Automatic Image White Balancing using Deep Learning

Blake Sanie
Georgia Institute of Technology
bsanie3@gatech.edu

Dylan Small
Georgia Institute of Technology
dylansmall@gatech.edu

Abstract

White Balancing is a photo-editing technique to change digital photos' colors to better reflect the realistic colors experienced at the scene itself. We create our own white-balancing dataset and introduce a modular way to train large pre-trained image models within this task. The performance of these models are compared to each other and a baseline custom non-pretrained Convolutional Neural Network (CNN) to explore how well each model correctly white balances given photos. We find that pre-trained transfer-learning-based image models best represent latent information about white balancing within images and thus are able to perform best on this task when compared to both pre-trained and non-pre-trained CNNs.

1. Introduction/Background/Motivation

1.1. White Balance

In the field of photography, graphic design, and greater image processing, one common problem is poor camera color calibration while in the field. As a result, images contain a visually-off color profile that do not reflect the true colors experienced at the scene itself. Professionals must rely on post-processing measures to recreate a color presence missed altogether in-camera, as if going back in time. The process of modifying an image's color profile to be neutral to the original scene is referred to by the industry as **White Balancing**.

In practice, adjusting an image's white balance is commonly achieved by tuning the temperature (yellow to blue) slider and tint (green to magenta) sliders in graphical image editing software, such as Adobe Photoshop. Alternately, these programs define another method to neutralize temperature and tint: manually selecting a scene-neutral tone, which then adjusts white balance such that the point also becomes image-neutral.

Scene-neutral tone: A color that may appear off-neutral in the image, but is neutral with respect to the scene captured.

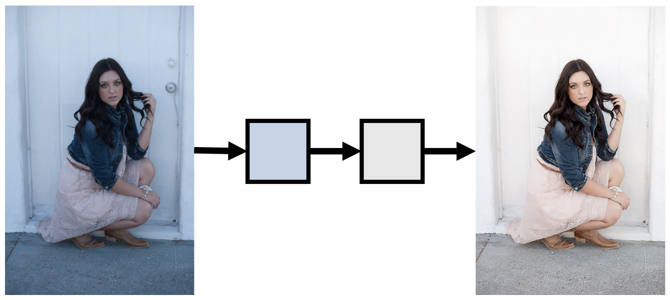

Figure 1. An illustration of computation white balancing by detecting a scene-neutral tone, defining a transformation from scene-neutral to image-neutral tone, and applying the transformation to the original image.

Image-neutral tone: A color that appears neutral when viewing the image, regardless of if the tone is neutral within the real-life scene itself.

In other words, when all colors in an image are simultaneously scene-neutral and image-neutral, the image is effectively white balanced.

1.1.1 Research Question

However, these methods rely on human color intuition or recollection of the scene's original colors. Thus, this becomes a problem highly subjective for humans, but perhaps one that can be made not only accurate and objective, but automated by Deep Learning methods.

This project aims to explore the feasibility of applying Deep Learning models and techniques towards white balancing within the image processing domain. Various models, methods, and frameworks are be tested and evaluated to extract greater insight surrounding the problem presented.

1.2. Surrounding Color Science

Before proceeding, a brief overview of relevant color science is in order. Inside a camera, a photosensitive sensor quantifies the photos hitting its receptive plane. Then, a processor encodes these readings into an image file, with color and luminance represented by an industry-standard spec, called a color model. Below are the color models discussed in this paper, each with a brief introduction:

1. **RGB** (Red/Green/Blue). Colors are represented as a 3-Tuple of red, green, and blue intensities. By this 3-Tuple model, an image is comprised of three channels. This is the most common, general purpose color encoding.

2. **XYZ**. Another 3-channel encoding with abstracted channel values that allow the span of pixel values, collectively known as color space, to closely resemble human perception and biases towards the spectral sensitivities of the human eye.

3. **LAB**. Color decomposition presented as a luminance channel, an abstract channel for temperature (A), and an abstract channel for tint (B). It is important to note that LAB color space has a neutral origin, the color (n,0,0), in the center, meaning negative values for A and B are present in cool-temperature or green-tinted colors.

Invertible transformations are possible between the above color models, allowing for color transformation operations to occur in a different spatial domain, though on the same original image. This choice of color space during transformation is critically relevant both when estimating an image-neutral tone, and later on applying white balance to a whole image.

2. Dataset

This project utilizes the Unsplash Lite Dataset. In its raw form, this dataset consists of a tabular view of 30k images, with image metadata, description, and download links to full-size previews. Focusing on images themselves, we downloaded 5k randomly sampled images (restricted by campus network bandwidth) to be used as our dataset for training, validation, and testing.

Note that images are downloaded and made available at high resolution, often at or above 4K. Since the models discussed later on require a lower resolution input, images are resized and stored as necessary to comply with these downstream specifications.

2.1. Preprocessing

We assume our dataset of images, having come from professional photographers and collectively low in color bias by Law of Averages, are color neutral (white balanced).

For each image, we randomly generate an off-neutral tone encoded in RGB to act as a scene-neutral one. This will become the model's target label. Each image is then transformed so that the intentional off-neutral tone becomes image-neutral. In other words, we proactively shift each image's color space from neutral to off-neutral, so that the model can complete the sought-after (and now known) inverse transformation from off-neutral to neutral.

We implemented a custom DataLoader module to apply intentional tints ad-hoc, adding dynamic augmentation and resisting overfitting.

3. Approach

3.1. Approach Overview

We formulate our research problem as the following task: build a model that will take an image input, and generate an RGB color output of a scene-neutral tone. Once this tone is estimated, the image can be transformed within a chosen color space to a white balanced state.

3.2. Model Considerations

With model development and underlying techniques and architecture pushing the frontier of Deep Learning today, we will utilize the following models to gain insight surrounding the feasibility of the given task.

Since the objective lies in the domain of Computer Vision, specifically as a regression problem, we applied several state-of-the-art models to effectively integrate structural, semantic, and composition information into inference. Our approach innovates from past work on White Balancing [1] by utilizing state-of-the-art pre-trained transfer models which allow our data to fine-tune a model that already has an implicit embedding of image features.

We also trained our own Convolutional Neural Network (CNN) to evaluate if the task is already feasible, and to what degree successful, with a simple architecture and fewer trainable parameters.

We believe these models will be successful due to the large and diverse amount of data we have procured in conjunction with using state-of-the-art transfer-based pre-trained models to minimize test loss.

3.2.1 Pre-trained Models

The Hugging Face platform allows Machine Learning engineers to download weights and underlying architectures of state-of-the-art models for direct use in PyTorch. We leverage this toolkit to fine-tune complex pre-trained models that already understand latent factors about images. We use both transformer-based models and CNNs. For each pre-trained model, the final layers are fine-tuned to fit the given task, with all other parameters gradient-fixed to leverage previously achieved feature extraction abilities. In our case, we detached the following transfer-learning-based models' classification head, and replaced it with a trainable $m \rightarrow 3$ linear layer to accomplish regression onto RGB values.

1. **google/vit-base-patch16-224**: ”Vision Transformer (ViT) model [5] pre-trained on ImageNet-21k (14 million images, 21,843 classes) [3] at resolution 224x224, and fine-tuned on ImageNet 2012 (1 million images, 1,000 classes) at resolution 224x224.” Source and Further Information.

2. **microsoft/beit-base-patch16-224-pt22k-ft22k**: ”BERT Pre-Training of Image Transformers (BEiT) model [2] pre-trained in a self-supervised fashion on ImageNet-22k - also called ImageNet-21k (14 million images, 21,841 classes) [3] at resolution 224x224, and fine-tuned on the same dataset at resolution 224x224.” Source and Further Information.

3. **microsoft/resnet-50**: ”ResNet (CNN) model [4] pre-trained on ImageNet-1k at resolution 224x224.” Source and Further Information.

3.2.2 Custom Convolutional Neural Network

We also train our data on a custom built, non-pre-trained CNN as a baseline. We use six convolutional layers each with kernel size of 3 and padding of 1, each layer is followed with ReLU activations, and a max-pooling layer (kernel size = 2) after the 2nd, 4th, and 6th convolutional layer. The final pooling output is passed into a neural network with one hidden layer, hidden size of 512, with sigmoid activations after each layer. The last layer of the neural network maps from the hidden size to our output size of 3.

3.3. Evaluation Measures

Evaluation plays a crucial step in defining differentiable criteria that the models can optimize though gradient descent, as well as the structural configurations that affect model prediction effectiveness with respect to the training and testing data.

3.3.1 Model Criterion

As a means of experimentation and analysis through comparison, the following model criteria are implemented and evaluated in combination with every other pipeline module.

1. **Mean Squared Error (MSE) Loss**: Attempt to minimize the L2 distance between predicted and ground-truth scene-neutral color.

2. **Delta E (ΔE) Loss**: The industry-standard color difference metric between display technologies. Effectively implemented as the MSE Loss of the predicted scene-neutral color, encoded in the LAB color space

The above criteria, for comparison purposes, are applied in each white balance transformation color space: RGB, XYZ, and LAB, which are all derived from the RGB encoded prediction.

3.4. Anticipated Problems

We expected a large time and space complexity of downloading, pre-processing, resizing, and training our data. We also anticipated some of our initial model choices to possibly change just in case we wanted to test out any other popular pre-trained models. To solve these problems, our code deliverables for downloading, pre-processing, and resizing the dataset images were multithreaded in order to speed up preparation times. When creating the framework for training each model, builder design pattern was embraced. This allows us to abstract any pre-trained model we want into a builder class containing a function that loads the model into a director class, which is itself generalized to train any built pre-trained model according to the PyTorch API.

4. Experiments and Results

4.1. Training and Inference Framework

At this point, inputs (images) and outputs (RGB scene-neutral colors) form feature-label pairs, and are batched for training and inference. The color space in which off-neutral color augmentation occurs, the model, the color space in which the white-balance transformation occurs, and the final criterion are all treated as modules in a pipeline that simultaneously form combinatorial problem. We developed a framework to execute every possible combination of these elements and collect relevant results to then analyze which methods are most suitable to the task, and why. This approach is also essential to future expandability, since more modules can be added as the Deep Learning field matures, and the experiment can be effortlessly reproduced and further benchmarked.

We run each model for 10 epochs of our dataset, and measure success by a lower criterion test loss, emphasizing the importance of a model's ability to generalize to unseen data. Our goal is to compare each of our models' test criterion to find the model which best encodes latent information that contributes to white balancing of a given image.

4.1.1 Hyperparameters

To ensure the transfer-learned models are adequately trained before final evaluation, the testing/training loop is performed with k-fold cross-validation with a 75%/25% split.

Our own custom CNN was manually hyperparameter tuned through explicit experimentation, adjusting number of hidden layers, feature map size, and convolution channel cardinality, to achieve minimal loss.

Model Name	Input Color Space	MSE Test Loss (RGB-encoded)	ΔE Test Loss
google/vit-base	RGB	**4.74e-3**	**5.39e-4**
google/vit-base	XYZ	**3.72e-3**	**2.97e-4**
microsoft/beit-base	RGB	5.34e-3	7.72e-4
microsoft/beit-base	XYZ	6.67e-3	1.11e-3
microsoft/resnet	RGB	3.52e-2	8.12e-3
microsoft/resnet	XYZ	3.47e-2	7.69e-3
non-pretrained CNN	RGB	2.01e-2	2.44e-3
non-pretrained CNN	XYZ	1.40e-2	1.94e-3

Table 1. Results for training across all models and color spaces for input and criterion. Model names have been shortened for reading purposes. Best results of each loss column are bolded for both XYZ and RGB input color space.

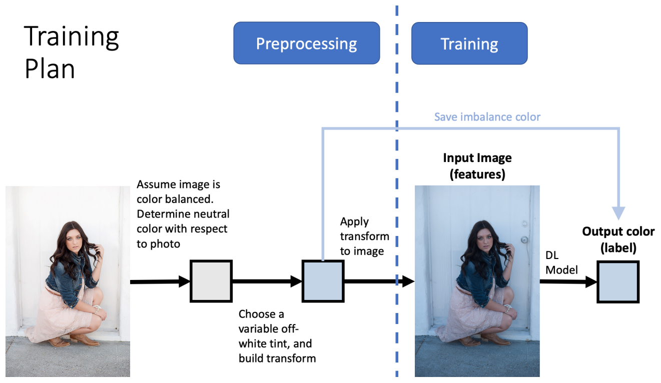
Training Plan
Preprocessing
Training
Save imbalance color
Input Image (features)
Assume image is color balanced. Determine neutral color with respect to photo
Apply transform to image
DL Model
Output color (label)
Choose a variable off-white tint, and build transform

Figure 2. A graphical overview of the project's training plan - highlighting the augmentation operation during prepocessing, and white balance fitting operation during training.

In all cases, learning rate is chosen to train the model near the criterion's local minimum within ten epochs for consistency between model experiments.

4.1.2 Result Collection

For each tested pipeline of color spaces, models, and criterion, the following metrics are tracked for further analysis: Augmentation Color Space, Criterion Color Space, Criterion, Final Train Loss, Final Test Loss.

4.2. Results Analysis

After training each model on different input and criterion color spaces, we are able to compare them using their minimum test loss after k-fold cross validation. Table 1 contains the minimum loss information for each model and respective input and criterion color spaces. From this table, **google/vit-base-patch16-224** stands out as the model with the minimum test loss for all input and criterion color spaces. Thus we can conclude that out of the pre-trained models, Google's Vision Transformer best encodes the features of images that are used for white balancing. Figure 3 shows examples of our pre-processed data, model output, and ground truth.

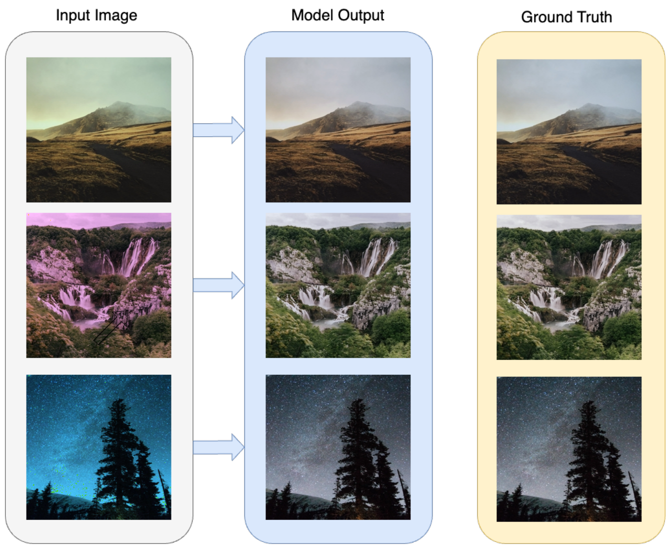

Figure 3. Input, output, and ground truth examples of some of the images in our dataset through the **google/vit-base-patch16-224** model.

However, a surprising result was that our custom non-pretrained CNN was able to outperform **microsoft/resnet-50** completely. We theorize this could be because the features that the pre-trained resnet CNN learned may not contribute as well to white balancing, and only adding one linear layer to these embeddings may not have been enough to predict the correct output, whereas our CNN upsamples image data and uses a hidden layer to predict its output. Nevertheless, the transformer-based pre-trained models are able to perform over 10 times better than the CNN-based models.

We recorded train loss to compare with test loss for each model over all 10 epochs of training to ensure that each model was not overfitting. Figure 4 shows steady convergence for the **microsoft/beit-base-patch16-224-pt22k-ft22k** and **microsoft/resnet-50** model, whereas **google/vit-base-patch16-224** and our custom non-pretrained CNN quickly converge. Our custom non-pretrained CNN exhibits slight overfitting and a jittery loss curve which can be attributed to the high number of datapoints with respect to a lower number of trainable parameters in our custom CNN when compared to the pre-trained models.

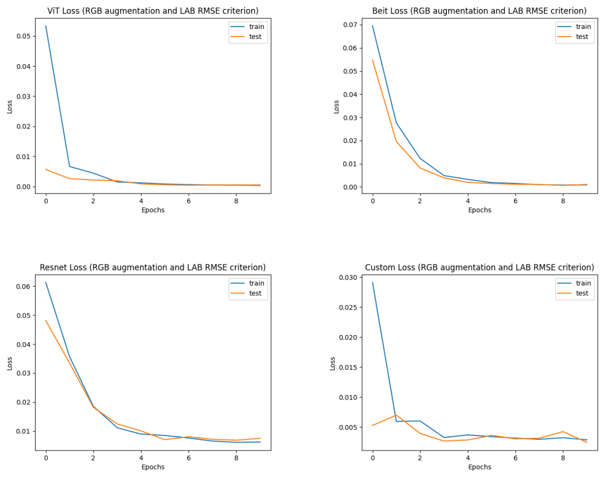

Figure 4. Train and Test ΔE loss for each model over 10 epochs of training.

5. Future Work

From our analysis of our experiment results, we recommend that future work on the White Balancing task be focused on transformer-based models. Not only do our results show that these models have better performance than their CNN counterparts, but they train faster as well and are thus able to be hyperparameter tuned more time efficiently. Additionally, this exploration of Deep Learning White Balancing raises a need for more testing to elaborate on the ability of pretrained models to encode white balancing information effectively. Through the builder design pattern in our code, we modulate this task and make it easy to connect any pre-trained model to be fine-tuned on our pre-processed dataset. Future work could also include making an even more effective custom model without transfer learning methods that is highly optimized to this task; even more effective than fine-tuned transfer-learning based models.

6. Contributions

Table 2 contains an overview of which team member contributed which work to our project.

7. Supplemental Materials

Our code can be found at `https://github.com/dylan-small/DeepColorBalancing`

References

[1] Mahmoud Afifi and Michael S. Brown. Deep white-balance editing. *CoRR*, abs/2004.01354, 2020. 2

[2] Hangbo Bao, Li Dong, and Furu Wei. Beit: BERT pre-training of image transformers. *CoRR*, abs/2106.08254, 2021. 3

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 3

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[5] Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Zhicheng Yan, Masayoshi Tomizuka, Joseph Gonzalez, Kurt Keutzer, and Peter Vajda. Visual transformers: Token-based image representation and processing for computer vision, 2020. 3

Student Name	Contributed Aspects
Blake Sanie	Data creation, pre-processing, resizing of dataset. Implementation of PyTorch custom Dataset and DataLoader code.
	Implementation of algorithm to train each model over various input and criterioncolor spaces.
Dylan Small	Implementation of model architecture and training/validation loop.
	Creation of modular classes to train different pre-trained and/or custom models

Table 2. Contributions of team members.