**Names:**

1. **Jyothi Vishnu Vardhan kolla**
2. **Karan Shah**

## Abstract:

In this project, we delved into the realm of color constancy, a critical preprocessing component in numerous computer vision applications. Color constancy mirrors a remarkable feature of the human visual system, allowing us to perceive object colors consistently, regardless of the prevailing illuminant. Recent years have witnessed a surge in the popularity of artificial neural networks, primarily owing to the groundbreaking accomplishments of deep neural networks in various recognition tasks. We embark on an exploration of color constancy through the lens of deep convolutional neural networks (CNNs) in this project.

## Introduction and prior work:

The driving force behind our project is the paper titled "Color Constancy Using CNNs" by Bianco et al, and the thesis Learning Colour Constancy Using Convolution Neural Networks by Hidir Yuzuguzel. The seminal work has served as a pivotal source of inspiration, shaping the core objectives and methodologies of our investigation into color constancy. The paper explores the application of Convolutional Neural Networks (CNNs) to address the challenging problem of color constancy, a fundamental aspect of computer vision. The described CNN takes image patches as input and works in the spatial domain without using any hand-crafted features that were employed by previous methods. The architecture, methodology, and results are presented well and demonstrate the stability of local illuminant estimation of CNN's. Our main objective was to understand how to reproduce this paper (extensive data preprocessing) and observe the results of this method.

## Methods:

### What is color constancy?

It is the process of estimating the scene illuminant and then correcting the Image based on its estimated to generate a new image of the scene as if it was taken under a reference light source.

### Data Preprocessing:

- Resize the images to 1200 X 1200.

- **If the image is Canon 5d, remove the black level from the image (black level = 129)**
- Find the corner positions of patches where each patch is of size 32 X 32 and there is 800+ patches from each image.
- Mask the patches that overlap with the MCC color checker in the image.
- Extract the top-100 brightest patches per image. co
- Perform contrast normalization using global histogram stretching for each patch
- Normalize the patches with zero-mean and unit-variance standardization.

**We use CNN to estimate illuminant of each patch and combine the patch scores to obtain an illuminant estimation for the image.**

**CNN Architecture**

- The Proposed network consists of 5 layers.
- Layer-1: Input -> 32 X 32 X 3-Dimensional Image patch.
- Layer-2: CONV -> 240(1 * 1 * 3) kernels, Stride 1 -> 32 X 32 X 240.
- Layer-3: Maxpool -> K = 8 X 8, Stride 8 -> 4 X 4 X 240.
- Layer-4: Flatten the 4 X 4 X 240 -> 3840 -> FFN with 40 neurons.
- Layer-5: A simple Linear regression layer with a three-dimensional output.

**CNN Parameters:**

- **Batch Size = 64**
- **Epoch = 8**
- **Learning Rate = 0.1**
- **Weight Decay = 0.0005**
- **Momentum = 0.9**
- **Optimizer = SGD**

**Dataset used:**

- Shi-Gehler Preprocessed Dataset.

**Learning process:**

- Train the Model in a 3-Fold cross validation setting.
- Assign illuminant ground truth to each patch associated to the Image which it belongs.

- Use the Euclidean loss as loss function in this process.
- Once the training is complete, get the predictions for each patch in the test set.
- We need to take mean or median of predictions of all the patches belonging to a particular Image.
- Then we apply Von-Kries Models to get the corrected Images.

**Evaluation:**

- **Following the thesis by** Hidir Yuzuguzel, angular error is used as the error metric.
- This metric is the angle between the RGB triplet of the ground truth illuminant e and the RGB triplet of the estimated illuminant e-hat.

$$angular\ error = \arccos\left(\frac{e^T\hat{e}}{\|e\|\,\|\hat{e}\|}\right)$$

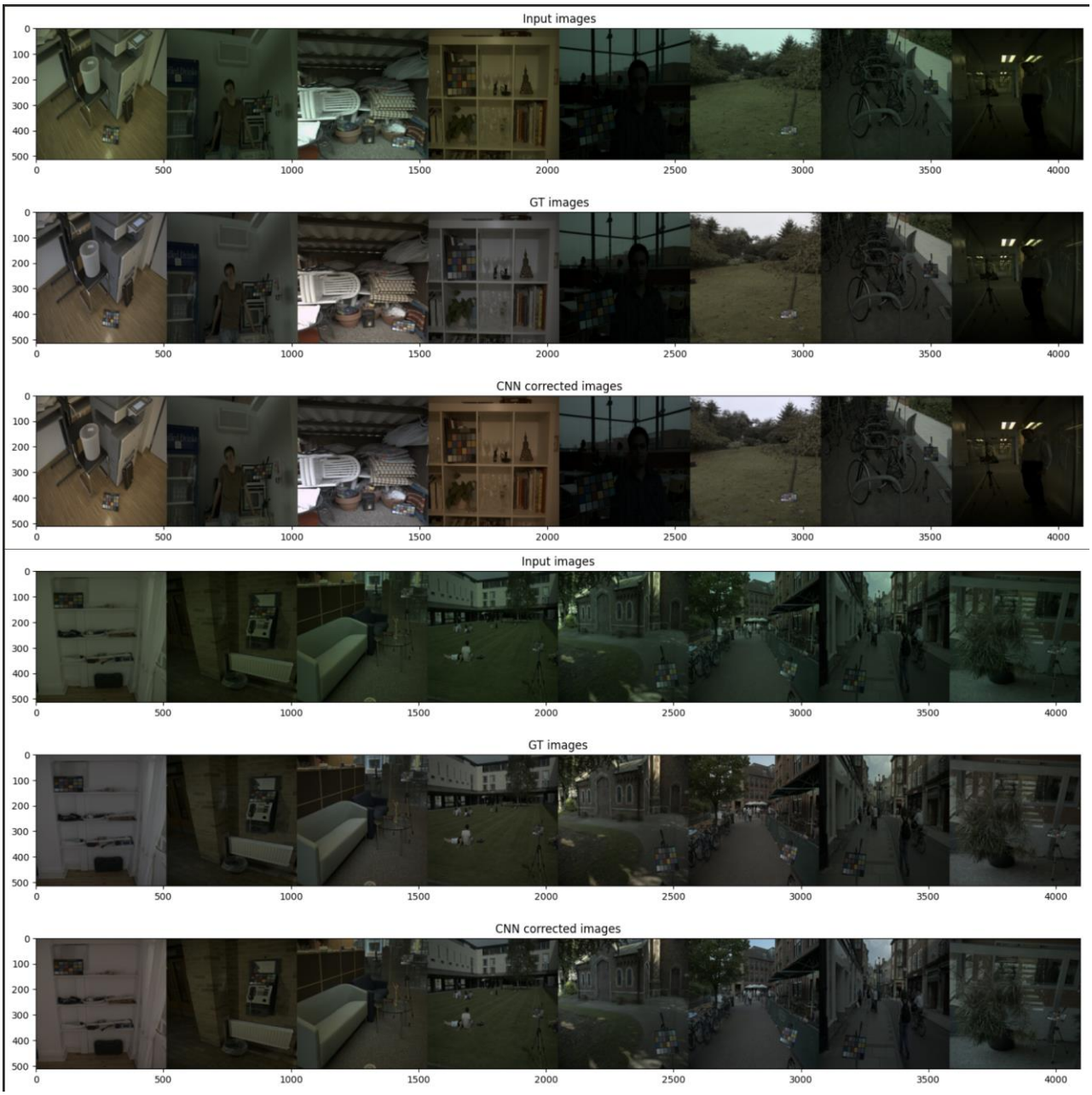
**Results:**

```
Patch-wise angular error statistics
min        0.000000
10prc      1.336181
median     5.185876
mean       8.391524
90prc     18.727731
max       44.382011
```

```
Median pooling angular error statistics
min        0.194231
10prc      1.148353
median     3.171899
mean       6.222008
90prc     16.563073
max       33.911500
```

```
Average pooling angular error statistics
min        0.398743
10prc      2.492947
median     5.606401
mean       6.777615
90prc     13.082627
max       30.597167
```

The statistics above show the angular error statistics from the CNN approach on the Shi-Gehler dataset. The reported angular error statistics are the minimum, $10^{th}$ percentile, median, average, $90^{th}$ percentile, and maximum. Using our pytorch implementation, these results are close but slightly worse than the results shown in Table 4.1 (page 34) of the thesis **by** Hidir Yuzuguzel.




**The visualizations above show randomly selected visuals with the input, ground-truth, and CNN corrected results. Visually, therefore, it does appear that CNN is able to correct the input images to a degree and match the ground-truth. However, more experimentation needs to be done to evaluate the performance of CNN by comparing it to other traditional methods and more advanced architectures. Our objective primarily was to reproduce the results, and we were able to get through the extensive amount of data preprocessing to produce these results.**

**Reflection and acknowledgements:**

**In this project, we learned how extensive amount of data preparation needs to be done to break each image down into patches, train our models, and build those patches back up to obtain the results. Also, I realized that the main bottleneck in this project was not the architecture, but instead, it was actually understanding how to transform and use the data properly to obtain the results. Originally, our Pytorch loss was performing horribly, and our results were coming out completely incorrect. This was not because of the architecture but rather a good portion of the data manipulation. In the end, the results were average using this CNN, and more experimentation needs to be done to explore the true benefits of using deep networks for color constancy.**