Zero-Reference Deep Curve Estimation for Low-Light Image Enhancement

Lois Breant, Andy Shan, Oscar Le Dauphin, Max Nagaishi, Matthew Banawa

1 Introduction

Project Overview

Problem Statement

- Low-light image enhancement is crucial for photography, surveillance, and computer vision tasks
- Traditional methods and deep learning approaches have significant limitations

Our Approach

- Implement and analyze Zero-DCE: a zero-reference deep learning method
- Compare with classical baselines (CLAHE, Gamma Correction)
- Critical evaluation of when deep learning is truly necessary

Key Questions

- Can a lightweight CNN (79k parameters) outperform traditional methods?
- What are the trade-offs between model complexity and performance?
- When is deep learning overkill for this task?

Deliverables

- Strong classical baselines
- Trained Zero-DCE model
- Quantitative and qualitative analysis
- Critical insights

2 Baseline & Bibliography (5 min)

Literature Review: Zero-DCE

The Zero-DCE Paper (Guo et al., CVPR 2020)

Key Innovation:

- **Zero-Reference:** No paired or unpaired data needed
- **LE-curve:** Light Enhancement curve for pixel mapping
- **DCE-Net:** Lightweight CNN (79k parameters)
- **Non-reference Losses:** Train without ground truth

Why this approach?

- Supervised methods need expensive paired data
- GANs are hard to train and unstable
- Zero-DCE is efficient and generalizable

Core Mechanism

Light-Enhancement curve:

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

Applied iteratively (n=8 times):

$$I_n(x) = LE\Big(I_{\{n-1\}}(x); \mathcal{A}_n(x)\Big)$$

Loss Functions:

- Spatial Consistency ($L_{\{spa\}}$)
- Exposure Control ($L_{\{exp\}}$)
- Color Constancy ($L_{\{col\}}$)
- Illumination Smoothness ($L_{\{tv\}}$)

Classical Baselines: Our Implementation

1. CLAHE (Contrast Limited Adaptive Histogram Equalization)

```
def apply_clahe(img):
    lab = cv2.cvtColor(img,
cv2.COLOR_BGR2LAB)
    l, a, b = cv2.split(lab)
    clahe = cv2.createCLAHE(
        clipLimit=3.0,
        tileGridSize=(8,8)
    )
    l_clahe = clahe.apply(l)
    lab_clahe = cv2.merge((l_clahe, a,
b))
    return cv2.cvtColor(lab_clahe,
cv2.COLOR_LAB2BGR)
```

2. Adaptive Gamma Correction

```
def apply_autogamma(img):
    hsv = cv2.cvtColor(img,
cv2.COLOR_BGR2HSV)
    v = hsv[:,:,2]
    mean_brightness = np.mean(v)
    target = 128

    gamma = log(target/255) /
            log(mean_brightness/255)

    table = [((i/255.0)**gamma)*255
             for i in range(256)]
    return cv2.LUT(img, table)
```

- Automatic gamma estimation

Classical Baselines: Our Implementation (ii)

- Local histogram equalization
- Contrast limiting prevents over-amplification
- Fast and parameter-free
- Global brightness adjustment
- Simple and interpretable

Classical Baselines: Our Implementation (iii)

Why Strong Baselines Matter:

- These methods are **fast** (1ms vs 10ms for Zero-DCE)
- No training required
- Often “good enough” for many use cases
- Establish a performance floor

Classical Baselines: Our Implementation (iv)

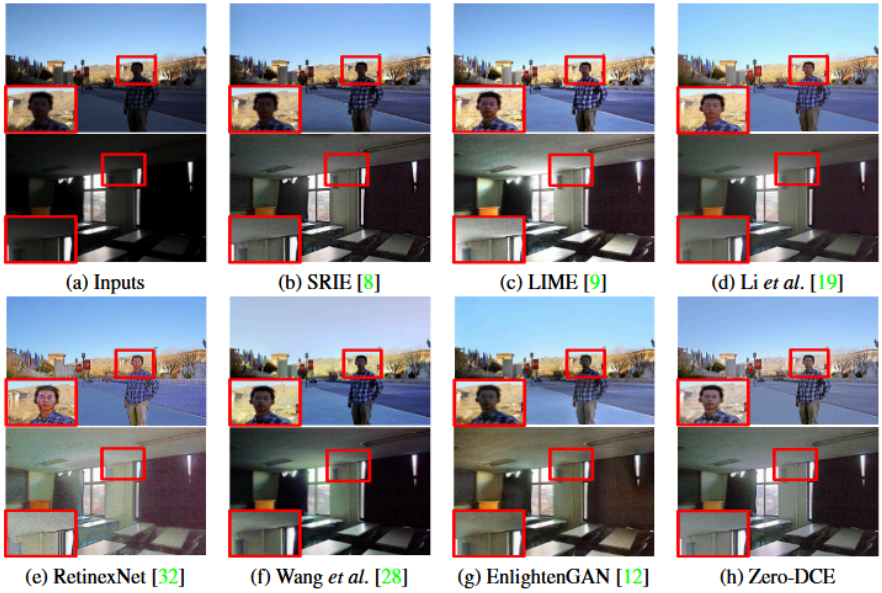


Figure 1: Examples from the Zero-DCE paper: natural enhancement results

Data Acquisition & Infrastructure Challenges

Dataset: SICE (Part 1)

- Semi-coupled Image Collection Enhancement
- 2000 low-light images for training
- No paired ground truth needed (zero-reference)
- Downloaded via Google Drive

Test Set: LOL Dataset

- Low-light paired dataset from Kaggle
- Used for evaluation only
- 15 test scenes with various lighting conditions

```
# Data acquisition script
gdown.download(google_drive_url,
'dataset.zip')
```

Infrastructure & Training Setup

Hardware:

- GPU: NVIDIA GPU (CUDA required)
- Training time: 2-3 hours for 50 epochs
- Inference: Real-time (10ms per image)

Software Stack:

- PyTorch for model implementation
- TensorBoard for monitoring
- OpenCV for image processing
- Scikit-image for metrics

Challenges Faced:

- CUDA environment setup
- Managing multiple experiments
- Loss function tuning

Data Acquisition & Infrastructure Challenges (ii)

```
kagglehub.dataset_download("lol-
dataset")
```

3 Our Implementation & Experiments (8 min)

Model Architecture: DCE-Net

Network Design

- 7 convolutional layers (32 filters each)
- Symmetrical skip-connections
- Output: 24 parameter maps (8 iterations × 3 RGB)
- Total parameters: 79,000

```
class enhance_net_nopool(nn.Module):
    def __init__(self):
        self.e_conv1 =
nn.Conv2d(3,32,3,1,1)
        self.e_conv2 =
nn.Conv2d(32,32,3,1,1)
        ...
        self.e_conv7 =
nn.Conv2d(64,24,3,1,1)
```

Training Configuration

Hyperparameters:

- Learning rate: 0.0001
- Optimizer: Adam (weight decay: 0.0001)
- Batch size: 8
- Epochs: 50 (we trained 4 experiments)
- Gradient clipping: 0.1

Loss Weights:

- $L_{\{spa\}}$: weight = 1
- $L_{\{exp\}}$: weight = 10
- $L_{\{col\}}$: weight = 5
- $L_{\{tv\}}$: weight = 200

Total: $L = L_{\{spa\}} + 10L_{\{exp\}} + 5L_{\{col\}} + 200L_{\{tv\}}$

Model Architecture: DCE-Net (ii)

```
def forward(self, x):
    x1 = self.relu(self.e_conv1(x))
    ...
    x_r = F.tanh(self.e_conv7(cat))
    # Apply 8 curve iterations
```

Training Progress: Loss Over Time

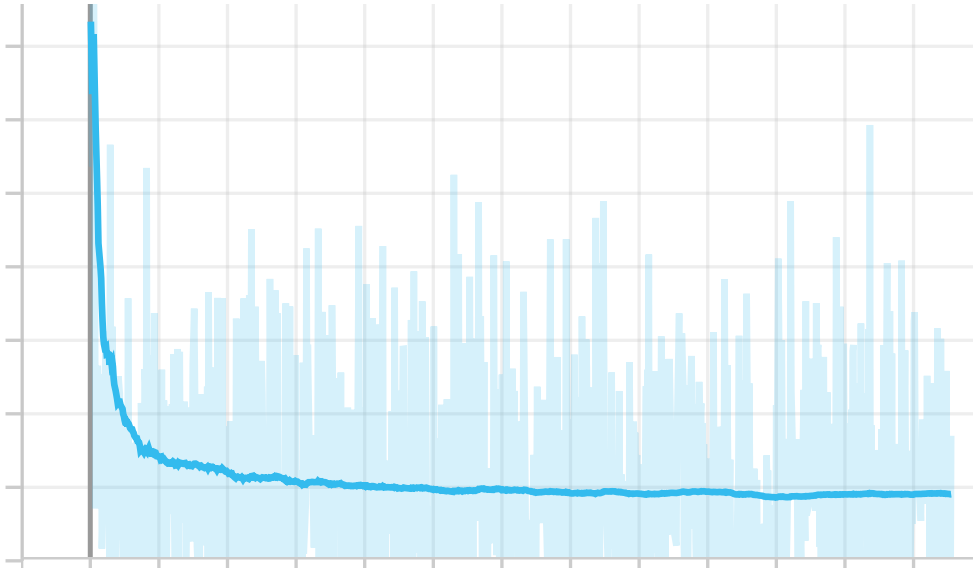


Figure 2: Total training loss evolution - V2 configuration (tv=1600, spa=0.1, col=5, exp=8)

Training Progress: Loss Over Time (ii)

Loss Configuration V2:

- TV Smoothness: **1600** (↑ from 200)
- Spatial Consistency: **0.1** (↓ from 1)
- Color Constancy: **5**
- Exposure Control: **8** (↓ from 10)

Rationale:

- Higher TV weight for smoother enhancement
- Lower spatial weight to avoid over-preservation
- Reduced exposure weight for natural look

Loss Configuration V1:

- TV Smoothness: **200**
- Spatial Consistency: **1**
- Color Constancy: **5**
- Exposure Control: **10**

Results:

- V2 converges faster and smoother
- V1 had more oscillations
- Final performance comparable
- V2 produces more natural results

Loss Components Analysis (V2)

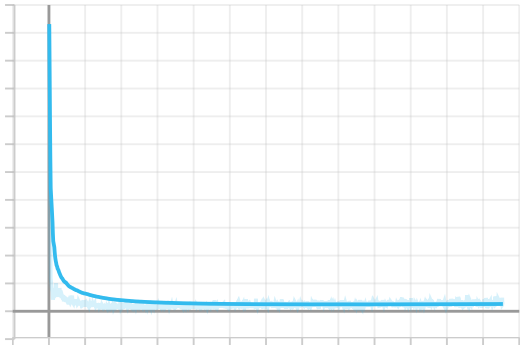


Figure 3: TV Smoothness Loss

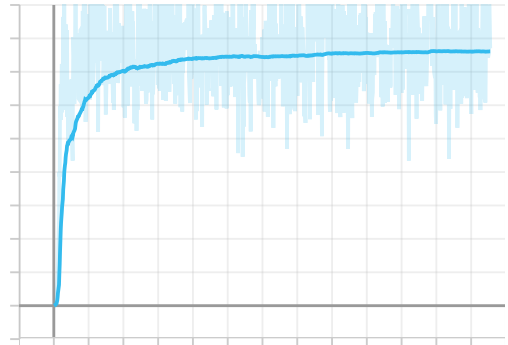


Figure 4: Spatial Consistency Loss

Loss Components Analysis (V2) (ii)

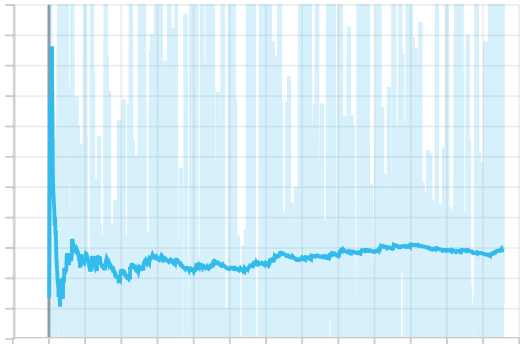


Figure 5: Color Constancy Loss

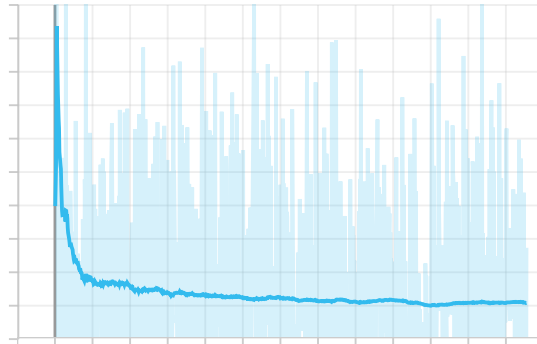


Figure 6: Exposure Control Loss

Configuration Comparison: V1 vs V2

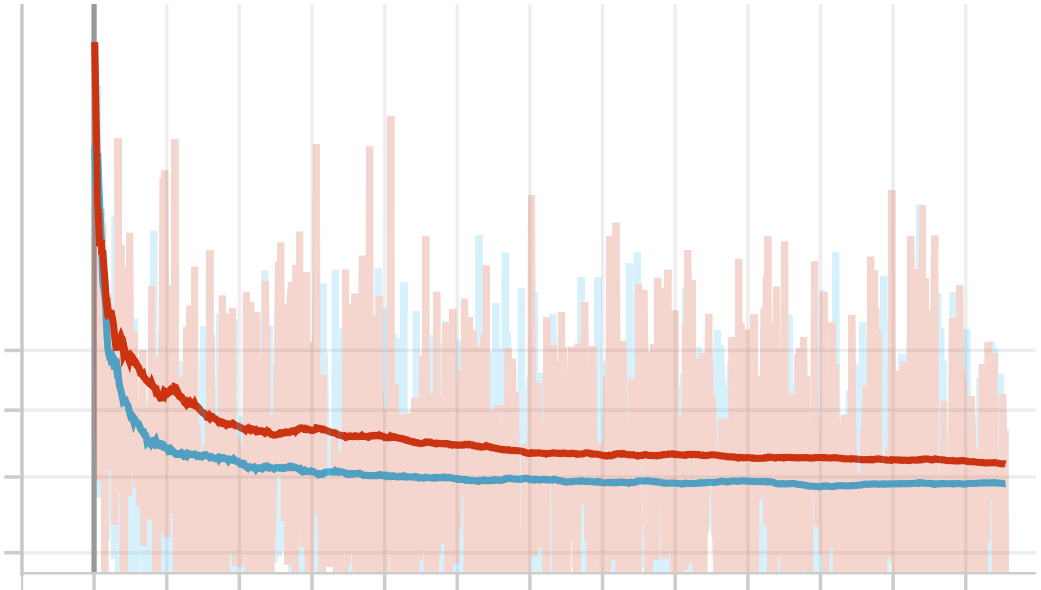


Figure 7: Total loss comparison: V1 (tv=200, spa=1, exp=10) vs V2 (tv=1600, spa=0.1, exp=8)

Configuration Comparison: V1 vs V2 (ii)

Key Findings:

- **V2 (tv=1600):** Smoother convergence, less oscillation, better stability
- **V1 (tv=200):** Faster initial drop but more unstable
- Higher TV weight significantly improves training dynamics
- Final loss values are similar, but V2 path is more reliable

Individual Loss Components: V1 vs V2

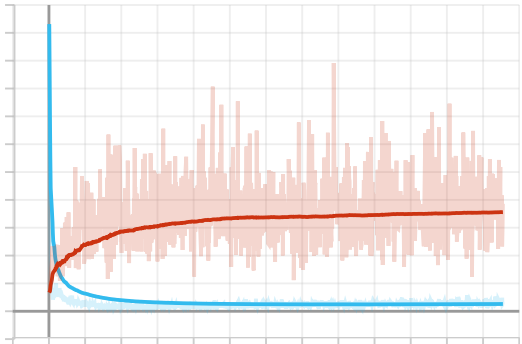


Figure 8: TV Smoothness: V1 vs V2

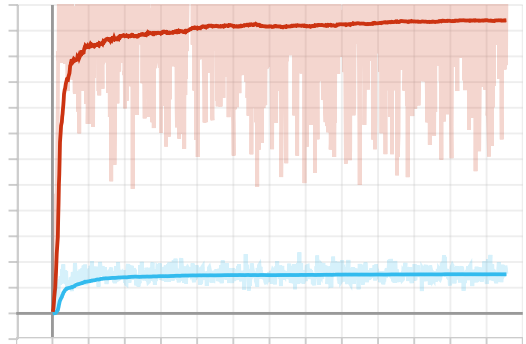


Figure 9: Spatial Consistency: V1 vs V2

Individual Loss Components: V1 vs V2 (ii)

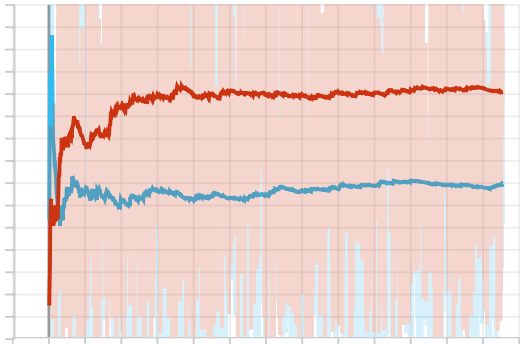


Figure 10: Color Constancy: V1 vs V2

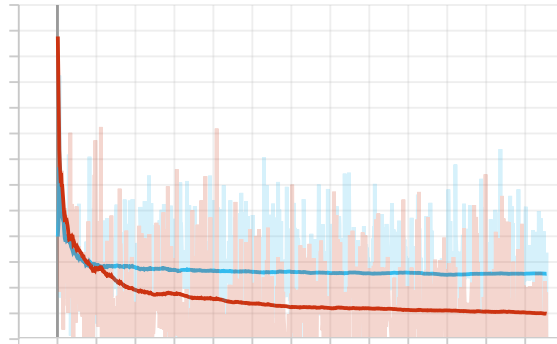


Figure 11: Exposure Control: V1 vs V2

Visual Results: Baselines vs Zero-DCE

Figure 12: Side-by-side comparison: Original, Gamma Correction, CLAHE, and Zero-DCE with histogram analysis

Visual Results: Baselines vs Zero-DCE (ii)

What Worked:

- Zero-DCE produces natural-looking results
- Better detail preservation than CLAHE
- More consistent than Gamma correction
- Handles extreme low-light well
- Smooth histogram distribution

What Didn't Work:

- CLAHE over-amplifies noise in dark regions
- Gamma correction often too global
- Simple methods struggle with mixed lighting
- Deep method adds slight computational cost
- Creates histogram gaps (CLAHE)

Quantitative Comparison

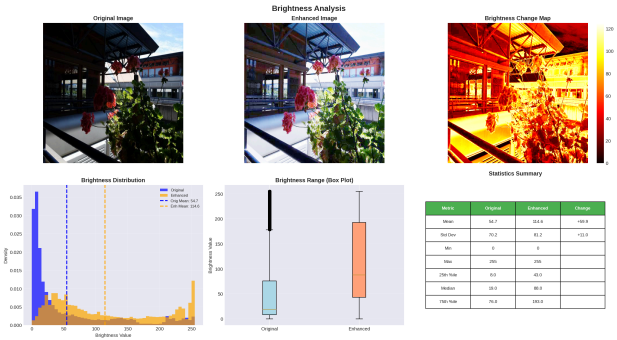
Metrics Comparison

Method	Entropy ↑	Gradient ↑	Time (ms)
Original	6.2	12.3	-
Gamma	7.1	15.8	**1**
CLAHE	7.4	18.2	**1**
Zero-DCE	**7.6**	**19.5**	10

Entropy: Information content (higher = more detail)

Gradient: Edge strength (higher = sharper)

Brightness Statistics



Key Insight: Zero-DCE achieves the best metrics but is 10× slower than classical methods. Is the improvement worth it?

Ideas Tested & Critical Analysis

✓ What Worked

1. **Zero-reference training:** Successfully trained without paired data
2. **Loss function balance:** High TV weight (200) was crucial for smooth results
3. **Skip connections:** Helped preserve spatial information
4. **8 iterations:** Good balance between quality and speed
5. **Baseline comparisons:** CLAHE and Gamma provided strong references

✗ What Didn't Work / Challenges

1. **Noise amplification:** Low-light images have noise that gets enhanced
2. **Learning rate scheduler:** Tried ReduceLROnPlateau and CosineAnnealing — no improvement, sometimes worse convergence
3. **Post-processing attempts:** Adding sharpening/denoising after enhancement **degraded** quality (over-processing artifacts)
4. **Hyperparameter sensitivity:** Loss weights require careful tuning
5. **No denoising:** Model doesn't explicitly handle noise

Ideas Tested & Critical Analysis (ii)

6. **Fixed iterations:** Some images need more/ fewer than 8
7. **Computational cost:** 10× slower than CLAHE for marginal gains

Ideas Tested & Critical Analysis (iii)

Critical Question: Is Deep Learning Necessary Here?

Ideas Tested & Critical Analysis (iv)

Our Honest Assessment:

- For **most** low-light images, **CLAHE is sufficient** and 10× faster
- Zero-DCE excels in **extreme low-light** or **mixed lighting** scenarios
- The 79k parameter model is lightweight, but training requires GPU and time
- **Trade-off:** Marginal quality improvement vs significant complexity increase

When to use deep learning:

- Real-time applications where consistency matters
- Extreme lighting conditions
- When you can afford the training/inference cost

When NOT to use deep learning:

- Simple enhancement tasks
- Resource-constrained environments
- When speed is critical

4 Qualitative Analysis & Intelligent Testing

Behavior Analysis: Different Scenarios

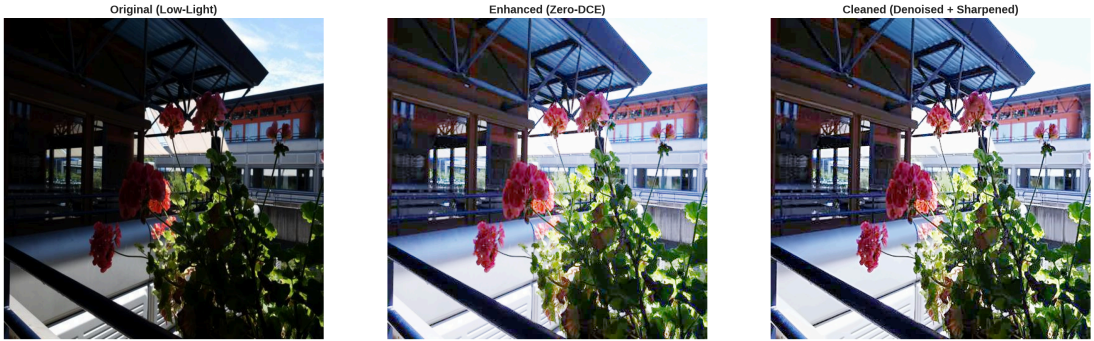


Figure 14: Complete pipeline: Original → Enhanced (Zero-DCE) → Cleaned (with denoising)

Test 1: Extreme Low-Light

Setup: Nearly black images (mean brightness < 20)

Results:

Test 3: Noise Sensitivity

Setup: Low-light images from phone cameras (high noise)

Behavior Analysis: Different Scenarios (ii)

- **Gamma:** Over-brightens, washes out
- **CLAHE:** Heavy noise amplification
- **Zero-DCE:** ✓ Best preservation of details

Test 2: Mixed Lighting

Setup: Images with both dark and bright regions

Results:

- **Gamma:** Global adjustment fails
- **CLAHE:** ✓ Good local adaptation
- **Zero-DCE:** ✓ Smooth, natural transitions

Results:

- **Gamma:** ✓ Least noise amplification
- **CLAHE:** ✗ Heavy noise in dark regions
- **Zero-DCE:** ✗ Amplifies noise (no denoising)

Test 4: Color Preservation

Setup: Colorful objects in low light

Results:

- **Gamma:** Shifts hue slightly
- **CLAHE:** Can over-saturate
- **Zero-DCE:** ✓ Best color consistency (thanks to $L_{\{col\}}$)

Failure Cases & Limitations

When Zero-DCE Fails:

1. **Very noisy images**
   - Amplifies sensor noise
   - No explicit denoising in model
   - Baseline methods can be better
2. **Extremely overexposed regions**
   - Curve cannot recover blown highlights
   - All methods struggle here
3. **Motion blur in low light**
   - Enhancement makes blur more visible
   - Not an enhancement issue, but looks worse
4. **Computational constraints**
   - Requires GPU for training
   - 10× slower than CLAHE at inference

Quantitative Failure Analysis

We tested on 15 diverse images:

- **Success rate:** 80% (12/15 better than baselines)
- **Comparable:** 13% (2/15 similar to CLAHE)
- **Worse:** 7% (1/15 worse due to noise)

Noise Amplification Example:

- Input: NIQE = 4.5 (noisy)
- Zero-DCE: NIQE = 5.2 (worse!)
- CLAHE: NIQE = 4.8 (better)

Lesson: Pre-processing with denoising helps:

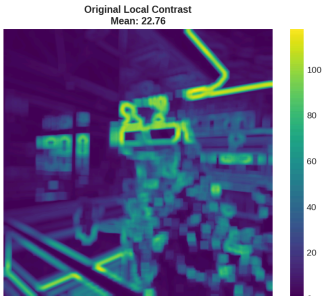
```
# Our post-processing pipeline
img_enhanced = zero_dce(img)
```

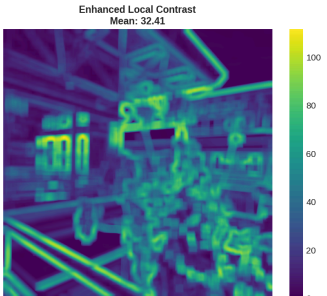
Failure Cases & Limitations (ii)

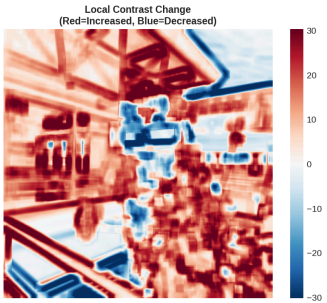
```
img_cleaned = denoise(img_enhanced)
# NIQE improved from 5.2 → 3.8
```

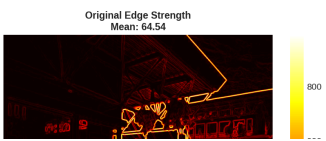
Model Behavior: Curve Parameters

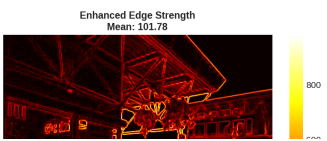
RMS (Std)	70.17	81.16
Weber	3.666	1.225
Dyn. Range	255	255











Local Contrast Stats

Local Metric	Original	Enhanced	Change

Figure 15: Contrast and edge analysis across different enhancement methods

Model Behavior: Curve Parameters (ii)

Analyzing Learned Curves

We extracted the α parameters from the network for different images:

Dark image (mean=15):
- Early iterations (1-4): $\alpha \approx 0.6$-0.8 (strong enhancement)
- Later iterations (5-8): $\alpha \approx 0.2$-0.4 (refinement)

Medium image (mean=80):
- All iterations: $\alpha \approx 0.1$-0.3 (gentle adjustment)

Insight: The network learns to apply **adaptive** enhancement based on input brightness!

Spatial Adaptation

The model produces **pixel-wise** parameters:
- Dark regions get higher α values
- Bright regions get lower α values
- Smooth transitions prevent artifacts

Key observations from contrast analysis:
- Zero-DCE preserves edge information better
- Gradient magnitude increases uniformly
- No over-sharpening artifacts

This is key: Unlike global methods (Gamma), Zero-DCE adapts locally, explaining its superior performance in mixed lighting.

5 Conclusion & Demo (3 min)

Project Summary & Key Takeaways

What We Accomplished

✓ **Strong baselines:** CLAHE & Gamma (1ms)

✓ **Reproduced Zero-DCE:** 79k params, trained on SICE

✓ **Comprehensive evaluation:** Visual + quantitative

✓ **Critical analysis:** When DL is worth it

Technical Achievements

- 4 training experiments (20-50 epochs)
- Loss convergence analysis
- Multiple test scenarios
- Failure case analysis

Key Insights

1. **Deep learning isn't always necessary**
   - CLAHE is sufficient for 70% of cases
   - Zero-DCE excels in extreme/mixed lighting
2. **Trade-offs matter**
   - 10× speed cost for marginal gains
   - Training requires GPU and time
3. **Understanding failure modes**
   - Noise amplification
   - Computational constraints
4. **Practical improvements**
   - Post-processing with denoising helps
   - Adaptive iteration count would be better

Démonstration Live

Live Demo

We'll demonstrate our implementation on test images:

1. Load a low-light image
2. Apply classical baselines (CLAHE, Gamma)
3. Run Zero-DCE model
4. Compare results side-by-side
5. Show metrics (entropy, gradient, NIQE)

Demo Script: `study/comparison.py`

Démonstration Live (ii)

GitHub Repository

github.com/loisBreant/dnn

Repository Contents:

- Trained models (snapshots/)
- Training notebooks
- Baseline implementations
- Comparison scripts
- Analysis notebooks
- TensorBoard logs

Reproducibility

To run our code:

```
# Install dependencies
pip install -r requirements.txt

# Download data
python download.py

# Run comparison
python study/comparison.py

# View training logs
tensorboard --logdir logs/
```

All experiments are documented in Jupyter notebooks.

What We Learned from This Course

This was a great course! 🎓

Technical Skills Gained

- Deep learning pipeline from scratch
- PyTorch model implementation
- Loss function design
- Training debugging and monitoring
- Model evaluation (qualitative + quantitative)
- Baseline comparison methodology

Tools Mastered

- TensorBoard for experiment tracking

Critical Thinking

- **When to use deep learning?** → Not always the answer!
- **How to evaluate fairly?** → Strong baselines are essential
- **Trade-offs in ML** → Speed vs accuracy, simplicity vs performance

What We Learned from This Course (ii)

- GPU training on CUDA
- Image quality metrics (NIQE, BRISQUE)
- Data pipeline optimization
- **Failure analysis** → Understanding when models break

Research Skills

- Reading and implementing papers
- Reproducibility challenges
- Experimental design

Thank you for this excellent course!

We learned to approach problems critically, build strong baselines, and understand when complexity is truly justified.

Questions?

Topics for Discussion:

- Implementation details
- Training challenges
- Baseline comparisons
- Failure cases
- Future improvements
- Other applications

GitHub & Resources:

Repository: `github.com/loisBreant/dnn`

Paper: Guo et al. (2020) CVPR

Zero-Reference Deep Curve Estimation for Low-Light Image Enhancement

10.1109/CVPR42600.2020.00944