ColorNet 2.0: Image Segmentation for Brand Color Correction in Video

Jake Liguori, Jeremy Spooner, Michelle Mayer, Ethan Tompkins,
Erica Walker, and Hudson Smith

Keywords: artificial intelligence, color management, segmentation, software, video

Abstract

Image segmentation involves partitioning an image into multiple regions based on the characteristics of pixels in the image. ColorNet 2.0, a machine learning model, uses segmentation to color correct images without needing to be retrained to correct each individual color. Users choose a target color and have areas of the image that perceptually match the target color selected for adjustment. ColorNet 2.0 improves on the limitations of previous models by allowing multiple colors to be targeted and adjusted by a single model. Initial tests of this model are promising, but some improvements are still needed before being used in a production setting. Future work will focus on improving both the color adjustment code and the efficiency of the neural network architecture to apply adjustments in real time.

Introduction

A commonly used technique in image processing is image segmentation, partitioning an image into regions of pixels with similar characteristics, such as similar textures, colors, or brightness (What Is Image Segmentation? 3 things you need to know, n.d.). For example, image segmentation could involve separating foreground from background or grouping pixels that belong to a common object. In segmentation, each pixel is classified into a distinct category. These categories can then be visualized using different colors, Figure 1.

Applications of image segmentation are widespread. Medical professionals use segmentation for highly accurate labeling of medical scans (Havaei, M. et al., 2017). Each pixel in the image that corresponds to a tumor can be labeled with a different color to help the doctor better identify the tumor's exact shape. Segmentation is also

Clemson University

an important component of autonomous driving systems. Numerous cameras and other sensors continually collect data that self-driving cars use to make decisions (Xu, H. et al., 2017). It is extremely important that the machine learning model can accurately detect and respond to objects that appear in the video feed, such as road markings, detour signs, or pedestrians.

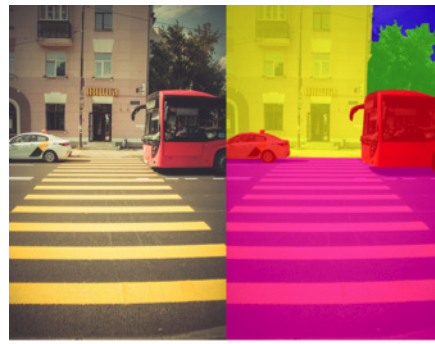

Figure 1. *An example segmentation where pixels labeled as vehicles are colored red, buildings are yellow, etc. (Palac, B, 2020).*

The ColorNet research team at Clemson University has recently developed a novel application of segmentation: color correcting live video feeds (Mayes, E. et al., 2020). One of the most significant issues with the current approach to color management in sports broadcasting is that color adjustments made to the frame impact all pixels. As a result, adjusting the RGB values to make a team's jerseys the appropriate brand color specification can negatively impact players' skin tones or the opposing team's brand colors (Walker, E.B. et al., 2020). The ideal solution would allow the technician to select certain regions of pixels on the screen for color adjustment. ColorNet 2.0 is a segmentation model that allows the user to input a target color and segment out portions of the screen containing pixels that are perceptually similar to the target color, Figure 2.

Figure 2. *Segmentation masks for selecting Clemson orange (middle) and Clemson purple (right) from the original video frame (left).*

The ColorNet 2.0 model builds upon ColorNet 1.5 which successfully used regression to predict the correct RGB values of pixels and perform automatic color correction on sequences of video frames. ColorNet 1.5 demonstrated the correction of Clemson Orange and Clemson Purple, but the model requires more training data to correct for each additional color. In addition, the technician did not have the opportunity to manually tweak the corrections with this previous version. ColorNet 2.0 fixes these limitations by introducing segmentation to create a model that can

correct any specified team color and eventually provide manual control adjustments for the technicians color correcting at the event.

Methods

ColorNet 2.0 uses a Convolutional Neural Network (CNN) to automatically select regions of the input image that correspond to a user-specified target color. The colors in the identified regions can then be modified in either RGB or HSV color space to produce the desired color representation. Importantly, only the identified target-color regions are modified.

The CNN is trained by providing a paired dataset containing unmodified frames taken from game clips and their corresponding segmentation masks. These masks are grayscale images indicating the true regions corresponding to the desired brand color. Through a conventional optimization procedure called Stochastic Gradient Descent, the CNN learns to estimate the probability that each pixel corresponds to the target brand color. The quality of this estimate is evaluated by comparing the predicted probability with the true grayscale mask.

To begin, we created an orange-only version of ColorNet 2.0. The major change from 1.5 was the shift from direct prediction of the correct color to prediction of the segmentation mask from which the correct color can be obtained. Given the model's predicted mask, additional code allows the user to apply a color shift of their choice to the segmented areas.

To progress to a model that could identify any target color beyond orange and purple, we needed a dataset consisting of colors from all across the spectrum. We developed a color augmentation process that uses a frame's segmentation mask to change the color values of the mask's selected pixels, Figure 3. This process is conducted for every input frame in the training data set using randomly selected colors, producing a dataset where each image has a distinct target color. The same is done for the test data set to determine if the model can perform well on colors it may or may not have seen before.

The entire purpose of the model is to adjust colors a broadcast technician or editor believes are displayed incorrectly. Therefore, it is important for the model to learn to segment pixels that are close to the range of the target color provided, even if not exactly equal. We can safely assume each frame of our dataset contains some deviation between the target color and the actual color displayed. When augmenting that dataset, we shift the colors from the original frame, allowing the deviation to persist. The augmented dataset then introduces the model to varying amounts of deviation of the colors within the image relative to the target colors.

Figure 3. *A grayscale mask (middle) can be used to shift the color of target pixels of an original frame (left), creating an augmented image (right). In this example, Clemson orange is changed to the University of North Carolina's blue.*

The next change from orange-only to multicolor exists at the beginning of the model architecture. In the orange-only version, the unaltered frame alone was sufficient for the model input, as it was only predicting orange. In our multicolor version, the model must also receive the target color it should segment.

To incorporate the target color into the model, it is subtracted from the original frame, creating a "difference" image. The result is an array of pixel data where the HSV difference values are closer to 0 for wherever the target color is located. Therefore, the model can learn that areas with HSV values closer to 0 are most likely to be the target color. It is important to note that the hue value is cyclic, meaning values of 0 and 360 degrees are equivalent. To capture the cyclic nature of hue, the hue difference is transformed by a sine function, which outputs values between -1 and 1 and ensures the data represents the shortest difference between hue values on the color wheel. The resulting difference image is then attached to the original, unaltered frame for further processing by the network. Together, the model can now see both the original values of each pixel and their distances from the target color.

In typical image segmentation, the target data takes the form of a binary mask with values of 1 indicating the segmented regions. In our case, however, our target data consists of grayscale masks indicating the confidence that the pixels correspond to the target color. Rather than using a threshold to project these values into a binary mask, we directly optimize our network to predict the target confidence values in the data. Our software shifts the colors of the selected regions proportional to the estimated confidence. As we will show, this produces higher quality segmentation masks. It also removes the burden of selecting a mask threshold.

Results

We used Delta E (ΔE), a measure of color difference, to measure the performance of ColorNet 2.0 against the manually corrected training dataset. A program was written to take an uncorrected frame and find all unique hue values present in the input image. Then, for each distinct hue, the program locates all pixels of this hue and computes ΔE between the original image and the model output image. From this, we can determine which colors the model is shifting and which colors are left alone. Figure 4 shows the performance of the orange-only segmentation model compared to our training data that was manually created in Adobe Premiere. As

expected, the model is shifting the color red by the largest amount, since Clemson Orange is most often incorrectly skewed towards a shade of red. The plots show the lower quartile, median, and upper quartile of the ΔE values for each hue. The lower quartile marks where a quarter of the measured ΔE values are lower and three quarters of the measured ΔE values are higher. The opposite is true for the upper quartile. Based on these graphs, the model is performing fairly similar to the training data and avoiding adjustments in unintended areas.

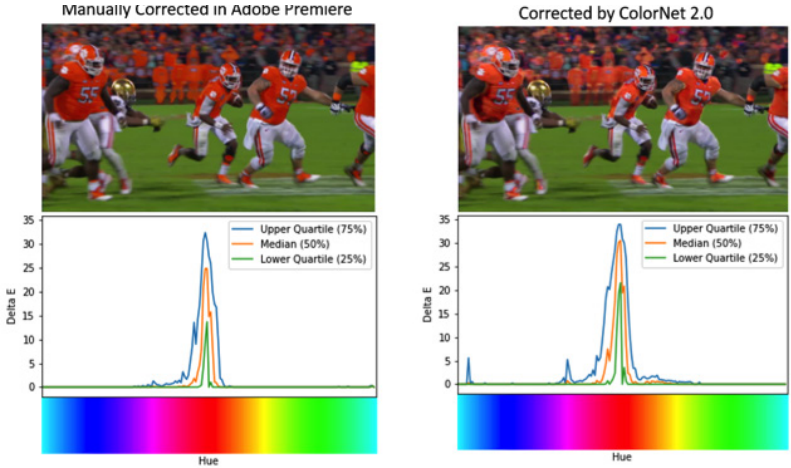

Figure 4. *Evaluating the performance of the ColorNet 2.0 orange-only model with binary segmentation masks. A measure of which hues are shifted in the training image (left) vs the model's output (right).*

With an optimal threshold of approximately 0.25, the binary multicolor model achieved an accuracy of 91.6%, a True Positive rate of 89.0%, and a True Negative rate of 98.8%. These results imply the model is correctly avoiding colors that are not within the target color range, similar to ColorNet 1.0 and 1.5. However, the lower True Positive rate also shows the model is not quite sensitive enough. This can create issues in a binary approach where an area of segmented color may contain holes, leaving a player's uniform or other near-solid block of target color inconsistent and noisy. Still, these results are encouraging that the model's architecture can successfully identify any target color.

To evaluate the grayscale multicolor model, we use metrics of expected accuracy, precision, and recall. Accuracy represents the percentage of correctly classified (masked or unmasked) pixels, precision indicates the fraction of pixels that were predicted to be masked that were actually masked, and recall measures the portion of actually masked pixels that were identified by the model. Table 1 shows the performances of three different model types, with the best version of the grayscale model achieving an accuracy of 94.6%, a precision of 58.9%, and a recall of 60.5%.

Model #	Frame Attached*	Final Kernel**	Accuracy	Precision	Recall
1	Yes	3	0.946 (0.004)	0.589 (0.071)	0.605 (0.073)
2	No	3	0.897 (0.008)	0.311 (0.025)	0.390 (0.032)
3	Yes	1	0.944 (0.006)	0.599 (0.036)	0.634 (0.055)
* Specifies if the original fram was attached to the different image.					
** Specifies the kernel size of the last convolutional layer.					

Table 1.

The "Final Kernel" column denotes the kernel size of the second and final convolutional layer. The convolutional layer performs an average of pixel values across a specific size (1 by 1 or 3 by 3). The size of the final kernel does not significantly affect results. The recall is lower due to a similar struggle as the binary model; the model is not yet sensitive enough to the target color. In many examples, the model does produce a spatially accurate prediction, resulting in the high accuracy score, but the white is not as intense as the original mask, Figure 5. Additionally, the precision appears to be low because the model is predicting low gray values in incorrect areas. The technician can still create good adjustments given a grayscale mask with high accuracy and lower precision and recall, but they would need to make larger adjustments live during the sporting event.

Figure 5. *The predicted mask (left) of an augmented frame (middle) is often spatially accurate but not as intense as the true mask (right). Certain locations of the prediction, such as the shoulders of the players in this example, may also be too gray.*

While statistical metrics are helpful in evaluating the model, a visual test is also required to determine if the model would be of sufficient quality in a broadcast setting. In the tests of the binary model, we noticed more noise in shaded areas. This is caused by the model's lower confidence in shaded areas, leading to some pixels moving from below the threshold to above it and vice versa between consecutive frames. Such artifacts would be unacceptable in a broadcast setting.

Recall the grayscale model does not require a threshold. Consequently, this noisy effect does not exist for this version of the model. If a pixel's confidence drops from one frame to the next, the adjustment will not be entirely removed as it may for the binary model, and instead, the adjustment loses intensity proportional to the loss in confidence. In our tests of the grayscale model, any changes in adjustments from frame to frame seemed seamless and generally undetectable. The tradeoff, however, is that this model is less sensitive and requires the technician to account for this by making larger manual adjustments. Another consideration is that adjustments will not always be the same across the image, unlike the binary model, Figure 6.

Generally, this was observed to occur in frames where part of a player is in the sun and part is shaded. While this effect is still preferable to the alternative of noisy shadows, optimally it should be improved so shaded areas are not partially neglected.

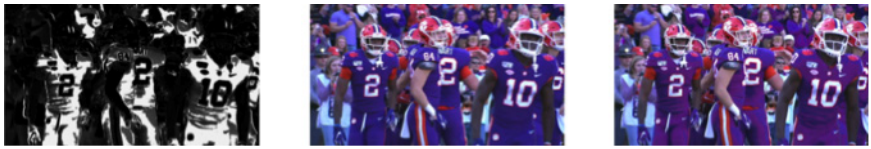

Figure 6. *Shading can produce an inconsistent grayscale mask (left). When applied to the original frame (middle), the adjusted frame (right) will not be noisy but adjustments may be inconsistent across the image.*

Ultimately, we decided the grayscale model is visually the best option for application in broadcast sports. While it provided encouraging results, we will continue improving the model to make definitive, consistent predictions. This will reduce the work of the technician further and provide additional confidence in the model's ability to segment a target color throughout the broadcast.

Conclusion

ColorNet 2.0 builds on the success of ColorNet 1.5, allowing for a more diverse model that no longer needs to be retrained for separate colors. The ColorNet 2.0 model evolved from a pixel-level regression approach to an image segmentation approach. This allows the user to input a target color and select regions of the image containing pixels similar to the target color. This enables technicians to correct an image for multiple colors by passing one frame through the model several times, each time with a different target color. Therefore, it is no longer necessary to create separate models for each color.

Although the initial results of the model have been promising, there are improvements needed prior to production implementation. Moving forward, we are improving the efficiency of the neural network architecture and the color adjustment code to support live adjustments at 60 FPS.

Acknowledgements

This project was supported by a Robert H. Brooks Sports Science Institute (RHBSSI) Seed Grant.

References

Havaei, M., Davy, A., Warde-Farley, D. , Biard, A., Courville, A., Bengio, Y., Pal, C., Jodoin, P., & Larochelle, H. (2017). Brain tumor segmentation with Deep Neural Networks. *Medical Image Analysis (Vol. 35)*. Retrieved October 18, 2021 from https://www.sciencedirect.com/science/article/abs/pii/S1361841516300330?via%3Dihub

Mayes, E., Lineberger, J.P., Mayer, M., Sanborn, A., Smith, H.D., & Walker, E.B. (2020). Automated Brand Color Accuracy for Real Time Video. *Proceedings of the 2020 NAB Broadcast Engineering and Information Technology Conference*. NAB Broadcast Engineering and Information Technology (BEIT) Conference, Las Vegas, NV. Retrieved October 18, 2021, from https://nabpilot.org/beitc-proceedings/2020/automated-brand-color-accuracy-for-real-time-video/

Palac, B. (2020). CC BY-SA 4.0 https://creativecommons.org/licenses/by-sa/4.0. Retrieved October 18, 2021 from https://commons.wikimedia.org/wiki/File:Image_segmentation.png

Walker, E.B., Smith, H.D., Mayes, E., Lineberger, J.P., Mayer, M., & Sanborn, A. (2020). Consistent Display of Clemson Brand Colors Using Artificial Intelligence. *Technical Association of the Graphic Arts (TAGA) Conference, June 2020, Oklahoma City, OK*. Retrieved October 18, 2021, from https://www.printing.org/taga-abstracts/t200074

What Is Image Segmentation? 3 things you need to know. (n.d.). Retrieved October 18, 2021 from https://www.mathworks.com/discovery/image-segmentation.html

Xu, H., Gao, Y., Yu, F., & Darrell, T. (2017). End-to-end Learning of Driving Models from Large-scale Video Datasets. *Computer Vision Foundation*. Retrieved October 18, 2021 from https://openaccess.thecvf.com/content_cvpr_2017/papers/Xu_End-To-End_Learning_of_CVPR_2017_paper.pdf