


Article

TM-Net: A Neural Net Architecture for Tone Mapping

Graham Finlayson[†] and Jake McVey^{*,†} 

School of Computing Sciences, University of East Anglia, Norwich Research Park, Norwich NR4 7TJ, UK

* Correspondence: j.mcvay@uea.ac.uk

† These authors contributed equally to this work.

Abstract: Tone mapping functions are applied to images to compress the dynamic range of an image, to make image details more conspicuous, and most importantly, to produce a pleasing reproduction. Contrast Limited Histogram Equalization (CLHE) is one of the simplest and most widely deployed tone mapping algorithms. CLHE works by iteratively refining an input histogram (to meet certain conditions) until convergence, then the cumulative histogram of the result is used to define the tone map that is used to enhance the image. This paper makes three contributions. First, we show that CLHE can be exactly formulated as a deep tone mapping neural network (which we call the TM-Net). The TM-Net has as many layers as there are refinements in CLHE (i.e., 60+ layers since CLHE can take up to 60 refinements to converge). Second, we show that we can train a fixed 2-layer TM-Net to compute CLHE, thereby making CLHE up to 30× faster to compute. Thirdly, we take a more complex tone-mapper (that uses quadratic programming) and show that it too can also be implemented — without loss of visual accuracy—using a bespoke trained 2-layer TM-Net. Experiments on a large corpus of 40,000+ images validate our methods.

Keywords: Histogram Equalization; Contrast Limited Histogram Equalization; tone mapping; contrast enhancement; unrolling

**Citation:** Finlayson, G.; McVey, J.TM-Net: A Neural Net Architecture for Tone Mapping. *J. Imaging* **2022**, *8*, 325. <https://doi.org/10.3390/jimaging8120325>

Academic Editor: Edoardo Provenzi

Received: 17 October 2022

Accepted: 7 December 2022

Published: 12 December 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Contrast enhancement plays a very important role in an image processing pipeline. There are many reasons why an image may have poor contrast including incorrect exposure setting, the problem of dynamic range compression [1], and that preferred image reproductions typically have more contrast than the original physical scene [2]. Often we can improve an image's contrast by simply darkening or brightening the image. Indeed, if an image appears dark then all the pixels must have small values and so, by definition, the contrast (or difference) between pixels must also be small. Increasing the image brightnesses effectively stretches the image histogram, which in turn leads to the average difference between pixels to increase. We can also increase contrast by, analogously, stretching the range of image brightnesses when the image histogram is predominantly skewed toward larger pixel values. Informally, contrast is said to be enhanced when detail that is hard to see in an input image is made more conspicuous in the reproduction.

A single function is applied to all pixel values in global contrast/tone enhancement, and spatially varying functions are applied in local tone adjustments [3–5]. The goal of contrast enhancement algorithms is to bring out more detail in an image, but this has to be balanced with the requirement to not introduce any artefacts. Moreover, if image contrast is increased too much then the image will not be preferred. It is important not to over enhance the image.

In this paper, we will only consider global tone mapping functions (although we will, briefly, discuss a tile-based local extension). The images used in this work are already-rendered SDR images (as opposed to HDR images that typically require more-significant enhancement [6]), and we enhance the image contrast to make the details more conspicuous. For our purposes we assume image brightnesses are in the interval [0, 1] and our tone

mapping functions are defined by a function $H()$ where $H : [0, 1], \rightarrow [0, 1]$. The functions $H()$ are increasing functions of brightness.

Tone mapping functions can be defined as a parametric adjustment e.g., $H(a) = \min(ka, 1)$, where k is a positive scalar. Other useful parametric forms of tone adjustment include the Naka Rushton function[7], the Michaelis-Menten Equations [8] and gamma adjustment algorithms [9–11]. However, the focus of this paper is non-parametric contrast adjustment. Here, the tone mapping functions are defined by input-output brightness pairs—that are to be mapped exactly—and a suitable interpolation function, e.g., [12].

Perhaps the most well-known non-parametric contrast enhancement algorithm is the venerable Histogram Equalization (HE) [13] algorithm. In **HE**, the cumulative histogram—of the input brightness histogram that is normalized so it sums to 1 (i.e., it is a probability density)—is used, directly, to define the tone map that takes the input to output pixel intensities. The tone map is the cumulative histogram of the probability density function. As the name suggests, after applying **HE** the output image has—more or less—a brightness histogram with uniform (equal) values.

From a conspicuity of detail point of view, an **HE** tone mapped image is in two senses optimal. First, when an image has a uniform brightness histogram and we compute the average absolute brightness difference (i.e., contrast) between random pairs of pixels then this average is maximized. Second, images that have a uniform brightness histogram have greater entropy/information[14] (i.e., they must take more bits to encode).

In Figure 1a, we show an example input image. The brightness histogram for the image is shown (as the solid blue line) in Figure 1b. In Figure 1c, the same image enhanced with **HE**. Clearly, the background of the **HE** enhanced image now has unnatural contrast and most of the details on the doorknob have been lost. These problems with the reproduction are understandable when we look at the shape of the tone map/curve—remember, for **HE**, defined to be the cumulative brightness histogram—mapping input pixel values to output counterparts. In the interval $[0.3, 0.4]$ the tone curve is very steep and conversely the brightness detail is overly stretched (directly leading to the unnatural contrast in the tone mapped door). The detail is overly compressed—the tone curve is flat—in the interval $[0.5, 1]$ (and the detail on the doorknob is reduced). The problems of overly stretching contrast and feature loss (due to compression of the tonal range) are commonly encountered when **HE** is used as a tone mapping algorithm.

A key property of **HE** is the relationship between the probability density function (the brightness histogram normalized to sum to 1) of an input image and the tone curve used for enhancement. The latter is the cumulative histogram—equally, the integral—of the former.

Now, returning to our example in Figure 1, suppose $H()$ is the tone mapping function—from **HE**—that resulted in a poor image reproduction. There were areas where contrast was too high and others where it was too low. Let us suppose there exists a *better* tone mapping function, $G()$. From our previous discussion, by differentiating the $G(a)$ we can return a density function $g(a)$. It turns out that finding $G(a)$ can be usefully expressed as the problem of finding $g(a)$ given $h(a)$. The aim is to find a $g(a)$ —as a proxy for $h(a)$ —that would integrate to a better tone map.

Let us choose $g(a)$ to be close to $h(a)$, but where its individual densities are neither too large nor too small. It follows that the corresponding tone curve—defined to be the integral of the density function—must have a bounded slope. This intuition is the central idea underpinning Contrast Limited Histogram Equalization (**CLHE**) [3], a widely deployed tone mapping algorithm. Importantly, **CLHE** also underpins the Apical Iridix tone mapper [15] that is embedded in many cameras and smartphones.

In Figure 1b, the red dotted line shows a probability density close to the original (solid blue line) but has a bounded density (in this example it is between 0.02 and 0.005, where the brightness range is divided into 100 bins). The corresponding cumulative histogram is shown in Figure 1f. Notice how the maximum slope is reduced and the minimum slope increased. Finally, in Figure 1e, we show the **CLHE** output. Here we have good detail

throughout the image and a pleasing reproduction where the image detail is rendered to be more conspicuous.

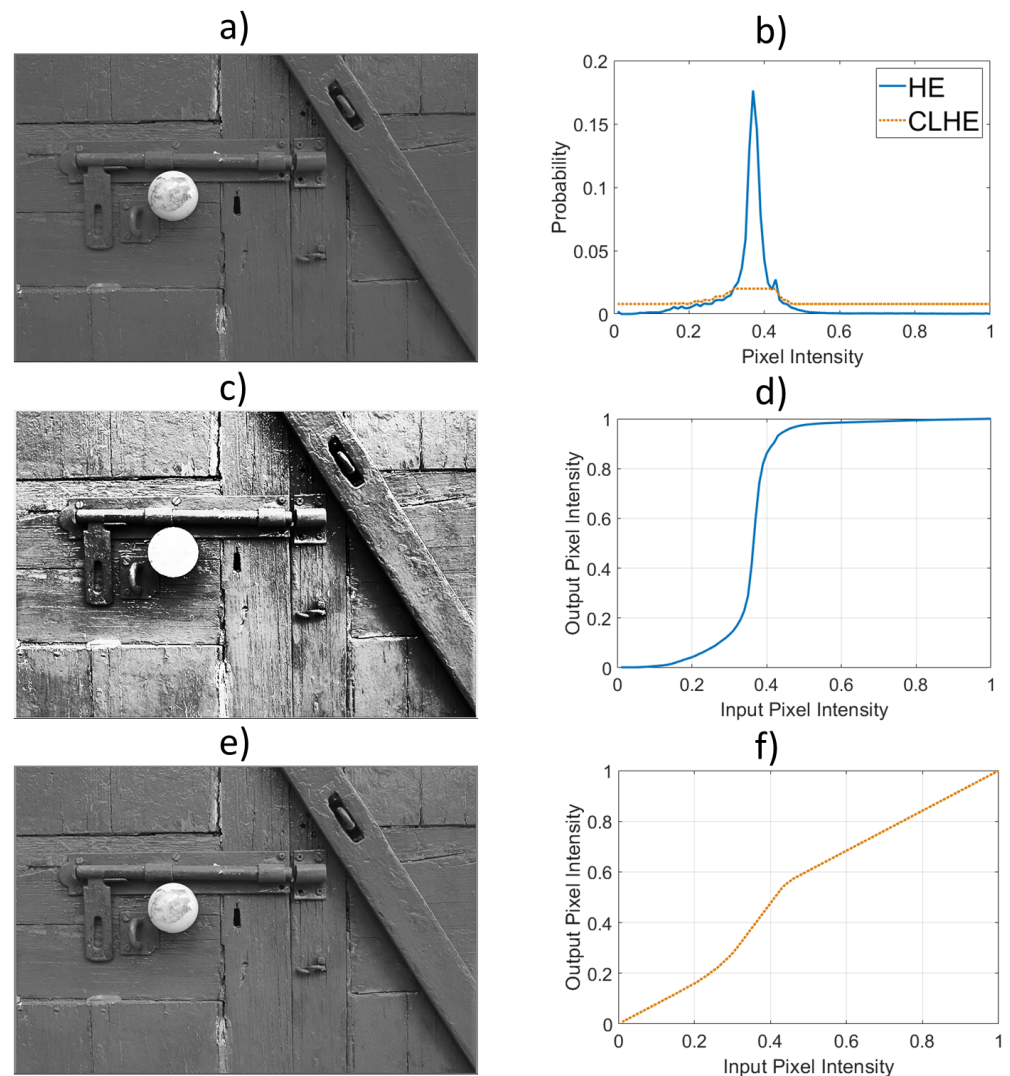


Figure 1. An illustration of HE and CLHE. (a) Original image. (b) Brightness histogram of (a) (solid blue), and CLHE proxy histogram (dotted red). (c) Image enhanced with HE. (d) HE tone curve. (e) Image enhanced with CLHE. (f) CLHE tone curve.

The first contribution of this paper is to show that each iteration (refinement) of the CLHE algorithm can be represented as a special tone mapping computational block in a neural network (where the internal structure—the connections—of the block are derived in Section 3). Of course unrolling iterative algorithms by means of a Neural Network is an established approach to accelerating iterative algorithms [16]. The important distinction here is that representing CLHE as a computational block lets us pre-define the parameters of the network (as opposed to random initialization). It will be shown that a deep net of a sufficient number of repeating layers exactly computes the CLHE histogram modification. We call the deep net that comprises repeating tone mapping layers a Tone Mapping Network, or a TM-Net for short. We illustrate the idea of unrolling an iterative histogram refinement algorithm by an equivalent TM-net architecture, in Figure 2.

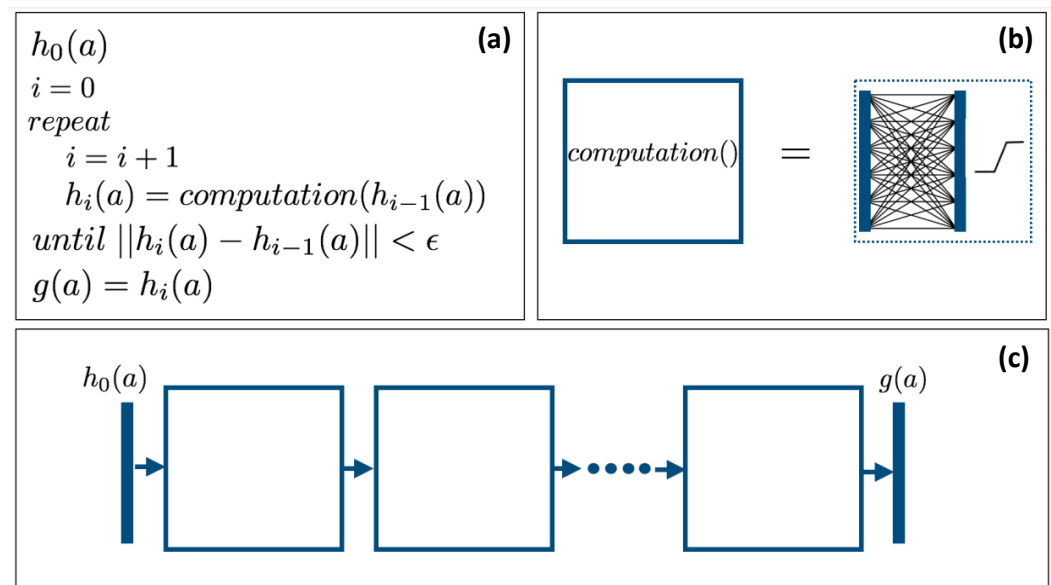


Figure 2. In panel (a), pseudo code representing the **CLHE** algorithm. In panel (b), the computation function can be represented as a single layer neural net block. In panel (c), by replicating the computational block enough times, we can calculate **CLHE** using a neural net.

In Figure 2a, we show the high level view of the **CLHE** algorithm that maps an input brightness density, $h_0(a)$ to the output density $g(a)$, where ultimately, the tone map will be defined as the integral of the latter.

We describe the function $\text{computation}()$ in the next section and its implementation as a neural computational block in Section 3. In each iteration, the density $h_i(a)$ is refined from $h_{i-1}(a)$. The iteration terminates when a convergence criterion is met.

On the right hand side of the Figure 2b, the solid blue vertical bars denote input and output densities (e.g., vectors of numbers). Inputs and outputs are connected by a network of connections. In the usual neural network way, each connection line takes an input value (one bin of the input histogram/density) and multiplies it by a weight. Per bin on the output layer we sum across all the connections (that input to that bin). The network connections shown here are for illustration only. The precise connectivity follows from the **CLHE** algorithm (and this is discussed in detail in Section 3). In the usual neural net way, the output layer feeds through an activation function, here piece-wise linear. Importantly, the piece-wise linearity follows exactly from how **CLHE** enforces height constraints on the probability densities.

So long as the network we define is built with a *sufficient* number of computational layers—as illustrated in Figure 2c—then the resulting deep net calculates **exactly** the same output as **CLHE**. The architecture shown in Figure 2 is called a tone mapping network, or **TM-Net** for short. Importantly—and unlike typical networks—our unrolling of **CLHE** here ensures the layers of the **TM-Net** are interpretable.

Of course, all we have done at this stage is re-represent an existing computation in a different form. In this paper we will also use our **TM-Net** architecture to make **CLHE** faster to compute, and also to compute tone maps that make images that are preferred to **CLHE**. In the second contribution of this paper, we replace our 60+ repeating block architecture with a simple 2-layer **TM-Net**. However, now, rather than adopting the default **CLHE** weights, we relearn them in the usual deep learning way (by Stochastic Gradient Descent [17]). The loss is determined by the difference between the 2-layer **TM-Net** output and the iterated-to-convergence **CLHE** output. We therefore seek—in just 2 layers—to compute the same output histogram as **CLHE** computes (in the worse case 60+ iterations). Effectively reducing the number of iterations allows us to overcome one of the primary drawbacks of **CLHE**, it is unpredictable and often significant computation time [18,19].

Of course, there are many other algorithms that determine a tone map dependent on the histogram of an image. So, in our third contribution we aimed to see whether our TM-Net could be used to learn non-CLHE tone maps. Here, we chose to investigate the algorithm of Arici et al. [20] both because it produces pleasant images, (often preferred to CLHE) and also because it has quite a complex formalism (including, in its general form, using quadratic programming). We found that we can retrain the TM-net to learn the outputs computed by [20], and in so doing make that method simpler to implement and faster to execute. Importantly, our TM-Net we define here is used to closely approximate (and speed up the execution for) existing algorithms, that is, we are not designing new tone curve algorithms here.

In Section 2 of this paper, we review the related background work. Our TM-Net is derived and presented in Section 3. In Section 4 we show—based on a large corpus of images—that a 2-layer TM-Net can be used to implement CLHE (which by default can take 60+ iterations to converge) thereby making it a much faster algorithm. We also demonstrate that the more complex tone mapper (from Arici et al) can also be implemented as a TM-Net. The paper concludes in Section 5.

2. Background

The Universal Approximation Theorem informs us that a neural network can be used to approximate any continuous function [21]. We can usefully think about the individual layers in a neural network by using the compact vectorized form described in Equation (1). We refer the reader to [22] for a more detailed derivation of each component, but for now let us borrow the notation. The network layer is expressed as:

$$\mathbf{a}^l = \sigma(w^l \mathbf{a}^{l-1} + \mathbf{b}^l). \quad (1)$$

where the output of layer l in the network is $\mathbf{a}^l \in \mathbb{R}^N$. $\sigma()$ is an activation function, and the weight matrix that scales the contribution of each neuron in the previous layer is $w^l \in \mathbb{R}^{N \times N}$. The output of the previous layer is denoted $\mathbf{a}^{l-1} \in \mathbb{R}^N$, and $\mathbf{b}^l \in \mathbb{R}^N$ denotes the bias value vector (one scalar bias for each neuron in layer l).

2.1. CLHE: Contrast Limited Histogram Equalization

Let us denote a scalar brightness image, $\mathbf{I}(x, y)$ and we assume that that image brightnesses are the interval $[0, 1]$. In the discrete domain, we denote the N -bin intensity histogram as $\mathbf{h} \in \mathbb{R}^N$. In the discussion that follows we will switch between using the histogram vector \mathbf{h} and the corresponding continuous representation $h(a)$ where appropriate (to make the exposition easier to follow). Here and henceforth, we assume that histograms are normalized to sum to one. So \mathbf{h} is a vector that sums to 1.

In simple Histogram Equalization (HE) an input image is mapped, using a tone mapping function, to an output counterpart so that the new image has an approximately uniform histogram (see Figure 1). In HE the tone map used is simply the cumulative histogram (actually the cumulative density since all our histograms sum to 1). In the discrete domain, a tone map can also be thought of as an N -vector, $\mathbf{H} \in \mathbb{R}^N$. By construction, each element of \mathbf{H} is also in the range $[0, 1]$. To visualize the tone-mapping, we can plot the HE tone curve, \mathbf{H} , against $(\frac{1}{N}, \frac{2}{N}, \dots, \frac{N-1}{N}, 1)$, e.g., see Figure 1d.

As introduced in the introduction, the tone curve in HE is the cumulative histogram sum, or equivalently in the continuous domain:

$$H(a) = \int h(a) da \quad (2)$$

where $H(0) = h(0)$ (a Dirichelet boundary condition to fix the constant of integration).

It follows that:

$$\frac{\delta}{\delta a} H(a) = h(a) \quad (3)$$

The equivalent corresponding vector calculation relating the tone map to image histogram:

$$\begin{aligned} H_1 &= h_1 \\ H_a &= \sum_{i=1}^a h_i \\ a &\in \{1, \dots, N\} \end{aligned} \quad (4)$$

Additionally, discretely, we can differentiate the tone curve to yield a brightness histogram:

$$\begin{aligned} h_a &= H_a - H_{a-1} & a > 1 \\ h_1 &= H_1 & a = 1 \end{aligned} \quad (5)$$

As for the continuous case, we understand that the slope of the tone map is proportional to the height of the histogram. In the discrete domain we must multiply the discrete difference in Equation (5) by the sample distance, here $\frac{1}{N}$. Where we remember that the image brightness a are in the interval $[0,1]$ and h_i records the frequency of brightnesses between $\frac{i-1}{N}$ and $\frac{i}{N}$. It follows that the slope of the tone curve is equal to

$$\text{slope}(a) = \frac{H_a - H_{a-1}}{1/N} \quad (6)$$

From which it follows that:

$$\frac{\text{slope}(a)}{N} = h_a \quad (7)$$

The importance of Equations (6) and (7) is that it makes clear how the height of a histogram relates to the slope of the corresponding tone curve (cumulative histogram). For a tone curve to have a slope greater than 0.5 and less than 2, then each bin of the histogram should be bounded:

$$\frac{0.5}{N} \leq h_a \leq \frac{2}{N} \quad (8)$$

In discussion around **CLHE**, the terms $\frac{0.5}{N}$ and $\frac{2}{N}$ are called ‘clip limits’. Though, generally, the link to the slope of the tone curve is often not made clear. We denote the limits $L = \frac{0.5}{N}$ and $U = \frac{2}{N}$ the **Lower** and **Upper** slopes.

In **CLHE**, for a given brightness histogram \mathbf{h} we calculate a proxy histogram, \mathbf{g} , where the bins of \mathbf{g} are close to that of \mathbf{h} , but the bins also meet the user defined slope limits. **CLHE** finds the proxy histogram according to an iterative 2-step algorithm (effectively we define the *computation()* function alluded to in Figure 2a,b).

In step 1, we ‘clip’ a given histogram so that it meets the **Lower** and **Upper** slope limits:

$$\hat{\mathbf{h}} = \min(\max(\mathbf{h}, L), U) \quad (9)$$

where the *min* and *max* functions are applied to each element of \mathbf{h} .

However, since our definition of a tone map is a cumulative histogram, it is important that the clipped histogram (which meets the slope constraints) integrates to 1. Thus, in a second step, we add a constant Δ to every bin so that the resulting histogram sums to one (and its cumulative sum can be used as a tone curve).

$$\mathbf{h} = \hat{\mathbf{h}} + \Delta \quad (10)$$

where,

$$\Delta = \frac{1 - \sum_{k=1}^N \hat{h}_k}{N} \quad (11)$$

This second step is sometimes referred to as a ‘redistribution step’. Now that we have added the Δ it is possible the new histogram, once again, does not meet the clip limits. So we iterate and step through Equations (9) through (11) (and repeat again until we arrive at a final histogram that meets the clip limits and sums to 1).

We remark that step 1 and step 2 both find histograms close to their inputs. For a given histogram the closest counterpart—in a least-squares sense—that adheres to the slope

limits is found by clipping, Equation (9). Similarly, the closest histogram (which sums to 1) to an input, in a least-squares sense, is found via Equations (10) and (11). While each individual step in **CLHE** is least-squares optimal they, it turns out, applied in iteration are not guaranteed to return an overall least-squares optimal solution [23]. However, generally, the **CLHE** histogram is very close to the optimal slope constrained density histogram.

The **CLHE** method is written as pseudo code in Algorithm 1. In Figure 3, we present a toy example of the **CLHE** computation. In our toy example, **CLHE** converges quickly. However, in our experiments we have found, in the worst case, it can take as many as 60 iterations to converge.

Algorithm 1 CLHE Algorithm

```

1: input brightness histogram:  $\mathbf{h}_0$ 
2:  $i = 0$ 
3: repeat
4:    $i = i + 1$ 
5:    $\hat{\mathbf{h}}_i = \min(\max(\mathbf{h}_{i-1}, \mathbf{L}), \mathbf{U})$ 
6:    $\Delta = \frac{1 - \sum_{k=1}^N \hat{\mathbf{h}}_i}{N}$ 
7:    $\mathbf{h}_i = \hat{\mathbf{h}}_i + \Delta$ 
8: until  $\|\mathbf{h}_i - \mathbf{h}_{i-1}\| < \epsilon$ 
9: output histogram:  $\mathbf{g} = \mathbf{h}_i$ 

```

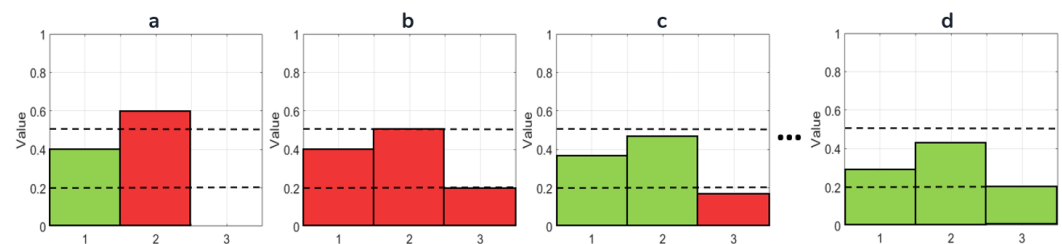


Figure 3. An illustration of **CLHE**. Dashed lines represent the slope bounds (for upper and lower slopes of 1.5 and 0.6). (a) Input histogram [0.4, 0.6, 0]. Sums to 1, but bins 2 and 3 do not obey slope constraints. (b) Histogram clipped to slope limits. All bins obey slope limit, but sum of histogram is 1.1, $\Delta = -0.1$ (c) Histogram with Δ evenly distributed (-0.33 to all bins). The result sums to 1, but bin 3 does not obey slope constraint and will be clipped again to 0.2. (d) Histogram after 5 clip and redistribution steps. Final values [0.35, 0.45, 0.2] satisfy slope bounds and sum to 1.

2.2. HMF: A Histogram Modification Framework

Any global tone mapping algorithm can be thought of as a histogram adjustment (like **CLHE**). Indeed, the derivative of a global tone map can be thought of a proxy histogram that encodes a brightness distribution useful for tone mapping. In **CLHE** the proxy—by construction—is close to the original histogram. However, it need not be. For example, a tuneable log-based encoding of image luminance has been used for tone mapping [24] with good results. Equally, optimization methods can be brought to generate a tone map that preserves the relationship between brightnesses in the (non tone-mapped) image [25]. Most relevant to this paper is the Histogram Modification Framework (**HMF**) by Arici et al [20] which, in effect, is a clever extension of **CLHE**. There, a proxy of a brightness histogram is found that has additional features that ensure the image reproduction (found from the cumulative histogram of the proxy) is preferable, such as presenting with ‘good’ whites and blacks.

The proxy histogram in **HMF** is found as the solution to an optimization problem with several weighted penalty terms. What is important for our purpose here is that the objective function in Equation (12) is—in its general form—solved using Quadratic

Programming (QP), and the values for the penalty terms (λ , γ , and α) are fixed (e.g., see the values suggested in the original work [20]).

$$J = \min_{\mathbf{g}} \|\mathbf{g} - \mathbf{h}\|_2^2 + \lambda \|\mathbf{g} - \mathbf{u}\|_2^2 + \gamma \|D\mathbf{g}\|_2^2 + \alpha \|S\mathbf{g}\|_2^2 \quad (12)$$

The first term of Equation (12) ensures that—like **CLHE**—the solved-for proxy histogram (\mathbf{g}) should be close to the original histogram (\mathbf{h}). The second term conditions (\mathbf{g}) to be similar to the uniform histogram, $\mathbf{u} = [\frac{1}{N}, \frac{1}{N}, \dots, \frac{1}{N}]^T$. When integrated to form a tone map, \mathbf{u} is a 45 degree line that maps each input to the same output, thus preventing any contrast enhancement. So, as \mathbf{g} moves closer to \mathbf{u} , the level of contrast enhancement in the final reproduction is reduced.

The third term conditions \mathbf{g} to be smooth. Here, D represents the discrete derivative operation (and so $D\mathbf{g}$ is the second derivative of the tone curve). This term can be usefully interpreted as a constraint on the ‘wiggly-ness’ of the tone curve. As the penalty term γ increases, the wiggly-ness of the tone curve decreases.

Finally, the term $S\mathbf{g}$ ensures that the darkest and brightest intensities of the image are mapped to darker and brighter intensities in the reproduction respectively. For this reason this term is sometimes referred to as ‘black and white stretching’. Mapping the darkest and brightest pixels in this way ensures that no unwanted details are introduced at either end of the brightness range.

In this work, we enforce some additional constraints in a modified optimization in Equation (13). First, our solved-for \mathbf{g} is a probability density and thus each element should be in the range $[0, 1]$ and the sum of \mathbf{g} should be 1. Next, as in **CLHE**, we would like the derived histogram to integrate to a tone map with bounded slope. We bound the height of the densities accordingly.

$$\min_{\mathbf{g}} J \text{ where } \mathbf{L} \leq \mathbf{g} \leq \mathbf{U} \text{ \& } \sum_{k=1}^N g_k = 1 \quad (13)$$

Equation (13) is also straightforward to implement in Quadratic programming.

2.3. Local Tone Mapping

CLHE—and indeed any global tone mapping method—can be applied locally in images. In **CLAHE** (Contrast Limited Adaptive Histogram Equalization) [3], an image is first separated into discrete non-overlapping tiles, and a histogram is calculated for each tile. Each tile histogram induces a local tone curve, and the per-pixel mapping is found using bilinear interpolation between the 4 nearest tiles (tone curves) to each pixel.

Other variations of local tone mapping include the sliding window approach originally presented in [26], where a per-pixel tone map is calculated using a predetermined number of the surrounding pixels. However, as images grow large this method becomes computationally expensive, making the tiled approximation approach preferable.

2.4. Global Tone Mapping

Global tone mapping remains an active topic of interest, e.g., see [27–29]. A global tone curve cannot be used to specifically enhance local contrast [30], however they can be simply implemented locally using the methods discussed in Section 2.3.

Spatially varying algorithms like bilateral filtering [31], **CLAHE** [3], **Retinex** [32], all aim to use local information to improve an image. However, often, the outputs calculated by these local algorithms can be well approximated by a global tone curve [33,34].

2.5. Image Enhancement with a Tone Curve

In this work, we present input and enhanced images in RGB space; however, all image manipulations occur in CIE $L^*a^*b^*$ space. First, the input RGB image is converted to $L^*a^*b^*$ and the brightness histogram calculated from the lightness channel (L^*). The tone curve

is then obtained (from the histogram) and applied to the lightness image to obtain the contrast enhanced version (L'^*).

Analogous to saturation preservation in CIE $L^*u^*v^*$ [35], we also aim to preserve the colour saturation of the enhanced images by normalizing the input chroma by the ratio of change in lightness as:

$$\frac{C_{ab}^*}{\hat{L}^*} = \frac{\sqrt{a^{*2} + b^{*2}}}{\hat{L}^*}. \quad (14)$$

where \hat{L}^* is calculated as $\frac{L'^*}{L^*}$. The a^* and b^* channels of the input $L^*a^*b^*$ image are first divided by \hat{L}^* , before the image is converted back to RGB to obtain the final output image.

3. Method

The starting point of this paper is to re-cast the **CLHE** algorithm into a form that we call a Tone Mapping Neural Network or TM-Net for short. Abstractly, the TM-Net implements each iteration of the **CLHE** algorithm as a single layer in a network computation, see Figure 2. Crucially, the network architecture and all the weights are defined by the **CLHE** algorithm (at this stage there is no learning). By building a deep net by repeating these layers we make an architecture that exactly calculates **CLHE**.

By translating an iterative algorithm to an equivalent deep net, we can ask new and novel questions. First, can we improve the speed efficiency of **CLHE**? Specifically, can we use a TM-Net with 2 layers to learn the **CLHE** computation? Secondly, can we use the TM-Net to learn other histogram-based tone-mappers?

3.1. The TM-Net

In Equation (1), we summarized the key components of a Neural Network layer. The 4 components include 2 known and 2 unknown variables. The known variables are the output of the previous layer in the network (\mathbf{a}^{l-1}), and the activation function ($\sigma()$). The unknown variables are the weight matrix (\mathbf{w}^l) and bias vector (\mathbf{b}^l). Training a Neural Network means that we need to solve for \mathbf{w}^l and \mathbf{b}^l that map the input vector to a desired output (\mathbf{a}^l). In this section, we marry **CLHE** and Equation (1), i.e., we derive the matrix, bias vector, and activation function that—when used in Equation (1)—would generate an output vector that exactly matches one iteration of **CLHE**.

We start with the observation that the clipping step of **CLHE** (Equation (9)) is analogous to the role of an activation function in a Neural Network and so the transcription is natural. Indeed, a simple piece-wise linear function [36] is illustrated in Figure 4a that computes the clip in Equation (9) and step 4 of the **CLHE** algorithm. The function is linear on the interval $[L, U]$, and flat everywhere else. Here, L and U denote the lower and upper slope bounds (clip limits) respectively. When this function is applied to the histogram in Figure 4b we find the ‘clipped’ histogram in Figure 4c.

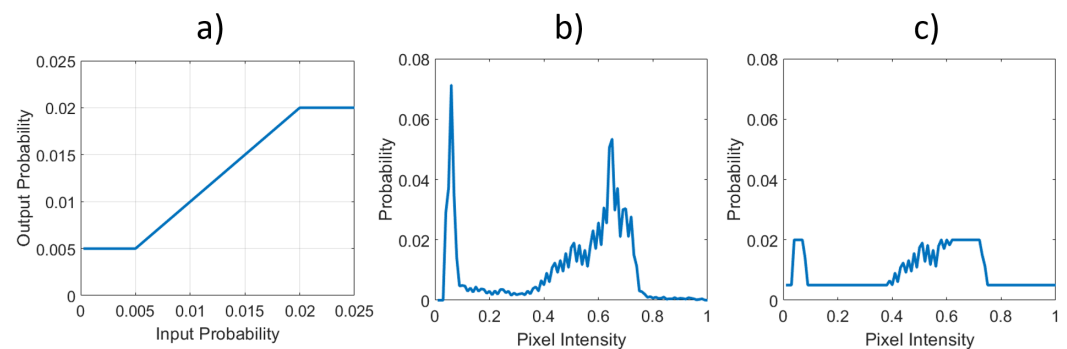


Figure 4. Visualising the piece-wise linear activation function’s impact on a discrete histogram vector. (a) A linear function between 0.005 and 0.02 (that clips to the boundary values for inputs outside this range). (b) An arbitrary histogram. (c) The histogram from (b) after it has been modified by (a).

The redistribution step of **CLHE** is visualized in Figure 5. We will conceptualize the redistribution process as two separate steps. First, we consider how to add an offset so a histogram sums to zero, Figure 5b. Given a zero-mean histogram, we can add the offset ($\frac{1}{N}$ to each bin) in Figure 5c to make a histogram that sums to 1. This two step view makes the redistribution step simple to write in matrix form.

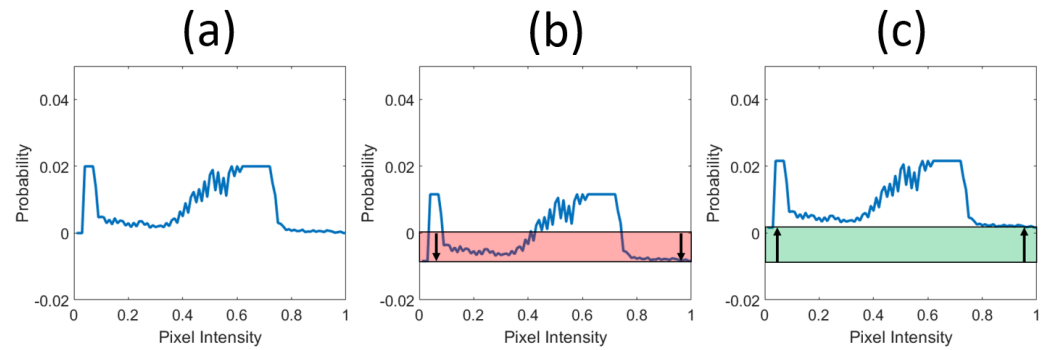


Figure 5. Manipulating a histogram (with N bins) until it sums to 1. (a) A histogram. (b) Histogram with all bins evenly decremented until it sums to 0. (c) All bins incremented by $\frac{1}{N}$. The histogram now sums to 1.

Let us define the $N \times 1$ vectors \mathbf{v} and \mathbf{w} where $v_i = 1$ and $w_i = \frac{1}{N}$ ($i = 1, 2, \dots, N$). Remembering that our N -bin histogram is denoted \mathbf{h} and denoting the $N \times N$ identity matrix by \mathcal{I} , we calculate:

$$\mathbf{h}^0 = [\mathcal{I} - \mathbf{v}\mathbf{w}^T]\mathbf{h} \quad (15)$$

which, effectively, adds the same value to each component of \mathbf{h} so that the resulting histogram sums to 0 (we use the superscript 0 to denote the histogram has a zero mean). Now we add the second offset. We define a fixed N -vector \mathbf{b} where each component $b_i = \frac{1}{N}$.

$$\mathbf{h}^1 = \mathbf{h}^0 + \mathbf{b} \quad (16)$$

Using the piece-wise linear function from Figure 4 as $\sigma()$, we can now express the histogram found by $i + 1$ iterations of **CLHE**, \mathbf{h}_{i+1} , as:

$$\mathbf{h}_{i+1} = [\mathcal{I} - \mathbf{v}\mathbf{w}^T]\sigma(\mathbf{h}_i) + \mathbf{b} \quad (17)$$

Returning to Equation (1) we are ready to transcribe the **CLHE** into a network computation. For $\sigma()$ we use the piece-wise linear function in Figure 4 with bounds at the desired clip limits. The w^l weight matrix is calculated as $\mathcal{I} - \mathbf{v}\mathbf{w}^T$ as defined above. Additionally, \mathbf{b}^l is a vector that holds $\frac{1}{N}$ in each element.

Our transposition from matrix equation to a Neural Network layer is further illustrated for a 5-bin example in Figure 6. Weighted connections between bins are represented as arrows (initialized to the shown values). From left to right, the input to the network first passes through the piece-wise linear clipping function. Next, the dashed orange and green lines represent multiplication of the input by \mathbf{v} and \mathbf{w}^T respectively, which is fed into the output layer. The input layer also feeds directly into the output layer, and—along with the bias vector (\mathbf{b})—the sum of all components define the output. This output—depending on the network architecture—is either fed into the activation function again, or is the network output.

Figure 6 shows that each iteration of **CLHE** can be thought of as a standard type of neural net computational block, where the input histogram is mapped to an output version by a matrix operation plus a bias, that is then rectified by a piece-wise linear function. Clearly, if we repeat the block structure enough times then the network must, by construction, calculate the same answer as **CLHE**. That is, if the conventional **CLHE** algorithm takes m steps to converge then the same answer can be found by replicating m layers of the form shown in Figure 6. In our experiments, we found **CLHE** always

converged in 60 iterations or fewer (mostly in less than 20 iterations). So, we need a 60 layer deep TM-Net to guarantee the same result as CLHE (to numerical precision). We call our repeating block architecture a Tone-Mapping Network or TM-Net for short.

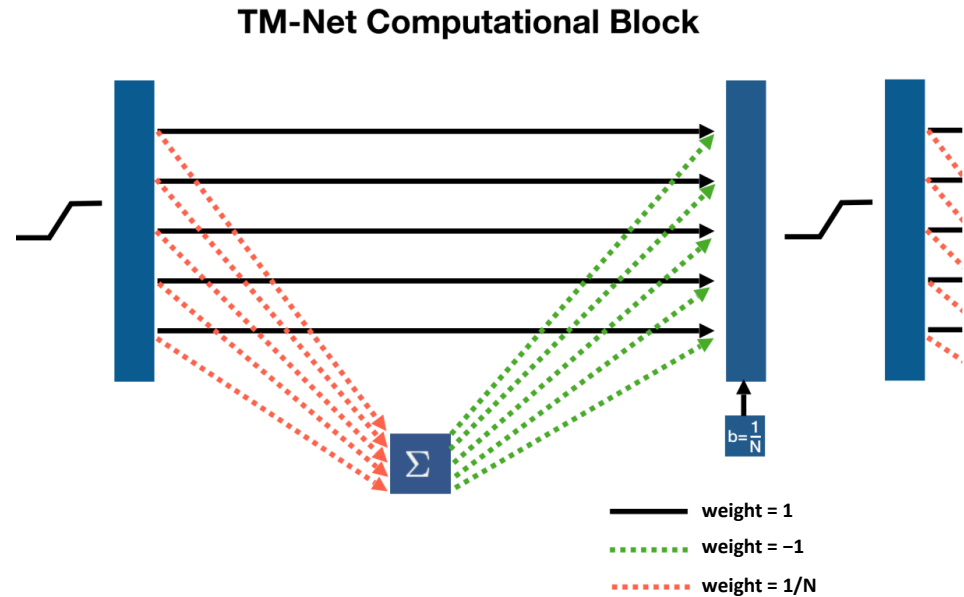


Figure 6. Graphical representation of a single layer in an N layer Tone-Mapping network(TM-Net). In this example, $N = 5$.

Finally, we note that, as the block is drawn, it actually resembles a residual neural network (albeit a very simple one). The actual computation is a simple summation followed by a propagation of the sum (follow the red and green dotted arrows). Then we add in the results from a previous layer.

3.2. Relearning CLHE

We hypothesize that a small network can be initialized (fewer layers than needed for CLHE convergence) to use a truncated TM-Net to learn the result of the fully convergent CLHE operation. Our hypothesis is that a truncated TM-Net with learned weights (we do not use the defaults that follow from the CLHE algorithm) will be able to learn the CLHE computation.

To be more concrete, suppose we have a network that has m layers built as described in the last section, and let us denote the output of a TM-Net as $TM(\mathbf{h})$. For the i th histogram in a dataset the network computes $TM(\mathbf{h}_i)$. We would like the network to produce histograms that are similar to the CLHE computation, $CLHE(\mathbf{h}_i)$, i.e., we would like $TM(\mathbf{h}_i) \approx CLHE(\mathbf{h}_i)$.

Suppose we use m computational layers of the form shown in Figure 6. In the k th (of m) computational block we have to learn $3N$ unknowns which we denote \mathbf{v}_k , \mathbf{w}_k and \mathbf{b}_k . Grouping the complete set of all the unknown \mathbf{v} 's, \mathbf{w} 's and \mathbf{b} 's as the $N \times m$ matrices V , W and B (each column of each matrix is respectively \mathbf{v} , \mathbf{w} and \mathbf{b}). Then for an m -layer network we need to minimize

$$\mathcal{J} = \min_{V, W, B} \sum_i ||CLHE(\mathbf{h}_i) - TM(\mathbf{h}_i)||^2 \quad (18)$$

where the Equation (18) is a simple least-squares *loss* function. It is implicit that to minimize Equation (18) that we need to find V , W and B (the network parameters) that makes the squared error as small as possible. In fact we would also like to place some constraints on these hidden variables. Given that \mathbf{v}_i —a set of network weights—is weighting a histogram, and a histogram has a natural order (bins range from a brightness level of 0 to 1 in increasing

order) we would like \mathbf{v}_i to be smooth in some sense. The intuition here is that we do not expect the k th weight w_k^l to be significantly different from w_{k+1}^l . That is, the weights and bias vectors, \mathbf{v}^l , \mathbf{w}^l , \mathbf{b}^l , (usefully visualised as functions plotted on a graph) should be smooth. Here we define the smoothness of all the parameters as

$$\mathcal{S} = (||DV||^2 + ||DW||^2 + ||DB||^2) \quad (19)$$

where D is the $N \times N$ linear operator that calculates the discrete derivative (of a column vector), e.g., see [20]. In Equation (20) we set forth the final form of our minimization (where λ is a user defined parameter weighting the importance of the smoothness of the network parameters).

$$\min \mathcal{J} + \lambda \mathcal{S}. \quad (20)$$

The weights and biases—underpinning the optimization in Equation (20)—can be found by Stochastic Gradient Descent (SGD) implemented using back propagation (BP) algorithm. The details of the SGD and BP algorithms (e.g., [17]) are not important here. What is important is we can optimize Equation (20) efficiently using standard techniques. By minimizing Equation (20) we relearn **CLHE**.

3.3. Training the Truncated TM-Net (Implementation Details)

We use a TM-Net with as few as two layers to approximate **CLHE**. The network was trained using 30,000 images randomly samples from the ImageNet dataset [37]. From each training image, we obtain the input histogram and the **CLHE** histogram (the network does not see the full image, only the histograms). These histograms serve as the input and target of the network respectively. The network was trained using the regularized mean squared error loss function (in Equation (20)) with $\lambda = 1 \times 10^{-4}$ and the following hyper parameters: batch size = 30,000, learning rate = 1×10^{-4} , and epochs = 500. The machine used for training uses PyTorch and an Intel i7 CPU and an NVIDIA RTX 2070S. Each training epoch resolved in 0.75 s for a total training time of 6 min.

As we will show in the experiments section, we also train networks (with the same hyper parameters) with more than 2 layers.

3.4. Cost of Learning and Applying a Tone Map

Our method—like **CLHE**—generates a tone curve from the histogram of brightnesses in an image. To build this histogram we need to visit each image pixel once. Once we have calculated the tone curve then to apply this curve we again need to visit each pixel once. Thus the cost of **CLHE** and our variant is proportional to the number of pixels in the image. However, the actual cost of calculating the tone curve is constant (independent of the size of the image).

4. Experiments

We evaluate the performance of our TM-Net by comparing the closeness of images enhanced with the TM-Net against two target algorithms: **CLHE** [3] and the Histogram Modification Framework (**HMF**) [20].

Effective comparison of images is a challenging problem [38]. Here, we use the ΔE distance metric to quantify closeness of the enhanced images. This was a deliberate choice because the tone curves used in this work are applied globally to each image. We also ran tests using the SSIM (structural similarity measure) and PSNR (peak signal to noise ratio) and found the results to follow the same trend as the ΔE errors (and so are not reported here). That said we include, for completeness, SSIM and PSNR results for the most challenging dataset.

To calculate ΔE error statistics, each test image is enhanced with the target algorithm, and then enhanced independently with the TM-Net tone curve. Each enhanced image is then converted to the CIE $L^*a^*b^*$ colour-space, and the per-pixel difference of respective channels in each image is calculated to generate an error image. From this error image here,

we calculate the mean, median, and 99-percentile error statistics for the input image, and do this for all images in the test datasets.

To be precise, the mean statistic is the mean of the individual means calculated per image. Similarly our median statistic is the median of the median errors again calculated per image. Similarly, for a given image we can calculate the 99-percentile CIE $L^*a^*b^*$ ΔE error. Then over a data set we can calculate the 99-percentile of the 99-percentile errors.

In our experiments, we will use four test datasets in this work. The well-known Kodak dataset [39] that contains 24 RGB images comprises our first image set. Remembering that 30,000 images from Image Net is used to train our TM-NET, the remaining 20,000 RGB images (not used for training) are used as our second image test set. [37]. In our third image set, we use 20,000 RGB images from the MIT Places database [40]. Our 4th image set draws challenging images from the first three datasets. Challenging examples are images that take more than 45 iterations for CLHE to converge.

4.1. Approximating CLHE: Contrast Limited Histogram Equalization

Here we wish to use our TM-NET to approximate CLHE. We will compare the outputs of the TM-Net according to our mean, median and 99-percentile statistics over our four image sets. In all cases, we found that the TM-Net always well approximated CLHE in three or fewer layers (and so we will only consider these approximations here).

In Figure 7, from left to right, we plot the mean of the mean, median of the median, and 99-percentile of the 99-percentile of ΔE for each of the test datasets used in this work, as a function of the number of layers in the TM-Net. We see that while the error in all instances is not 0, it is indeed close to 0 and, naturally, becomes closer to 0 as the number of layers in the TM-Net increases. A common heuristic approach to determining the point of diminishing returns is to identify the ‘elbow’ of the error curve. For all examples in the figure the elbow occurs at two layers.

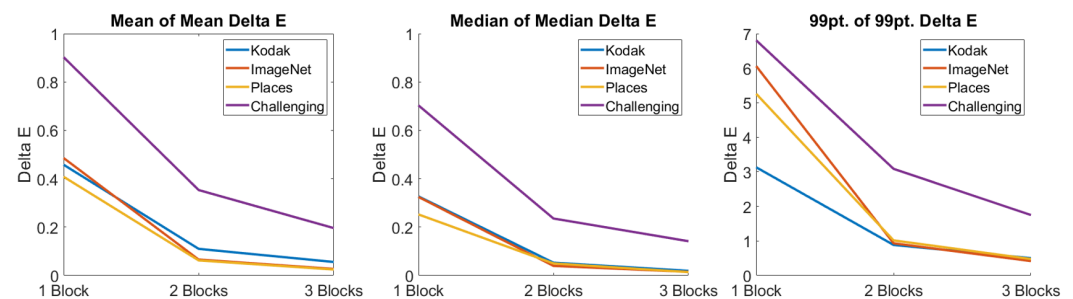


Figure 7. Change in ΔE as the number of blocks in the CLHE TM-Net increases. Left, mean of mean ΔE . Middle, median of median ΔE . Right, 99-percentile of 99-percentile ΔE .

Moreover, in complex images (e.g., photographic pictures like the ones used in this work) an average ΔE —where the error is distributed throughout the image—of between 3 and 5 [41,42] is generally thought to be visually not noticeable. For our data, this—like the elbow—points to a 2-layer TM-Net sufficing.

In the first three results columns of Table 1, we compare the performance of a 2-layer TM-Net against fully converged CLHE. Notice the mean of the mean ΔE is close to zero (visually indistinguishable in most cases). Significantly, for the most challenging images, the 99 percentile error is just 3.09 which is visually not significant for complex images. The final column in the table reports the 99 percentile ΔE error for the 2-iteration CLHE (hrer, we manually limit the permitted number of iterations to 2). Notice that for the first 3 rows/datasets the errors for 2-iteration CLHE are fairly low (though higher than the 2-Layer TM-Net approximation). This is because in most cases CLHE converges in few iterations. However, for the challenging dataset, the 99 percentile ΔE error is 5.81 (significantly larger than the TM-Net approximation and visually noticeable).

Table 1. ΔE (\pm standard deviation) between the 2-layer TM-Net and CLHE. The final column compares the 2-layer TM-Net to 2-iteration CLHE.

2-Layer TM-Net	Mean ΔE	Median ΔE	99pt. ΔE	2-Iteration CLHE 99pt. ΔE
Kodak	0.11 (± 0.04)	0.05 (± 0.03)	0.88 (± 0.31)	1.3 (± 0.41)
ImgNet	0.07 (± 0.02)	0.04 (± 0.03)	0.93 (± 0.28)	1.42 (± 0.36)
Places	0.06 (± 0.05)	0.05 (± 0.03)	1.02 (± 0.23)	1.14 (± 0.52)
Chall.	0.35 (± 0.12)	0.24 (± 0.19)	3.09 (± 0.61)	5.81 (± 1.13)

In Figure 8, we present an example of an output generated using an image drawn from the challenging dataset. We compare the output of the fully converged CLHE with the 2-layer TM-Net. As expected from the error stats presented, the outputs are visually identical. For comparison, we also run CLHE but terminate it, arbitrarily, after two iterations. Here, there is a significant residual difference. Pay close attention the image outputs corresponding to the input region bounded by the red rectangle.

In Figure 8 (right), we show the input histogram and the modified histograms for the three algorithms. We see that the TM-Net output is almost identical to CLHE iterating to convergence. However, the 2-iteration CLHE has a noticeable delta.

Additionally, we highlight the differences between the 2-layer TM-Net (left) and 2-iteration CLHE (right) outputs with a heat-map of ΔE in Figure 9. The colours in the figure moving from blue to yellow represent increasing difference between the output images from the target image. The erroneous pixels for the limited CLHE error image cluster around the diver. The mean of the mean ΔE error for the TM-Net and limited CLHE outputs are low, 1.1 and 1.9, respectively. While the 99-percentile of the 99-percentile ΔE error tells us the same respective difference is 2.6 and 3.9. Clearly, the TM-Net output is much closer to the target.

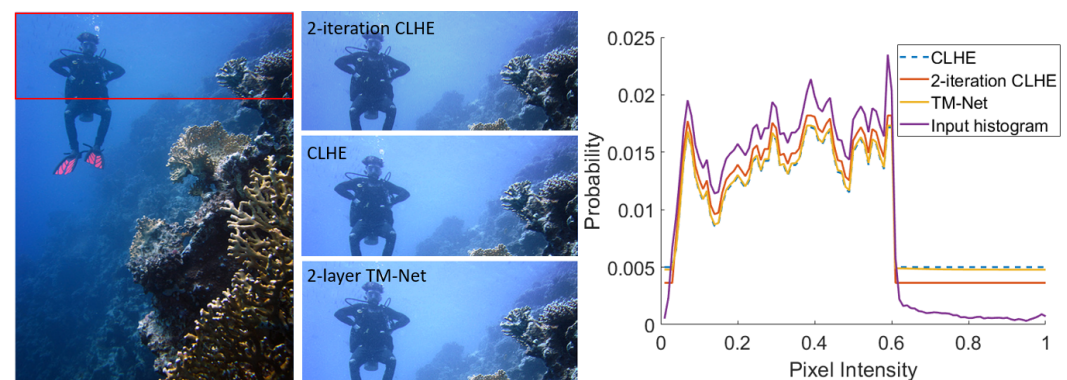


Figure 8. Left, a standard RGB image. Middle, the highlighted section enhanced with CLHE, a 2-layer TM-Net, and CLHE limited to 2 iterations. Right, input and modified histograms found by each method.

Finally, in Figure 10 we show several images from the Kodak dataset (left) enhanced with full-iteration CLHE (middle) and the 2-layer TM-Net (right). The ΔE for each image in the set is shown in the top right. The mean ΔE error is close to 0 for all images and there are close to zero noticeable differences even under very close observation.

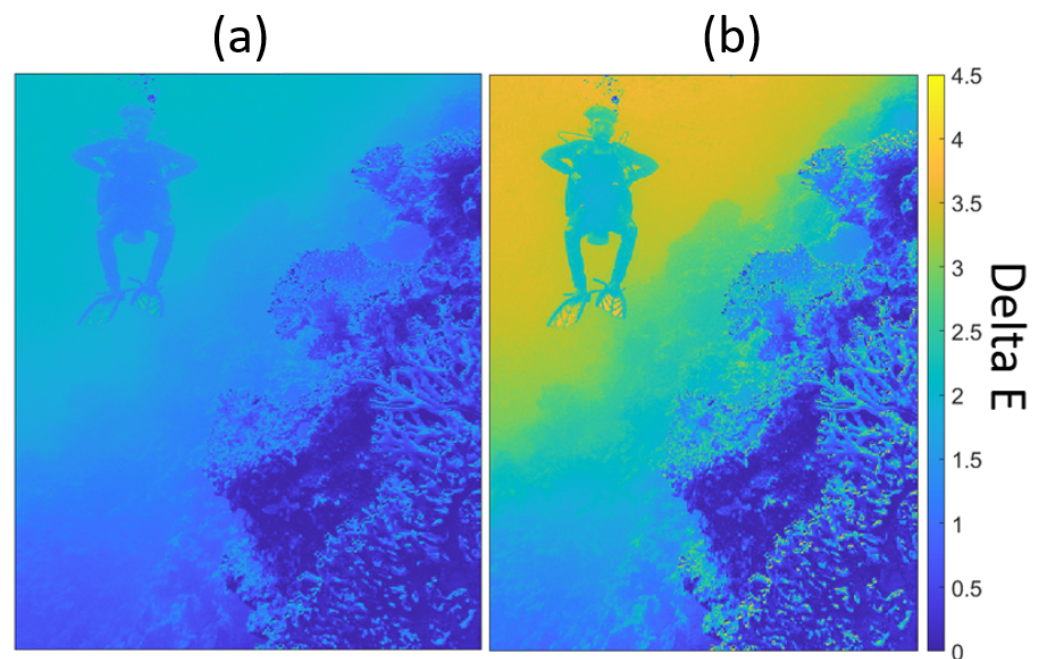


Figure 9. Error image for (a) 2-layer TM-Net. (b) CLHE limited to 2 iterations.

4.2. HMF: A Histogram Modification Framework

The original Quadratic Programming (QP) based implementation of **HMF** is complex (see Equation (12)). Here, we seek to reformulate the algorithm into the proposed TM-Net form. Essentially, we ask if this more complex algorithm be made as simple as **CLHE**. To retrain the TM-Net, we use the same architecture and training dataset as the **CLHE** TM-Net, and extract from each image the original and HMF modified histograms. As before, we can summarize our network with Equation (18)—replacing $CLHE(h_i)$ with $HMF(h_i)$ —and also including the smoothing constraints on the parameters of Equations (19) and (20). As before, we initialize the parameters (weights) of the network to the **CLHE** weights.

In Table 2, we compare the performance of our bespoke 2-layer TM-Net against the Histogram Modification Framework [20]. Clearly, the ΔE is not zero, but the numbers are small. As for approximating **CLHE**, a 2-layer TM-Net suffices to approximate the **HMF** framework.

Table 2. Mean (\pm standard deviation) of the mean, median, and 99-percentile of ΔE for enhanced images compared to **HMF**, averaged over each image in the datasets.

2-Layer TM-Net	Mean ΔE	Med. ΔE	99pt ΔE
Kodak	1.54 (± 0.38)	1.49 (± 0.39)	3.46 (± 0.81)
ImageNet	1.65 (± 0.41)	1.27 (± 0.33)	3.88 (± 0.79)
Places	1.37 (± 0.35)	1.10 (± 0.29)	3.74 (± 0.83)
Chall.	1.98 (± 0.52)	1.95 (± 0.37)	4.02 (± 0.91)

4.3. PSNR and SSIM

We present Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) statistics for the challenging images dataset in Table 3. To obtain these statistics, we used MATLAB's respective `ssim` and `psnr` functions to compare the TM-Net reproductions against the respective **CLHE** and **HMF** reproductions. The final row in the table compares the **CLHE** TM-Net against 2-iteration **CLHE**. Mean PSNR and SSIM results are shown and their standard deviations.

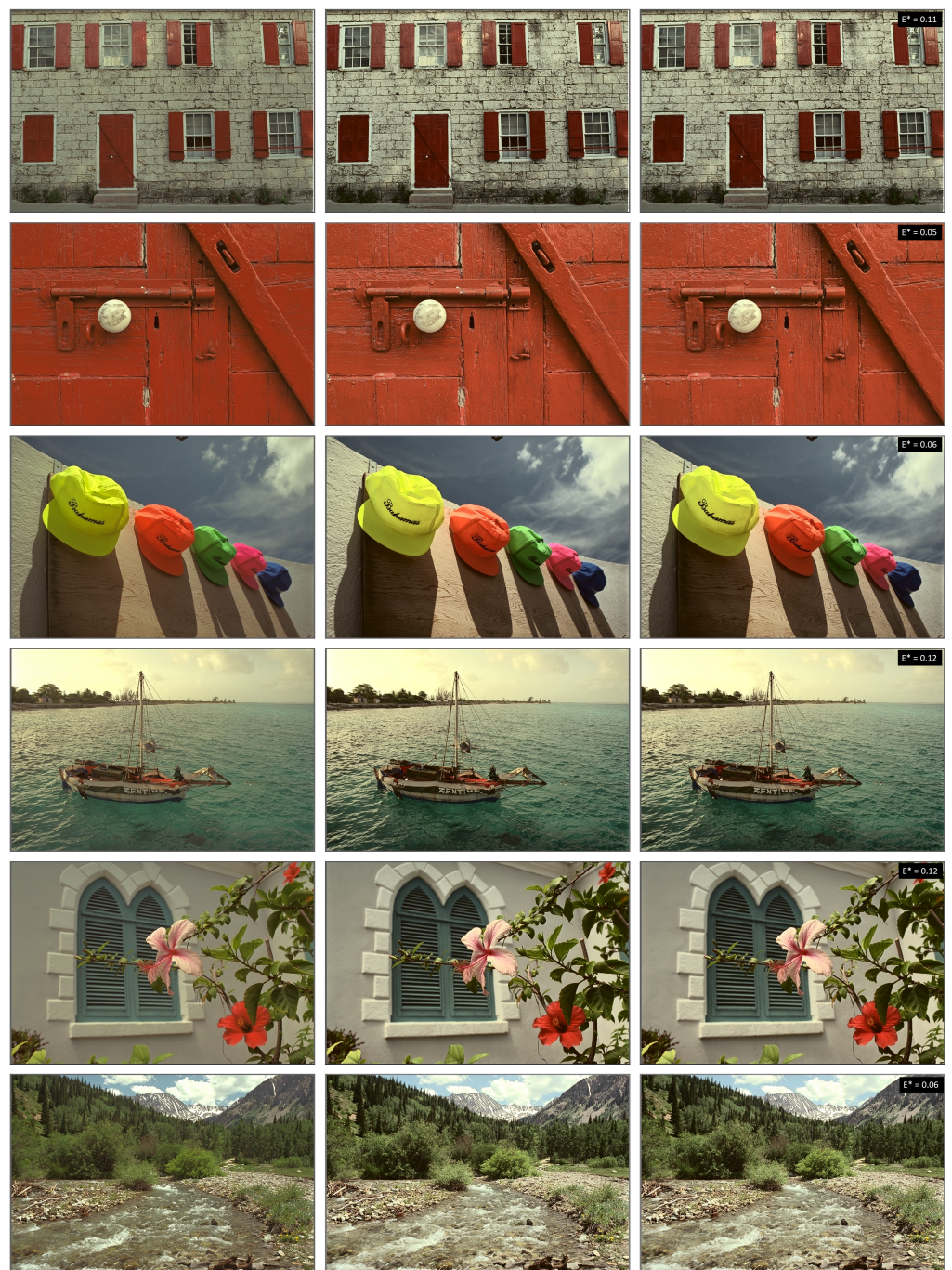


Figure 10. For each image in the set: First, original image. Middle, image enhanced with CLHE. Right, image enhanced with a 2-layer TM-Net. Mean of mean ΔE between the enhanced images are shown in top right of each image.

Table 3. Mean (\pm standard deviation) PSNR and SSIM statistics for both networks using the Challenging Images dataset.

	PSNR	SSIM
2-Layer TM-Net CLHE	47.52 (± 8.79)	0.97 (± 0.17)
2-Layer TM-Net HMF	32 (± 2)	0.99 (± 0)
2-iteration CLHE	38.13 (± 10.17)	0.95 (± 0.23)

For the 2-layer TM-Net approximation to **CLHE**, both the PSNR and SSIM results show that the 2-layer TM-Net delivers a better approximation to the full **CLHE** algorithm compared to running **CLHE** for two iterations. The PSNR is 47.52 for the 2-layer TM-Net but only 38.13 when **CLHE** is run for two iterations. Analogously, the SSIM (where 100 means perfect similarity), the mean SSIM for the 2-layer TM-Net is 0.97 and 0.95 when **CLHE** is run twice.

4.4. Execution Time of CLHE vs. TM-Net

Finally, we compare the speed of the TM-Net compared to **CLHE**. Since the tone curves here are applied to images in exactly the same way, when we measure timings we only consider the execution of the histogram modification steps (and not the application of the tone curve to images). In the introduction, we stated **CLHE** in the worst case we found converges in 60 iterations, and our TM-Net has two layers, thus from the perspective of modification steps the TM-Net is up to 30 times faster.

Next, we measured our MATLAB implementation of **CLHE** on the ImageNet and Places datasets (40,000 images total). The total computation time was 1063 s, or 26.6 ms per image. The TM-Net on the same dataset converged in 712 s, or 17.8ms per image. That is, running **CLHE** on the dataset took 49.3% longer. These timing experiments were performed on a machine running an Intel i7 CPU and an NVIDIA RTX 2070S.

5. Conclusions

Contrast Limited Histogram Equalization (**CLHE**) [3] is a widely used tone adjustment algorithm that ships in cameras and smartphones (including, the Apical IRIDIX algorithm [15]). **CLHE** is an iterative algorithm which iteratively refines a histogram so that its cumulative distribution (which defines the **CLHE** tone curve) has bounded slope (neither too small nor too large). In our experiments, we found that **CLHE** often converged quickly but that it was not unusual for it to take 20 iterations or, in the worst case, 60 iterations to converge.

In this work, we show how **CLHE** can be exactly transcribed into a neural network framework that we call TM-Net. The TM-Net has as many layers as there are iterations in **CLHE** with the weights in the network prescribed by the **CLHE** algorithm.

That we define **CLHE** as a neural network allows us to take advantage of the fact that a network can learn new parameters. That is, we need not use the weights that follow from the **CLHE** algorithm but we can relearn them in the usual network learning manner, i.e., by stochastic gradient descent. Surprisingly, we show that the outputs from a 2-layer TM-Net visually approximate the images generated by the **CLHE** algorithm running to convergence. The 2-layer TM-Net always runs much faster than the **CLHE** algorithm that it approximates.

Next, we were interested whether our TM-Net architecture could be used to learn other tone mapping algorithms. To test this idea, we took the Histogram Modification Framework (**HMF**) [20] as an exemplar algorithm. The general **HMF** framework finds a tone curve with an optimization defined as Quadratic Program. In **HMF**, various objective functions are minimized, including terms that ensure the tone curve maps, respectively, blacks to blacks and whites to whites, and that the tone curve should be smooth. Significantly, overall the **HMF** framework produces preferred tone-renderings to **CLHE**.

Again, we find that a trained 2-layer TM-Net is able to well-approximate **HMF**. Here the efficiency gains are even more stark. Quadratic Programming is an expensive algorithm to implement and run. Implemented as a 2-layer TM-Net, the complexity of **HMF** is found to be no greater than **CLHE**.

Author Contributions: Conceptualization, G.F.; methodology, G.F. and J.M.; software, J.M.; validation, G.F. and J.M.; formal analysis, G.F. and J.M.; investigation, G.F. and J.M.; resources, G.F. and J.M.; data curation, J.M.; writing—original draft preparation, J.M.; writing—review and editing, G.F.; visualization, G.F. and J.M.; supervision, G.F.; project administration, G.F.; funding acquisition, G.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by EPSRC grants EP/S028730/1 and EP/P007910/1.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The Kodak Image Dataset (<http://www.cs.albany.edu/~xypan/research/snr/Kodak.html>, accessed on 1 February 2020.) The ImageNet dataset (<https://image-net.org/download-images.php>, accessed on 1 February 2020.) The MIT Places Database (<http://places.csail.mit.edu/downloadData.html>, accessed on 2 January 2021).

Conflicts of Interest: The authors declare no conflict of interest.

References

- Fattal, R.; Lischinski, D.; Werman, M. Gradient Domain High Dynamic Range Compression. *ACM Trans. Graph.* **2002**, *21*, 249–256. [CrossRef]
- Barten, P.G. *Contrast Sensitivity of the Human Eye and Its Effects on Image Quality*; SPIE Optical Engineering Press: Bellingham, WA, USA, 1999.
- Pizer, S.M.; Amburn, E.P.; Austin, J.D.; Cromartie, R.; Geselowitz, A.; Greer, T.; ter Haar Romeny, B.; Zimmerman, J.B.; Zuiderveld, K. Adaptive histogram equalization and its variations. *Comput. Vision Graph. Image Process.* **1987**, *39*, 355–368. [CrossRef]
- Kim, Y.T. Contrast enhancement using brightness preserving bi-histogram equalization. *IEEE Trans. Consum. Electron.* **1997**, *43*, 1–8.
- Wang, Y.; Chen, Q.; Zhang, B. Image enhancement based on equal area dualistic sub-image histogram equalization method. *IEEE Trans. Consum. Electron.* **1999**, *45*, 68–75. [CrossRef]
- Reinhard, E.; Heidrich, W.; Debevec, P.; Pattanaik, S.; Ward, G.; Myszkowski, K. *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting*; Morgan Kaufmann: Burlington, MA, USA, 2010.
- Shapley, R.; Enroth-Cugell, C. Chapter 9 Visual adaptation and retinal gain controls. *Prog. Retin. Res.* **1984**, *3*, 263–346. [CrossRef]
- Valeton, J.; van Norren, D. Light adaptation of primate cones: An analysis based on extracellular data. *Vis. Res.* **1983**, *23*, 1539–1547. [CrossRef] [PubMed]
- Rahman, S.; Rahman, M.M.; Abdullah-Al-Wadud, M.; Al-Quaderi, G.D.; Shoyaib, M. An adaptive gamma correction for image enhancement. *EURASIP J. Image Video Process.* **2016**, *35*, 1–13. [CrossRef]
- Huang, S.C.; Cheng, F.C.; Chiu, Y.S. Efficient contrast enhancement using adaptive gamma correction with weighting distribution. *IEEE Trans. Image Process.* **2012**, *22*, 1032–1041. [CrossRef]
- Singnoo, J.; Finlayson, G.D. Understanding the gamma adjustment of images. *Color Imaging Conf.* **2010**, *2010*, 134–139.
- De Boor, C.; De Boor, C. *A Practical Guide to Splines*; Springer: New York, NY, USA, 1978; Volume 27.
- Gonzales, R.C.; Woods, R.E. *Digital Image Processing*; Prentice Hall: Hoboken, NJ, USA, 2002.
- Shannon, C.E. A mathematical theory of communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423. [CrossRef]
- Chesnokov, V. Image Enhancement Methods and Apparatus Therefor. U.S. Patent 7302110, 27 November 2007.
- Monga, V.; Li, Y.; Eldar, Y.C. Algorithm Unrolling: Interpretable, Efficient Deep Learning for Signal and Image Processing. *IEEE Signal Process. Mag.* **2021**, *38*, 18–44. [CrossRef]
- Robbins, H.; Monro, S. A stochastic approximation method. *Ann. Math. Stat.* **1951**, *22*, 400–407. [CrossRef]
- Reza, A. Realization of the Contrast Limited Adaptive Histogram Equalization (CLAHE) for Real-Time Image Enhancement. *VLSI Signal Process.* **2004**, *38*, 35–44. [CrossRef]
- Pizer, S.; Johnston, R.; Ericksen, J.; Yankaskas, B.; Muller, K. Contrast-limited adaptive histogram equalization: Speed and effectiveness. In Proceedings of the First Conference on Visualization in Biomedical Computing, Atlanta, GA, USA, 25 May 1990; pp. 337–345.
- Arici, T.; Dikbas, S.; Altunbasak, Y. A histogram modification framework and its application for image contrast enhancement. *IEEE Trans. Image Process.* **2009**, *18*, 1921–1935. [CrossRef] [PubMed]
- Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Syst.* **1989**, *2*, 303–314. [CrossRef]
- Nielsen, M.A. *Neural Networks and Deep Learning*; Determination Press: San Francisco, CA, USA, 2015; Chapter 2.
- McVey, J.; Finlayson, G. Least-Squares Optimal Contrast Limited Histogram Equalisation. *Color Imaging Conf.* **2019**, *2019*, 256–261. [CrossRef]
- Qiu, G.; Guan, J.; Duan, J.; Chen, M. Tone mapping for hdr image using optimization a new closed form solution. In Proceedings of the 18th International Conference on Pattern Recognition, Hong Kong, China, 18 September 2006; Volume 1, pp. 996–999.
- Ward, G.; Shakespeare, R. *Rendering with Radiance: The Art and Science of Lighting Visualization*; Morgan Kaufmann: Burlington, MA, USA, 1998.
- Ketcham, D.J.; Lowe, R.; Weber, J. Real-time image enhancement techniques. *Image Process.* **1976**, *74*, 120–125.
- Fu, X.; Cao, X. Underwater image enhancement with global-local networks and compressed-histogram equalization. *Signal Process. Image Commun.* **2020**, *86*, 115892. [CrossRef]
- Veluchamy, M.; Subramani, B. Image contrast and color enhancement using adaptive gamma correction and histogram equalization. *Optik* **2019**, *183*, 329–337. [CrossRef]

29. Celik, T.; Li, H.C. Residual spatial entropy-based image contrast enhancement and gradient-based relative contrast measurement. *J. Mod. Opt.* **2016**, *63*, 1600–1617. [[CrossRef](#)]
30. Stark, J.A. Adaptive image contrast enhancement using generalizations of histogram equalization. *IEEE Trans. Image Process.* **2000**, *9*, 889–896. [[CrossRef](#)] [[PubMed](#)]
31. Tomasi, C.; Manduchi, R. Bilateral filtering for gray and color images. In Proceedings of the Sixth International Conference on Computer Vision, Bombay, India, 6 August 1998; pp. 839–846.
32. Land, E.H. An alternative technique for the computation of the designator in the retinex theory of color vision. *Proc. Natl. Acad. Sci. USA* **1986**, *83*, 3078–3080. [[CrossRef](#)] [[PubMed](#)]
33. Singnoo, J.; Finlayson, G.D. Optimal global approximation to spatially varying tone mapping operators. In Proceedings of the Conference on Colour in Graphics, Imaging, and Vision, Amsterdam, The Netherlands, 6 May 2012; pp. 182–188.
34. Finlayson, G.D.; Singnoo, J. Global Approximation to Spatially Varying Tone Mapping Operators. U.S. Patent No. 9129388 B2, 2 January 2018.
35. Schanda, J. *Colorimetry: Understanding the CIE System*; John Wiley & Sons: Hoboken, NJ, USA, 2007.
36. Jin, X.; Xu, C.; Feng, J.; Wei, Y.; Xiong, J.; Yan, S. Deep learning with s-shaped rectified linear activation units. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12 February 2016.
37. ImageNet Image Database. Available online: <http://image-net.org/index> (accessed on 1 February 2020).
38. Artusi, A.; Banterle, F.; Carra, F.; Moreno, A. Efficient Evaluation of Image Quality via Deep-Learning Approximation of Perceptual Metrics. *IEEE Trans. Image Process.* **2020**, *29*, 1843–1855. [[CrossRef](#)] [[PubMed](#)]
39. Kodak Lossless True Color Image Suite. Available online: <http://r0k.us/graphics/kodak/> (accessed on 1 February 2020).
40. Places, The Scene Recognition Database. Available online: <http://places.csail.mit.edu/> (accessed on 1 February 2020).
41. Meyer, G.W. Reproducing and synthesizing colour in computer graphics. *Displays* **1989**, *10*, 161–170. [[CrossRef](#)]
42. Stokes, M.; Fairchild, M.D.; Berns, R.S. Precision Requirements for Digital Color Reproduction. *ACM Trans. Graph.* **1992**, *11*, 406–422. [[CrossRef](#)]