***Effects of Image-Transformed Data Augmentation on Detecting Traffic Lights***

*By: Manav Gurnani*

## *Abstract*

Traffic light detection has been well studied, with large automotive manufacturers heavily investing in the technology. One important parameter to note when developing a model to detect traffic lights is the quality of the input data. Traffic light detection works far differently during the daytime and nighttime due to the differences in light and noise that the images may be contaminated with. Hazardous driving conditions with lower visibility can be especially hard to work around.

In an attempt to formulate a new situation, this paper discusses the viability of data augmentation in training datasets, and its impact on the accuracy of a computer vision model in predicting traffic lights. The 4 data transformations utilized in the experiment include histogram equalization, logarithmic-inverse transformation, gamma transformation, and histogram specification. Using these, there are 15 possible combinations of experiments, all of which compare the final accuracy of the model. Out of the 15, the model presented a higher accuracy in detecting traffic lights when all 4 transformations were used simultaneously.

## 

## **Introduction**A close up of a car dashboard Description automatically generated with medium confidence

In the United States, approximately 50% of all car accidents occur at intersections[1]. These accidents can have many root causes, e.g., DUI collisions, traffic violations, and speeding. However, often these accidents can be attributed to misjudgment of traffic lights. Over the past decade, safety technologies have been embedded into vehicles, including smarter airbags, blind spot detection, and radar-guided cruise control. 

More importantly, the innovation of DAS (Driverless Assistance Systems) has been predicted to mitigate traffic collisions by at least 90%[3]. Automotive manufacturers such as Tesla, Audi, and BMW have incorporated technologies such as traffic sign detection, pre-collision sensing, and traffic signal detection[3]. These technologies are crucial to improving road safety in today’s faster vehicles. Moreover, they serve as a vital part of autonomous vehicle testing. One such type of technology is traffic signal detection, a vehicle’s ability to detect traffic lights and their color state.

Instead of cameras, Audi **[Figure 1]** uses signals and communication to read and interpret traffic lights. However, ever since the introduction of Machine Learning and AI technology into the automotive industry, researchers have hypothesized a new method to detect traffic lights without having to build an external connection: computer vision.

Computer vision is a segment of Machine Learning that includes interpreting images through neural networks[4]. An example could be an object detector, which could take an image of an object as input, and it successfully identifies what the object is and the state of the object.

In this case, a camera would take input of an image of a traffic light into a trained machine-learning model and would receive the state of the traffic light (i.e., stop, go, warning)[5]. Based on this information, the computer would react differently. The accuracy of this model is key as it determines how successful the computer would be at predicting and reacting to traffic signals, and demonstrates its viability in a public environment.

The purpose of this experiment is to optimize traffic light detection in order to improve accuracy. We could use data augmentation, performance analysis, and statistics to manipulate different parts of the neural network to make it better. However, this experiment focuses on the training data used by the network to learn. We take the given dataset, apply a set of image transformations and use it to train the model. We performed histogram equalization, logarithm transformations, inverse transformations, and histogram specification on each of the images to create a more challenging dataset to learn from[6].

Several papers have attempted to optimize traffic light detection, and have done so using better model architectures and data augmentation. For example, the paper describing VATLD **[Figure 2]**, a new method to measure the performance of a traffic light detection model, describes software that not only presents statistics to the users but also suggests improvements to better the model being analyzed[7]. Another paper concerning semantic consistency aims to enhance the data provided to the model for training. Instead of modifying the model like VATLD, this research manipulates the data fed into the model for it to learn. It uses GANs (generative adversarial networks) to re-crop, scale, and physically transform images to challenge further, forcing the model to learn about many extremes regarding data.Graphical user interface, application

Description automatically generated

The layout of my amendments to the model are located in different parts of the model. In this experiment, traffic light detection is improved by appending the normal dataset with transformed images.

Chart, line chart

Description automatically generated

## **Results**

### Experiment 0

Operations: None

Training Stage

Accuracy: 0.x316 (31.6%)

### Experiments 1-15

Key:

* H: Histogram Equalization
* L: Logarithm and Inverse
* G: Gamma Transform
* S: Histogram Specification

| **#** | **Operations** | **Accuracy** |
| --- | --- | --- |
| 1 | H | 0.316 |
| 2 | L | 0.347 |
| 3 | G | 0.312 |
| 4 | S | 0.303 |
| 5 | H | L | 0.300 |
| 6 | H | G | 0.314 |
| 7 | H | S | 0.256 |
| 8 | L | G | 0.348 |
| 9 | L | S | 0.288 |
| 10 | G | S | 0.312 |
| 11 | L | G | S | 0.341 |
| 12 | H | G | S | 0.342 |
| 13 | H | L | S | 0.307 |
| 14 | H | L | G | 0.314 |
| 15 | H | L | G | S | 0.365 |

## 

## **Discussion**

There were certain parts of the experiments that could have been improved. Firstly, the organization of the files from the raw zip file could’ve been automated by moving images around before cropping them. A system where the images could be directly pulled out, cropped, and allocated in one traversal could’ve been more efficient, saving both storage and processing time.

Despite this, numerous other algorithms and image transformations could have a similar, if not better impact on the accuracy of the model. A similar version of this experiment could include a greater number of transformations (i.e., thresholding) for a broader range of results. Moreover, other model-improving methods could be embedded with these filters, such as the methods described in IV.

An extended version of this experiment could be reflected in road signs. Instead of traffic lights, focusing on traffic signs might be relatively simpler and a smaller step toward automotive technology research. Furthermore, it is a technology that would be better adapted by the automotive industry today, as many manufacturers have already implemented speed limit readers and stop sign readers in their vehicles.

## **Methods**

### Pre-processing

#### Adding Images to Drive Folder

Once the dataset was been [downloaded (or linked)](https://www.kaggle.com/datasets/mbornoe/lisa-traffic-light-dataset), we started by getting the paths of all the images[8]. We aimed to perform a one-time copy operation to construct the folders we need rather than moving individual images into organized folders.

#### Adding All Annotations

We traverse through all of the image folders including the day/nightTrain and the day/nightSequence folders **[See Figure 3]**, getting the paths and storing them for later reference. Simultaneously, we traverse through the annotation folders to get organized data for each image. We do not rename any files as that would impact our search in each image's correspondence later. However, we sort both lists alphanumerically to allow for easier relative access to the image files from the annotations.Graphical user interface, text, application

Description automatically generated

#### Cropping the Images

Each image has a specifically listed space on two corresponding CSV files, one for the outlined traffic light box at the bulb itself. As we are only attempting to detect traffic light boxes, we only utilize the files labeled as BOX. The BOX CSV files include important information about each image such as the name, coordinates for the traffic light, the labeled class, and the frame number from the video it was pulled. To fit in our experiment, we only use the labeled class and the coordinates of the traffic light **[See Figure 4]**.

To enable quicker retrieval, we cropped the images from the annotation files. This is achieved by first creating a data frame out of each accessed annotation file. We then use the snippet out of the path of the file to locally access the file. We then use the coordinates to crop the whole image and place it into one of the 7 folders (corresponding to each possible class) in a folder consisting of all the cropped images.

#### Constructing the Training/Test/Validation Datasets

To feed data into our model, we created three key folders for our use: train, test, and val. Furthermore, we set the amount of data stored in these folders. 80% of the entire data went into the train folder, 10% went into the test folder, and the remaining 10% went into the validation folder.

Since the current images in the folders are sorted, we shuffle the list containing all image paths to randomize the split of the data. Following the shuffle, calculate the approximate size of each dataset (based on the ratio presented above) and use those indices to access the image paths and move image files to their respective train, test, or val folders. It is important to note that the train, test, and val folders consist of class subfolders to allow for automatic scanning by TensorFlow.

To construct a valid dataset that the neural network can utilize, we use TensorFlow’s Preprocessing directory from a dataset to construct our three datasets with the image height and width set to 180px and a batch size of 32.

### Varying Factors

#### Applying Image Transformations[9]Diagram Description automatically generated

* + - 1. **Experiment 1: Histogram Equalization**

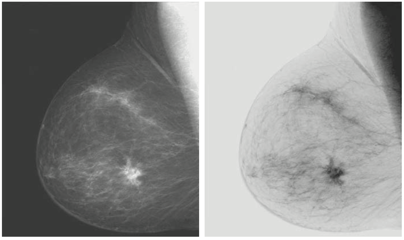
Histogram processing/equalization depends on the probability of a certain intensity level occurring in the pixels of an image. Each intensity level occupies a number from 0 to 255-pixel scale with 0 corresponding to low intensity (darker image) and 255 corresponding to the greatest intensity (brighter image).

For each pixel intensity value, a histogram is made plotting the relative frequencies of each intensity level, also known as the graph of probabilities. This graph determined whether the majority of the image is dark or bright **[Figure 5]**.

In histogram processing, we take this graph and equalize these intensity values using calculus operations to achieve a balanced level of intensity, producing a similar number of pixels for each intensity level. This, in turn, aims to provide a higher-contrast image, improving clarity.

* + - 1. **Experiment 2: Logarithm and Inverse [Figure 6]**
         1. LogarithmA picture containing text, person

            Description automatically generated

S is the final intensity value of the pixel, *c* is a constant calculated to be the greatest intensity level (rescaled to the 0-1 spectrum), and *r* is the value of the pixel being looked at. The ultimate goal of the transformation is to highlight the high-intensity pixels and dull the low-intensity pixels for more clarity. 

* + - * 1. Inverse (Negatives)

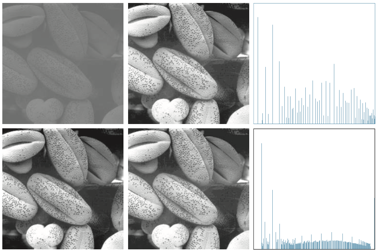


*L - 1* is the maximum possible intensity value, and *r* is the intensity value of the pixel going to be transformed. This operation aims to make the low-intensity parts of the image shine to bring more clarity to the otherwise darker parts of the image.

* + - 1. **Experiment 3: Gamma [Figure 7]**

Look at (b)(2) for labels of *s, c,* and *r*. Just like the logarithm transformation, the aim of the gamma (power) transformation is to bring high contrast into the image by creating a greater difference in the lowest and highest intensity pixels. However, the is a constant set by the user to define the corresponding function. In our case, we set it to 0.25 for optimal results, making our equation .

* + - 1. **Experiment 4: Histogram Specification [Figure 8]**

Histogram specification is a more concentrated version of histogram equalization. It still performs operations on the ‘graph of probabilities,’ but it uses calculus and float rounding to construct an image that has been equalized with a reference image. The reference image provides a basis for comparison to enhance the contrast of the image using another image.



### Training

#### Training Environment

* + - 1. Pre-Checklist

Prior to building the model, we normalized the node values to fit all 255 potential values of a pixel into a scale from 0 to 1. This enabled the first layer, the flattened layer, to flatten the image by extracting every single pixel value in a one-dimensional list. This process was repeated a total of three times for each of the RGB (Red, Green, Blue) pixel channels for color images.

* + - 1. Foundational Model Architecture

After consideration of the different types of convolutional neural networks set for image classification, the ResNet model was chosen[10]. Firstly, it has over 23 million parameters embedded into the model for training and a relatively deep network architecture, all pointing to more computations in losses and a more accurate model[11]. Secondly, the model has been known for its ability to maintain low error rates during training, which also corresponds to greater accuracy/precision values. Lastly, the model is comparatively simpler to construct compared to its competitors.

The model was compared to alternatives such as GoogLeNet, AlexNet, and the VGG-16 architectures. The ResNet model performed better than the AlexNet and VGG-16 models due to the increased number of parameters and loss-based computations involved in the training cycle. Although the GoogLeNet has a similar number of trainable parameters, the ResNet model achieved greater accuracy than the GoogLeNet.

#### Model Structure [Figure 9]Text Description automatically generated

* + - 1. **ResNet50**

This part of the architecture is 2048 nodes long with 2 channels and concerns the primary filter applied to each of the images.

* + - 1. **Flatten**

Once an image has been through the filters as part of the ResNet architecture, it is then flattened pixel by pixel to then be inputted into the nodes of the neural network.

* + - 1. **Dense**

As the image has now been flattened, the next two layers will be responsible for the classification of the image. This layer will have nodes that will be calculating algorithms for image classification.

* + - 1. **Activation**

This layer is the output layer, a layer with 7 nodes (for each of the 7 classes) that spits out a list of numbers for us. The number with the highest value also referred to as the ‘brightest’, will be the predicted value of that image.

## **Acknowledgments**

I would like to express great appreciation to my mentor Dr. Ross Greer. I would like to express my gratitude to Polygence and the opportunity to research as a part of their community.

## **References**

1. Gasparian, Martin. “Accidents at Intersections in California.” *Maison Law*, 11 Aug. 2022, <https://maisonlaw.com/personal-injury/car-accidents/intersections/>.
2. Brown, Molly. “Self-Driving Cars Could Reduce Accidents by 90 Percent, Become Greatest Health Achievement of the Century.” *GeekWire*, 29 Sept. 2015, <https://www.geekwire.com/2015/self-driving-cars-could-reduce-accidents-by-90-percent-become-greatest-health-achievement-of-the-century/>.
3. Kottasová, Ivana. “Cars and Traffic Signals Are Talking to Each Other | CNN Business.” CNN, Cable News Network, 29 Oct. 2018, <https://www.cnn.com/2018/10/29/business/volkswagen-siemens-smart-traffic-lights/index.html>.
4. “But What Is a Neural Network? | Chapter 1, Deep Learning.” YouTube, YouTube, 5 Oct. 2017, [www.youtube.com/watch?v=aircAruvnKk](http://www.youtube.com/watch?v=aircAruvnKk).
5. R. Kulkarni, S. Dhavalikar and S. Bangar, "Traffic Light Detection and Recognition for Self Driving Cars Using Deep Learning," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018, pp. 1-4, doi: 10.1109/ICCUBEA.2018.8697819.
6. E. T. Hassan, N. Li and L. Ren, "Semantic Consistency: The Key to Improve Traffic Light Detection with Data Augmentation," 2020 IEEE Intelligent Vehicles Symposium (IV), 2020, pp. 1734-1739, doi: 10.1109/IV47402.2020.9304653.
7. L. Gou et al., "VATLD: A Visual Analytics System to Assess, Understand and Improve Traffic Light Detection," in IEEE Transactions on Visualization and Computer Graphics, vol. 27, no. 2, pp. 261-271, Feb. 2021, doi: 10.1109/TVCG.2020.3030350. Yabuuchi K, Hirano M, Senoo T, Kishi N, Ishikawa M. Real-Time Traffic Light Detection with Frequency Patterns Using a High-Speed Camera. Sensors. 2020; 20(14):4035. <https://doi.org/10.3390/s20144035>
8. Jensen, Morten Bornø “LISA Traffic Light Dataset.” Kaggle, 28 Feb. 2018, [www.kaggle.com/datasets/mbornoe/lisa-traffic-light-dataset](http://www.kaggle.com/datasets/mbornoe/lisa-traffic-light-dataset).
9. Woods, Richard Eugene, and Gonzalez, Rafael C. *Digital Image Processing*. India, Pearson India, 2018.
10. “Convolutional Neural Networks (CNNs) Explained.” YouTube, YouTube, 9 Dec. 2017, [www.youtube.com/watch?v=YRhxdVk\_sIs](http://www.youtube.com/watch?v=YRhxdVk_sIs).
11. Danielsen, Nina. “Simple Image Classification with Resnet 50.” Medium, Medium, 26 Nov. 2019, <https://medium.com/@nina95dan/simple-image-classification-with-resnet-50-334366e7311a>.