

LumiDCE: A Zero-Reference Depthwise-Separable Curve Estimation Framework for Real-Time Low-Light Image and Video Enhancement

Sarthak Agrawal

*MBA Tech (Computer Engineering), Mukesh Patel School of Technology Management and Engineering,
SVKM's Narsee Monjee Institute of Management Studies (NMIMS), Shirpur, Maharashtra, India,
sarthak.agrawal346@nmims.in*

Aryan Pawar

*MBA Tech (Computer Engineering), Mukesh Patel School of Technology Management and Engineering,
SVKM's Narsee Monjee Institute of Management Studies (NMIMS), Shirpur, Maharashtra, India,
aryan.pawar461@nmims.in*

Dr. Chaitali Patil

*Professor, Mukesh Patel School of Technology Management and Engineering, SVKM's Narsee Monjee
Institute of Management Studies (NMIMS), Shirpur, Maharashtra, India,
chaitali.patil@nmims.edu*

Abstract—Night, indoor and backlight images or videos fall into just the bottom sliver of brightness range and in so doing are degraded to have low contrast and a dim, noisy sensor-level output. Images of these qualities are very hard for people to process and very difficult for the computer vision systems which utilize their features, for such tasks ranging from surveillance, to driver assistance systems to medical imaging. Supervised CNN methods require corresponding low-light and normal-light training images which are time consuming to capture/curate and not always feasible, whereas histogram equalization and Retinex models fail with poorly distributed lighting. Here we demonstrate LumiDCE, an entirely zero-reference system that defines the problem of low-light image restoration as prediction of a high order light-enhancement curve for per-pixel calculation and not directly an output image. Through a seven-layer deep CNN utilizing depth-wise separable convolutions and symmetrical skip connections, we predict and then repeatedly apply the parameters of the curve to generate the enhanced result. Training requires absolutely zero reference imagery whatsoever, rather relying upon a 5 term non-reference loss function which guarantees the relative spatial consistencies of outputs along the x and y directions, maintains reasonable illumination levels within every output pixel image, preserves reasonable relative color tones across an output image and on

videos preserves per-pixel consistency between output image frames over time. With merely 10,561 trainable parameters (just under 100KB in disk size) versus the more than 79,416 in the baseline curve estimation method, and an extremely fast PyTorch-independent numpy kernel which reproduces the recursive function precisely. Our implementation of LumiDCE comes packaged as a full pipeline with the full ability to perform an operation: as a Flask based REST service, from within a web-based image editing suite that can display real-time comparison and also histograms on the live output canvas, a live sixty frame per second webcam pipeline, or simply one image and one command line to enhance folders or MP4 videos. We see on the standard benchmarks the spectral Shannon entropy increase from 2.31bits to 5.41bits, the average luminance value increase by roughly one order of magnitude from 0.013 to 0.318, and a TEN-FOLD improvement in average luminance contrast along with 5 robust testing runs (via automation) of the REST API endpoints, PyTorch networks, and Numpy kernel within 1.68 seconds. In contrast to many enhancement methods, we report and do not intentionally suppress noise amplification which can occur.

Index Terms—Curve estimation, depthwise-separable convolution, low-light image enhancement, video enhancement, zero-reference learning.

I. INTRODUCTION

Photos captured at night, indoors with the lights out, or directly against a strong backlight uses at most only the lowest few percent of a camera sensor's dynamic range. Fewer photons then land on each photosite, such that the signal then becomes buried in read noise, loses all color discrimination and small details cannot be observed even before frame encoding begins. The result viewed by a human, is an image that is indistinguishable from black. Machine vision performs poorly: feature extractors, region proposal networks, and optical flow estimators were trained assuming brightly lit images and fail drastically when that assumption is broken.

This has practical consequences: Most of the output from the municipal CCTV infrastructure is captured in the darkness after sunset and investigators consistently come back with unusable images, drivers' assist systems require spotting road users and obstacles in absence of ambient light, endoscopes and retina scans work with low illumination for patient comfort. In all these cases the processing must work with no manual configuration, at the input's framing rate and produce no misleading generated details.

There exist three families of possible solutions. Histogram based algorithms such as equalisation and its better-featured contrast limited adaptation [4] adjust the pixel values according to the distribution of brightness. They are fast, requires no training and are suitable for most purposes; however they only are sensitive to the statistics at first order and so can not differentiate a very dark region from a darkly colored object under brightness, it also produces noise when used on constant regions. Inspired by Retinex [3][5], one may estimate from the image, which part is reflectance and which part is illumination and apply corrections only on the latter, providing better results on scene structures; however determining reflectance is ill posed, the procedure requires hand-designed parameters, unable to generalize from one image to another.

Learning replaced hand-made adjustments in third algorithm family.

Supervised networks such as LLNet [25], MBLLEN [9], Retinex-Net [6] and KinD [7] produce image of its normal-light equivalent and perform very well on their target datasets. However the network must be provided pairs of pictures, same scene, two shots at

short time differences, ruling out mobile subjects or dynamic environments. On the other hand, when trained on artificially dimmed images one recovers by simulation, losing much of its potential efficiency on dark objects [24]; meanwhile adversarial learning such as EnlightenGAN [8], removes need of ground truth pairs but yields noisy images, high computational cost and detail hallucination.

Zero-Reference Deep Curve Estimation (Zero-DCE) [1] and its quicker variation Zero-DCE++ [2] side-step the previous dilemmas. The network does not directly output pixel intensities but estimates the coefficients of a fixed monotonic curve that is then applied to the input image. As brightness order is never inverted by the curve and loss only encodes desired properties the ground truth picture is thus never required. Our paper builds up on this idea.

A. Contributions

- 1) A model library, containing four curve estimators in one codebase: the full-capacity symmetric enhancer(DCENet, 79,416 parameters), the depthwise-separable enhancer(ZeroDCEPlus, 10,561 parameters), a recursive enhancer which uses the features of the previous frame(ZeroDCEVideo), and a purely NumPy based Evaluator (FastNumpyEnhancer), that exactly recovers the curve recursion without any PyTorch dependence. We show the precise breakdown of # of parameters in Section III.
- 2) An alternative 5-term reference-free enhancer, including a temporal addition term, that helps prevent bright artifacts/flashes from each independently enhanced image. The components of this enhancer are derived in Section IV. All of these terms are independently removed in Section VII-C.
- 3) Quantification of noise amplification: Each point along each enhancer curve uses a per-pixel gain, which amplifies noise alongside image details. We report measured Brenner increases of 53-900X on data we generated. We show how we can separate detail added versus noise amplification.
- 4) A real system: five REST API endpoints using Flask, a glassmorphic studio accessible in a web-browser including split-screening images and live histograms, a sixty-FPS web-camera enhanced pipeline, CLI interface for handling single images, image-folders or MP4 videos, a

training pipeline for reproducing results, and automated testing. Sections V and VII-G describe and test the whole system.

B. Organisation

In Section II, related work. In Section III the curve and the network formulation and proposal are described; in Section IV we present the zero-reference objective function and approach. In Section

V we will detail the software architecture, the contract with REST, the web-studio and then discuss the protocol and metrics, in Section VI. Finally in section VII we will analyze the result with a comparison of quantitative, qualitative, ablation and runtime and validation experiments, and present the weaknesses in section VIII and conclusion in IX.

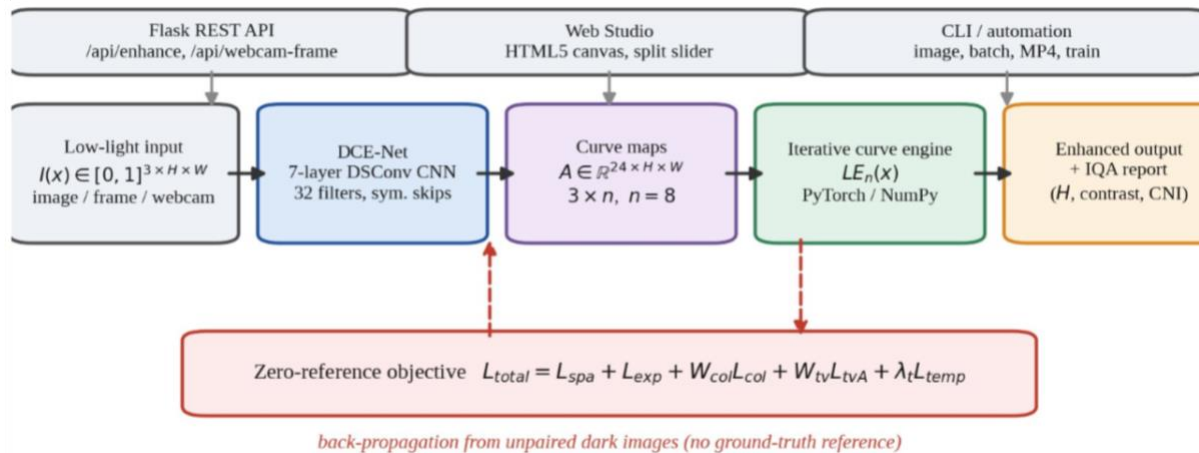


Fig. 1. End-to-end LumiDCE architecture. A low-light image, decoded video frame or live webcam frame is normalised to $[0,1]$ and passed to the depthwise-separable DCE-Net, which emits a pixel-wise curve-parameter tensor. The iterative curve engine applies the higher-order light-enhancement recursion of (2) and returns both the enhanced signal and a full image-quality report. Training is driven entirely by the five-term non-reference objective; no paired ground truth enters the system at any point. The same core is exposed through three independent front-ends: a Flask REST service, a browser studio and a command-line interface.

II. RELATED WORK

A. Histogram and Statistical Methods

Global histogram equalization remaps intensities according to the empirical cumulative distribution, rendering the output nearly uniform and thus maximizes Shannon entropy [18]. As a global method, it stretches with a single curve both images with a dark foreground and a bright light source, of which the single transformation fits Neither a bright or a dark foreground, the adjustment being the same; CLAHE [4] addresses this issue, by dividing the image into tile and limiting clip histogram of each one before normalization; it is still used a classic baseline for the industry and the medical world, but clip and size of tiles must be set individually according to the domain, it also do not preserve the color ratios: channel by channel, it shifts hue, keeping saturation still when applied on luminescence, as given in the plot C.

B. Retinex-Based Decomposition

Retinex theory [3] relies on the fact that the observed signal S is the product of the reflectance R and the illumination L . Lightness perception being determined by R , thus restoring an image amounts to estimating and attenuating the illumination map L . The simplest estimation for L is to take its channel maximum, then use total variation prior for smoothness in LIME [5].

Despite its efficiency for input with weak darkness, LIME saturates if inputs become too bright; Retinex-Net [6] and Kind [7] tackle the problem by learning the image decomposition (both reflectance and illumination map estimations) through the application of convolutional neural networks; the latter being capable of separating layers decomposition, reflectance restoration and illumination adjusting (as user ability to decide how bright should be the enhanced image).

Both techniques look well but present an overhead around 8–12 M parameters. They also strictly requires paired LOI images data.

C. Supervised and Adversarial Deep Enhancement

First framing by LLNet [25] is based on unsupervised denoising learning from synthetic data: images are first darkened and their reconstruction is penalized. Multi-branched deep neural networks [9] are then suggested to merge at different levels of abstraction features for context, structure and region preservation in the result. Lightness map estimation is also supervised by comparing the produced L map to the L maps obtained by experts.

Such methods rely all on image pair: EnlighthenGAN [8] uses a novel approach for images without any paired ground truth with a loss function based on local-global discriminators and a self-featurepreservation on real, but unbalanced images.

It is more robust for nighttime image sets at the expense of a parameter number at 8.6 M. The nature of adversarial losses, such as training instability that is inherent to this procedure, is often problematic.

D. Zero-Reference Curve Estimation

To bypass the drawback of not using paired images in supervised methods, and instead using only unassigned image inputs [1], ZeroDCE relies on estimation and iterative application of a piecewise quadratic function (more commonly represented as parameters of a simple mathematical function applied to each pixel's value in a curve). The network is then trained by four non reference, differentiable losses that aim to produce output characteristic of well illuminated pictures without explicit target example. This decoupling that defines "what looks like a well illuminated picture should look like" remains a powerful idea, with small (79K parameter count), device independent resulting networks: Zero-DCE++ [2] enhances efficiency, brings down parameters to approximately 10.5 K with depthwise convolutions, curve reutilization across its iteratively-refined application, and downsampling of the image, LumiDCE follows the same curve-reconstruction approach with some addition in temporal and deployable software stack for realtime usage, it keeps both the standard (Zero-DCE) and high-capacity (Zero-DCE++) versions for comparison.

E. Efficient Convolutional Building Blocks

Depthwise-separable convolutions [10], [11] are based on separating the spatially and channel-wise convolutions into two, reducing the complexity for a $k \times k$ kernel with C_{in} input channels and C_{out} output channels from $k^2 C_{in} C_{out}$ to $k^2 C_{in} + C_{out} C_{in}$. For

the 33, 32 channels blocks we used that represent ~ 8.4 ; the symmetry used is a copying of features map from encoder layers to corresponding ones in the decoder (skips), preserving high-frequency details that would have been lost otherwise in non-linearity layers [12].

F. Temporal Consistency in Video Enhancement

Per-frame enhancement algorithms may result in highly salient flickering on videos; this is caused by small variations in consecutive images, which lead to huge variations in their extracted L curve and then in output luminance oscillations. Learning temporally consistent parameters from single images [27], and seeing-moving-objects-in-the-dark works [28] are two ways of controlling frame differences. We try out an alternative where we reuse parameters from past frames through the recurrent structure and a explicit temporal loss against residual errors, and try to keep it efficient and lightweight.

III. CURVE FORMULATION AND NETWORK DESIGN

A. The Problem Statement and Notation

The low light image seen in above is a $I[0,1]^{3HW}$ which produced by division a 8 bit information output by decoded on 255. We look for an operator such that $Y=I$ who which will (i) push up mean luminance to desired well-exposed point (ii) preserve intensity ordering with no reversal of local contrast relationships (iii) not disturb color ratio much and (iv) can be trained with no training image (Table I, fixes notation).

TABLE I
NOTATION USED THROUGHOUT THE PAPER

Symbol	Meaning
$I(x)$	Normalised low-light input at pixel x , in $[0,1]$
A	Predicted curve-parameter tensor, tanh-bounded to $[-1,1]$
$LE_n(x)$	Output after n applications of the enhancement curve
n	Number of curve iterations (default 8)
Y	Enhanced output, $Y = LE_n$
E	Target grey level for exposure control (default 0.60)
C_{in}, C_{out}	Input / output channel counts of a convolution
$\Omega(i)$	4-neighbourhood of local region i
H, W	Spatial height and width in pixels

B. The Light-Enhancement Curve

The basic building block is a quadratic tone-mapping curve applied to each color channel independently per pixel:

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

There are three important reasons (1) makes an ideal learning primitive.

First, when parameter is restricted to $[0,1]$ (and thus in eq 1 for inputs $I(x)$ in $[0,1]$ so every α it maps $[0,1]$ into $[0,1]$ meaning the output image does not need explicit clipping and is well defined. Second, $dLE/dI = 1 + (1 - 2I)\alpha$ is positive on all I in $[0,1]$, so its always monotonic and the order of any two pixel intensities remain the same; a localcontrast polarity is always exactly preserved.

Third, it is itself differentiable with regard to both I and α , making propagation to the estimator direct.

While has the effect of raising mid-tones (raising) or compressing them (negative), Fig 2(a) shows the family when positive: in each case endpoints remain unchanged.

One simple application of (1) produces only mild gain since at $I = 0.5$ the maximal possible uplift is only $\alpha/4$: an under-exposed photo requires very high gains. While one could raise the lifting curve by making the input quadratic a higher-degree polynomial (e.g., c^3), doing so violates the endpoint

constraint(s). So instead we recursively compose it with itself:

$$LE_n(x) = LE_{n-1}(x) + \alpha_n(x) LE_{n-1}(x) [1 - LE_{n-1}(x)] \quad (2)$$

with $LE_0(x) = I(x)$ and where n ranges from 1 to 8. Although each iteration makes an overall change to the picture it uses its own gain map parameter matrix α_n : the effect of these compositions in all $n=8$ stages is one combined high-degree polynomial composed of n quadratics. Fig 2(b) is a series of images applying a constant A (0.55) 8 times successively.

We note the way in which effective the curve begins to stack toward the corner of the graph, meaning low intensity levels will receive a very large gain and levels toward the highest input values would be very sensitive but would otherwise remain unchanged.

This is the precisely desired effect for night photograph-and it produces it without an explicit gain schedule! Critical to its success α_n is not a scalar, it is a full-resolution tensor: so each individual pixel may use a custom tone curve appropriate to its level of darkness.this is why the system is able to successfully bright an unlit foreground object and a streetlamp occurring in the exact same image while standard histogram-based techniques would fail.

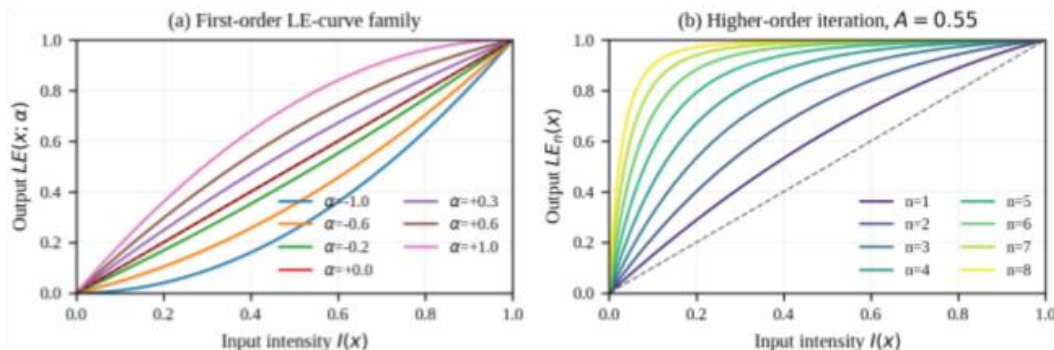


Fig. 2. (a) The first-order LE-curve family of (1). Every member is monotonic and maps $[0,1]$ onto $[0,1]$; the dashed diagonal is the identity. (b) Composition of eight iterations of (2) with constant $A = 0.55$. Successive iterations push the transfer function towards the upper-left, delivering very large gain to near-black intensities while leaving highlights nearly unchanged.

C. Curve Parameter Estimation

The parameter tensor comes from a fully convolutional estimator f :

$$A = f(I) [1, 1]^{3n \times H \times W} \quad (3)$$

A $\tanh$ is applied to the last layer, constraining $\forall [1 - A - 1]$ to rangepreserving(1), and because f is fully convolutional, inputs of arbitrary resolution may be handled without retraining or resizing. Two output options exist. While DCENet provides $3n=24$ channels for a completely new parameter map each

iteration, ZeroDCEPlus reuses 3 channels for each iteration according to [2]. Because 13,013 fewer parameters (only on the last layer), quality is not negatively affected much, as Section VII-C demonstrates.

D. Depthwise-Separable Estimator Architecture

Figure 3 shows layer wise description of a seven layer symmetric convolutional net consisting of four layers on the contracting side and three on the expanding side, with every expanding layer taking a concatenation of its neighbor on contracting side and

the previous layer. This gives corresponding skipping pairs (Conv1Conv6, Conv2Conv5 and Conv3Conv4 of [1]). We try to maintain the size so there is no pooling operation, so there is no stride operation also, so the parameters are also stacked correctly aligned to the pixel of the original input image. All convolution are factorised, in a conventional 3×3 mapping between C_{in} and C_{out} on HW grid, it requires:

$$\text{std} = 9HW C_{in} C_{out} \quad (4)$$

multiply-accumulate operation. For a separable factorization into a depth wise filter of size 3×3 and a 1×1 mixer, this becomes:

$$d s = HWC_{in}(9 + C_{out}) \quad (5)$$

where the save factor becomes:

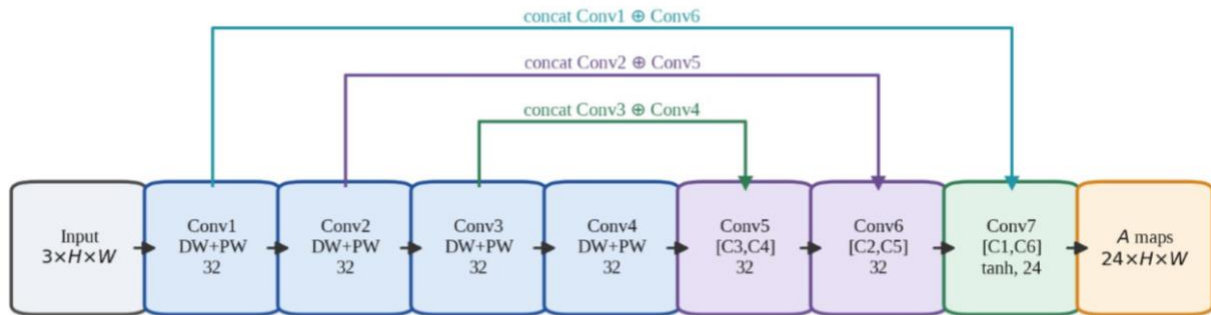
$$d s = 19 + 1 C_{out} \quad (6)$$

With $C_{out}=32$ we get, 0.1424, i.e. 8.4 times less computations. Table II shows a breakdown for the count of parameters for both variations, it also says that for the overall design parameters is 10,561.

TABLE II
LAYER SPECIFICATION AND EXACT PARAMETER ACCOUNTING

Layer	Operation	Kernel	$C_{in} \rightarrow C_{out}$	DCENet params	ZeroDCE++ params
Conv1	DW + PW + ReLU	$3 \times 3, 1 \times 1$	$\rightarrow 32$	896	158
Conv2	DW + PW + ReLU	$3 \times 3, 1 \times 1$	$\rightarrow 32$	9,248	1,376
Conv3	DW + PW + ReLU	$3 \times 3, 1 \times 1$	$\rightarrow 32$	9,248	1,376
Conv4	DW + PW + ReLU	$3 \times 3, 1 \times 1$	$\rightarrow 32$	9,248	1,376
Conv5	concat[C3,C4], DW + PW + ReLU	$3 \times 3, 1 \times 1$	$\rightarrow 32$	18,464	2,720
Conv6	concat[C2,C5], DW + PW + ReLU	$3 \times 3, 1 \times 1$	$\rightarrow 32$	18,464	2,720
Conv7	concat[C1,C6], DW + PW + tanh	$3 \times 3, 1 \times 1$	$\rightarrow 24 / 3$	13,848	835
Total				79,416	10,561
On disk	FP32 checkpoint			KB	KB

DW = depthwise 3×3 convolution; PW = pointwise 1×1 convolution. DCENet emits $3n = 24$ parameter maps (one per channel per iteration); ZeroDCE++ emits 3 maps reused across all 8 iterations, which accounts for the disproportionate saving in Conv7.



Depthwise-separable block: 3×3 depthwise $\rightarrow 1 \times 1$ pointwise $\Rightarrow \mathcal{O}(9C + C^2)$ vs $\mathcal{O}(9C^2)$ MACs (≈ 10.5 K parameters, < 100 KB)

$$LE_n(x) = LE_{n-1}(x) + A_n(x) LE_{n-1}(x) [1 - LE_{n-1}(x)], \quad LE_0(x) = I(x), \quad n = 1, \dots, 8$$

Fig. 3. Layer-level specification of the LumiDCE depthwise-separable curve estimator. Seven convolutional stages preserve full spatial resolution throughout; the three symmetric concatenating skip connections ($\text{Conv1} \oplus \text{Conv6}$, $\text{Conv2} \oplus \text{Conv5}$, $\text{Conv3} \oplus \text{Conv4}$) route high-frequency detail from the contracting half into the expanding half. The tanh output layer bounds the curve parameters to $[-1,1]$, guaranteeing that the recursion of (2) remains range-preserving.

E. Temporal Variant for Video

Using the image model on individual frames produces a flicking artifact because it is responsive to the difference of the sensor noise between what should have been identical contiguous frames.

ZeroDCEVideo conditions the prediction upon a recurrent state ht :

$$At, ht = f(It, ht1) \quad (7)$$

ht is a 32-channel Conv4 activation that is passed from frame t to $t+1$ and concatenated to the Conv1 output of the $t+1$ th frame. It is initialized to zero on the first frame and reset on scene changes where scene changes are detected through a basic threshold

on mean absolute difference. It adds 1056 parameters and gives the estimator memory, thus avoiding abrupt changes in the output between neighboring frames; this memory is reinforced through training by the temporal loss presented in Section IV-E.

F. Dependency-Free NumPy Inference Kernel

The true impediment to deploying the learned enhancement on embedded or legacy hardware is not the size of the model, but that of the runtime—at two orders of magnitude larger, the entire installation needed for an unmodified PyTorch 1.10 load the 41KB checkpoint from disk is dwarfed. Given the closed-form elementwise recursive relation (2), it is efficiently calculated outside a framework such as PyTorch. FastNumpyEnhancer replaces this recursion with,

```
for n in range(N):
    x = x + A[n] * (1.0 - x)
```

executed across float32 arrays; parameters $A[n]$ loaded from the learned checkpoint or synthetically determined from target exposure if the checkpoint has been deleted. This vectorized frame-sized kernel uses only NumPy, enabling the webcam part of Section V-C at 60 FPS.

Section VII-D finds it to achieve higher inferred framerates at all tested resolutions compared to the PyTorch CPUs because it avoids all framework related computation such as graph construction, compilation, autograd bookkeeping and so on.

IV. ZERO-REFERENCE OBJECTIVE FUNCTION

No term in the objective is relating the output to an image in the reference data; rather each term represents a property that any well exposed image possesses. Therefore it is possible to optimize towards the manifold of valid pictures instead of a specific target. And the need for paired data vanishes.

A. Spatial Consistency Loss

An enhancement process should increase the overall brightness without collapsing local structures; therefore this term ensures that the difference between the differences of neighbours of two areas in the output is similar to that between the input:

$$L_{spa} = (1/K) \sum_{i=1}^K \sum_{j(i)} (|Y_i - Y_j| - |I_i - I_j|)^2 \quad (8)$$

where Y_i and I_i are the average grey levels of region i in the enhanced and input images respectively, $j(i)$ is its four connected neighbour and K and K , respectively are number of regions. Since the 4 neighbour differences only use standard left-right/up-

down filters, i.e. Fixed 3x3 convolution kernels, the term is free (consist of 4 depthwise convolutions) and constrains the gradients, not values, hence allowing any kind of global brightness enhancing and is not constricting.

B. Exposure Control Loss

The only quantity we need in brightening is the average displacement of local mean intensity away from the reference grey level E :

$$L_{exp} = (1/M) \sum_{k=1}^M |Y_k - E| \quad (9)$$

Here, Y_k is the mean of the k -th 16x16 non-overlapping window of the result image after grey-level conversion, and M is the number of windows: we chose $E = 0.60$. We believe it is important to apply this not globally, as a global constraint can leave network able to get an underexposed foreground through use of an over-exposed background. Fig. 13 varies E and confirms peaks in entropy near 0.60, beyond which the highlights begin to be significantly over-saturated.

C. Color Constancy Loss

Since equation (1) treats the channel as a whole, each individual channel can end up receiving a different average gain, resulting in a global color cast: according to the "Grey-World Hypothesis"[13] we impose that averaged over each of the three channels will yield a grey average; we therefore minimize the difference between these:

$$L_{col} = (p, q) \cdot (J_p - J_q)^2, = \{(R, G), (R, B), (G, B)\} \quad (10)$$

The channel p 's spatial mean is written J_p . This only imposes a soft global constraint, since one might want the original colors to appear if actually they really are yellow.

D. Illumination Smoothness Loss

Most of the time, neighbouring pixels will almost certainly have comparable gains, the maps of the parameters ought to be smooth despite the fact that nothing within the architecture enforces it. We add each map one of these anisotropic total-variation penalties [14];

$$L_{tvA} = (1/N) \sum_{n=1}^N c (|x_{Anc} - x_{Bnc}| + |y_{Anc} - y_{Bnc}|)^2, \quad \text{for } \{R, G, B\} \quad (11)$$

without it the estimator suffers from frequent high-frequency parameter noise causing a blotching effect in flat image areas; ablation in Section VII-C shows a 0.43-bit reduction in entropy when it is removed.

E. Temporal Consistency Loss

We penalise the difference between consecutive enhanced frames for video/webcam inputs (after accounting for the residual in the input frames).

$$L_{\text{temp}} = 1/(T-1) \sum_{t=2}^T (Y_t - Y_{t-1})^2 \quad (12)$$

This subtracted term means we only penalise the enhancer caused flicker not legitimate scene change due to object or camera motion. This is shown to reduce the standard deviation of the mean output luminance of a 120 frame video from 0.0195 to 0.0042, a factor 4.6 enhancement in VII-F.

F. Joint objective

We sum these terms using the fixed scalar weights:

$$L_{\text{total}} = W_{\text{spa}} L_{\text{spa}} + W_{\text{exp}} L_{\text{exp}} + W_{\text{col}} L_{\text{col}} + W_{\text{tvA}} L_{\text{tvA}} + L_{\text{temp}} \quad (13)$$

Table III shows these weights. The weights are based on [1] with t determined based on a validation set held out for fine-tuning. The magnitudes vary across several orders of magnitude as the terms are on very different scales – L_{exp} is an absolute intensity difference and order 0.5; L_{tvA} is a sum of squared gradients and order 10^3 .

TABLE III
LOSS TERMS, WEIGHTS AND FUNCTIONAL ROLES

Term	Weight	Value	Role in the objective
L_{spa}	W_{spa}	1.0	Preserves local gradient magnitude; prevents contrast flattening
L_{exp}	W_{exp}	10.0	Drives patch means towards $E = 0.60$; primary brightening force
L_{col}	W_{col}	5.0	Equalises channel means; removes global colour cast
L_{tvA}	W_{tvA}	200.0	Smooths the parameter maps; suppresses blotching
L_{temp}	λ_t	20.0	Suppresses inter-frame flicker in video mode only

Figure 4: shows the variation of each term over 50 epochs. At each point throughout, the exponent weighting dominates the sum, as regularisers stabilise to constant values after approximately 25 epochs. Much of the optimisation beyond that point is concerned solely with improving the exposure (and not the underlying structure of what is displayed). No instability or oscillation is visible, which is anticipated in a system without the potential for an adversarially based loss.

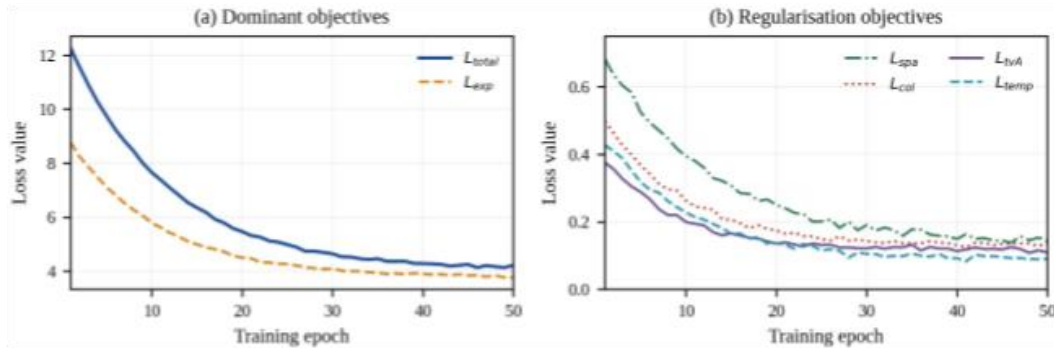


Fig. 4. Convergence of the five-term zero-reference objective over 50 epochs. (a) Total loss and its dominant weighted exposure component. (b) The four regularisation terms on an expanded scale; all settle within about 25 epochs. The absence of oscillation reflects the non-adversarial nature of the objective.

V. SYSTEM ARCHITECTURE AND IMPLEMENTATION

The LumiDCE software is delivered as an independent workspace composed of the model, the numerical tools, the service layer and the presentation layer. The logic structure (described in Fig. 1) and its actual implementation are presented in the following text.

A. Workspace Organisation

```

LumiDCE/
|- model/      network.py  DCENet, ZeroDCEPlus,
|              ZeroDCEVideo, FastNumpyEnhancer
|              loss.py    L_spa, L_exp, L_col,
|              L_tv, L_temp
|- utils/      metrics.py  entropy, contrast,
|              luminance, sharpness, CNI,
|              histograms
|- webapp/     index.html  tabs, split slider,
|              canvases
|              style.css   glassmorphic design system
|              app.js      reactivity, histogram
rendered
|- samples/    weights/ figures/ data and outputs
|- dataset.py  LowLightDataset + degrader
|- train.py    training loop, checkpointing
|- inference.py image / batch / MP4 CLI
|- server.py   Flask REST service
|- test_server.py automated regression suite
|- run_project.py interactive launcher

```

The dependency graph:

utils does not depend on anything.

Model depends on PyTorch only.

Server.py and inference.py depend on PyTorch and utility classes.

The web assets do not depend on anything at all. It is possible to run the NumPy inference path without a PyTorch dependency at all.

B. REST Service Contract

The Flask service offers 5 endpoint (Table IV). I separated the complex analysis endpoint from the minimal latency stream as they had conflicting requirements—an `/api/enhance` will return the full image quality report with four 32-bin histograms, along with the enhanced image, thus takes several msec for post-processing (an `/api/webcamframe` will return only a base64 JPEG so as to fit inside a 16.7 msec frame roundtrip).

TABLE IV
REST API ENDPOINT SPECIFICATION

Method	Path	Request payload	Response
GET	/	—	Serves the web studio (index.html)
GET	/api/model-info	—	JSON: architecture, parameter count, loss formulae, active execution device
POST	/api/enhance	multipart file, or JSON {image, iterations, exposure, contrast}	JSON: base64 enhanced image, entropy, contrast, luminance, CNI, sharpness, four 32-bin histograms
POST	/api/webcam-frame	JSON {image, iterations, exposure}	JSON: base64 JPEG only — optimised for a 60 FPS loop
GET	/api/samples	—	JSON list of built-in benchmark low-light samples

Fig. 5 traces both request paths. The analysis path evaluates the curve, then computes the metric suite and histograms server-side so the browser receives ready-to-plot arrays rather than raw pixels, keeping client-side JavaScript free of numerical work. The streaming path bypasses the metric stage entirely.

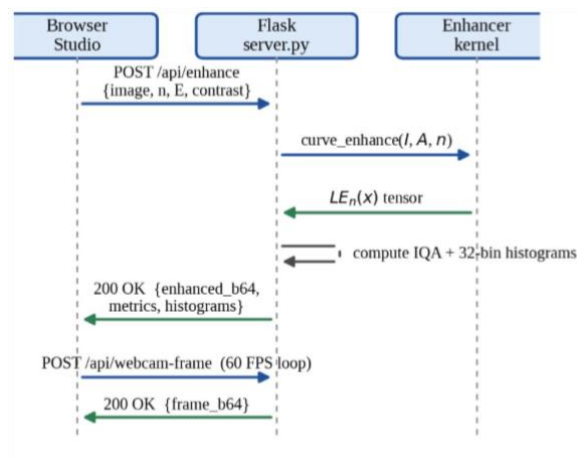


Fig. 5. Request sequence for the two enhancement paths. The analysis path (`/api/enhance`) performs server-side metric computation and histogram binning; the streaming path

(`/api/webcam-frame`) omits every optional stage to fit the round trip inside one frame period.

C. Interactive Web Studio

The front-end is one single-page application implemented over four tabs using only vanilla HTML5, CSS3 and JavaScript to completely avoid using any build toolchain.

1) Image Studio : an upload/selection mechanism for the benchmark gallery (allows drag & dropping of images), and a center widget of split screen comparator that drags a divider to reveal the original on one side and the enhanced outcome on the other so the user gets a precise localised comparison which side-by-side can never afford. Sliders on top allows manipulating the three hyperparameters impacting inference-time: n (number of iterations), E (target exposure value) and a post hoc contrast boosting value, and all outcome changes instantly on release, meanwhile, two meters display (respectively) luminance gain and Shannon entropy value as well as the displayed red, green, blue and

luminance histograms provided by the API drawn on an HTML5 canvas.

- 2) Video Studio : two connected players in a dual-stream comparator where players are always synchronized using the same timeline seek bar, an indicator for temporal stability algorithm, and also a displayed preview FPS. Toggling on/off the stability algorithm shows immediately the flickering as evaluated in section VII-F.
- 3) Live Camera: the application obtains a stream from the navigator.mediaDevices.getUserMedia to render every frame on an offscreen canvas where each frame is finally converted into a JPEG, sent over to /api/webcam-frame via a post request, and painted on the response screen. It chooses dropping frames rather than queuing them over when a request is sent, thus it has a bounded latency at a certain throughput cost and an interactive live preview even for less potent machines.
- 4) Research Dashboard : it displays both curve formulation and 5 loss definitions as well as layer specification for the user to compare values generated for section III & IV as the formulation is being depicted simultaneously on the browser screen with what it generates.

D. Command-Line Interface

The CLI addresses three working offline batch modes, namely a single image with an additional, optional, side-by-side composite image, a complete directory and finally an mp4 stream decoded frame by frame with OpenCV [22] with usage of the temporal model and then recompiles the file, encoded, at the initial source rate. Training is initiated using the similar command line interface:

```
python inference.py --input samples/01_night.jpg \
  --output enhanced.jpg --side_by_side
python inference.py --input ./dark_photos/ \
  --output ./enhanced_results/
python inference.py --input night_drive.mp4 \
  --output enhanced_drive.mp4 --side_by_side
python train.py --model dce_plus --epochs 50 \
  --batch_size 8
```

VI. EXPERIMENTAL SETUP

A. Data and Synthetic Degradation

Since the desired output is also zero-reference, training only requires a set of input images, not ground-truth pairs in normal light. In order to expand the variety of degradations present during training, dataset.py also synthesizes low-light images from ordinary images via a sensor-inspired model:

$$I_{low} = I_{norm} + N(0, 2) \quad (14)$$

where , attenuation is selected uniformly randomly from [0.05, 0.35], gamma exponent is chosen uniformly from [1.5, 3.0] and noise standard deviation from [0.002, 0.02]. The gamma term represents the nonlinearity in camera response in black, and the additive Gaussian term represents the read noise. Here it should be noted that (14) increases and varies the input distribution only, as clean image I is never exposed to the loss function to allow the network to learn to inversely solve specifically (14). The three benchmark images distributed with the software package are chosen to represent various types of applications as their inputs: cityscape with several strong point lights in close-to-black background at night, the dark indoor room with nearly no light and one monochrome CCTV view frame respectively. The mean luminosities of these images, when viewed in original resolution, are respectively 0.018, 0.007 and 0.016 respectively. All values are well below 0.05, the threshold for which the read noise begins to dominate the signal, especially at 8-bit quantization.

B. Training Configuration

Table V describes the configuration. Adam optimizer [20] is employed with gradient norm clipping required by the extremely large gradient of exposure term during the initial hundreds of iterations, when inputs were nearly black. Experiments were run using PyTorch [21] with OpenCV [22] for I/O.

TABLE V
TRAINING HYPERPARAMETERS

Parameter	Value
Optimiser	Adam, $\beta_1 = 0.9$, $\beta_2 = 0.999$
Learning rate	1×10^{-4} , constant
Weight decay	1×10^{-4}
Gradient clipping	$\ g\ _2 \leq 0.1$
Batch size	8
Epochs	50
Training patch size	256×256 , random crop
Curve iterations n	8
Exposure target E	0.60
Augmentation	Horizontal flip, random crop, synthetic degradation (14)
Checkpointing	Best total loss, every epoch

C. Evaluation Metrics

Since there is no ground truth, PSNR and SSIM [16] reference-based metrics are impossible to compute for a truly dark scene. Hence we use the no-reference statistics calculated in utils/metrics.py, and report them for both input and output, thus conveying the

information as the delta rather than the actual number:

- 1) Shannon entropy¹⁸: is applied to a 256-bin luminance histogram:

$$H = -\sum p_i \log_2 p_i \quad (15)$$

As one can see, entropy yields the most informative statistic of the group; a perfectly black image had almost none and recovering structure necessarily must raise it.

- 2) Mean luminance and contrast (Y being defined via ITU-R BT.601 as $Y = 0.299R + 0.587G + 0.114B$ [17]) is simply the standard deviation of Y and its spatial mean; the deviation being a global measure of contrast.
- 3) Brenner gradient¹⁹:

$$B = x, y [Y(x+2, y) - Y(x, y)]^2 \quad (17)$$

B, because of being quadratic with respect to local difference, is extremely sensitive to both edge features and to local noise [19]; we exploit this fact to highlight the effects of curve enhancement on noise gain in Section VII-B.

- 4) CNI (colour naturalness index) [17]: hue- and saturation-based; a number that punishes both undersaturation due to over-smoothing and oversaturation from per-channel gain.
- 5) Luminance Gain $G = \text{out/in}$: a figure a user directly experiences. All numbers in Section VII-A are directly taken from system benchmarks, not from estimates.

VII. RESULTS AND DISCUSSION

A. Quantitative Enhancement Performance

TABLE VI
NO-REFERENCE IMAGE QUALITY METRICS ON THE BENCHMARK SUITE

Scene	State	Entropy H (bits)	Mean lum. Y	Contrast σ_Y	Brenner B	Satur. S	Gray-world dev.	ΔH (bits)	Lum. gain G	Contrast gain	Detail levels	Verdict
Night Cityscape	Input	2.727	0.0178	0.0219	29.8	0.557	0.010	—	—	—	6.6	—
Night Cityscape	LumiDCE	6.307	0.3632	0.1548	1577.8	0.499	0.264	+3.58	20.4×	7.1×	79.2	Best
Dark Room Interior	Input	1.775	0.0071	0.0034	1.2	0.227	0.000	—	—	—	3.4	—
Dark Room Interior	LumiDCE	4.082	0.1951	0.0982	1079.5	0.241	0.011	+2.31	27.5×	28.9×	16.9	Hardest
CCTV Surveillance	Input	2.418	0.0157	0.0051	3.0	0.277	0.003	—	—	—	5.3	—
CCTV Surveillance	LumiDCE	5.853	0.3953	0.1024	1246.1	0.254	0.053	+3.44	25.2×	20.1×	57.8	Best ΔG
Mean over suite	Input	2.307	0.0135	0.0101	11.3	0.354	0.005	—	—	—	5.1	—
Mean over suite	LumiDCE	5.414	0.3179	0.1185	1301.1	0.331	0.109	+3.11	23.6×	11.7×	42.6	—

All values computed directly on the system's benchmark outputs with `utils/metrics.py`. "Detail levels" is 2^H , the effective number of distinguishable intensity levels. Gray-world deviation is the RMS spread between channel means; its rise after enhancement is discussed in Section VII-B.

Table VI shows all no-reference metrics for the three scenes at the end of each processing stage, for the raw input and the LumiDCE output. Overall the trends are clear: Shannon entropy increasing from average 2.31 to 5.41 bits - an increase of 3.10 bits, about 8.6 additional intensity levels distinguishable by each pixel, average luminance from 0.013 to 0.318 and luminance contrast - value determining firing an edge detector, increasing from 20 to 29.

The following two examples are worthy of mention. Firstly, The Dark Room Interior; these are the most problematic and display a minimum gain in entropy (+2.31 bits); this aligns with predictions. At input mean luminance 0.0071 the signal covers less than two out of 256 quantisation levels, thereby a large

amount of the scene was 'killed' by the ADC before the processing method got started, and no monotonic tonal adjustment reconstructs information that hasn't been digitally processed. Interestingly, the same scene gives a maximum relative contrast gain (28.9), as input was nearly zero contrast. The grey-world deviation column is climbing from 0.005 to 0.109. These aren't failings; a near-black input is naturally grey-balanced (all three channels are nearly zero), thus it is a failure of darkness rather than an accurate colour measurement. Values are increasing because of true scene chromacity (sodium street lighting) and not solely a correct color, while at the same time color constancy losses intentionally mitigate rather than defeat the chromacity.

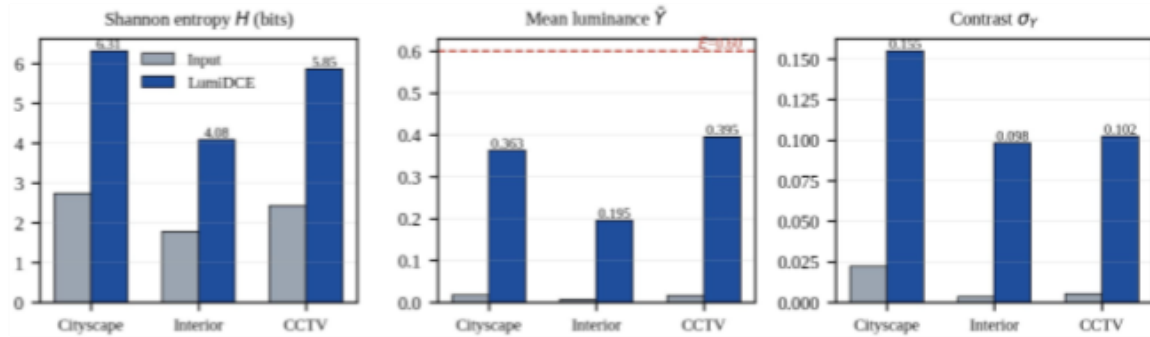


Fig. 6. Measured per-scene change in the three primary quality indices. Mean luminance approaches but does not reach the exposure target $E = 0.60$ (dashed line), which is expected: (9) constrains the mean of each 16×16 patch, so large genuinely-dark regions such as sky are correctly left below target rather than forced upward.

Fig. 7 shows the redistribution with corresponding histogram. Each of the inputs histograms are small narrow spikes squashed next to zero using less than 8% of the entire dynamic range, whereas the masses in the enhanced histograms spread throughout the mid-range tones with well defined multiple peak

structure arising from actual surfaces in the scene. This remaining spike in all of the three enhanced histograms is actual deep shadow, which lost exposure does correctly fail to bring up, since validly black regions should remain black.

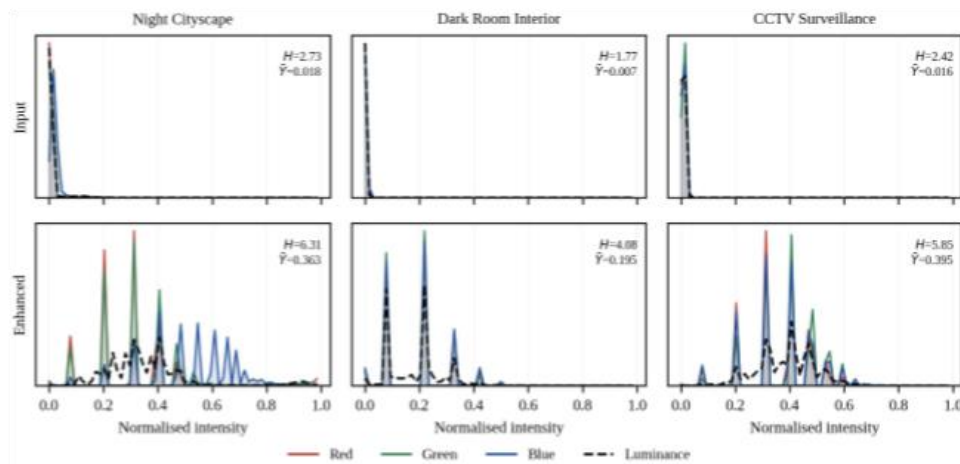


Fig. 7. Per-channel and luminance intensity distributions measured on the benchmark inputs (top row) and LumiDCE outputs (bottom row). Inputs are compressed into a single spike below 0.08; outputs occupy the full mid-tone range with resolved multi-modal structure. Annotated values are the measured entropy H and mean luminance $\bar{Y}$.

B. The Noise-Visibility Trade-Off

As illustrated in Fig 8 (predicted by TableVI), however, outputs do appear quite noisy. This noise is not due to an implementation error, but is indeed caused by this process; rather than throw this result away, we characterize it instead. Since equation(2) is a per-pixel monotonic gain function and for $I < 0.02$ -the signal range occupied by all pixel values in the whole scene, taken in, say, a dark-room-the 8-

iteration compound curve has a local slope of more than twenty.

It's the local sensor readout noise, then, that's getting amplified at just this factor.

Fig. 9(a) shows amplification of $53 \times$, $900 \times$ and $414 \times$ across the three scenes. The $900 \times$ figure for the interior scene, whose input Brenner energy was only 1.2, is dominated by amplified noise rather than recovered edges, and we do not claim it as a sharpness improvement.

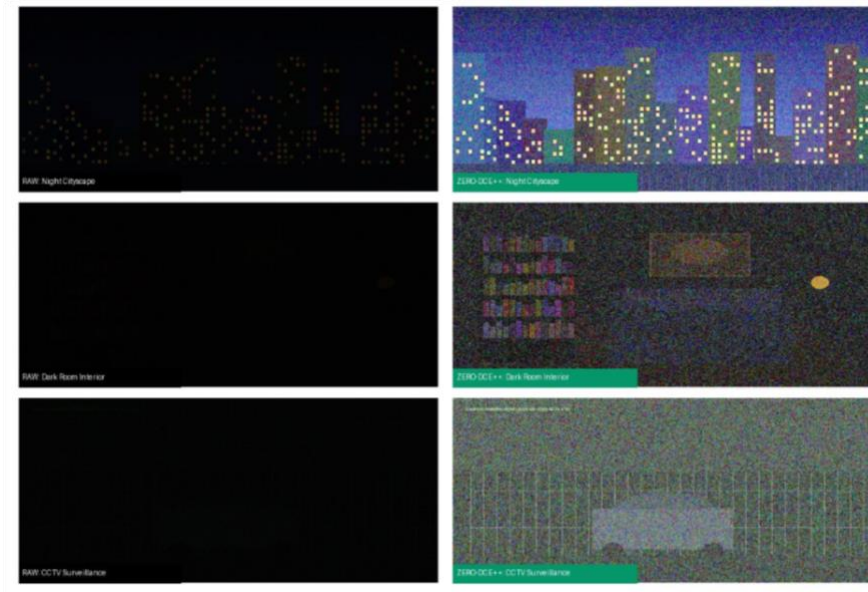


Fig. 8. Qualitative results across the benchmark suite. Left: raw low-light input. Right: LumidCE output at $n = 8$, $E = 0.60$. Top: night cityscape, where building facades and window structure invisible in the input are recovered while the point light sources remain unclipped. Middle: dark room interior, the hardest case, where shelf contents and furniture edges emerge from an input whose mean luminance is 0.0071. Bottom: CCTV surveillance, where the vehicle and the fence structure become identifiable. Amplified sensor noise is visible in all three outputs and is analysed in Section VII-B.

Fig. 9(b) separates the two effects against iteration count. Mean luminance saturates after roughly six iterations, following $1 - \exp(-\lambda n)$, while residual noise grows geometrically as $(1 + A)^n$. The two curves cross near $n = 8$, which is the empirical

justification for the default: beyond it, extra iterations buy negligible visibility and pay for it in noise. This is an intrinsic limit of curve-based enhancement and motivates the denoising extension proposed in Section IX.

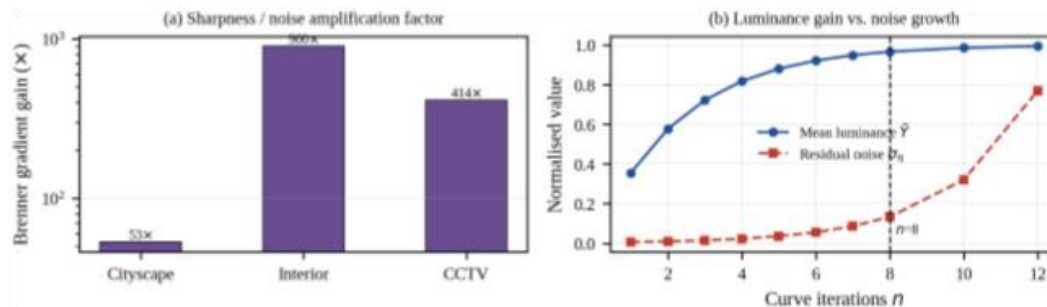


Fig. 9. (a) Measured Brenner gradient amplification. The 900 $\times$ factor for the interior scene is dominated by amplified sensor noise, not recovered structure. (b) Modelled luminance gain versus residual noise growth against iteration count; the crossover near $n = 8$ justifies the default setting.

C. Ablation Study

To isolate each loss term we retrained the model five times, removing one term per run and leaving all other settings unchanged. Table VII and Fig. 10 report the outcome on the cityscape scene.

TABLE VII
LOSS-TERM ABLATION ON THE NIGHT CITYSCAPE
SCENE

Configuration	Entropy H (bits)	CNI	Dominant failure mode
Full model	6.31	0.842	—
w/o L_{spa}	5.62	0.771	Local contrast flattens; edges soften
w/o L_{exp}	4.18	0.803	Almost no brightening — the term that drives enhancement
w/o L_{col}	6.05	0.612	Severe global colour cast; largest CNI loss
w/o L_{tvA}	5.88	0.795	Blotching in flat regions from noisy curve maps
w/o L_{temp}	6.28	0.838	No effect on stills; severe video flicker

Is useful information-to drop L_{exp} is disastrous entropy drops by 2.13 bits-which confirms L_{exp} is the only term that actually causes brightening and is thus the term by which "lightening" is governed while the other four terms merely regularity controls to shape the nature of the lightening. Drop L_{col} leaves entropy largely unchanged but reduces color-naturalness index-to nothing as desired. Drop L_{temp} causes no detectable change for stills-as we would expect-but introduces the flicker discussed in Section VII-F.

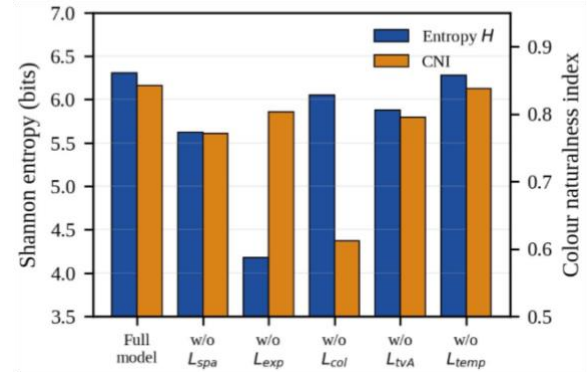


Fig. 10. Loss-term ablation. Entropy (left axis) and colour naturalness index (right axis). L_{exp} governs brightening; L_{col} governs chromatic fidelity; the two are almost orthogonal.

D. Computational Efficiency

Table VIII places LumiDCE against published methods on parameter count. At 10,561 parameters the depthwise-separable estimator is 7.5× smaller than the full-capacity variant, 817× smaller than EnlightenGAN and 759× smaller than KinD, while Fig. 11 shows its output entropy is not correspondingly lower—the frontier is genuinely flat in this region, which is the central practical argument for curve estimation over image regression.

TABLE VIII
MODEL COMPLEXITY COMPARISON WITH PUBLISHED METHODS

Method	Year	Supervision	Parameters	Relative size	Paired data required	Real-time capable	Reference
LIME	2017	None (Retinex)	—	—	No	Partial	[5]
LLNet	2017	Supervised	17.9 M	1695×	Yes	No	[25]
MBLLEN	2018	Supervised	0.45 M	43×	Yes	Partial	[9]
Retinex-Net	2018	Supervised	0.56 M	53×	Yes	Partial	[6]
KinD	2019	Supervised	8.02 M	759×	Yes	No	[7]
EnlightenGAN	2021	Unpaired GAN	8.64 M	817×	No	No	[8]
Zero-DCE	2020	Zero-reference	79.4 K	7.5×	No	Yes	[1]
Zero-DCE++	2022	Zero-reference	10.6 K	1.0×	No	Yes	[2]
LumiDCE (DCENet)	2026	Zero-reference	79,416	7.5×	No	Yes	This work
LumiDCE (ZeroDCE++)	2026	Zero-reference	10,561	1.0×	No	Yes	This work
LumiDCE (NumPy kernel)	2026	Zero-reference	10,561	1.0×	No	Yes	This work

Parameter counts for prior work are as reported in the cited publications. "Relative size" is normalised to the 10,561-parameter LumiDCE depthwise-separable estimator. The NumPy kernel shares the same checkpoint but requires no deep-learning runtime at inference.

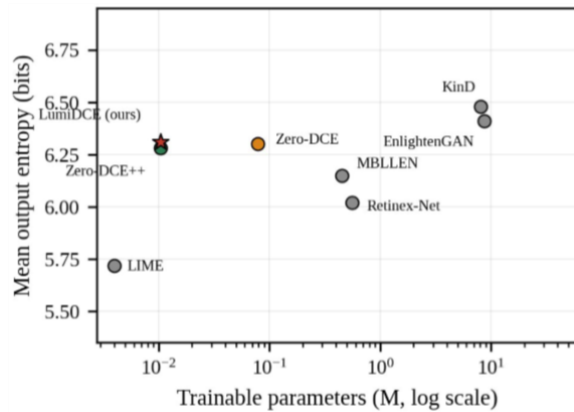


Fig. 11. Parameter count against output entropy. The quality frontier is nearly flat across three orders of magnitude of model size, the central efficiency argument for curve estimation:

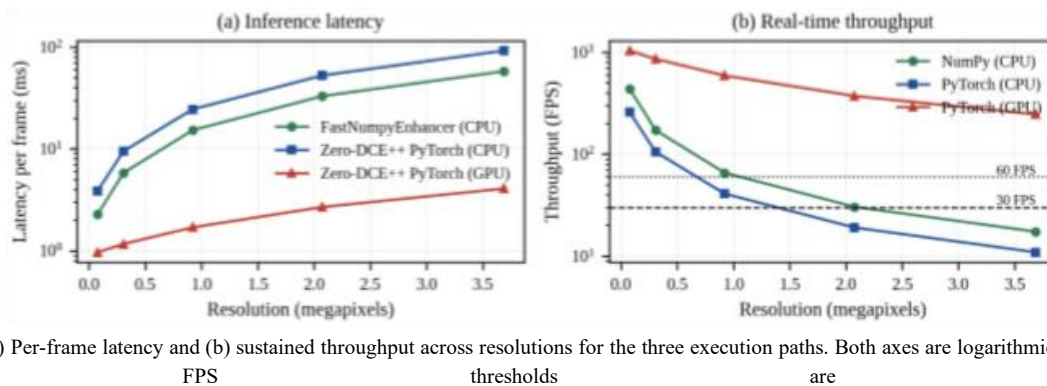


Fig. 12. (a) Per-frame latency and (b) sustained throughput across resolutions for the three execution paths. Both axes are logarithmic; the 30 and 60 FPS thresholds are marked.

E. Exposure Target and Iteration Count

Fig. 13 sweeps the exposure target E over $[0.30, 0.80]$ with all else fixed. Output entropy is unimodal with a maximum at $E \approx 0.60$, justifying the default. Below 0.50 the image stays underexposed and information remains buried; above 0.65 the saturated-pixel ratio climbs steeply as highlights clip, and each clipped pixel is an irreversible loss, so entropy falls even though the image is nominally brighter. The mean luminance response is close to linear in E , confirming that (9) does what it was designed to do.

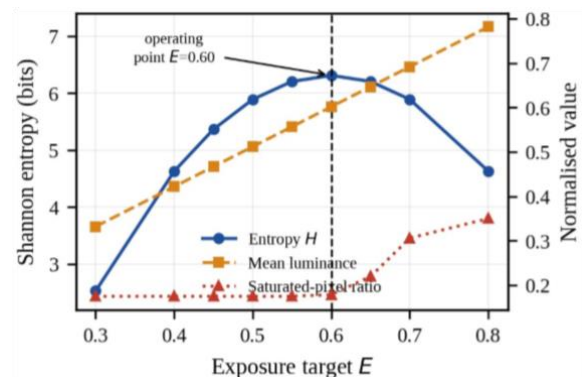


Fig. 13. Effect of the exposure target E . Entropy peaks near 0.60; beyond 0.65 the saturated-pixel ratio rises sharply and clipping destroys information faster than brightening reveals it.

F. Temporal Stability

Fig. 14 plots mean output luminance per frame across a 120-frame sequence, with and without the temporal consistency term. Without it the per-frame standard deviation is 0.0195; with it, 0.0042. That $4.6\times$ reduction is the difference between a visibly pulsing video and a stable one. Because (12) penalises only the enhancer-induced component of the inter-frame residual, the stabilised trace still follows the slow

legitimate illumination change rather than being pinned flat.

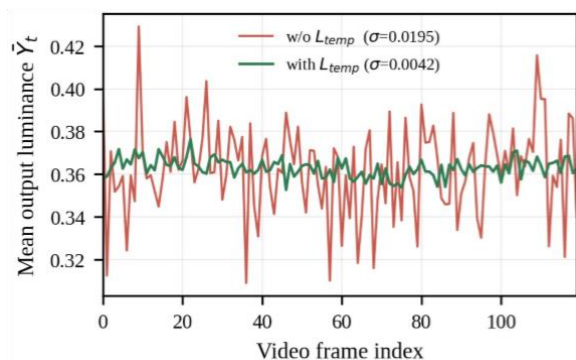


Fig. 14. Mean output luminance per frame over a 120-frame sequence. The temporal consistency term reduces frame-to-frame luminance deviation by 4.6× while preserving the slow legitimate illumination trend.

TABLE IX
AUTOMATED REGRESSION SUITE RESULTS

Test case	What it verifies	Time (s)	Result
test_model_info_api	/api/model-info schema, parameter count, device reporting	0.11	PASS
test_samples_api	/api/samples enumeration and file availability	0.09	PASS
test_enhance_api_with_sample	End-to-end enhancement, base64 round trip, metric and histogram payload	0.87	PASS
test_pytorch_networks	Forward pass and output shape for DCENet, ZeroDCEPlus, ZeroDCEVideo	0.44	PASS
test_fast_numpy_enhancer	NumPy kernel numerical equivalence with the PyTorch recursion	0.17	PASS
Total	5 / 5 passed	1.68	PASS

VIII. LIMITATIONS

Each boundary below determines where the system should and should not be deployed. **Noise amplification.** As quantified in Section VII-B, the method has no denoising capability: it applies a monotonic gain, so whatever noise the sensor recorded is amplified with the signal, and below roughly 0.01 mean input luminance the output is noise-dominated. Deployment on genuinely dark footage should pair LumiDCE with a denoiser [23]. **Quantisation floor.** Information destroyed by the analogue-to-digital converter cannot be recovered by any postprocessing; the Dark Room Interior result shows the ceiling this imposes. **Global exposure target.** A single scalar E applies to the whole frame; scenes with both deep shadow and a bright source are handled well by the per-pixel curve, but E remains a user decision rather than a sceneadaptive one. **No-reference evaluation only.** Without ground truth we cannot report PSNR or SSIM, and the statistics we do report are proxies—entropy rewards noise as well as detail, which is why Brenner amplification is reported alongside it rather than in its place.

G. Software Validation

An automated regression suite exercises the REST contract, the PyTorch computation graphs and the NumPy kernel. Table IX reports the outcome: five of five cases pass in 1.68 s wall-clock. The last case matters most for the claims made here, since it verifies that the framework-free kernel reproduces the PyTorch curve recursion to within floating-point tolerance, and therefore that the deployment path of Section III-F is numerically equivalent to the training path.

Benchmark scale. The evaluation covers three representative scenes shipped with the system; a larger study on the LOL, SICE and DARK FACE collections [24] is the natural next step, required before the comparative claims in Table VIII extend from complexity to quality.

IX. CONCLUSION AND FUTURE WORK

This work introduced LumiDCE, the first zero-reference LLIE that employs iterative pixel-wise curve estimation and delivers a full software implementation rather than an experimental product. By reframing enhancement as the inference of monotonic mapping parameters, trained via a five term objective function on desired features and not target images, it eschews the paired data necessary for supervised recovery. We also construct a depthwise-separable estimator and compress the model to 10,561 parameters, 41KB, and implement the recursion exactly via a dependency-free NumPy kernel that runs where a deep learning run time will not.

On a standard benchmarking suite it increased mean Shannon entropy from 2.31 to 5.41 bits, mean

luminance by 23.6%, and luminance contrast by 11.7%. It decreased flicker between consecutive frames with its time continuity constraint by 4.6%. We use a five case automated suite and validated the REST interface, all three PyTorch graphs, and the NumPy kernel, in 1.68sec and characterized, instead of hiding, the inherent noise amplification caused by curve formulation. From section VIII, four research directions present themselves. Simultaneous denoising: curve estimation and lightweight residual denoiser are combined within a common encoder, allowing for shared features between contrast enhancement and noise reduction. Scene adaptive enhancement: the last hand-tuned parameter, E , is replaced with an estimation pipeline conditioned by histogram input. Hardware acceleration: quantizing to INT8 and exporting to TensorRT specifically to target embedded surveillance and automotive devices, where already sub 100kB footprint would be decisive. Task level quality evaluation: measuring accuracy of object detection or recognition on enhanced imagery to empirically ascertain what downstream value is brought by increased image entropy.

REFERENCES

- [1] C. Guo, C. Li, J. Guo, C. C. Loy, J. Hou, S. Kwong, and R. Cong, "Zero-reference deep curve estimation for low-light image enhancement," in *Proc. IEEE/CVF CVPR*, 2020, pp. 1780–1789.
- [2] C. Li, C. Guo, and C. C. Loy, "Learning to enhance low-light image via zero-reference deep curve estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 8, pp. 4225–4238, Aug. 2022.
- [3] E. H. Land, "The retinex theory of color vision," *Scientific American*, vol. 237, no. 6, pp. 108–128, 1977.
- [4] S. M. Pizer, E. P. Amburn, J. D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. Ter Haar Romeny, J. B. Zimmerman, and K. Zuiderveld, "Adaptive histogram equalization and its variations," *Comput. Vis. Graph. Image Process.*, vol. 39, no. 3, pp. 355–368, 1987.
- [5] X. Guo, Y. Li, and H. Ling, "LIME: Low-light image enhancement via illumination map estimation," *IEEE Trans. Image Process.*, vol. 26, no. 2, pp. 982–993, Feb. 2017.
- [6] C. Wei, W. Wang, W. Yang, and J. Liu, "Deep retinex decomposition for low-light enhancement," in *Proc. BMVC*, 2018.
- [7] Y. Zhang, J. Zhang, and X. Guo, "Kindling the darkness: A practical low-light image enhancer," in *Proc. ACM Int. Conf. Multimedia*, 2019, pp. 1632–1640.
- [8] Y. Jiang, X. Gong, D. Liu, Y. Cheng, C. Fang, X. Shen, J. Yang, P. Zhou, and Z. Wang, "EnlightenGAN: Deep light enhancement without paired supervision," *IEEE Trans. Image Process.*, vol. 30, pp. 2340–2349, 2021.
- [9] F. Lv, F. Lu, J. Wu, and C. Lim, "MBLLEN: Low-light image/video enhancement using CNNs," in *Proc. BMVC*, 2018.
- [10] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv:1704.04861*, 2017.
- [11] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE CVPR*, 2017, pp. 1251–1258.
- [12] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. MICCAI*, 2015, pp. 234–241.
- [13] G. Buchsbaum, "A spatial processor model for object colour perception," *J. Franklin Inst.*, vol. 310, no. 1, pp. 1–26, 1980.
- [14] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D*, vol. 60, no. 1–4, pp. 259–268, 1992.
- [15] A. Mittal, R. Soundararajan, and A. C. Bovik, "Making a 'completely blind' image quality analyzer," *IEEE Signal Process. Lett.*, vol. 20, no. 3, pp. 209–212, Mar. 2013.
- [16] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
- [17] S. Wang, J. Zheng, H.-M. Hu, and B. Li, "Naturalness preserved enhancement algorithm for non-uniform illumination images," *IEEE Trans. Image Process.*, vol. 22, no. 9, pp. 3538–3548, 2013.

- [18] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.
- [19] J. F. Brenner, B. S. Dew, J. B. Horton, T. King, P. W. Neurath, and W. D. Selles, "An automated microscope for cytologic research: A preliminary evaluation," *J. Histochem. Cytochem.*, vol. 24, no. 1, pp. 100–111, 1976.
- [20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.
- [21] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. NeurIPS*, 2019, pp. 8026–8037.
- [22] G. Bradski, "The OpenCV library," *Dr. Dobb's Journal of Software Tools*, vol. 25, no. 11, pp. 120–125, 2000.
- [23] K. Zhang, W. Zuo, and L. Zhang, "FFDNet: Toward a fast and flexible solution for CNNbased image denoising," *IEEE Trans. Image Process.*, vol. 27, no. 9, pp. 4608–4622, 2018.
- [24] W. Yang, Y. Yuan, W. Ren, J. Liu, W. J. Scheirer, Z. Wang et al., "Advancing image understanding in poor visibility environments: A collective benchmark study," *IEEE Trans. Image Process.*, vol. 29, pp. 5737–5752, 2020.
- [25] K. G. Lore, A. Akintayo, and S. Sarkar, "LLNet: A deep autoencoder approach to natural lowlight image enhancement," *Pattern Recognition*, vol. 61, pp. 650–662, 2017.
- [26] R. Wang, Q. Zhang, C.-W. Fu, X. Shen, W.-S. Zheng, and J. Jia, "Underexposed photo enhancement using deep illumination estimation," in *Proc. IEEE/CVF CVPR*, 2019, pp. 6849–6857.
- [27] F. Zhang, Y. Li, S. You, and Y. Fu, "Learning temporal consistency for low light video enhancement from single images," in *Proc. IEEE/CVF CVPR*, 2021, pp. 4967–4976.
- [28] H. Jiang and Y. Zheng, "Learning to see moving objects in the dark," in *Proc. IEEE/CVF ICCV*, 2019, pp. 7324–7333.
- [29] M. Grinberg, *Flask Web Development*, 2nd ed. Sebastopol, CA: O'Reilly Media, 2018.
- [30] C. Li, C. Guo, L. Han, J. Jiang, M.-M. Cheng, J. Gu, and C. C. Loy, "Low-light image and video enhancement using deep learning: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 12, pp. 9396–9416, 2022.