Retrieval of Color Space Conversion Matrix via Convolutional Neural Network

Larry Pearlstein, Alexander Benasutti, Skyler Maxwell, Matthew Kilcher, Jake Bezold, and Warren Seto

***Abstract*—The problem of proper identification of the color space associated with digital luma and chroma data has been widely reported by video processing professionals. The problem arises from confusing, and sometimes conflicting, statements regarding color space usage and description. Although standards allow for the carriage of descriptive metadata regarding color space, some applications do not require the presence of such metadata. Those standards typically recite assumptions on color space that should be followed in the absence of embedded color space descriptors. The unfortunate result of this approach is a state of confusion in the industry, and consequently the possibility of errors in rendering the output of decoded video and images. Our work represents the first known attempt to determine color space directly from luma/chroma pixel data, and provides an alternative to sole reliance on potentially missing or incorrect metadata, or weakly followed defaults. Although a color space is defined by many parameters, such as primary chromaticity, transfer characteristics and matrix coefficients, we chose to focus on determining which standard for matrix coefficients had been used to create a given luma/chroma image. We addressed the problem via deep convolutional neural networks (DCNNs), trained on millions of images. Our results are encouraging, and suggest that DCNNs can be used to solve this ill-posed problem.**

***Index Terms*—BT.601, BT.709, color space, convolutional, neural network.**

I. Introduction

The representation of color imagery via conversion from intensities of red, green, and blue primaries, to luma plus two chroma components, has been extensively applied in consumer electronics and related imaging systems. Early applications were in devices and systems related to compatible color television [1], and current applications abound in the areas of digital video and image compression [2]. Standards have been developed for representing color imagery to allow for interoperability between content producers and consumers. Such standards generally cover at least three areas: the chromaticity of primary color light emitters, the (typically) non-linear transfer characteristic between light intensity and represented intensity, and the conversion matrix coefficients. We use the term *color space* as an aggregation of these three areas of standardization. There are two standards for color space in common use, both developed by the International Telecommunication Union - Radiocommunication Sector (ITU-R), BT.601 [3] and BT.709 [4]. Following Poynton [2] we refer to a digital representation of color in terms of the non-linear mapping of intensities of red, green and blue primaries as *R'G'B'*, and the results of matrix conversion to luma plus chroma as *Y'CbCr*. In addition, following general usage, we refer the process whereby pixel sample values are converted from R'G'B' to Y'CbCr as *matrixing*, and the inverse process as *dematrixing*. Because matrixing and dematrixing are linear operations on real vectors we use the generic term *space* when convenient, to refer to the set of R'G'B' values or Y'CbCr values.

Digital image and video compression systems typically represent color components using the Y'CbCr space, and the results of decompression must ultimately be converted to R'G'B' space before reproduction by a display. In order to perform the conversion, the video device or system must determine whether the matrix specified by BT.601, or BT.709 was used to produce the Y'CbCr representation. Unfortunately, in practice this determination is often made with low confidence. The problem of proper identification of the color space associated with Y'CbCr data has been widely reported in online discussion boards, and arises from confusing, and sometimes conflicting, statements regarding color space usage and description. Although standards allow for the carriage of descriptive metadata regarding color space, some applications do not require the presence of such metadata. Those standards typically recite assumptions on color space that should be followed in the absence of embedded color space descriptors. The lack of strict requirements on metadata usage and color space selection has resulted in a state of confusion in the industry, and consequently the possibility of errors in rendering decoded video and images.

Our work represents the first known attempt to determine color space directly from luma/chroma pixel data, and provides an alternative to sole reliance on potentially missing or incorrect metadata, or an algorithm for defaulting to a color space standard based on pixel resolution or other factors. We explored the use of deep convolutional neural networks (DCNNs), trained on millions of images, for determining the matrix coefficients directly from Y'CbCr pixel data. Our results are encouraging, and suggest that a DCNN-based approach can lead to improved decisions for dematrixing.

This paper is organized as follows. The history of color space handling in JPEG, MPEG, and ATSC is reviewed in Section II. We present details of matrix conversion and potential clipping of RGB values in Section III. Our

Manuscript received March 13, 2019; revised June 19, 2019.

L. Pearlstein, S. Maxwell (email), A. Benasutti (email), M. Kilcher (email), J. Bezold (email) are with Department of Electrical and Computer Engineering, The College of New Jersey, Ewing, NJ, 08628 USA (e-mail: pearlstl@tcnj.edu, maxwels2@tcnj.edu, benasua1@tcnj.edu, kilchm2@tcnj.edu, bezoldj1@tcnj.edu).

W. Seto was with Department of Electrical and Computer Engineering, The College of New Jersey, Ewing, NJ, 08628 USA (e-mail setow1@tcnj.edu).

doi: 10.18178/ijmlc.2019.9.4.816

experiments and results are described in Section IV and conclusions are presented in Section V.

II. Color Space Handling in JPEG, MPEG and ATSC

The JPEG File Interchange Format (JFIF) was published in 1992 [5]. Its goal was to enhance JPEG interoperability, and included a restriction that the color space conversion matrix must be based on the BT.601 model. On the other hand, the MPEG-1 standard for video compression, published in 1993 [6], did not include any means to specify color space information, or guidance regarding color spaces. The original MPEG-2 standard, ratified in 1995 [7], provided an option for embedding color space metadata, and required the decoder to assume BT.709 matrix coefficients if this metadata were not included. However, the updated MPEG-2 standard (2000) states that, in the absence of relevant metadata, "the matrix coefficients are assumed to be implicitly defined by the application" [8]. In 1995, the Advanced Television Systems Committee (ATSC) Digital Television Standard [9] specified that its application default color space should be assumed to be BT.709.

In 1999, Charles Poynton presented the following at the SMPTE Technical Conference [10]:

The coefficients of Rec. 601 are ubiquitous in conventional 525/59.94 video, 625/50 video, and computing. But according to recently-adopted SMPTE and ATSC standards, ATV and HDTV will use a new, different set: the luma coefficients of SMPTE 240M (BT.709). This introduces a huge problem: There will be one flavor of Y'CbCr for small (SDTV) pictures, and another for big (HDTV) pictures.

In 2006, ATSC Recommended Practice A/54A stated that "broadcasters should understand that some receivers will display 480-line formats according to SMPTE 170M (BT.601) colorimetry and 720- and 1080-line formats according to SMPTE 274M (BT.709) colorimetry," confirming Poynton's concern [11].

III. Matrix, Dematrix and Clipping Analysis

The process of matrixing from R'G'B' space to the Y'CbCr can be represented as:

$$q_k = M_k p \tag{1}$$

where $p_{3\times1}$ represents an R'G'B' pixel, M_k represents the 3×1 conversion matrix for system k, and $q_{3\times1}$ represents the pixel p in luma/chroma space. For convenience, and without loss of generality, we assume that the ranges of values are as follows:

$$\begin{bmatrix}0\\0\\0\end{bmatrix} \le p \le \begin{bmatrix}1\\1\\1\end{bmatrix} \tag{2a}$$

$$\begin{bmatrix}0.0\\-0.5\\-0.5\end{bmatrix} \le q_k \le \begin{bmatrix}0.0\\0.5\\0.5\end{bmatrix} \tag{2b}$$

Based on these definitions, the conversion matrices corresponding to Rec. BT.601 and Rec. BT.709 are:

$$M_{601} = \begin{bmatrix}0.2990 & 0.5870 & 0.1140\\-0.1687 & -0.3313 & 0.5000\\0.5000 & -0.4187 & -0.0813\end{bmatrix} \tag{3a}$$

$$M_{709} = \begin{bmatrix}0.2126 & 0.7152 & 0.0722\\-0.1146 & -0.3854 & 0.5000\\0.5000 & -0.4542 & -0.0458\end{bmatrix} \tag{3b}$$

respectively. The process of matrixing is illustrated in Fig. 1.

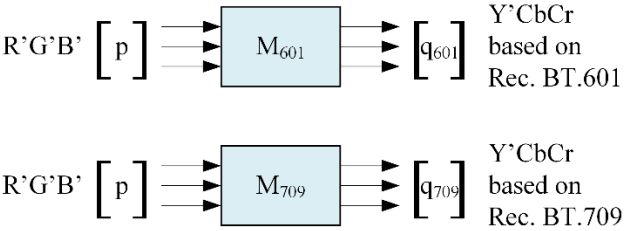


Fig. 1. Matrixing from R'G'B' to Y'CbCr spaces.

The matrices M_{601} and M_{709} are well-conditioned and easily inverted. Clearly, the original R'G'B' values can be recovered by dematrixing via application of the appropriate matrix inverse. However, as outlined in Section I, there are situations where one cannot determine with certainty which set of matrix coefficients were used to generate an image in Y'CbCr space. Thus, the incorrect matrix inverse may be applied at times, as illustrated in Fig. 2.

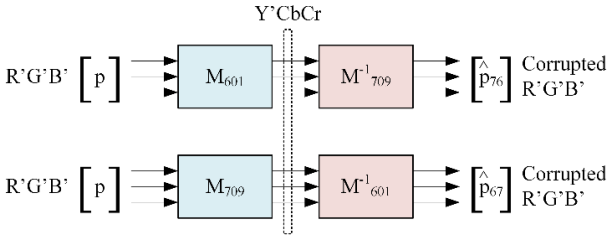


Fig. 2. Improper dematrixing from Y'CbCr to R'G'B' spaces.

Mathematically we represent these scenarios as:

$$\hat{p}_{76} = M_{709}^{-1} \cdot M_{601}\, p = M_{76} p \tag{4a}$$

$$\hat{p}_{67} = M_{601}^{-1} \cdot M_{709}\, p = M_{67} p \tag{4b}$$

where $M_{76} \triangleq M_{709}^{-1} \cdot M_{601}$, and $M_{67} \triangleq M_{601}^{-1} \cdot M_{709}$, which are the effective improper transformations from input R'G'B', through Y'CbCr, to dematrixed R'G'B'.

Simple numerical analysis yields:

$$M_{67} = \begin{bmatrix}0.9136 & 0.0785 & 0.0079\\-0.1050 & 1.1722 & -0.0671\\0.0096 & 0.0322 & 0.9582\end{bmatrix} \tag{5a}$$

$$M_{76} = \begin{bmatrix}1.0864 & -0.0723 & -0.0141\\0.0965 & 0.8451 & 0.0584\\-0.0141 & -0.0277 & 1.0418\end{bmatrix} \tag{5b}$$

Examination of (5a,b) suggests that the differences between the effective improper transformations and an identity matrix are not insignificant. In particular, M_{67} boosts saturated green colors, and reduces the intensities of saturated reds, blues, and purples. The transformation M_{76} results in a complementary set of errors. When $p = [c \quad c \quad c]^t$, *i.e.*, a

gray-shade pixel, then $\hat{p}_{kl} = p$, and no distortion is introduced as a result of the incorrect choice of matrix inverse.

We observe that the transformations M_{67} and M_{76} can result in a conversion to R'G'B' values that violate the range constraints of (2a), even when the original R'G'B' values satisfy these constraints. The upper range constraints can be violated in cases of maximum intensity, highly saturated colors that align with coefficients of M_{67} and M_{76} that are greater than unity, and the lower range constraints can be violated for any intensity of highly saturated colors that align with the coefficients that are less than zero. In such cases clipping will occur as part of the dematrixing process, due to enforcement of the constraints. We further observe that, when the proper matrix inverse is applied to q_k with infinite precision, the results will always satisfy the range constraints.

These observations suggests that, for images where clipping would occur when the improper matrix inverse is applied, one could test both matrix inverses, and simply choose the one that does not produce R'G'B' values outside of the legal range. We formulate the problem in terms of a hypothesis test, where the hypotheses are:

$$H_{601}: M_k = M_{601} \tag{6a}$$

$$H_{709}: M_k = M_{709} \tag{6b}$$

Here we assign H_{601} to represent the hypothesis that the matrix M_k is based on BT.601 and H_{709} for the hypothesis that the matrix is based on BT.709. We denote the hypothesis test described above by H_1, which can be expressed more formally as a function of a Y'CbCr image $G = q(x, y)$ as:

$$H_1[C(G)] = \begin{cases} H_{601}, C < 0 \\ H_{709}, C > 0 \\ \text{erasure}, C = 0 \end{cases} \tag{7}$$

where

$$C(G) = \sum_{x,y} I\left(\begin{bmatrix}0\\0\\0\end{bmatrix} \le M_{709}^{-1} q(x,y) \le \begin{bmatrix}1\\1\\1\end{bmatrix}\right) - \sum_{x,y} I\left(\begin{bmatrix}0\\0\\0\end{bmatrix} \le M_{601}^{-1} q(x,y) \le \begin{bmatrix}1\\1\\1\end{bmatrix}\right)$$

and $I(\cdot)$ represents the Boolean indicator function. The statistic C is the difference between the counts of pixels that remain in the legal range after conversion to R'G'B' when Rec. BT.709 is assumed vs. when Rec. BT.601 is assumed. If the pixels $q(x, y)$ were obtained directly via application of (1) then at least one of these counts is equal to the total number of pixels in the image. It is entirely possible that both counts will be such, which would lead to $C(G) = 0$, and $H_1[C(G)] =$ erasure (no basis for decision).

The detector H_1 has a serious weakness. It provides no basis for a decision when there would be no clipping under either hypothesis. As noted above, clipping to the upper limit can only occur where there is both maximal intensity and maximal saturation of certain colors. To study this possibility we created modified versions of our datasets where the contrast was artificially deflated by 0.9 to dramatically reduce, or eliminate, this sort of clipping.

IV. DCNN

In recent years, deep convolutional neural networks (DCNNs) have been widely applied to problems in computer vision and image processing [12]. Most recently, there has been work on addressing the color constancy problem via DCNN [13]. In that work, Afifi used an architecture similar to AlexNet [14], with an added input semantic mask, to recover color constancy parameters. Considering the ability of DCNNs to extract subtle information regarding ill-posed color problems, it seems reasonable to ask whether a DCNN can be used as the basis for a detector of the Y'CbCr conversion matrix based on converted pixel data. In particular, a DCNN detector might be able to use some semantic understanding of a scene to be able to work effectively, even when detector H_1 would declare 'erasure'.

A wide variety of DCNN architectures have been proposed, which vary in number of free parameters, computational complexity, and effectiveness for computer vision applications [15]. We selected some notable DCNN architectures for exploration, as listed in Table I.

TABLE I: DCNN ARCHITECTURES USED FOR EXPLORATION

DCNN Network Architecture
AlexNet [14]
Inception [16]
Inception with Batch Norm. [17]
ResNet [18]
ResNext [19]
MobileNet [20]

Development of a DCNN generally involves the following:

1) Select/acquire a dataset. Perform data format conversion, as necessary, according to the requirements of the tool-chain used.
2) Randomly partition the dataset into a training subset, and a validation subset.
3) Select a network architecture, which includes the organization of layers of various types, and various additional hyper-parameters, such as the number of feature maps per layer, convolution kernel sizes, sizes of fully connected layers, learning rates, number of epochs for training and the loss function used.
4) Train the network by repeatedly applying the entire training subset, in units of mini-batches, and allowing the neuron weights to adapt based on gradient descent to minimize the loss function.
5) Evaluate the quality of the trained network by applying the validation subset, and measuring the quality of network results on this previously unseen data.
6) If results are acceptable then stop, otherwise go to Step (3).

Our DCNN work relating to each of these steps is described further below.

A. Dataset and Partitioning

We used the popular ILSVRC 2012 portion of the ImageNet dataset as the basis for training and validation [21]. We used the standard partitioning into a training set of 1,281,167 images and a validation set of 50,000 images, although we made no use of any labels provided with the datasets. For each image we cropped to the maximal centered square sub-image, and then scaled to obtain a pixel resolution

256 × 256. Each formatted R'G'B' image was then converted into Y'CbCr space via both M_{601} and M_{709}, and the resulting images were labeled according to the conversion matrix used. Thus, our total training set contained about 2.5 million images, and our validation set contained 100,000 images. The process of data formatting is illustrated in Fig. 3.

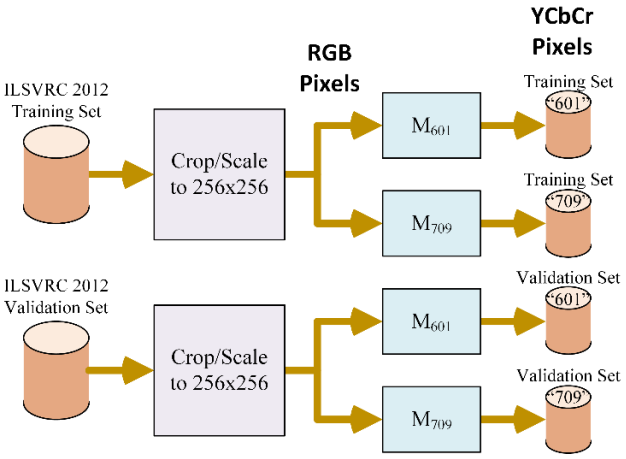


Fig. 3. Block diagram of the formatting process applied to ImageNet data.

In our experiments, we represented image data components using unsigned eight-bit integers, representing values in the range [0,255]. It should be noted that it is common practice to use the so-called *studio* range for Y'CbCr data, which is defined as [16,235] for Y' data and [16,240] for Cb and Cr data. We chose to use the full range instead, to attain a slight reduction in quantization effects.

B. DCNN Architecture and Training

The DCNN architectures listed in TABLE were selected, and modified to accommodate two output classes, corresponding to the hypotheses H_{601} and H_{709}. Our experimental programming environment was based on the OpenSource Apache MXNet framework [22]. We trained our networks with 30 epochs of data, using mini-batches of 32 images, with the AdaDelta optimizer [23]. All experiments were carried out under Ubuntu 16.04 on an AMD Ryzen Threadripper 1900X 8-Core Processor, with dual NVidia Titan V processing units. Training typically required about 2-4 days of platform time for each network, but application of a trained network could be performed at a rate of many tens of frames per second.

C. Evaluation

Each DCNN studied was configured to have two, one-hot encoded outputs from a softmax layer, as shown in Fig.. The softmax converts neuron activation values into estimates of the a posteriori probabilities of each class, *i.e.*, $Pr\{H_k|G\}$. The networks were trained to minimize a cross-entropy loss function between the softmax output and the desired class. Classification was accomplished based on which of the two softmax outputs was larger. Since the softmax outputs must be non-negative and sum to unity, this can be represented as:

$$H_2\{G\} = \begin{cases} H_{601}, o(2) < 0.5 \\ H_{709}, o(2) > 0.5 \end{cases} \quad (8)$$

For convenience we declare H_{709} when $o(2) = 0.5$, but in practice this case rarely, if ever, arises.

During training, we monitored the evolution of the loss function and classification accuracy over both the training and validation sets of images. After selecting the best DCNN based on validation data, we ran additional experiments on new data, to test the ability of our network to transfer its learning to classify data from completely unrelated sources.

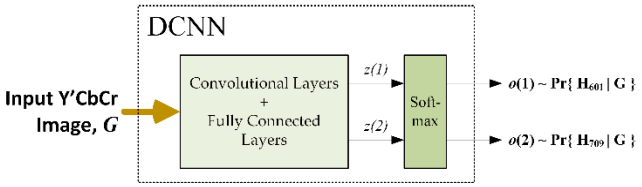


Fig. 4. Block diagram of generic deep convolutional neural network, highlighting Y'CbCr inputs and softmax output layer.

V. Experiments and Results

The datasets used for evaluation and comparison of detectors H_1 and H_2 are listed in Table II. We selected datasets from a variety of sources. As mentioned above, for each of the datasets we created an additional version of the images where the contrast was artificially reduced by a factor of 0.9, to explore the effect of reducing or eliminating the possibility of exceeding the allowed maximum values in R'G'B' space under either hypothesis, H_{601} or H_{709}.

The 'imgnet' dataset was derived from the ILSVRC 2012 validation set, which includes images from an enormous range of scenes. For illustration, some of the images from 'imgnet' are shown in Fig. 5.

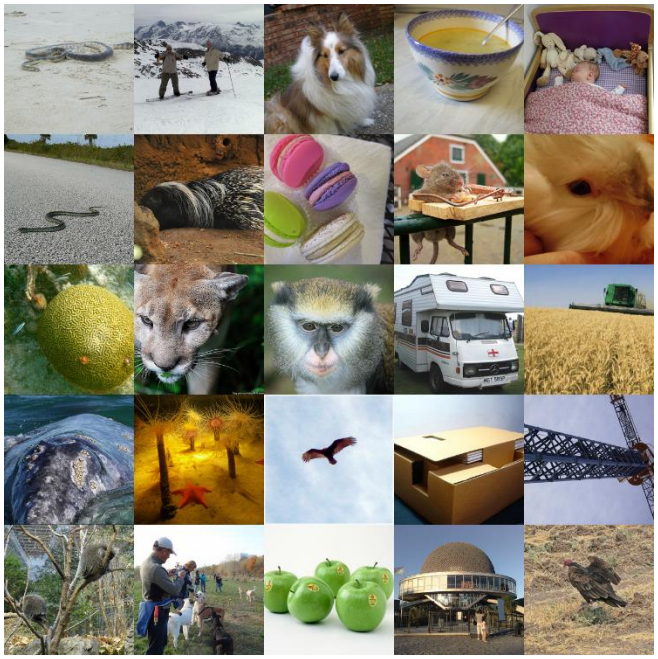


Fig. 5. Representative subset from 'imgnet' dataset.

The 'flow_gdn' and 'suzie' datasets were derived from publicly available moving picture sequences used by the ISO/IEC WG11 SC29 standards body for evaluating MPEG compression. For illustration, frame 60 from each of these sequences is shown in Fig. 6. The 'flow_gdn' dataset contains highly saturated colors due to bright flowers and leaves. The dataset 'suzie' is a sequence of head-and-shoulders images of a woman using a landline style telephone handset, which contains a relatively narrow range of colors. The 'suzie' dataset proved to be particularly challenging.

Fig. 6. Example images from datasets derived from standard MPEG video test sequences: frame 60 from 'flow_gdn' (left), and 'suzie' (right).

To further explore cases with foliage and flowers, we downloaded a relevant clip from the YouTube-8M set of public videos, and extracted frames to create the 'garden' dataset. Example frames from garden are shown in Fig. 7.

Fig. 7. Example images from 'garden' dataset, derived from a selected YouTube-8M video clip.

TABLE II: DATASETS USED FOR EVALUATION

Dataset	# of Images	Range Scaling	Source
imgnet	200	none	ILSVRC 2012
imgnet_0.9	200	0.9	ILSVRC 2012
flow_gdn	360	none	MPEG-2 std. tests
flow_gdn_0.9	360	0.9	MPEG-2 std. tests
suzie	150	none	MPEG-2 std. tests
suzie_0.9	150	0.9	MPEG-2 std. tests
garden	100	none	YouTube-8M vids.
garden_0.9	100	0.9	YouTube-8M vids.

A. *Detector H_1 – Clipping Analysis*

We studied detector H_1 by processing Y'CbCr images, and counting the number of pixels that would have *clipped*, *i.e.*, exceeded the allowed range of R'G'B' values, when converted via the use of M_{601}^{-1} and M_{709}^{-1}. Since no clipping occurs when the correct dematrixing transform is applied, we only tallied the counts of clipped pixels if the incorrect transform were applied. Images where there would have been no clipping under either dematrixing transform were recorded as erasures. The results from application of detector H_1 are shown in Table III.

Examination of Table III reveals that clipping is a common occurrence, and would be a potentially powerful indicator for detecting the conversion matrix. It also indicates that images with reduced contrast may have a significantly lower rate of clipping, as evidenced by the generally high proportion of erasures for the datasets where contrast was scaled by 0.9.

B. *Detector H_2 – DCNN*

Seven different DCNN architectures were trained using a dataset based on the entire set of ILSVRC 2012 Training images (about 1.25 million Y'CbCr training images for each of M_{601} and M_{709}). We trained for 30 epochs and computed validation loss and accuracy after each epoch. For each network, we selected the best epoch based on validation accuracy, and validation accuracy values thus obtained are shown in Table IV. The best network architecture was ResNet34 [18], [24], and its learning curves are shown in Fig. 8. The ResNet approach permits very deep networks by providing bypass paths that reduce the problem of vanishing gradients during training. A ResNet has a first convolutional layer followed by four convolutional *stages*, as shown in Fig. 9.

TABLE III: RESULTS FOR DETECTOR H_1

Dataset	Transf.=M_{76}			Transf.=M_{67}			Eras.
	Clip Lo	Clip Hi	Clip Both	Clip Lo	Clip Hi	Clip Both	
imgnet	32%	79%	29%	65%	38%	29%	22%
imgnet_0.9	30%	2%	2%	64%	0%	0%	53%
flow_gdn	100%	100%	100%	100%	100%	100%	0%
flow_gdn_0.9	100%	0%	0%	100%	0%	0%	0%
suzie	0%	97%	0%	0%	0%	0%	51%
suzie_0.9	0%	0%	0%	0%	0%	0%	100%
garden	66%	100%	66%	100%	81%	81%	0%
garden_0.9	66%	0%	0%	100%	0%	0%	17%

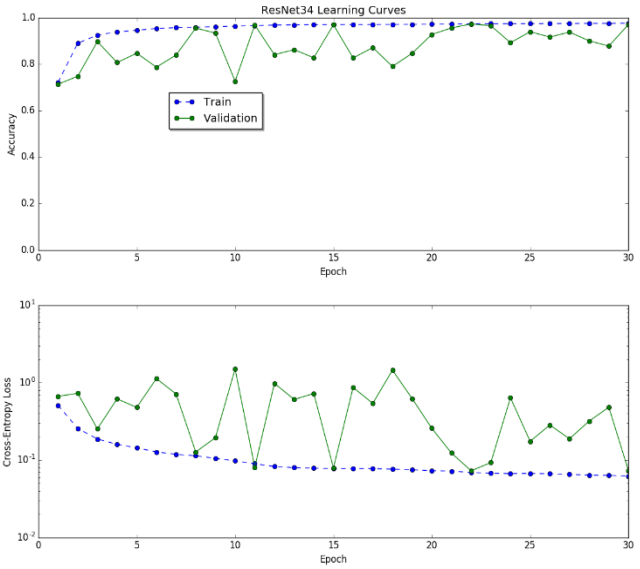


Fig. 8. ResNet34 learning curves, training, and validation using datasets based on entire training and validation sets from ILSVRC 2012. Accuracy (top) and Cross-Entropy Loss (bottom) are shown.

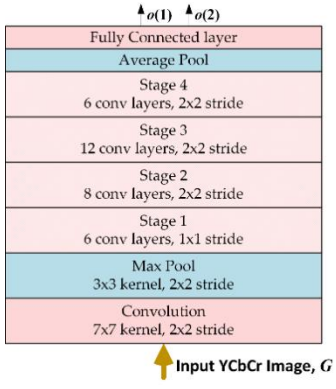


Fig. 9. ResNet34 convolutional neural network architecture, which includes 33 convolutional layers plus one fully connected layer.

Referring to Fig. we see that the best validation accuracy was obtained for Epoch 22, and the Epoch 22 parameter values were used to obtain all of the ResNet34 results described here. The relatively large fluctuations in validation metrics are notable, and are indicative of the ill-posed nature of the problem at hand, and the relatively little separation between the decision classes.

We evaluated our ResNet34 over the eight datasets, and the accuracy results are presented in Table V. The DCNN performed excellently on all of the datasets without contrast scaling, and fairly well on sets with contrast scaling. We compared the performance of the ResNet34 DCNN (hypothesis test H_2), against hypothesis test H_1, by plotting H_1 erasures vs. H_2 accuracy, as shown in Fig.. Note that dashed red line in the figure represents the effective accuracy that would be produced if H_1 erasures were treated as having 50% accuracy. Results that are to the right of the line correspond to datasets where the DCNN was superior, and those to the left of the line are where H_1 was superior.

TABLE IV: VALIDATION ACCURACY VS. DCNN ARCHITECTURE FOR IMAGENET DATASET.

Network	Validation Accuracy (Best Epoch)
MobileNetV1	66.2%
AlexNet	70.9%
ResNet18	75.0%
ResNext34	75.0%
Inception V1	75.4%
Inception BN	91.4%
ResNet34	97.3%

TABLE V: RESNET 34 ACCURACY RESULTS FOR VARIOUS DATASETS

Dataset	DCNN Accuracy $M_k = M_{601}$	DCNN Accuracy $M_k = M_{709}$	DCNN Accuracy Average
imgnet	98%	93%	95%
imgnet_0.9	99%	89%	94%
flow_gdn	100%	99%	99%
flow_gdn_0.9	77%	95%	86%
suzie	96%	92%	94%
suzie_0.9	100%	20%	60%
garden	100%	100%	100%
garden_0.9	98%	71%	84%

We see that the DCNN detector H_2 was superior to clipping-based detector H_1 for four out of the eight datasets considered. This suggests that the DCNN was able to take advantage of semantic-level understanding of the pictures and apply prior understanding of valid object colorization. The internal working of deep networks are notoriously difficult to understand, however certain types of visualizations can produce satisfying insights into their operation [25], [26].

One avenue for gaining insight into the network is to look for common characteristics among the images where the network exhibits its extremes of high/low accuracy. High accuracy is associated with cases where the output probability corresponding to the correct class is high and low accuracy is obtained where the output probability corresponding to the incorrect class is high. We looked at the extremes of network performance across all 100,000 validation images. Examples of images producing the highest accuracy are shown in Fig. 11, and of those producing the lowest accuracy are shown in Fig. 12. Referring to Fig. 11 we note some common themes associated with high accuracy: blue sky, human flesh tones and green plants. Each of these themes represent a particular range of colors found in the natural world, and are related to many examples found in the training data. On examination of Fig. 12, we see some unusual blue tones which might be incorrectly interpreted as sky, but also many images that are dominated by tan or brown tones, especially foxes. We note that foxes have a similar appearance to dogs, are much less common, and have a coloration that is distinctly different from that of most brownish-colored dogs.

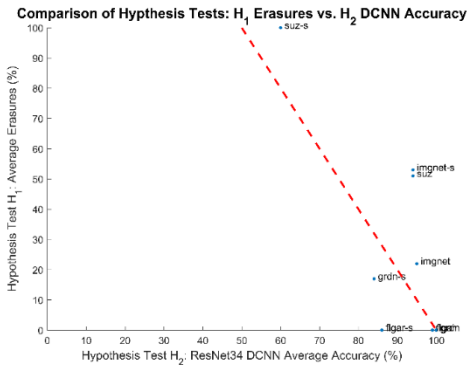


Fig. 10. Comparison between hypothesis test H_1 erasures and H_2 (ResNet34 DCNN) accuracy. Note that dashed red line represents the effective accuracy that would be produced if H_1 erasures were treated as having 50% accuracy. Results that are to the right of the line are where the DCNN was superior, and those to the left of the line are where H_1 was superior.

Fig. 11. Examples of the images where the DCNN output probability corresponding to the correct class is maximized. Note some common color themes -- blue sky, skin tones, green foliage.

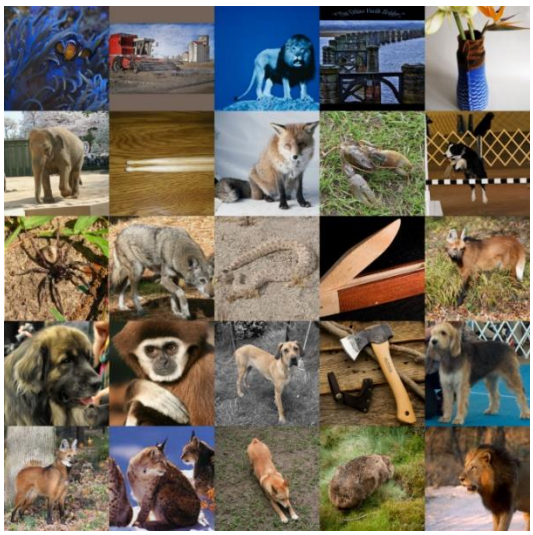


Fig. 12. Examples of the images where the DCNN output probability corresponding to the correct class is minimized. Note the unnatural or uncommon blues and greens. Less obvious is any explanation for the large number of tan and brown tones, especially foxes and dogs.

Another common approach for understanding DCNNs

involves identifying which images produce the maximum activation values in various feature maps. Convolutional layers closer to the input tend to extract simple features and, in our ResNet34 DCNN, they respond very similarly to images converted with either matrix. Examining activation values in later stages we find increasing specificity for certain types of content, responding very differently depending on which matrix was used for conversion. By the final convolutional layer, many of the feature maps are quite specific detectors of conversion matrix.

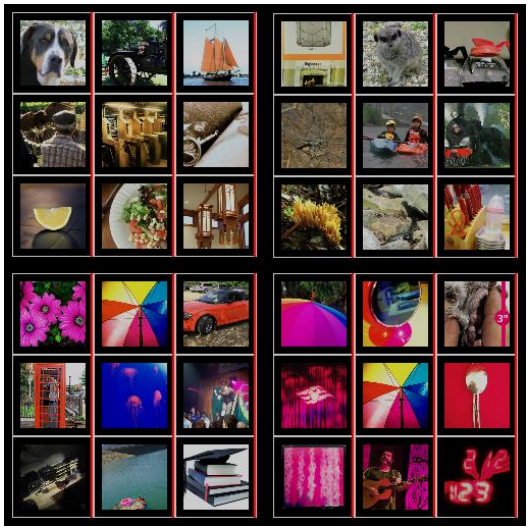


Fig. 13. Sets of images producing the maximum activation values for four of the 256 feature maps at the output of Stage 4. For each of the feature maps the nine images producing the maximum activation values are shown. (Top-left) Images contain browns and oranges. (Top-right) Images contain reds and oranges. (Bottom-left) Images contain pinks and reds. (Bottom-right) Images contain reds. The full-length red stripes at the right side of each box indicate that the corresponding image was ultimately assigned a probability of nearly unity at the output corresponding to the correct class.

For illustration, we present the findings relating to an intermediate convolutional layer in Fig. 13. The figure shows sets of images producing the maximum activation values for four of the 256 feature maps at the output of Stage 4. For each of the feature maps the nine images producing the maximum activation values are shown. It appears that the nine images at the top-left are distinguished by brown and orange colors. The nine images at the top-right contain reds and oranges. At the bottom-left, the nine images contain pinks and reds. At the bottom-right, the images are distinguished by large red regions. The full-length red stripes at the right side of each box indicate that the corresponding image was ultimately assigned a probability of nearly unity at the output corresponding to the correct class.

The finding that a deep convolutional layer in our color space detection network focuses on color swatches stands in contrast to the behavior of typical networks, which are trained for image classification according to objects. Deep convolutional layers in networks trained for the latter task tend to respond to specific types of patterns, *e.g.* snout-looking patches of pixels, text-looking patches, flowery patches, *etc.*

Referring to Table V we see that the one notable failure was on the suzie_0.9 dataset, when the ground truth was the Rec. 709 conversion matrix. For this particular case, the accuracy was only 20%, which is far worse than a simple coin flip, which would yield an accuracy of 50%. The first 144 frames of the suzie_0.9 dataset are shown in Fig. 14. Since all 150 frames of the suzie_0.9 dataset were very similar, we examined the frame-by-frame output of the network on the entire suzie dataset, when the ground truth was H_{709}, to better interpret the network's behavior. A plot of the network output for each frame of suzie_0.9 is shown in Fig. 15. Here, frames where the output value is above the reference line are decided correctly, and those where the output value was below the reference line were decided in error. We note that all of the frames between Frame 45 and Frame 58 were decided correctly. In these frames, the model is tilting her head, and closing her eyes. Ultimately, however, we are unable to determine the root cause for the failures on most of the Rec. 709 images for suzie_0.9, or why the networked performed well on a certain subset of the frames.

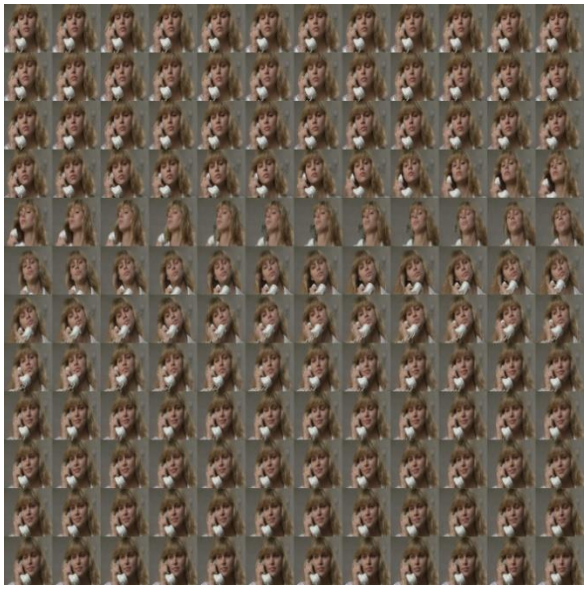


Fig. 14. The first 144 frames of the suzie_0.9 dataset. Note that there are very little frame-to-frame differences in colors or objects depicted.

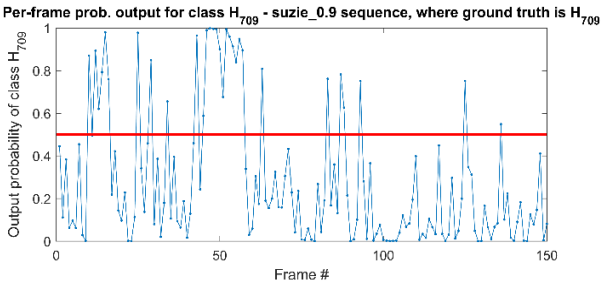


Fig. 15. Per-frame output probabilities corresponding to H_{709} for suzie dataset, restricted to the cases where the ground truth was H_{709}. Frames where the output value is above the reference line are decided correctly, and those below the reference line were decided in error.

VI. Discussion/Conclusion

We presented two novel approaches to determining the color space conversion matrix directly from Y'CbCr pixel data. The method of counting pixels that would result in clipping after conversion can be effective in many cases, and is very simple. The DCNN is more complex but it appears to be capable of significantly better performance than the first approach in many cases, and one can imagine modifications to training that would produce even more robust behavior. Although our work is only a first attempt at solving this

problem, we believe that the results obtained thus far suggest that a DCNN could be used to implement a practical improvement to current consumer electronics products and professional video production tool-chains. Going forward, we plan to train more advanced networks, and to expand our training set to include a number of additional video sequences, including popular television programs and theatrical films, with the expectation that a DCNN could become even more robust.

ACKNOWLEDGMENT

The authors would like to thank Charles Poynton for his help in developing Section II.

REFERENCES

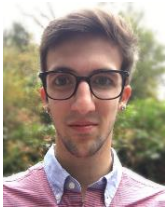
[1] DH Pritchard, "US color television fundamentals: A review," *SMPTE Journal*, vol. 86, no. 11, pp. 819–828, 1977.
[2] C. Poynton, *Digital video and HD: Algorithms and Interfaces*, Elsevier, 2012.
[3] International Telecommunication Union - Radiocommunication Sector, "Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios," *Recommendation ITU-R BT.601-7*, 2011.
[4] International Telecommunication Union - Radiocommunication Sector, "Parameter values for the HDTV standards for production and international programme exchanges," *Recommendation ITU-R BT.709-6*, 2015.
[5] E. Hamilton, *JPEG File Interchange Format*, 2004.
[6] ISO/IEC 11172-2, "Information technology-coding of moving pictures and associated audio for digital storage media up to about 1.5 Mbit/s: Part 2 video," 1993.
[7] ITU-T, "Rec. H.262-ISO/IEC 13818-2:1995 Information technology–generic coding of moving pictures and associated audio: Video," 1995.
[8] ITU-T, "Rec. H.262-ISO/IEC 13818-2:2000 Information technology–generic coding of moving pictures and associated audio: Video," 2000.
[9] Advanced Television Systems Committee, *A/53: ATSC Digital Television Standard*, A/53, pp. 21–26, 1995.
[10] C. Poynton, "Color in 1080p24 and electronic cinema: Converting between R'G'B' and 4:2:2," in *Proc. 141st SMPTE Technical Conference and Exhibition*, 1999, pp. 1–8.
[11] Advanced Television Systems Committee, "Recommended practice: Guide to the use of the ATSC digital television standard," A/54A, 2006.
[12] Y. LeCun, Y. S. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.
[13] Mahmoud Afifi, "Semantic white balance: Semantic color constancy using convolutional neural network," arXiv preprint arXiv:1802.00153, 2018.
[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.
[15] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," arXiv preprint arXiv:1605.07678, 2016.
[16] C. Szegedy, W. Liu, Y. Q. Jia *et al.*, "Going deeper with convolutions," in *Proc. CVPR*, 2015.
[17] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," arXiv preprint arXiv:1502.03167, 2015.
[18] K. M. He, X. Y. Zhang, S. Q. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. European Conference on Computer Vision*, Springer, 2016, pp. 630–645.
[19] S. N. Xie, R. Girshick, P. Dolla´r, Z. W. Tu, and K. M. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5987–5995.
[20] A. G. Howard, B. Chen *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint arXiv:1704.04861, 2017.
[21] O. Russakovsky, J. Deng *et al.*, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
[22] Apache MXNet: A flexible and efficient library for deep learning. (2017). [Online]. Available: https://mxnet.apache.org
[23] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," arXiv preprint arXiv:1212.5701, 2012.
[24] K. M. He, X. Y. Zhang, S. Q. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
[25] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. European Conference on Computer Vision*, Springer, 2014, pp. 818–833.
[26] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," arXiv preprint arXiv:1506.06579, 2015.

Larry Pearlstein received the BSEE from Drexel University in 1982, and the Ph.D. degree in electrical engineering from Princeton University in 1987. He served as the chairperson of the Video Specialists Group within the Advanced Television Systems Committee (ATSC), and led the effort to document the video compression portion of the ATSC Digital Television Standard. While working for ATI, AMD, and Broadcom, he architected video subsystems for consumer electronics chips. He holds 71 US patents in the areas of digital television and consumer electronic. He is currently an associate professor of electrical and computer engineering at The College of New Jersey.

Alexander Benasutti is a sophomore in the Department of Electrical and Computer Engineering at The College of New Jersey, and is an undergraduate researcher. His current research interests involve deep learning, software development, and digital design. He worked as a student researcher for deep convolutional neural networks at The College of New Jersey during the summer of 2018.

Skyler Maxwell is a junior in the Department of Electrical and Computer Engineering at The College of New Jersey. He is an undergraduate researcher at The College of New Jersey and his research interests include image and video processing, and deep convolutional neural networks. He held a summer internship at the US Air Force Research Laboratory, Rome, NY in 2017.

Matthew Kilcher is a junior computer engineering student at The College of New Jersey. He is a member of the Tau Beta Pi honor society, and does undergraduate research in the fields of computer vision and image processing. He has previously worked at Gatekeeper Intelligent Security in Newtown, PA, and will be interning at AT&T as a part of their Technology Development Program (TDP) in Los Angeles, CA for summer 2019.



Jake Bezold is a computer engineer with major in the Department of Electrical and Computer Engineering at the College of New Jersey with an expected graduation date of May 2020. He is an undergraduate student researcher, and he worked as a researcher, applying deep convolutional neural networks to the real world, at the College of New Jersey during summer 2018.

Warren Seto received his bachelors of science in computer engineering from The College of New Jersey in 2018. After graduation, he joined Apple Inc as a human interface device engineer and he is working on the User Experiences Team. His interests include low level systems programming and toying with embedded and edge-computing devices.