https://doi.org/10.2352/issn.2694-118X.2021.LIM-93

Towards a Generic Neural Network Architecture for Approximating Tone Mapping Algorithms

Jake McVey and Graham Finlayson; University of East Anglia; Norwich, England.
J.McVey@uea.ac.uk, G.Finlayson@uea.ac.uk

Abstract

Tone curves are a key feature in any image processing pipeline, and are used to change the pixel values of an input image to find an output image that looks better. Perhaps the most widely deployed tone curve algorithm is Contrast Limited Histogram Equalisation (CLHE). CLHE is an iterative algorithm that tone maps an input image so that the histogram of the output is (approximately) maximally uniform subject to the constraint that the tone curve has bounded slope (neither too large or too small).

In this paper, we build upon a neural network framework [1] that was recently developed to deliver CLHE in fewer iterations (each layer in the neural network is analogous to a single iteration of CLHE, but the network has fewer layers than the number of iterations needed by CLHE). The key contribution of this paper is to show that the same network architecture can be used to implement a more complex (and more powerful) tone mapping algorithm. Experiments validate our method.

INTRODUCTION

Using a tone curve to enhance an image is one of the oldest image processing techniques. A tone curve is a 1-to-1 mapping function that transforms the pixel values of an image. These new pixel values define an output image that should look better than the original.

A well-known tone adjustment algorithm is Histogram Equalisation (HE) [2]. In HE, the tone curve is obtained directly as the cumulative sum of the image histogram. For a standard 8-bit greyscale image, the histogram has 256 bins, where each bin counts the occurrences of pixel intensity k in the image ($k \in [0, 255]$). This histogram is normalised so that the sum of the histogram is 1 (it is a probability density function, PDF) before the CDF (cumulative density function), or cumulative histogram, is calculated.

In Figure 1, left, we see a greyscale image of one of the 24 widely used Kodak reference images [3]. On the right, we see the output of naive Histogram Equalisation. Clearly the output image looks terrible. In the middle panel we see an output image that is more pleasing. The image has more contrast and the details are more conspicuous, while the image retains a natural appearance.

Contrast Limited Histogram Equalisation (CLHE) [4] is a well known and widely used histogram modification algorithm that produces tone curves that have bounded slopes (neither too high or too low). While CLHE is an improvement on HE it is still far from perfect. Known problems are that the generated tone curves can be 'wiggly', and blacks and whites in the image can become too compressed (resulting in loss of detail). The Histogram Modification Framework (HMF) [5] is an example of a more modern tone curve method which solves the problems in CLHE but is more complex to implement. The middle image in Figure 1 shows the output of the HMF. In essence, in this paper we will show how complex methods such as [5] can be made as simple as CLHE.

In Figure 2, left panel (a) we show the histogram (dashed lines) of the input image (left, Figure 1). In the right panel (b) we show the corresponding Histogram Equalised tone-curve (the cumulative histogram), also a dashed line. The HMF algorithm tries to find a histogram (that sums to one) that is close to the actual input histogram, and that has the additional property that its cumulative histogram will result in a more 'reasonable' tone curve (see red curve in panel (b)).

Neural networks can approximate many functions and have been used successfully in many applications including image processing [6]. In this paper, we will use a neural net to map an input image histogram to an output histogram that, when integrated, will make a good tone curve. Our starting point will be the network derived in [1] which was designed to approximate CLHE. Like many networks, the CLHEnet has the same layer repeated several times. Figure 3 summarises a single layer in the CLHEnet.

The solid blue columns in Figure 3 represent the values in the bins of a histogram. In this toy example the histogram has just 5 bins (hence 5 arrows), but the analogy extends to any N-bin histogram. The values of the input are directly connected to an output layer. The values of the input are also weighted and connected to a single summation block (follow the red arrows) and a bias term also feeds into this block. The calculated sum is then weighted and added to the output layer (follow the green arrows). With the exception of the final block in the network, the new values in the output layer are then combined and fed through an activation function. A piece-wise linear activation function is used. The output of the network is the final output block (that does not pass through the activation function before output).

The CLHEnet has the property that it can be instantiated (the weights fixed) so that with > 50 layers the CLHE algorithm can be simulated exactly. The novelty of [1] is to show that CLHE - for a large corpus of images - can be computed in just two or three layers so long as we train the network (and do not use the CLHE default weights).

The prior work [1] shows how the individual algorithm steps can be mapped one-to-one to the architecture shown in Figure 3. In this paper we wonder whether more complex histogram based tone mapping algorithms can be learnt using the CLHEnet. We focus on the Histogram Modification Framework (HMF) of [5]. In HMF a modified histogram is sought that, like CLHE, is close to the original but where additional constraints are also added to ensure a smooth tone curve and good whites and blacks.

While the HMF tone mapped images are reliably preferred over CLHE, the HMF algorithm is significantly more complex. In this paper we show that we can re-train the CLHEnet and implement the HMF framework much more efficiently. Our work may suggest that the CLHEnet can be used to implement histogram based tone mapping algorithms more efficiently.

The Histogram Modification Framework is discussed more in section 2. In section 3, we use a trained instantiation CLHEnet to implement HMF. Section 4 details the results of our experiments,

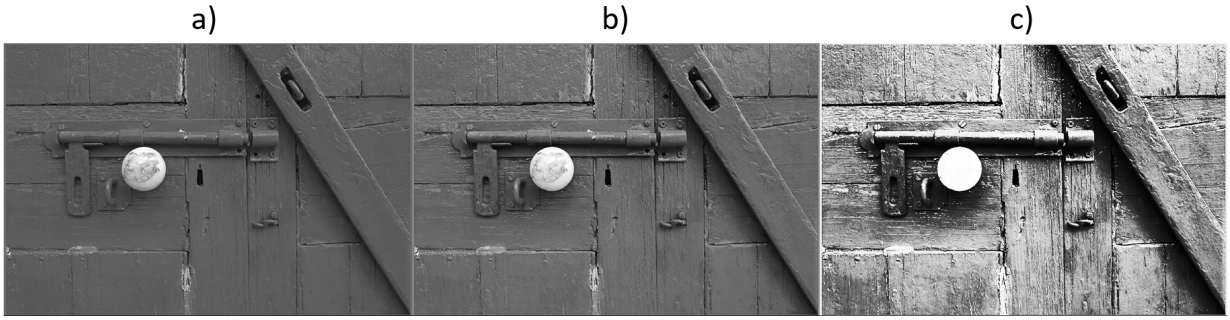

Figure 1: Comparison of Histogram Modification Framework (HMF) and Histogram Equalisation (HE). **a)** Input greyscale image. **b)** HMF enhanced image. $\lambda = 1, \gamma = 100, \alpha = 5$ **c)** HE enhanced image. The tone curves used to enhance these images are shown in Figure 2b.

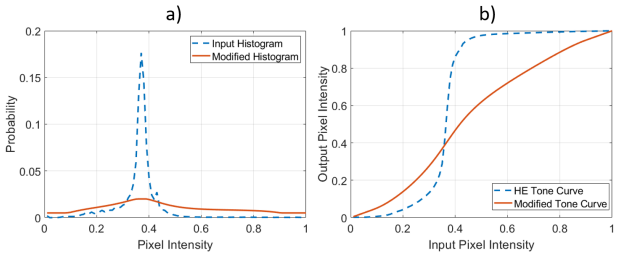

Figure 2: Histograms and associated tone curves of a greyscale image. **a)** Normalised (and modified) histogram of pixel intensities. (image PDFs) **b)** Tone curves that are cumulative sum (CDFs) of the histograms.

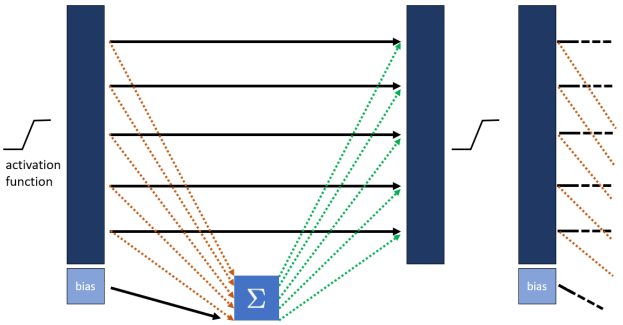

Figure 3: A graphical representation of a 'block' in the CLHE feed-forward neural network. Arrows indicate weighted connections in the network.

and the paper concludes in section 5.

Background

Revisiting the tone curves in Figure 2b reveals the cause for the poor quality of the HE image. The slope of the HE tone curve is very steep on a narrow interval. Consequently, pixel intensities in that interval are over-emphasised in the output image. In 1c this manifests as unnatural details on the door. The HE tone curve is also very shallow over wide intervals. Shallow slopes cause details to be compressed as intensities are mapped close together [7]. This can result in loss of detail in the brightest and darkest regions. Finally, consider the speed with which a slope changes from steep to shallow. When this happens too quickly the image details can become unevenly enhanced, which often manifests as contour-like patchiness artefacts, seen throughout the HE image.

A Histogram Modification Framework (HMF) was introduced in [5] and improves on the HE and the CLHE methods. Both CLHE and HMF start with an input image and the histogram for this image. In CLHE - through an iterative algorithm - we find a derived histogram that has 'clip' limits on the bin counts (each has to be larger than and less than respective lower and upper bounds). Because the tone-curve is the integral of the derived histogram we, by bounding the bin counts, also bound the slope of the tone curve (to be neither too steep or too shallow).

However, CLHE can produce wiggly tone curves (leading to contouring artefacts) and blacks and whites are often overly compressed. The Histogram Modification Framework (HMF) attempts to mitigate these problems. The HMF tone curve is also the integral (cumulative sum) of a derived histogram. The derived histogram is found by minimizing:

$$min_{\mathbf{h}} ||\mathbf{h} - \mathbf{h_i}||_2^2 + \lambda ||\mathbf{h} - \mathbf{u}||_2^2 + \gamma ||D\mathbf{h}||_2^2 + \alpha ||S\mathbf{h}||_2^2. \qquad (1)$$

The first squared error term teaches that the derived histogram ($\mathbf{h}$) should be close to the original $\mathbf{h_i}$. The second term guides the derived histogram towards the identity histogram, $\mathbf{u} = [\frac{1}{N}, \frac{1}{N}, \cdots, \frac{1}{N}]^T$ (where ^{T} represents vector transpose). As λ - a user defined weight - grows large the histogram becomes closer to the uniform histogram. The integral (cumulative sum) of the uniform histogram is a tone curve that is a line at 45 degrees. This is the identity tone curve where output brightness equals input brightness. Therefore, as the solved-for histogram is pushed closer to the uniform histogram, the level of contrast enhancement in the output image is reduced.

The third error term controls the smoothness of the histogram, here D is the discrete derivative operation ($D\mathbf{h}$ is the second derivative of the tone curve). We can usefully think of γ as a constraint on the 'wiggly-ness' of the tone curve. As γ grows large so the wiggly-ness of the tone curve decreases.

The final error term is used to map a predetermined range of the darkest and brightest intensities to darker and brighter intensities respectively. This is sometimes referred to as 'black and white stretching'. The last term ensures unwanted details are not introduced to the deepest blacks and whites of the image. See [5] for a more in depth explanation.

Additionally, the solved for derived histogram $\mathbf{h}$ should have all bins in the range [0,1] and sum up to one (the histogram is a PDF). As in CLHE we would like the derived histogram have a bounded minimum and maximum bin counts (so we constrain slope bounds of the tone curve). In this work the upper and lower bounds are defined as $\frac{2}{N}$ and $\frac{0.5}{N}$ respectively, where N represents the number of bins (values) in the histogram. This results in tone-curves whose slope is between 0.5 and 2. We can solve Equation 1 - with these additional constraints - using Quadratic Programming.

As a final detail, the penalty terms in Equation 1 are weighted using values suggested in [5] (and also used in Figure 1).

Method

Let us consider a generic description per-block of the computation carried out in a feed-forward neural network [8]:

$$\mathbf{a}^l = \sigma(W^l \mathbf{a}^{l-1} + \mathbf{b}^l). \tag{2}$$

Where $\mathbf{a}^l \in \mathbb{R}^N$ is a vector that is the output of layer l in the network. Next, $\sigma()$ is an activation function. $W^l \in \mathbb{R}^{N\times N}$ is a matrix of weights that scales the contribution of each neuron in the previous layer, $l-1$. $\mathbf{a}^{l-1} \in \mathbb{R}^N$ is a vector that is the output of the previous layer in the network, and $\mathbf{b}^l \in \mathbb{R}^N$ is a vector that holds the bias values for each neuron in layer l.

In this work, we adopt the same framework as CLHEnet to define a neural network version of the HMF algorithm. So we need to marry Figure 3 with the notation of Equation 2. Let us begin by initialising the network using the CLHE parameters, that are the values for the weights and biases such that > 50 blocks is CLHE.

At initialisation, the weight matrix W is written as $\mathbf{v}\mathbf{w}^T$ when $\mathbf{v}, \mathbf{w} \in \mathbb{R}^N$ are two vectors of 1 and $\frac{1}{N}$ respectively. Here $\mathbf{v}\mathbf{w}^T$ models the single block summation (where the red arrows feed in and the green ones leave, shown in Figure 3). The bias vector $\mathbf{b} \in \mathbb{R}^N$ is defined at initialisation as a vector of $\frac{1}{N}$'s ($\mathbf{b} = [\frac{1}{N}, \frac{1}{N}, \cdots, \frac{1}{N}]^T$). The activation function, $\sigma()$, is a piece-wise linear function that bounds the input to upper and lower bounds, in this case $\frac{2}{N}$ and $\frac{0.5}{N}$ where N is the size of the input and target histograms. This bounding of the histogram bins has the effect of making the min and max slopes of the associated tone curve .5 and 2 respectively. We illustrate this function (for a histogram of size $N = 100$) and it's effect on a histogram in Figure 4.

Of course, a clipped histogram will not sum to one. So we need to calculate the 'delta' $\triangle$ (from 1) and add Δ/N to the clipped histogram (so that it does sum to one). This is exactly what the architecture in Figure 3 computes (with the initialisation) proposed in the last paragraph. Previous research has shown that CLHE (which can take up to 50 iterations to converge) can be implemented with a 3 layer net (using the blocks shown) where we do not use the defualt initialisations but rather learn the optimal weights.

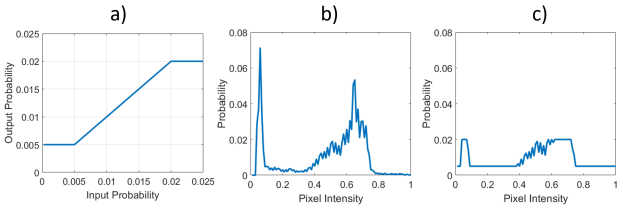

Figure 4: A piecewise linear activation function and it's effect of a histogram vector. **a)** A function that is linear between at 0.005 and 0.02 (and clips to the these values for inputs outside this range). **b)** A histogram. **c)** This histogram after it has been modified by the plateau function.

Here, we will attempt to use the same CLHEnet architecture to learn the HMF algorithm. As before we will initialise with CLHE weights. And, arbitrarily, we will use a 3 layer network. We call our learnt network (even though it is based on CLHEnet), HMFnet because the learned weights are different. Indexing over a large corpus of histograms we seek to find the weights and biases that minimize:

$$\mathscr{J} = \min_{\mathscr{P}} \sum_i ||HMF(\mathbf{h}_i) - HMFnet(\mathbf{h_i}; \mathscr{P})||_2^2. \tag{3}$$

In Equation 3 $\mathscr{P}$ denotes the set of all weights and biases that define the 3 layers of the network. In the neural-net literature, $\mathscr{J}$ is called as a loss function. We seek to learn the weights and biases that minimize the loss.

We actually minimize a loss function that is slightly more complex:

$$\min_{\mathscr{P}} \mathscr{J} + \beta \mathscr{S}. \tag{4}$$

where $\mathscr{S}$ controls the smoothness of the discovered weights and biases. While the details of what smoothness means are discussed elsewhere, we recapitulate the intuition. In the lth level of the network we have the weights $\mathbf{v}^l$, $\mathbf{w}^l$ and the bias $\mathbf{b}^l$. These vectors are in one-to-one correspondence to the histogram bins. These in turn have a natural order (e.g. they record the relative frequency of the pixels in the image from darkest to brightest). We would like this natural order to be accounted for in the minimization. Indeed, we do not expect the kth weight w_k^l to be significantly different from w_{k+1}^l. That is, we expect the weight vectors $\mathbf{v}^l$, $\mathbf{w}^l$ and the bias $\mathbf{b}^l$ (viewed as functions we might plot on a graph) to be smooth. Thus, the term $\mathscr{S}$ in Equation 4 is the sum of the squared derivatives of the weights and biases calculated at the different levels. This in turn is weighted by the controllable parameter β, where β represents the importance of smoothing. See [1] for more details.

The specific weights and bias values that underpin the optimization in Equation 4 are found using stochastic gradient descent (SGD) and back propagation (BP). The precise details of the SGD and BP [6] are not important for the purpose of this work. What is important that we can use these standard techniques to optimize Equation 5 efficiently.

Training the network

To train the network we used a randomly sampled set of 30,000 images from the ImageNet dataset [9]. From each image we obtained the input histogram and modified histogram found by the HMF. These served as the input and target of the network respectively. We trained the network using a regularised mean squared error loss function (Equation 4) with $\beta = 0.001$, and the following hyper parameters: batch size = 30,000, learning rate = 1e-4, and epochs = 250.

Experiments

To evaluate the success of our HMFnet we compare the *closeness* of images enhanced with our method against images enhanced with the target algorithm HMF [5]. For the purpose of this work closeness is quantitatively measured using the CIELAB Delta E [10] distance metric. To obtain the Delta E statistics we take a reference image and enhance it with the target algorithm. We then enhance the same reference image independently with the proposed method. Both enhanced images are then converted to the CIELAB colour-space, and the per-pixel difference between the proposed and target image is calculated to generate an error image. Using this error image we calculate the mean Delta E per image. We then calculate the mean, median, and 99 percentile of the mean error for all the images in the dataset.

We evaluate our method on three datasets: the well-known Kodak dataset [3] that contains 24 RGB images. A randomly sampled set of 20,000 RGB images from the ImageNet dataset [9] that were not used to build the model, and a final 20,000 RGB images from the MIT Places database [11].

Results and Analysis

In Table 1 we show the Delta E error statistics for the proposed HMFnet framework when compared to HMF [5]. Clearly, the errors shown are not zero. However, the numbers are small. In tightly controlled viewing conditions a Delta E of 1 correlates with a just noticeable color difference. However, in complex images (e.g. photographic pictures) an average Delta E of between 3 and 5 [12, 13] is generally thought to be visually not noticeable.

In Figure 5 we show a series of images from the Kodak dataset that have been enhanced with HMF (middle column) and HMFnet (right column). For each image in the set we show the mean Delta E in the top right corner. From these images we learn that a target for Delta E of around 3 is sensible, as the image in the set that is 'most-different' (flower) shows only few noticeable differences even under very close observation.

3 block HMFnet	*Mean* ΔE^*	*Med.* ΔE^*	*99pt* ΔE^*
Kodak	1.49 ($\pm$ 0.34)	1.22 ($\pm$ 0.41)	3.74 ($\pm$ 0.96)
ImageNet	1.32 ($\pm$ 0.37)	1.48 ($\pm$ 0.28)	3.31 ($\pm$ 0.89)
Places	1.16 ($\pm$ 0.23)	1.24 ($\pm$ 0.31)	3.12 ($\pm$ 0.88)

Table 1: Mean, median, and 99-percentile ($\pm$ standard deviation) of the mean ΔE^* for enhanced images, averaged over each image in the datasets.

Figure 5: *For each image in the set:* First, original image. Middle, image enhanced with HMF [5]. Right, image enhanced with proposed HMFnet.

Conclusion

In this paper we demonstrate that a neural network framework that was designed to approximate a specific histogram based tone mapping algorithm (CLHE [4]) is also able to be retrained to approximate the Histogram Modification Framework (HMF) of [5]. Significantly, HMF is a complex algorithm. By reformulating it in network form it is orders of magnitude faster (than the original QP-based implementation). Experiments demonstrate that the HMFnet delivers images that are visually very similar to those delivered by the original HMF algorithm.

Acknowledgements

The authors are grateful for funding from EPSRC grants EP/S028730/1 and EP/P007910/1.

References

[1] G. Finlayson and J. McVey, "A neural network framework for approximating histogram adjustment algorithms," *Currently in review*, 2021.

[2] R. C. Gonzalez and R. E. Woods, *Digital image processing*. Prentice-Hall, 2002.

[3] "Kodak lossless true color image suite." http://r0k.us/graphics/kodak/. Accessed: 01-06-19.

[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," *Computer vision, graphics, and image processing*, vol. 39, no. 3, pp. 355–368, 1987.

[5] T. Arici, S. Dikbas, and Y. Altunbasak, "A histogram modification framework and its application for image contrast enhancement," *IEEE Transactions on Image Processing*, vol. 18, no. 9, pp. 1921–1935, 2009.

[6] J. A. Freeman and D. M. Skapura, *Neural networks: algorithms, applications, and programming techniques*. Addison Wesley Longman Publishing Co., Inc., 1991.

[7] J. McVey and G. Finlayson, "Least-squares optimal contrast limited histogram equalisation," *Color and Imaging Conference*, vol. 2019, pp. 256–261, 2019.

[8] M. A. Nielsen, *Neural Networks and Deep Learning*. chap. 2, Determination Press, 2015.

[9] "Imagenet image database." http://image-net.org/index. Accessed: 01-06-19.

[10] A. R. Robertson, "The cie 1976 color-difference formulae," *Color Research & Application*, vol. 2, no. 1, pp. 7–11, 1977.

[11] "Places, the scene recognition database." http://places.csail.mit.edu/. Accessed: 02-01-21.

[12] G. W. Meyer, "Reproducing and synthesizing colour in computer graphics," *Displays*, vol. 10, no. 3, pp. 161–170, 1989.

[13] M. Stokes, M. D. Fairchild, and R. S. Berns, "Precision requirements for digital color reproduction," *ACM Trans. Graph.*, vol. 11, no. 4, pp. 406–422, 1992.

Author Biography

Jake McVey is a PhD student at the University of East Anglia (UEA). His work focuses on colour image processing, and machine learning approaches to image enhancement.

Professor Graham Finlayson is the Director of the Colour & Imaging Lab at the UEA. He has published over 200 conference papers and over 75 journal papers. He is also the inventor of 30+ patents many of which are used in commercial products. His interests span perception, colour image processing and physics-based computer vision.