https://doi.org/10.2352/issn.2169-2629.2019.27.29

Color Space Transformation using Neural Networks

Lindsay MacDonald, University College London, UK *and* ***Katarina Mayer, ESET, Bratislava, Slovakia***

Abstract

We investigated how well a multilayer neural network could implement the mapping between two trichromatic color spaces, specifically from camera R,G,B to tristimulus X,Y,Z. For training the network, a set of 800,000 synthetic reflectance spectra was generated. For testing the network, a set of 8,714 real reflectance spectra was collated from instrumental measurements on textiles, paints and natural materials. Various network architectures were tested, with both linear and sigmoidal activations. Results show that over 85% of all test samples had color errors of less than 1.0 ΔE_{2000} units, much more accurate than could be achieved by regression.

1. Introduction

The signals produced by a digital camera are encoded at each pixel position in the image as red, green and blue (*R,G,B*). The relationship between the scene's reflected light and each signal depends on the spectral sensitivity of the corresponding channel. Human color vision, on the other hand, responds according to the sensitivity of the retinal photoreceptors ('cone cells') in long, medium and short (*L,M,S*) wavebands. The light reflected from the object, as it reaches the camera or observer, is a continuous distribution of power as a function of wavelength over the visible spectrum 380–780 nm. Both camera and observer apply a projective mapping from N to 3 dimensions, but because their primaries (i.e. spectral basis functions) are different there is no simple way of defining correspondence between the two triplets of signals (Fig. 1).

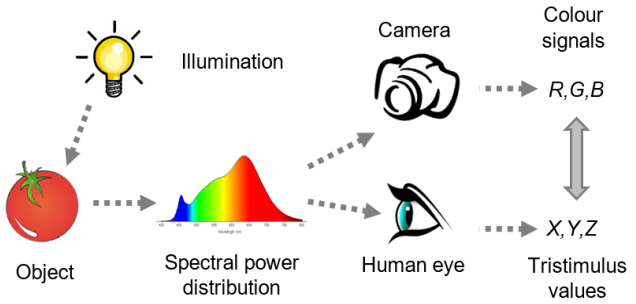

Figure 1. *Correspondence of color signals from a camera and observer.*

This is restatement of input device characterization, i.e. the process of establishing the relationship between the color of an object seen by a human observer, usually expressed in CIE tristimulus values, and signals generated by the camera. We seek to represent the camera signals as a function of color: $(R,G,B) = f(X,Y,Z)$. There is generally no analytical function available, and neither set of spectral sensitivity functions can be represented as a linear combination of the other. Hence an approximation is frequently used, by fitting the data with a low-order polynomial through a regression procedure.

A large dataset of 8,714 reflectance spectra was collated from readily available spectral reflectance measurement sets [1] over the wavelength range 380 to 780 nm, and interpolated to 1 nm intervals by the Matlab function `interp1` with the cubic spline option. Measurement data was at intervals of 10nm or less. Where the original wavelength range was smaller, for example with no data provided below 400 nm or above 700 nm, the reflectance was set to zero. *R,G,B* and *X,Y,Z* triplets corresponding to all reflectance spectra were calculated at 1nm wavelength intervals, multiplying by the spectral power distribution of the D65 illuminant, and the interpolated sensitivity curves for the Nikon D200 digital camera [3] and the tristimulus functions of the CIE 2° Standard Observer, as plotted in Fig. 2. All calculations were performed in Matlab over the wavelength range [380,780] nm, using data vectors with 401 elements. Computed values for *R,G,B* were normalized to the range [0,1], while *X,Y,Z* values were normalized to $Y_w = 100$.

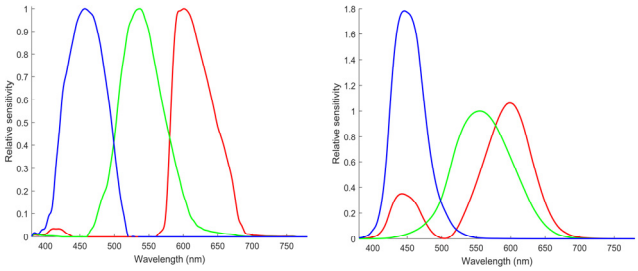

Figure 2. *Spectral sensitivity functions of: (left) Nikon D200 camera; and (right) the CIE 2° Standard Observer.*

As a baseline for the accuracy of the color transform, a regression procedure was employed to find coefficients m_{ij} that would minimize error in the *X,Y,Z* domain over the full dataset:

$$\begin{aligned} X_i &= m_{11}R_i + m_{12}G_i + m_{13}B_i \\ Y_i &= m_{21}R_i + m_{22}G_i + m_{23}B_i \\ Z_i &= m_{31}R_i + m_{32}G_i + m_{33}B_i \end{aligned} \tag{1}$$

where R_i, G_i, B_i are normalized signal values from the camera sensors for each sample. The system of equations can be written more compactly in matrix form as:

$$[\boldsymbol{X} \quad \boldsymbol{Y} \quad \boldsymbol{Z}] = [\boldsymbol{R} \quad \boldsymbol{G} \quad \boldsymbol{B}]\,\mathbf{M} \tag{2}$$

where for n wavelength intervals, $\boldsymbol{X} = [\,X_1 \ldots X_n\,]^T$ is the nx1 response vector (and similar for $\boldsymbol{Y}$ and $\boldsymbol{Z}$), $\boldsymbol{R} = [\,R_1 \ldots R_n\,]$ is the nx1 signal vector (and similar for $\boldsymbol{G}$ and $\boldsymbol{B}$) and the 3x3 matrix $\mathbf{M} = [\,m_{ij}\,]$ contains the coefficients.

A regression fitting over the full set of 8,714 real samples yielded the matrix:

$$\mathbf{M_R} = \begin{bmatrix} 74.32 & 29.43 & 3.26 \\ 41.36 & 104.61 & -6.87 \\ 13.79 & -4.52 & 154.76 \end{bmatrix} \tag{3}$$

The distribution of color errors is shown in Fig. 3, with mean of 1.68 ΔE^*_{ab}. This is considered a good performance, but note that more than 50% of errors are above the visibility threshold (assuming JND = 1.0) and over 5% of color errors exceed 5 ΔE^*_{ab}.

To validate the fitting procedure, the collated dataset was partitioned by random selection into two subsets, each containing 4,357 samples. A matrix was computed by regression fitting over the first subset, and applied to the other subset to obtain the predicted *X,Y,Z* values. The performance of this linear fitting of camera *R,G,B* to tristimulus *X,Y,Z* was evaluated by converting both the reference color and estimated color of each sample to *L*a*b** and calculating the color difference as ΔE^*_{ab}. The distribution of errors for fitting was very similar to that for testing with median 1.05 (1.03), mean 1.71 (1.67), and 95th percentile 5.47 (5.24).

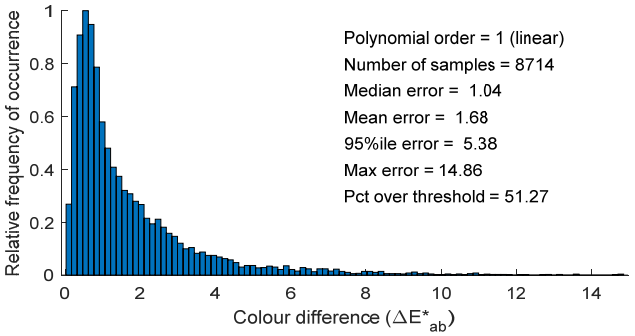

Figure 3. *Distribution of color errors (ΔE^*_{ab}) from fitting of 3x3 matrix to a set of X,Y,Z triplets derived from 8,714 samples of real (measured) reflectances.*

Error statistics for second- and third-order polynomial fitting to all real data are shown in Table 1. These represent the baseline of performance that can be achieved by conventional means, using transforms of colorimetric values (in this case *R,G,B* to *X,Y,Z*).

Table 1. Error statistics for regression fitting of real samples

Polynomial order	1	2	3
Median	1.04	0.90	0.80
Mean	1.68	1.37	1.12
95th percentile	5.38	4.20	3.18
Maximum	14.86	14.48	10.91
% errors above visual threshold	51.27	45.31	39.79

In a recent study by Jackman *et al* of camera characterization in a machine vision application, conversion of *RGB* into *L*a*b** was achieved by combining a linear color space transform and multiple regression methods. They found that it was possible to predict *L*a*b** color coordinates with errors less than 2.2 ΔE^*_{ab} units by transforming the *R,G,B* color components into new variables that better reflected the structure of the *L*a*b** color space. Color features derived by this system were able to discriminate between three classes of ham with 100% correct classification, whereas color features measured on a conventional colorimeter were not [6].

A problem with regression is that the transformation should be optimized over all reflectance spectra that could be encountered. If the usage is always limited to a specific population of materials or objects, for example photographic prints or paints or textiles made with known dyes, it may be sufficient to take a selection of test samples from that population. But for a digital camera that could be used for any real-world materials, it is not sufficient to train it on a small number of samples with a limited range of reflectance spectra, e.g. by using a standard color test target, and then expect the transform to predict accurately all possible spectra.

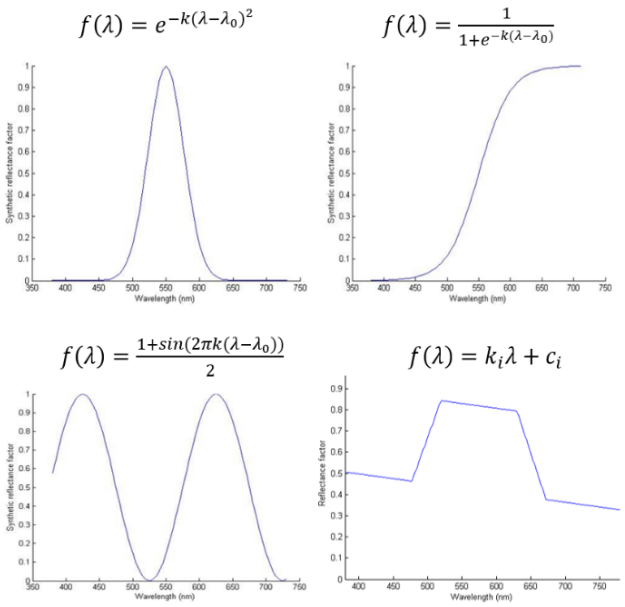

Figure 4. *Four generating functions for synthetic spectra: (top left) Gaussian; (top right) logistic; (bottom left) sine; (bottom right) piece-wise linear ramp.*

2. Synthetic reflectance spectra

To simulate the widest possible range of reflectance spectra that might be encountered by a camera, a set of synthetic spectra was generated. They were constrained to be continuous, single-valued functions of wavelength at 1 nm intervals over the range 380 to 780 nm, with a maximum slope (first derivative) not exceeding ±0.024 nm^{-1}. The latter limit was derived from analysis of real spectra, which showed that the first derivatives fall largely within the range from -0.01 to +0.02 nm^{-1}. These gradient values can be related to the cross-sectional area of the color gamut [2].

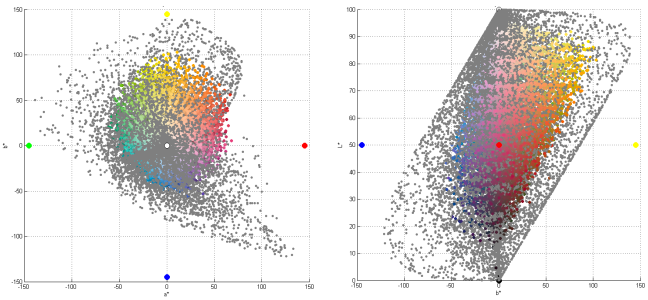

Figure 5. *Cross-sections of (left) a*-b* plane and (right) L*-b* plane of color gamut of 8,714 real (colored) and 20,000 synthetic (grey) spectra in CIELAB space for the D65 illuminant.*

The synthetic reflectance spectra were based on eight generating functions: Gaussian, inverted Gaussian, logistic, inverted logistic, sine, upward ramp, downward ramp, sum of three Gaussians, and piecewise linear (Fig. 4). The central wavelength λ_0 and width parameter k were randomized to produce a family of 100,000 curves for each function, giving 800,000 synthetic spectra in total. Central wavelengths ranged over the visible spectrum from 400 to 700 nm, with the full range of reflectance factors [0,1]. Limits

were set on the slope and width parameters to restrict the maximum slope of the curves in all cases within the range ±0.024 per 1 nm interval, based on an analysis of the reflectance of naturally occurring materials [1]. $L^*a^*b^*$ coordinates of all real and synthetic spectra were calculated for the D65 illuminant (Fig. 5). The real samples fit within cuboidal bounds of approximately $10<L^*<95$, $-60<a^*<70$ and $-55<b^*<105$. The synthetic spectra, however, fill a much larger volume of $0<L^*<100$, $-130<a^*<135$ and $-125<b^*<140$. They are more demanding, therefore, as a test of the robustness of any color transformation.

Regression was applied, as described in the previous section, fitting the corresponding *R,G,B* and *X,Y,Z* triplets derived from the full set of 800,000 synthetic spectra, giving the matrix:

$$\mathbf{M_S} = \begin{bmatrix} 78.23 & 31.50 & 4.30 \\ 36.23 & 102.02 & -9.77 \\ 16.83 & -3.16 & 157.16 \end{bmatrix} \quad (4)$$

Comparison with $\mathbf{M_R}$ in Eq. (3) shows that the blue component was slightly under-represented in the real samples. The fitting errors were larger because of the greater variance of the synthetic dataset, and higher orders of polynomial did not improve performance, actually causing the median error to increase (Table 2).

Table 2. Error statistics for fitting of synthetic spectra

Polynomial order	1	2	3
Median	1.43	1.45	1.53
Mean	3.08	2.95	2.52
95th percentile	11.26	10.40	8.00
Maximum	73.37	70.16	63.78
% errors above v.t.	58.08	58.59	61.51

The polynomial transforms derived from fitting the synthetic spectra were tested on corresponding triplets derived from the real (measured) reflectance spectra. The results in Table 3 are slightly worse than for fitting on the real reflectance data alone (Table 1). But because the synthetic spectra densely sample the whole space of realistic reflectance spectra (within the given constraints) the transformations can be regarded as independent of any particular set of samples. Thus the matrix $\mathbf{M_S}$ could be considered optimal for the given camera (Nikon D200) and illuminant (D65) over all possible reflective surfaces that might be encountered.

Table 3. Error statistics for testing with real samples

Polynomial order	1	2	3
Median	1.11	1.06	1.01
Mean	1.83	1.74	1.41
95th percentile	5.76	5.43	3.85
Maximum	14.02	11.72	9.51
% errors above v.t.	52.81	51.76	50.46

3. Neural networks

An earlier study by Fdhal et al. [4] trained a neural network to convert *R,G,B* to *L*,a*,b** in the context of ICC output profile generation for an Epson inkjet printer. The network had an input and output layer, each with 3 nodes, and one hidden layer with 360 nodes. The training dataset had 909 samples and the test set had 256 samples. The mean and maximum errors (ΔE^*_{ab}) achieved by the network were claimed to be 0.28 and 2.29 respectively. One reason for the smallness of these values may have been that the samples were limited to a printed test chart, the color gamut of which occupied a relatively small volume in color space.

We have studied whether a neural network of any topology can produce better results for the two-way color transformation between *R,G,B* and *X,Y,Z* than a system of polynomials fitted by regression. The network was trained using the 800,000 synthetic reflectance spectra, as described in Section 2, and was tested using the dataset of 8,714 real reflectance spectra. A previous project for processing *R,G,B* signals from a color laser scanner [1] had demonstrated that some performance improvement over regression could be achieved by an adaptive color lookup table (CLUT) method, and we wanted to see what could be achieved by a deep learning network.

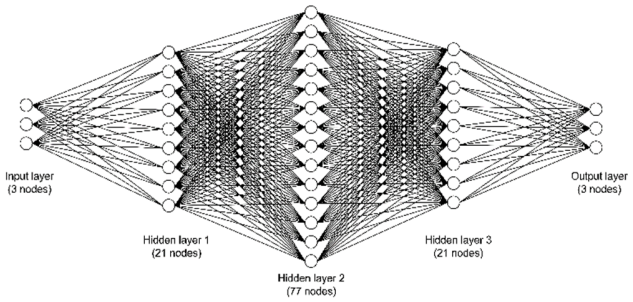

Figure 6. *Schematic representation of a fully-connected network with three hidden layers.*

Initially, we decided on a network with nonlinear activations. To identify the optimal architecture of our network, we ran a grid search over the space of plausible parameter values as follows:

- Activations: {sigmoid, relu, tanh}
- Number of nodes: {3, 21, 54, 77, 96, 114, 132}
- Optimizer selection {adam, sgd, rmsprop, adagrad, adadelta}
- Kernel initializer {'uniform', 'lecun_uniform', 'normal', 'zero', 'glorot_normal', 'glorot_uniform', 'he_normal', 'he_uniform'}
- Batch size {50, 100, 150, 200, 250}
- Epochs {25, 50, 75, 100, 125}

In the first round of grid search we considered a network architecture with just one hidden layer, with number of nodes ranging from 3 to 132. The optimal number of nodes selected by the algorithm was 21 but the fit was not very impressive with R^2 of 0.982 and ΔE^*_{ab} around 1.46. We repeated this procedure for the number of nodes in the second hidden layer, and the algorithm selected 77 nodes as optimal. The performance was R^2 of 0.995 and ΔE^*_{ab} just below 1.0, much better than with only one hidden layer. Finally, we checked whether a third hidden layer would improve our results (trying out the same values). The algorithm selected 21 nodes. Next we used grid search to select the activations, optimizer, kernel initializer, number of epochs and batch size. The best architecture was a network with 3 nodes in both input and output layers and three hidden layers with 21, 77 and 21 nodes respectively (Fig. 6). Other optimal parameters were as follows: activation function: sigmoid; optimizer: adam; kernel initializer: uniform; batch size: 50. We trained the network over 50 epochs using back-propagation, i.e. an algorithm for learning the weights at each node.

The data derived from synthetic spectra was scaled by a min-max scaler to vary between 0 and 1. Before training the model, we split the synthetic data into a training set (80%) and cross-validation set (20%) using stratified sampling, so that each generating function was equally represented in the training process.

4. Results for sigmoidal network

The network with sigmoid activations was trained, taking *R,G,B* triplets as input and producing *X,Y,Z* triplets as output. We trained two models: the first using ΔE^*_{ab} and the second using ΔE_{2000} as a loss function. From the results listed in Table 4, it is clear that errors are smaller when the model is optimized over ΔE_{2000} rather than ΔE^*. Both the mean error and its standard deviation are lower in this case. Differences between the two distributions are visually compared in Fig. 7. The model optimized over ΔE_{2000} has smaller errors and 85% of the error distribution is below the nominal visibility threshold (just noticeable difference) of $\Delta E_{2000} = 1.0$, compared with 68% in the case of the model optimized over ΔE^*_{ab}. These results are significantly better than those of regression methods, as reported in Tables 1 and 3.

Table 4. Error statistics for network trained by two methods

Loss function	ΔE^*_{ab}		ΔE_{2000}	
Error metric	ΔE^*_{ab}	ΔE_{2000}	ΔE^*_{ab}	ΔE_{2000}
Mean	1.048	0.886	0.699	0.580
Std dev	0.776	0.716	0.563	0.494
Minimum	0.042	0.029	0.028	0.022
1st quartile (25%)	0.549	0.362	0.335	0.254
2nd quartile (50%)	0.866	0.707	0.579	0.453
3rd quartile (75%)	1.247	1.156	0.827	0.737
Maximum	8.860	10.602	5.322	5.002
% errors below v.t.	60%	68%	82%	85%

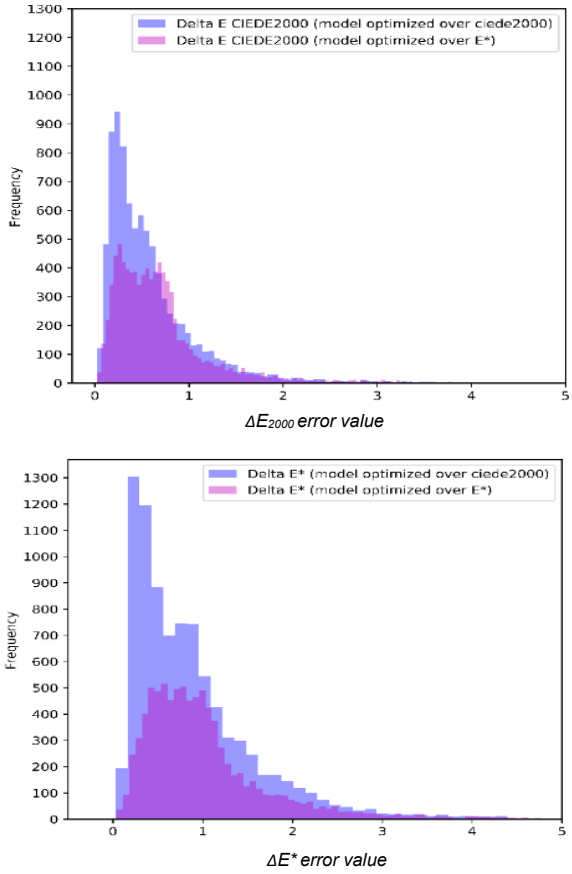

Figure 7. *Occurrence of errors for two forms of optimization*

5. Scalability and noise performance

Despite the symmetrical architecture of our chosen network, commutativity of this model is not automatic, meaning that A→B does not imply B→A. Once the model has been trained, it works only in that direction, i.e. *R,G,B* → *X,Y,Z*. Simply swapping inputs with outputs leads to suboptimal performance of the model with correlation $R^2 = 0.7189$ and mean $\Delta E_{2000} = 0.9468$. However the model with identical architecture can be retrained with inputs swapped for outputs, leading to an equally tight fit: $R^2 = 0.9986$ and mean $\Delta E_{2000} = 0.5980$.

Ideally the data should be scalable, meaning that any scaling factor applied to the input data should cause the output data to change by the same factor (at least within the operating range of the input and output values). For camera characterization this is equivalent to linear tracking of exposure, which does not hold true for most non-linear characterizations [5]. For scaling factors in range [0,1] our sigmoidal network gives approximate linearity (Fig. 8).

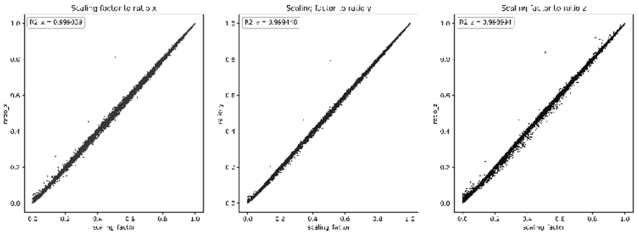

Figure 8. *Ratio of output/input vs input scaling factor for X,Y,Z*

Replotting the difference using the Bland-Altman convention [10] (defined as [prediction with scaled data] / [prediction with original data] - scaling factor) vs scaling factor shows a substantial scatter of results (Fig. 9). The behavior of the three output channels (*X,Y,Z*) is similar but not identical.

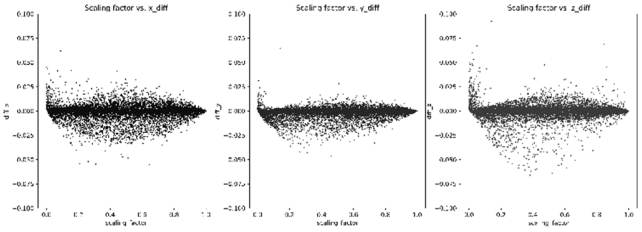

Figure 9. *Ratio of output/input vs input scaling factor for X,Y,Z*

We added Gaussian noise to each *R,G,B* observation, i.e. random noise from a normal distribution with mean 0 and variance 1, with amplitude multiplied by 0.05 to simulate a signal-to-noise ratio of 20:1. Fig. 10 shows noised difference = [model prediction with noised data – model prediction with original data] vs noise.

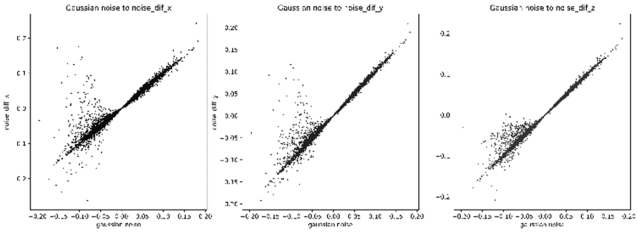

Figure 10. *Differences vs Gaussian noise amplitude for X,Y,Z*

6. Linear neural network

The network with sigmoid activations gives the smallest color errors for the training and test datasets, but performs poorly for scaling of the data and in the presence of noise. We therefore sought to find an alternative network with linear activations by running a grid search over the space of plausible parameter values as follows:

- Number of nodes in each layer {3,33,63,93,123,153,183,213}
- Optimizer selection {adam, sgd, rmsprop, adagrad, adadelta}
- Kernel initializer {'uniform', 'lecun_uniform', 'normal', 'zero', 'glorot_normal', 'glorot_uniform', 'he_normal', 'he_uniform'}
- Batch size {50, 100, 150, 200, 250}
- Epochs {25, 50, 75, 100, 125}

The first round of search resulted in 63 as the optimal number of nodes and, even if all other hyper-parameters were selected optimally, the network could not be trained to achieve a fit with R^2 above 0.98. Hence another hidden layer was added, in which the optimal number of nodes was found by grid search to be 33. The search was repeated for other numbers of nodes around 63 and 33, but did not further improve performance. In each search, the model fit was evaluated using KFold cross-validation with cv=3 folds, meaning that the training data was split into 3 parts and a model was fitted to each. The optimality was determined by averaging the model's performance on these 3 folds. The network was trained over 50 epochs with a callback, so that at each epoch of the training procedure, the model was saved only if it was the best so far. Thus the resulting model was the one that performed best on the validation dataset (not necessarily the one trained in epoch 50). The final selected linear network has four layers with 3-63-33-3 nodes.

Table 5. Performance of sigmoidal and linear networks

	Sigmoidal		Linear	
	Loss	R^2	Loss	R^2
Training set	0.7532	0.9984	0.9586	0.9974
Validation set	0.7490	0.9985	0.9610	0.9974
Test set	0.5802	0.9985	0.6617	0.9978

The performance of the linear network is poorer than that of the sigmoidal network, in terms of both greater mean loss (ΔE_{2000}) and lower coefficient of correlation (Table 5). But this is compensated by the very much better scalability, with a reduction of the standard deviation by factors of at least 10x (Table 6). The linear model is also quite stable for scaling factors above 1, even greater than 10, and more resistant to the presence of noise (Fig. 11).

Table 6. Difference errors for scaling of sigmoidal network

	Sigmoidal		Linear	
	Mean	Std dev	Mean	Std dev
X	-0.001127	0.008851	-0.000521	0.000836
Y	-0.001798	0.006540	-0.000225	0.000382
Z	-0.000523	0.009126	0.000428	0.000643

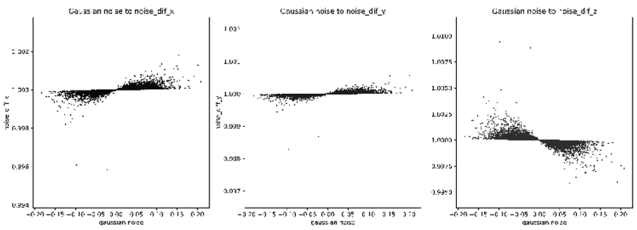

Figure 11. *Differences vs Gaussian noise amplitude for X,Y,Z (linear model)*

7. Discussion

We chose a network architecture with three hidden layers rather than one very wide hidden layer, as was used in [4]. With our training data, one hidden layer was not sufficient to achieve a good fit. Two hidden layers performed better but the best results were achieved with three hidden layers. Four or more hidden layers led to over-fitting. We tried four types of activation functions: linear, relu (rectified linear unit), tanh and sigmoid. Of the four, sigmoid activations achieved the best fit in the training. Linear activations led to comparatively slower learning of the network and the results eventually stopped improving despite many learning epochs, suggesting that the relationship between inputs and outputs is nonlinear. Relu activations led to faster learning during initial epochs but later to over-fitting. Tanh activations performed only slightly worse than sigmoid activations. Our summary conclusions are that sigmoid activations enable the best error performance but that linear activations enable the best scalability.

When the logistic function was removed, giving linear input to each node, the model did not fit the data so well. It achieved rather good correlation, with $R^2 = 0.9976$, but the losses were higher than in the selected model and, to make matters worse, there were clear signs that the model was trying to memorize training data and did not generalize even to cross-validation data. We conclude that, in general, the nonlinearity introduced by the sigmoid helps the model to train better and, more importantly, to generalize better. In effect the linear input has to clip over-range values to min or max, whereas the sigmoid avoids this problem by its asymptotes.

The sigmoid model architecture we selected has 3 nodes in the input layer, followed by 21, 77, and 21 nodes respectively in the subsequent hidden layers, and 3 nodes in the output layer (Fig. 6). Since we have used a fully-connected dense network, the model has 3482 parameters to be estimated: 4x21 (between the input and first hidden layer, where 4 stands for 3 inputs plus 1 bias unit), 22x77 parameters between the first and second hidden layers, 78x21 parameters between the second and third hidden layers and finally 22x3 parameters in the output layer.

The optimal architecture of a network depends on how well the models fit the data. The empirical procedure is to start with simple models and continue to make them more sophisticated until their performance is satisfactory. While there has been some research on using data characteristics to determine the network architecture [8], there is still no standard approach in deep learning, and the performance of such models is not yet very stable.

The minimum size of the training dataset was investigated by fixing the model architecture to 3-21-77-21-3 and iteratively reducing the number of training samples in units of 8,000 (choosing equal-sized random samples from the training data derived from synthetic reflectance spectra). The performance of the network, measured as the percentage of test sample errors below the visibility threshold of $\Delta E_{2000} = 1.0$), is plotted against the number of training samples in Fig. 12, showing that the performance is maintained down to about 100,000 training samples. Only below this number does the performance fall away. The noise in the graph arises from the random weight initialization in the training phase of the network.

The networks were trained over 50 epochs. We experimented with different numbers of epochs and stopped training when the performance stopped improving. If training were to continue beyond this point, the model would not learn more about the relationships within the data, but just start to memorize the training set [11].

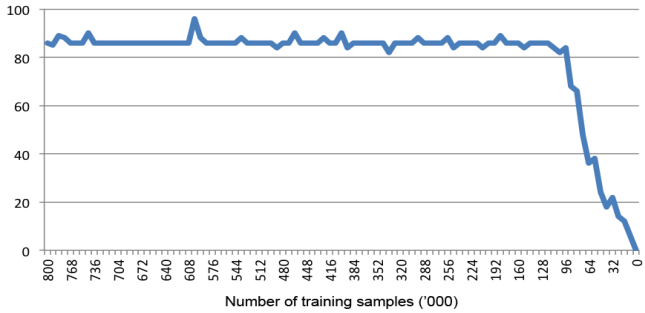

Figure 12. *Performance of sigmoidal model vs number of training samples*

It is not obvious that any meaning can be inferred from the estimated weights applied to the data passed at each layer. Deep learning models are known as 'black-box models' for good reason, because their internal structure is concealed, and it is difficult to interpret how the model does what it does and why it works. Since we have so many nodes in so many layers, it is almost impossible to assign any meaning to what is happening at each layer. The beauty of a neural network (as opposed to a linear regression model, for example) is that it's the model that decides which features in the data to use, by assigning weights. In a regression you need to say only: use first-, second- or third-order, fit the data and estimate all the coefficients. But when training a neural network, the model itself creates the features, or variables, it uses by linear or nonlinear transformations of the inputs and their combinations.

It is interesting to speculate on what the topology of our network suggests about color vision and human neural processing. The dense five-layer interconnected network with sigmoidal activations is not unlike the layered organization of neurons in the retina (literally a neural network). It is reckoned that for a human observer the total number of discernible colors is approximately 2 million [9]. This is a measure of the span of the retinal mapping from an infinite number of incoming spectra to the perceived triplets encoded in a luminance-chrominance domain (akin to *Yuv*) in the optic nerve to the visual cortex. So if we can say that a certain network architecture (numbers of layers and nodes) is necessary and sufficient to encompass this level of complexity then we have established something about retinal neuronal architecture.

The number of possible pathways through our neural network is the product of the number of nodes at each layer. For the sigmoidal network architecture it is 3x21x77x21x3 = 305,613 and this is one measure of the complexity of the network. But this number of training data items is not necessarily needed, because the number of paths is not crucial. A more reasonable indicator is the number of parameters (weights on the connections between nodes), which in our case is 3,482 (with biases), but in general the measure of complexity depends on the problem [7].

Our models do not use any convolutional layers, because the color stimulus at each point in the scene is assumed to be independent of its neighbors. In real environments, the retinal array is viewing the whole scene, so the colors of neighboring regions have an effect on perception. Retinal neurons are interconnected through multiple layers of the network, particularly at the amacrine and ganglion levels. For example, in the well-known phenomenon of simultaneous contrast the effect is to increase local color contrast between center and surround. Hence a spatial aggregating process, implemented in image processing by convolution, is required.

In computer vision algorithms, convolution is often employed to identify features in the scene, and to decrease the number of parameters the model has to learn. In machine learning, convolutional networks are justified by the efficiency of estimation [12]. In a dense network all weights are independent (so many more parameters need to be estimated), whereas in a convolutional network the weights of neighboring pixels in each convolution are assumed to be the same, no matter where in the picture the convolution filter is moved. Thus dense networks are much less efficient because they have many more degrees of freedom. In very large problems, it is not possible to use dense networks because of limits on the computational power available.

In conclusion, we have shown that a dense neural network can give very good performance in mapping from a camera *R,G,B* domain to a device-independent *X,Y,Z* domain. The key to its success is the availability of a very large training dataset, generated from synthetic reflectance spectra that span the gamut of all physically realizable colors. The training and testing datasets used in this study were computed from the spectral sensitivity of the Nikon D200 camera to generate *R,G,B* and from the Standard Observer and D65 illuminant to generate *X,Y,Z*. Any other camera, observer and illuminant could be used to compute corresponding datasets, but these would be metamers of the data for the reflectance samples. We assert that the performance of the neural network would be little affected by these parameters.

References

1. MacDonald, L.W. (2012) 'Colour Laser Scanner Characterisation by Enhanced LUT', *Proc. 6th IS&T European Conf. on Colour in Graphics, Imaging and Vision* (CGIV), Amsterdam, 137-142.
2. Buchsbaum, G. and Gottschalk, A. (1984) Chromaticity coordinates of frequency-limited functions*, J. Opt. Soc. Am. A*, 1(8):885-887.
3. MacDonald, L.W. (2015) Determining Camera Spectral Responsivity with Multispectral Transmission Filters, *Proc. IS&T Color Imaging Conf*., Darmstadt, 12-17.
4. Fdhal, N., Kyan, M., Androutsos, D. and Sharma, A. (2009) Color space transformation from RGB to CIELAB using neural networks, *Proc. Pacific-Rim Conf. on Multimedia*, pp. 1011-1017. Springer, Berlin.
5. Vazquez-Corral, J., Connah, D. and Bertalmío, M. (2014) Perceptual color characterization of cameras. *Sensors*, 14(12):23205-23229.
6. Jackman, P., Sun, D.W. and El Masry, G. (2012) Robust color calibration of an imaging system using a color space transform and advanced regression modelling. *Meat science*, 91(4):402-407.
7. Brownlee, J. (2017) How Much Training Data is Required for Machine Learning? *Machine Learning Process.* https://machinelearningmastery.com/much-training-data-required-machine-learning/
8. Tanno, R., Arulkumaran, K., Alexander, D., Criminisi, A. and Nori, A. (2019) Adaptive Neural Trees. *Proc. 7th Intl. Conf. on Learning Representations (ICLR)*. arXiv preprint 1807.06699.
9. Pointer, M.R. and Attridge, G.G. (1998) The number of discernible colours. *Color Research & Application*, 23(1):52-54.
10. Bland, J. M. and Altman, D. (1986) Statistical methods for assessing agreement between two methods of clinical measurement. *The lancet*, 327(8476), 307-310.
11. Arpit, D., Jastrzębski, S., Ballas, N., et al. (2017) A closer look at memorization in deep networks. *Proc. 34th Intl. Conf. on Machine Learning,* 70:233-242, JMLR.
12. Snoek, J., Larochelle, H., and Adams, R.P. (2012) Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems* (NIPS 25), 2951-2959.