for automatic dataset processing.

Custom Datasets: Create `images/input/NEW_DATASET/`, add images (`.jpg`, `.png`, `.bmp`), run `inference_batch`. Results saved to `images/output/`.

E. Additional Notes

The original typst file for this report is available upon request, please contact me via email. A PDF version can be generated using the Typst compiler, and a precompiled PDF is included in the repository.

References

[1] C. G. Guo *et al.*, "Zero-reference deep curve estimation for low-light image enhancement," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, June 2020, pp. 1780-1789.

[2] S. Rakshit, "LOL Dataset." [Online]. Available: https://www.kaggle.com/datasets/soumikrakshit/lol-dataset

[3] A. Mittal, R. Soundararajan, and A. C. Bovik, "Making a "Completely Blind" Image Quality Analyzer," *IEEE Signal Processing Letters*, vol. 20, no. 3, pp. 209-212, 2013, doi: 10.1109/LSP.2012.2227726.

[4] V. N., P. D., M. C. Bh., S. S. Channappayya, and S. S. Medasani, "Blind image quality evaluation using perception based features," in *Twenty First National Conference on Communications, NCC 2015, Mumbai, India, February 27 - March 1, 2015*, IEEE, 2015, pp. 1-6. doi: 10.1109/NCC.2015.7084843.