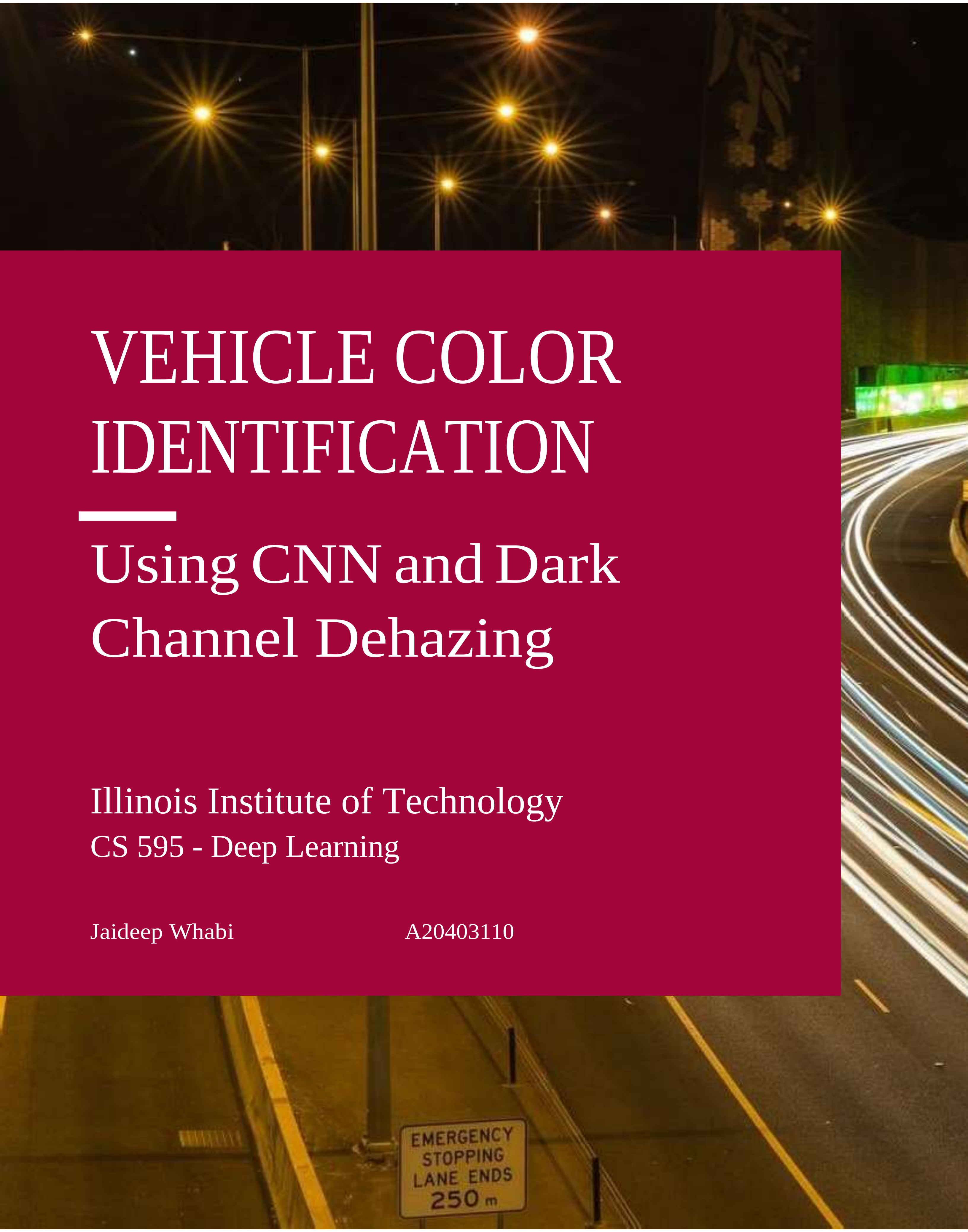


VEHICLE COLOR IDENTIFICATION

Using CNN and Dark Channel Dehazing

Illinois Institute of Technology
CS 595 - Deep Learning

Jaideep Whabi A20403110

Contents

Problem Statement

In recent years the number of cars on roads has increased exponentially and, identifying them has become a very big task. Vehicle color information is one of the 3 important elements in ITS (Intelligent Traffic System), the other two being– Make and model of the car and license plate recognition. Vehicle color is an important property for vehicle identification and provides visual cues for fast action law enforcement. Recognizing the color of a moving or even a still vehicle can be a very challenging task because of several factors including weather conditions, quality of video/image acquisition, and strip combination of the vehicle. Different ensemble algorithms can be used to give us color recognition.

Proposed Solution

Dehazing of images

A major part of this project will deal with the removal of haze from images. This algorithm can be applied to many other problems too. This algorithm uses atmospheric light to give us clearer images. As cities grow bigger, we are encountering more air and light pollution. Both contribute to hazy images that are difficult to process in real time. We will be using transmission maps and atmospheric light to generate a clear image.

Classifiers

The haze free images are fed to a convolutional network for feature engineering and the output is fed to a fully connected network for classification. Multiple color spaces (RGB, HSV and CIE LAB) are tested to determine the color space which is easiest to interpret and leads to higher performance. The final model, when given an image of a vehicle, will be able to predict the color of the vehicle with high accuracy.

Implementation Details

The Research papers we referred to various color spaces and various classification techniques. We implemented an ensemble method which dehazed the images first and then creating a fully connected neural network.

Dataset Used

The Vehicle Color Recognition Dataset contains 15601 vehicle images in eight colors, which are black, blue, cyan, gray, green, red, white and yellow. The images are taken in the frontal view captured by a high-definition camera with the resolution of 1920×1080 on the urban road. The collected data set is very challenging due to the noise caused by illumination variation, haze, and over exposure. The dataset is available at -

http://122.205.5.5:8071/~pchen/project.html

Haze Removal

We use Dark channel prior, which is an assumption based on their examination of haze-free outdoor images. It represents the finding that, in most non-sky images, at least one color channel(R/G/B) has pixels with very low intensity. The dark channel is defined as:

$$I^{dark}(x) = \min_{c\in\{r,g,b\}}(\min_{y\in\Omega(x)} I^c(y))$$

I^c is a color channel of I, and $\Omega(x)$ is a local area centered at x. He et al[4] examined 5000 daytime haze-free images and found that about 75% of the pixels in the dark channels had zero values and the intensities of 90% of the pixels were below 25.

The dark channel of the foggy images is not that dark. Hence, a foggy image will be darker than a clear image due to added air light. This means that the intensity of the dark channel represents the fog density and the object's distance from the camera in a way.

The dehazing algorithm we use is based on dark channel prior.

$$I^{dark}(x) = J^{dark}(x)t(x) + A^c\ (1 - t(x))$$

$I^{dark}(x)$ is the dark channel for the foggy image, $j^{dark}(x)$ is the dark channel for the clear image, and A^c represents the color channel of atmospheric light. As most of the intensity values of the dark channel for the clear image are zero, the equation can be rewritten as:

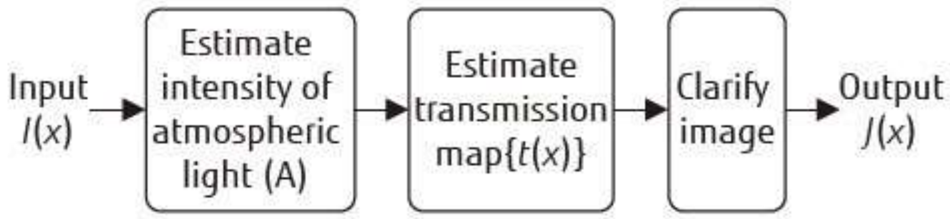
$$I^{dark}(x) \approx A^c(1 - t(x))$$

The transmission map can be estimated using a transformed version of this equation:

$$t(x) \approx 1 - I^{dark}(x)/A^c.$$

The intensity of the atmospheric light is estimated by first identifying the top 0.1% brightest pixels in the dark channel ($I^{dark}(x)$). From among these pixels, the one with highest intensity is selected as the one representing the atmospheric light.

The steps for clearing an image will be as follows:


Input
$I(x)$
Estimate intensity of atmospheric light (A)
Estimate transmission map{$t(x)$}
Clarify image
Output
$J(x)$

Now, to estimate the atmospheric light, we first identify the top 0.1% brightest pixels in the dark channel. And, then the highest value is identified as the atmospheric light.
We then identify a coarse map of a local area, after which we create an image edge information map which are then combined to create a precise transmission map representing fog density for each pixel.
We use a simpler method to solve this problem. A fine map with minimum values for the R, G and B channels is derived to represent the edge detail information of the foggy image.
The coarse map is then refined by selecting the pixel with the minimum one compared with the value of the pixel at x in the fine map.

$$M^t(x)=\min(\max_{y\in\Omega(x)}M^{coarse}(y), M^{fine}(x))$$

The transmission map can now be obtained by:

$$t(x)=1-\omega\cdot M^t(x)/A$$

ω is the defogging parameter, which lies in the range of 0 to 1. It represents how much an image must be defogged. The smaller the value, the hazier the image.

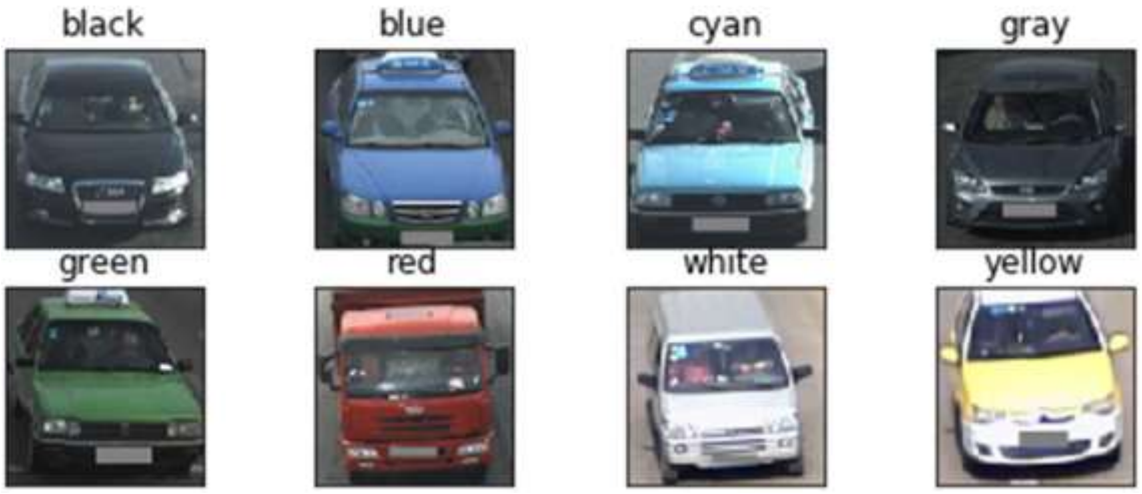
Finally, this formula is used to clarify the image:

$$J(x)=(I(x)-A)/\max(t(x),t_0)+A,$$

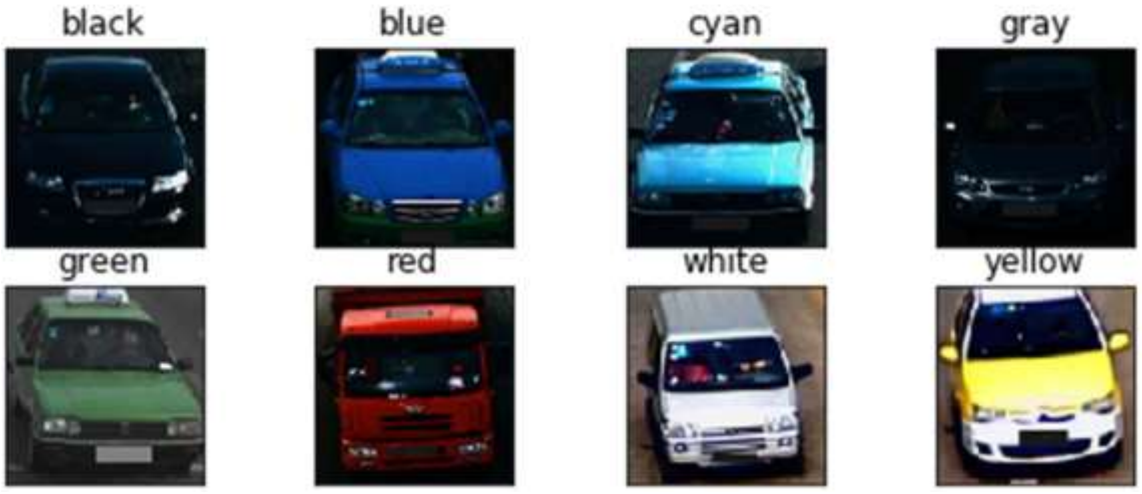
We created four functions that give us clear images as outputs:

- atmospheric_light
  Finds the atmospheric light by scanning through the image.
- Dark_channel_find
  Finds the dark channel for each pixel in the image.
- Coarse
  Finds the coarse map for image
- Dehaze
  Uses input from the 3 functions mentioned above and generates a dehazed image which is finally used to train the models.

Hazy Images



Hazefree Image



Importing and standardizing the Data

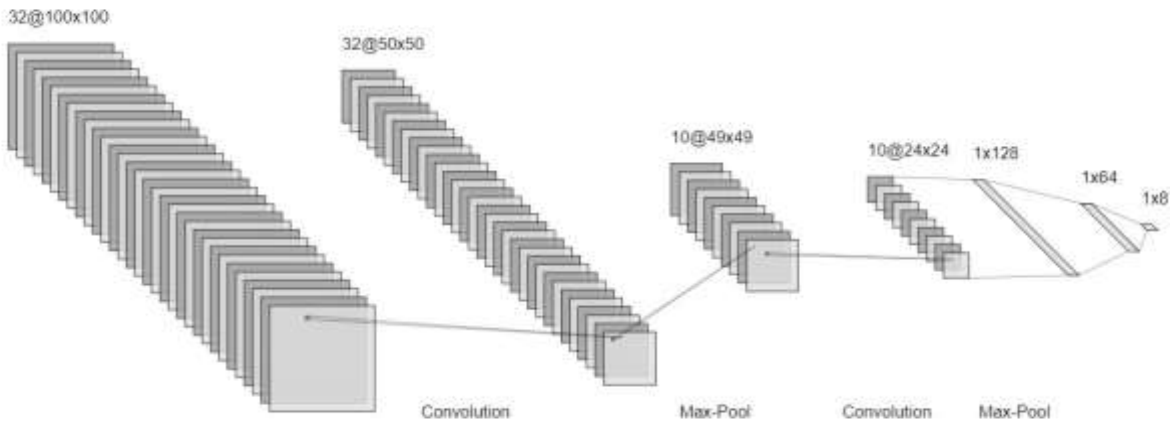
The input data is imported using the OpenCV imread function, which reads an input file and converts it to RGB channels which are stored as a numpy array. This array will have a value for the Red channel (R), Blue Channel (B) and Green channel (G). The values for each of these channels are between 0-255. Thus reading 15,601 images will give us a 4D Tensor of data. Class labels of each image are created by looking at the folder name the image was present in. By default, OpenCV reads and stores an image in BGR format instead of RGB, so if the image is to be visualized in python, the channel orders will have to be shuffled. To convert the image data into a different color space, you can call the cvtCOLOR function. Each image is resized to (100x100) pixels to keep a standard data size for training input.

Once all the images have been read, we split our data into training, validation and test samples (sklearn's train-test split used) [training – 9452, validation – 1000, testing - 5149]. Since the pixel values for each color channel are between 0-255, the scope of variance in data increases. To compensate for this, we divide each value by 255 thus getting the range of each pixel-channel between 0-1. This will lead to easier interpretation of the data by the model.

Classification

Once all the images have been treated for haziness with the algorithm mentioned above, we will use a Convolutional network for feature extraction and then a fully-connected layer to classify based on the features extracted by the previous CNN. Color spaces are very important to color recognition applications, like vehicle color recognition. The selection of color space will impact the recognition performance. As such, we will train our model using 3 different color spaces: RGB, HSV and CIE Lab to check which color space is most useful in training a CNN to classify vehicle images. We will test our network on both hazy and haze free data.

Proposed Architecture



The proposed architecture for classification is shown above. It consists of 2 Convolutional layers with pooling for each layer, followed by a dense network of 2 hidden layers and an output layer for classification. This architecture was zeroed in after multiple iterations using different architectures and hyperparameter tuning for having consistent performance. The Convolutional and dense layers use the ReLU (Rectified Linear Unit) activation provided by keras, except for the output layer which uses the softmax activation function. The first convolution has 32 filters with a kernel size of (3,3) while the second convolution has 10 filters with a kernel size of (2,2). There is a Max-pooling layer after each convolution with a stride of 2 and a window of (2,2), thus decreasing the size post each convolution by half. Between the CNN and dense network there is a Flatten layer to flatten the data from a 4D Tensor to a 2D Tensor.

The loss function used is categorical cross entropy, with RMSprop as the optimizer. Alternatively, a Stochastic Gradient Descent with momentum (0.9) optimizer can also be used if you wish, since across tests, both these optimizers had similar performances, which were best in class. The metric used is accuracy. The model will be trained across a large range of epochs using validation data, and the validation loss vs training loss curves will be used to assess the optimum number of epochs.

Color Spaces

The RGB Color Space

The RGB colorspace has the following properties

- It is an additive colorspace where colors are obtained by a linear combination of Red, Green, and Blue values.
- The three channels are correlated by the amount of light hitting the surface.

The LAB Color-Space

The Lab color space has three components.

1. L – Lightness (Intensity)
2. a – color component ranging from Green to Magenta.
3. b – color component ranging from Blue to Yellow.

The Lab color space is quite different from the RGB color space. In RGB color space the color information is separated into three channels, but the same three channels also encode brightness information. On the other hand, in Lab color space, the L channel is independent of color information and encodes brightness only. The other two channels encode color.

The HSV Color Space

The HSV color space has the following three components

1. H – Hue (Dominant Wavelength).
2. S – Saturation (Purity / shades of the color).
3. V – Value (Intensity).

The advantage of HSV is that it uses only one channel to describe color (H), making it very intuitive to specify color although it is device dependent.

The 3 color spaces mentioned above will be tested and a comparative study is mentioned in the results section.

Results

The machines we used for processing were-

1. Intel (R) Core (TM) i5-7300HQ CPU @2.50GHZ (8 GB Memory) dual core with Nvidia GTX 1050(8GB Memory)

2. Intel (R) Core (TM) i7-7500U CPU @2.70GHZ (8 GB Memory) dual core

Dehazing of images on these machines took a couple of days even after diving them into smaller tasks.

The proposed architecture was run on the images generated after the haze free module was run on the dataset. To compare the results, the same model was also trained on the original dataset which has hazy images. The model was trained across 30 epochs and the results are shown below:

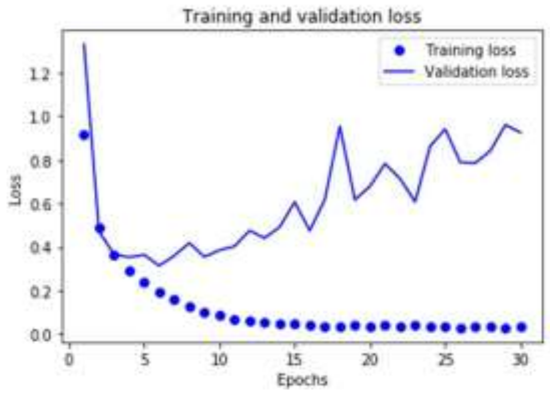


Figure 1 - Training vs validation loss for haze free images

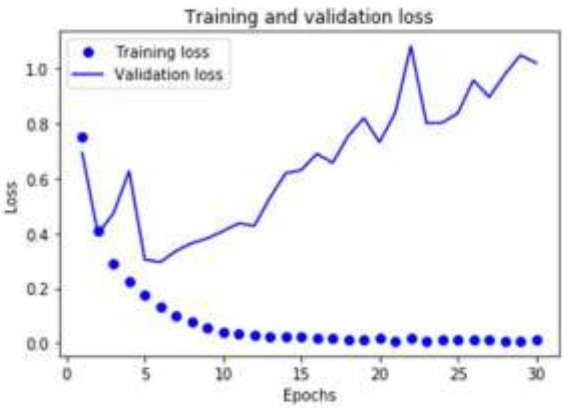


Figure 2 - Training vs validation loss for hazy images

As you can see, for haze free images, the model converges quicker than in comparison to hazy images. Moreover, the initial loss value is comparatively much less (0.6931 vs 1.3304 for haze free vs hazy images). Moreover, if we see accuracy, haze free images consistently reach accuracy values above 90% whereas the for hazy images accuracy is usually around 86%. The difference is evident on test data. The table below shows the accuracy and F1 scores for hazy and haze free training. The accuracy is calculated per class so as to check against the accuracy metrics mentioned in the research paper we refer to.

Color	Accuracy for hazy images	Accuracy for haze-free images	F1 Score for hazy images	F1 Score for haze free images
black	96%	95%	95.00%	94.00%
blue	90%	91%	93.00%	92.00%
cyan	93%	90%	94.00%	92.00%
gray	71%	76%	78.00%	82.00%
green	67%	99%	74.00%	98.00%
red	97%	97%	97.00%	96.00%
white	96%	95%	89.00%	90.00%
yellow	92%	94%	94.00%	96.00%
Average:	87%	92%	89%	93%

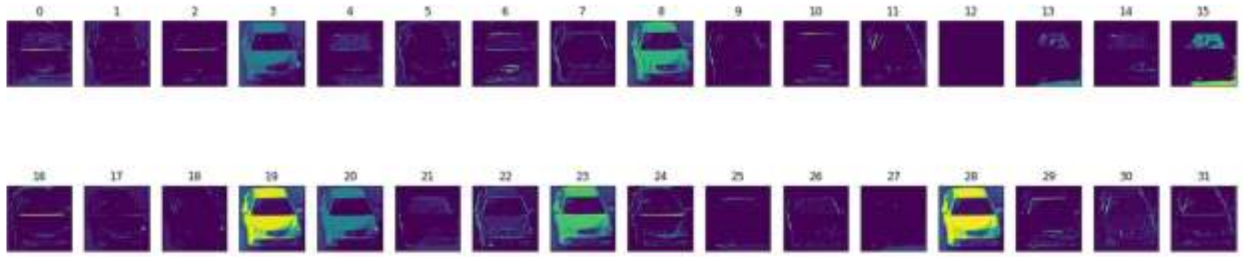
The original research paper has an average accuracy of 94% with a much more complex network. If we implement the haze free algorithm to treat the images before training, we are able to arrive to a relatively similar model with a much simpler architecture. (the original paper has 5 convolutional layers as compared to 2 convolutional layers in this network).

Also, if we analyze the tables, the average accuracy if we use haze free images as training is 92% which is 5% more than when we use hazy images. The F1 score for the model also increases when using haze free images for training indicating a higher precision and higher recall.

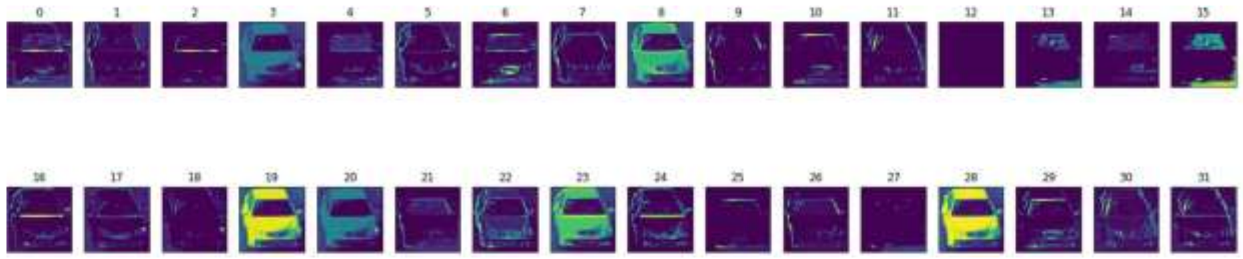
In conclusion, treating the images to be haze-free prior to training the model leads to a higher accuracy while keeping a much simpler architecture.

To understand how the convolution layers, perform feature engineering, refer to the figure below:

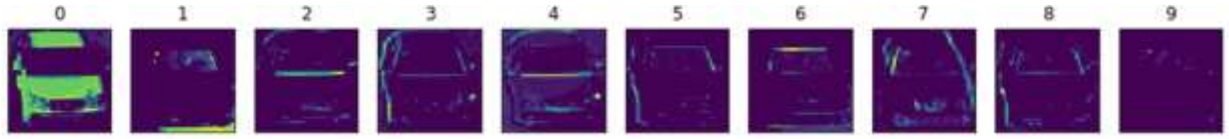
Convolution 1:



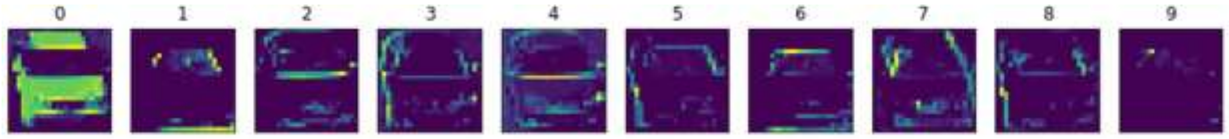
Max pool:



Convolution 2:



Max pool 2:



As you can see, the first layer learns high level features like the car outline, and the next layer learns the different edges of the car. In the second convolution layer, features start becoming much more abstract and it was found that nothing new is learnt if additional convolutions are added.

Finally, the model architecture was tested against different color spaces, more specifically, RGB (which is the default color space when reading through OpenCV), CIE LAB and HSV. We get the latter 2 color spaces by converting from RGB to the particular color space. The same model was trained and tested against each of these color spaces and the results are below:

Training:

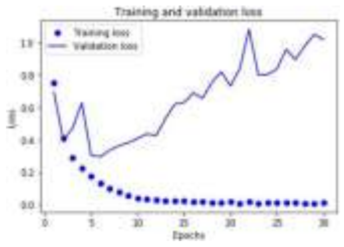


Figure 3 - Training for RGB

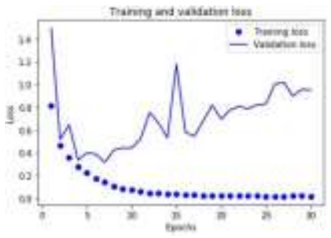


Figure 2 - Training for HSV

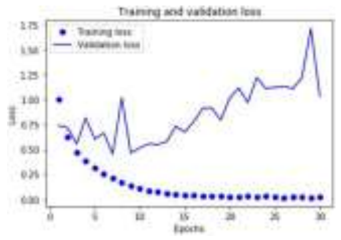


Figure 1 - Training for CIE LAB

Testing:

Color	Accuracy for RGB	Accuracy for HSV	Accuracy for LAB
black	95%	95%	95%
blue	91%	90%	90%
cyan	90%	89%	89%
gray	76%	85%	85%
green	99%	99%	99%
red	97%	95%	95%
white	95%	84%	84%
yellow	94%	85%	85%
Average:	92%	90%	90%

Conclusion

- Using haze-free(clean) images gives us simpler models that converge faster and give us 5% higher accuracy.
- As seen from the training epochs and testing accuracy for the optimal model for each color space, the RGB color space looks to be the best way to define this dataset and leads to better model interpretation and performance. Precision and accuracy scores for each color space can also be calculated to confirm this.

References

1. Reza Fuad Rachmadi: “Vehicle Color Recognition using Convolutional Neural Network”, 2015; [http://arxiv.org/abs/1510.07391 arXiv:1510.07391]

2. K. S. Aarathi and A. Abraham, "Vehicle color recognition using deep learning for hazy images," 2017 International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, 2017

3. C. Hu, X. Bai, L. Qi, P. Chen, G. Xue and L. Mei, "Vehicle Color Recognition With Spatial Pyramid Deep Learning," in IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 5, Oct. 2015

4. K. He. Et al.: Single Image Haze Removal Using Dark Channel Prior. Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009

5. J.-W. Hsieh, L.-C. Chen, S.-Y. Chen, D.-Y. Chen, S. Alghyaline, and H.-F. Chiang, Vehicle Color Classification Under Different Lighting Conditions Through Color Correction, IEEE Sensors Journal, 2015,

6. P. Chen, X. Bai and W. Liu, Vehicle Color Recognition on an Urban Road by Feature Context, IEEE Transactions on Intelligent Transportation Systems (TITS), 2014,

7. E. Dule, M. Gokmen, M. S. Beratoglu, A convenient feature vector construction for vehicle color recognition, Proc. 11th WSEAS International Conference on Neural Networks, Evolutionary Computing and Fuzzy systems, pp: 250255, (2010)

8. N. Baek, S.-M. Park, K.-J. Kim, and S.-B. Park, Vehicle Color Classification Based on the Support Vector Machine Method, International Conference on Intelligent Computing, ICIC 2007

9. J.-W. Son, S.-B. Park, and K.-J. Kim, A convolutional kernel method for color recognition, International Conference on Advanced Language Processing and Web Information Technology (ALPIT 2007)

10. http://alexlenail.me/NN-SVG/LeNet.html , Used to create neural network visualizations.

11. Fast Single Image Defogging by Zhiming Tan, Xianghui Bai, Bingrong Wang and Akihiro Higashi

12. https://www.learnopencv.com/color-spaces-in-opencv-cpp-python/