Project Results Summary and Citations

Created by Unknown User (barrelchester), last modified on Apr 29, 2023

Code

My code is available on github: https://github.com/barrelchester/masters-project.log-space-color-constancy

The work is present in 3 notebooks in the "dev" folder:

color_constancy_experiments-regression.ipynb
color_constancy_experiments-contrastive_model.ipynb
color_constancy_experiments-histogram_models.ipynb

Citations

B. A. Maxwell, R. M. Friedhoff and C. A. Smith, "A bi-illuminant dichromatic reflection model for understanding images," 2008 IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK, USA, 2008, pp. 1-8, doi: 10.1109/CVPR.2008.4587491.

The primary text on log RGB illuminant detection.

Ershov, E., Savchik, A., Semenkov, I., Banić, N., Belokopytov, A., Senshina, D., … Lončarić, S. (2020). The Cube++ Illumination Estimation Dataset. IEEE Access, 8, 227511–227527.

Reference paper for the dataset I used.

Ershov, E., Savchik, A., Semenkov, I., Banic, N., Koscevic, K., Subašic, M., … Nikolaev, D. (2020). Illumination Estimation Challenge: experience of past two years. doi:10.48550/ARXIV.2012.15779

An overview of the recent methods used for color constancy.

Flachot, A., Akbarinia, A., Schütt, H. H., Fleming, R. W., Wichmann, F. A., & Gegenfurtner, K. R. (2020). Deep Neural Models for color discrimination and color constancy. doi:10.48550/ARXIV.2012.14402

An overview of deep learning methods applied to color constancy.

Lo, Y.-C., Chang, C.-C., Chiu, H.-C., Huang, Y.-H., Chen, C.-P., Chang, Y.-L., & Jou, K. (2021). CLCC: Contrastive Learning for Color Constancy. ArXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2106.04989

The paper inspiring my contrastive model experiments.

Y. Hu, B. Wang and S. Lin, "FC^4: Fully Convolutional Color Constancy with Confidence-Weighted Pooling," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 330-339, doi: 10.1109/CVPR.2017.43.

An exploration of a deep convolutional model for color constancy.

Project Results Summary

I used the SimpleCube dataset for all experiments. In each experiment I compare performance on a linearized version of the data and the log of the linearized data. I use the standard color constancy statistics on the test set; the mean, median, trimean, best25, and worst25 angular errors.

One of the highest baselines I used for comparison with my models is that of the Fast Fourier Color Constancy (FFCC) model by Barron et al. on the Gehler-Shi dataset:
Mean: 1.61, Median: 0.86, Trimean: 1.02, Best25: 0.23, Worst25: 4.23

My best model was a GoogLeNet Regression model trained for 200 epochs on linear-expand-log* data, which is competitive or better than this baseline:

Mean: 1.420396, Median: 0.869077, Trimean: 1.242575, Best25: 0.286096, Worst25: 3.530126

*After the linear transform the data is re-expanded to it's original range and then log is applied.

Pre-Trained GoogLeNet Regression Model

These experiments use a pre-trained GoogLeNet model with a regression output layer for predicting the illuminant RGB values. I tested models on data normalized to 0..1, linearized data, the log of the raw pixel values, the linearized values expanded to original range and then log applied ("linear_expand_log"), and linearized data with log applied ("linear_log"). For the other experiments I only used linear and linear_log.

The model with the best test metrics is competitive with the SOTA, and was trained for 200 epochs on "linear_expand_log" data:

Mean: 1.420396, Median: 0.869077, TriMean: 1.242575, Best25: 0.286096, Worst25: 3.530126

These were the best results for any experiment I did.

The model performance for linear and linear_log was quite similar, although I noticed that the training and cross validation curves for linear_log tended to be smoother, which generally leads to better long term convergence stability. It's plausible that continued training would improve the results even more.

I believe that the "linear_expand_log" performed the best because it retained more information in log space due to spreading out the linear values first, preventing them from collapsing into each other when log was applied.

Regression Model with Contrastive Pre-Training

These experiments use the contrastive pre-training method outlined in the paper "CLCC: Contrastive Learning for Color Constancy" by Lo, et al. Again, pre-trained GoogLeNet models were used to encode the images, whose representations were pushed towards those of the same illuminant but different scene, and away from those of the same scene with different illuminants. After pre-training, the weights are copied to a regression model similar to the first experiments. Differential learning rates are employed to discourage "catastrophic forgetting" of the pre-trained weights, with the shallow layers having a very small LR and the deeper layers having a normal LR.

The contrastive pre-training is difficult to control and exhibits some odd behavior. It's sensitive to the distance metric used and the learning rate. I observed some instances where the loss would barely change for 50 epochs and then plunge downward. It would be interesting to examine the activation maps of the network before and after a critical point like this is hit.

The end results were close to but a little less than those of the plain regression model. It's possible that more work on stabilizing the pre-training could yield improvements.

Histogram-based Models

My final experiments were done using various forms of histogram data. I first used concatenated RGB 1D histograms with linear and linear_log data in a simple linear model of 3 layers. The performance was just about equal, with the log version scoring slightly better test metrics. The test results were very poor (mean 9.8 compared to 1.4) compared with direct image processing models. The 1D histograms are unlikely to represent the important 3D structures in color space though, so it's to be expected that the test results would be weak.

Next I projected the 3D RGB points onto a set of 5 2D planes with randomly varying orientations which were globally fixed for all the data. This approach is meant to encode the 3D structures in a compressed manner, in the form of channels provided to a CNN. The CNN consisted of 6 conv layers and 2 fully connected linear layers with an RGB regression output layer.

The training exhibited some very interesting critical points and was generally unstable, varying greatly between runs and sometimes resulting in NaN values on the test set. Despite this, some of the runs resulted in surprisingly strong test scores, the strongest being linear data trained for 100 epochs and yielding **Mean: 1.672695, Median: 0.986252, TriMean: 1.456510, Best25: 0.329702, Worst25: 4.238528**. This is very close to my best score from the GoogLeNet model which took days to train on full image data, whereas this method only took about an hour! This result is very exciting and it would be very worthwhile to continue to experiment and find ways to stabilize the learning.

Future Work

I believe the most exciting result came from the histogram model and would greatly benefit from further investigation, specifically different types of histograms and point projections as well as investigation into stabilizing the training. I also believe that the contrastive model method could be improved with careful tuning of the training procedure to prevent instability. For instances in the contrastive model and histogram training where the loss suddenly drops, it would be interesting to examine the feature activations to determine what critical breakthrough the model may be learning.

No labels