Detection of Variable Message Signs from Image Input

Chase Howard
University of Washington Bothell
CSS 581 Machine Learning
Fall 2021

Contents

Abstract

There are an increasing number of partially autonomous vehicles (AV) on the road which require the most up to date information on the environment for appropriate and safe decision making. Temporarily relevant information regarding road conditions, detours, traffic conditions, and closures are often displayed on Variable Message Signs (VMS) on the side of roads. This information can be critical for AV to react appropriately in each situation, which then gives rise to a need for the capability f extracting information those VMS to be interpreted by AV. The goal of this research was to build a system to extract Regions of Interest (ROI) from an image and classify whether that ROI contains a VMS or not. Specifically, this project is looking to build and label a database of images containing VMS, implement a method to segment the images based on ROI likely containing VMS, evaluate various features by which a VMS may be distinguished, and train a Machine learning model to accurately classify a Variable message sign. The XGBoost model built off the metrics extracted from the generated database showed a precision, recall, and F1 score of approximately 0.87 when classifying variable message signs in a 10-fold cross validation scoring of the extracted features. An additional item of note, most failures to classify VMS could also be seen as exceptionally poor instances by segmentation method identifying accurate ROI, leading to an improper classification. This lends to the idea that the machine learning model may perform better with an improved segmentation method, additional metrics allowing for proper classification of VMS within improperly segmented images, or a model which is decoupled from the segmentation method and can accurately identify VMS with imperfect segmentations.

Introduction

Problem statement

Autonomous vehicles are always observing surroundings through various sets of sensors to make decisions over vehicular controls. Traffic and road conditions are temporal component that is currently unable to be updated via traditional methods. Variable Message Signs (VMS) often contain information updated to reflect the current road conditions which may not reach autonomous vehicles via the traditional update mechanisms. This information may indicate changes in emergency road conditions (potholes, bridge failures, disabled vehicles, temporary detours in place, etc.) or extreme weather conditions (black ice, snow, rain, etc.) The purpose of this project is to design a framework for segmenting and detecting VMS within images acquired from autonomous vehicles to accurately detect when a VMS is in the scene. The segmented image can then be passed to a separate module for tracking and text detection for data extraction. A tracking module which has been passed a Region of Interest (ROI) can then predict the VMS position in the next frame, reducing computational requirements. A text analyzing module will be capable of extracting the information from the VMS and adapting the information to a proper format for processing by the autonomous vehicle. The scope for this project is explicitly centered around the detection of VMS in a still image.

Some ethical considerations, these VMS have been compromised by malicious actors in the past. There is a possibility a malicious actor may hack a VMS to pass incorrect or possibly dangerous information to autonomous vehicle networks. This must be considered when processing VMS data and what actions are acceptable from information extracted from a VMS.

Properties of Variable Message Signs (VMS)

There are two properties of VMS which allow for recognition in image data. These generally center around the shape and colors on the signs. The VMS are designed to stand out to human vision with high contrasting colors, easily visible in most environments. This makes VMS a good candidate for detection using visible wavelength camera data. Some common features found within VMS:

- Common positional locations on the road (although not standardized)
- Rectangular or Square in shape
- Black or dark background color
- Orange or Yellow text color
- Common fonts used, although not standardized with variable character heights

While there are many similarities with respect to the signs, there are also a fair number of differences which can make processing more difficult such as:

- Can have damage to signs (pixels out or weathering)
- Varying weather conditions obscuring signs (snow, rain)
- Varying backgrounds which may decrease contrast of signs
- Lighting conditions due to time of day
- Lighting conditions due to orientation of the Sun to Sign to Camera (Glare / Saturation)
- Shape modification due to sign orientation
- Color shifting due to Lighting conditions

These factors will need to be considered when detecting Regions of Interest as well as extracting features from each of the regions to more suitable identify signs.

Related Work

Ellahyani, A., Jaafari, I., Charfi, S. (2021). Traffic Sign Detection for Intelligent Transportation Systems: A Survey. *ICCSRE'2020,* E3S Web of Conferences 229, 01006

Fleyeh, H., Dougherty, M. (2014) Road and Traffic Sign Detection and Recognition

Most related work in the field has been in relation to traffic sign detection. In essence, this is a similar problem to be solved. Both are attempting to identify signs through various segmentation methods and extract meaningful information from the sign. The main difference being the type of sign being detected.

Both articles mentioned above touch upon similar concepts regarding varying methods by which to detect signs, either using color-based detection methods or shape-based detection methods. The article list the difficulties associate with each task as well as possible mitigation strategies which have been implemented. With respect to color-based segmentation, difficulties arise with color shifting dueto illumination levels and angle of camera / signs. The color of the sign will change depending on the time of day, its age and physical condition, its reflective properties and orientation with respect to the sun and camera. This makes a difficult challenge of creating a global model to address all concerns and provide acceptable performance. There is more variation in the color-based detection methods according to the linked articles. However, there are methods by which to mitigate this through

implementation of color constancy through normalizations and mean shifts to preserve color information using the HSV color space, as Hue is invariant to lighting conditions; however, this isn't always effective as there can be multiple images within a scene the same color as the sign.

Discussions involving difficulties with shape-based methods include the presence of similar shapes within a field of view which may provide false positives – such as a mailbox, windows, cars, etc. This is often handled by the next processing phase which determines properties of the object segmented for classification. One study in 2014 [Timofte et al.] showed the implementation of Histogram Oriented Gradients (HOG) (through an SVM) with a boosting classifier (Adaboost) for detection of signs. Where these methods tend to have difficulties is with translations, scaling, and rotation of signs.

It should also be noted that recently (2019-present), Neural networks and other machine learning models have been used for both the sign recognition and detection processing phases. Some of the studies have used several layers of cascading machine learning models to properly set thresholding and extraction parameters to properly segment out ROI containing VMS. Convoluted Neural Networks and Deep Learning models appear popular in this space. These were not investigated in this project due to timing constraints.

Data

Notes regarding the Segmentation methods are listed in the Appendix and not within the report itself; however, they bare a large impact on the overall performance of the model and outputs generated.

Database

The database was created by scraping images off Google Images using a tool 'SerpAPI' [https://serpapi.com/] to collect several images containing Variable Message signs. This tool worked well; although, there was manual post processing required to remove images not relating to roadways or traffic. All images included in the database contain a Variable Message sign. This doesn't guarantee that each image will have a ROI Classified as containing a VMS, as there will be multiple ROI exported from the Segmentation method. This resulted in a more or less balanced case of positive to negative labels withing the database. This is all described in more detail below. There was a total of 621 images in this database and the distribution of positive and negative labels is described in the image below [isVMS == 0 is negative class, isVMS == 1 is positive class]. Due to the time limitations imposed by the class and the time required for scraping images and manually labeling them, the database remained somewhat small. This resulted in a database of images less representative of real-world weather and lighting conditions which may hurt the model performance in the real world. This is something to be investigated further.

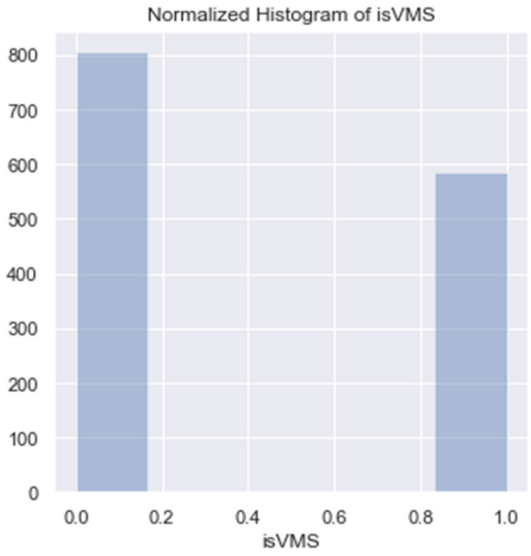


Figure 1: Class Label Distribution in dataset

Workflow / Data Pipeline

The general workflow can be seen in the diagram below. Steps outlined generally follow the procedure:

1. **Image Acquisition:** Camera output or loading from local storage
2. **Image Segmentation:** Adaptive thresholding and color detection for ROI extraction
3. **Feature Extraction:** Analysis on ROI, outputting metrics of interest
4. **Classification and Output:** Model prediction and classification
5. **External Analysis (Tracking + Text Analysis Modules):** out of scope for this project

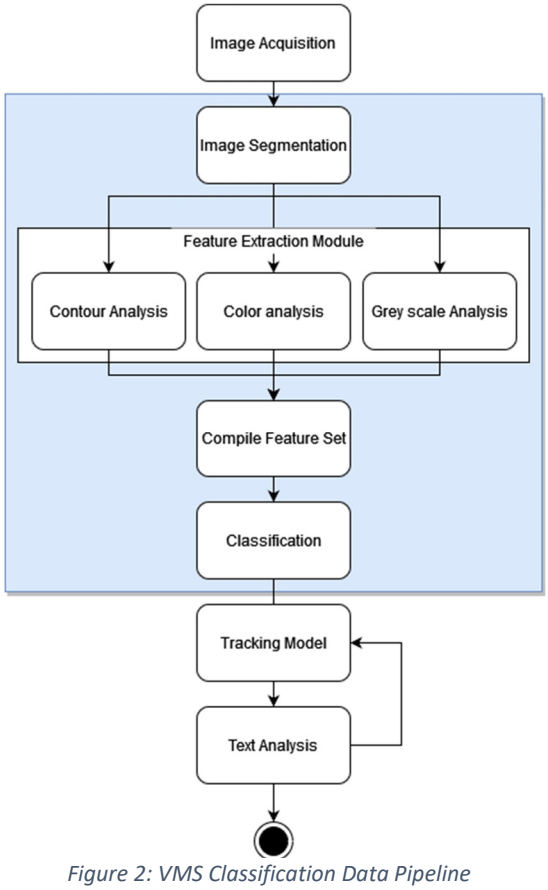


Figure 2: VMS Classification Data Pipeline

All items within the Blue Rectangle in the diagram above would be considered in the scope of this project. The Image Acquisition will be simulated by loading images from a local storage location to simulate the acquisition of an image by a camera. The initial database was populated by scraping images from Google Images using SerpApi. These images were then processed using a series of segmentation methods listed in the Appendix.

Feature Extraction

A total of 22 feature were extracted from each ROI for characterization. Feature selection is described below from each of 3 separate methods of analysis including: Shape, Color, and Greyscale. For brevity, Color and Greyscale have been combined as the same metrics are calculated; the only difference is the metrics are gathered from a color (HSV) image and a Greyscale image. This is to identify the potential upside of using color versus greyscale, or if the simpler (Greyscale) processing method can be used.

Each of the features should provide benefit in differing situations. Shape-based methods should be able to avoid difficulties related to lighting conditions and color shift for VMS detection while capable of handling transformations such as rotation for ease of reading text in post classification processing.

Color based methods will contain more noise, this is an inherent restriction of differing lighting conditions as well as camera acquisition settings. Additionally, color will be influenced by environmental conditions such as weather, time of day, and VMS orientation. These factors will most likely result in a weaker classifier – although color methods do demonstrate strengths in detection as noted before.

Shape Feature Extraction

Shape analysis was executed on each of the ROI identified by the segmentation methods discussed earlier. The initial set of metrics are as follows:

- **Centroid (Cx, Cy):** location with respect to local region of interest, not global image
- **Area**: total pixel count within the identified ROI
- **Perimeter**: pixel count along outer bound of ROI
- **Aspect Ratio**: ratio of bounding rectangle's width to height
- **Extent**: percentage of area filled by ROI when fit with a bounding rectangle
- **Solidity**: percentage of area filled by ROI when fit with a convex hull
- **Orientation**: angle of major and minor axis when an ellipse is fit to the ROI

Color / Greyscale Feature Extraction

Color and Greyscale Analysis were performed on each of the Histograms generated within a single ROI. Analysis includes the following metrics

- **Area**: integral of entire Histogram, bin difference and count number
- **AreaPercent**: percent of histogram within defined area of histogram (defined thresholding)
    - Area_subset / Area_total
    - Greyscale looks pixels with values in the top ¼ of values [0, 65]
    - Color looks at pixels with values indicating orange to yellow inclusion [10,50]
- **Mean**: mean pixel value
- **Peak**: highest occurring pixel value

Feature Distributions are in Appendix B

No feature transformations were included as initial model performance was sufficient without binning the data or transforming into categorical values. This would have included an additional step in processing which was not needed, although it may make decision boundaries clearer. This may be a point to investigate in future iterations of the project.

Feature Correlation

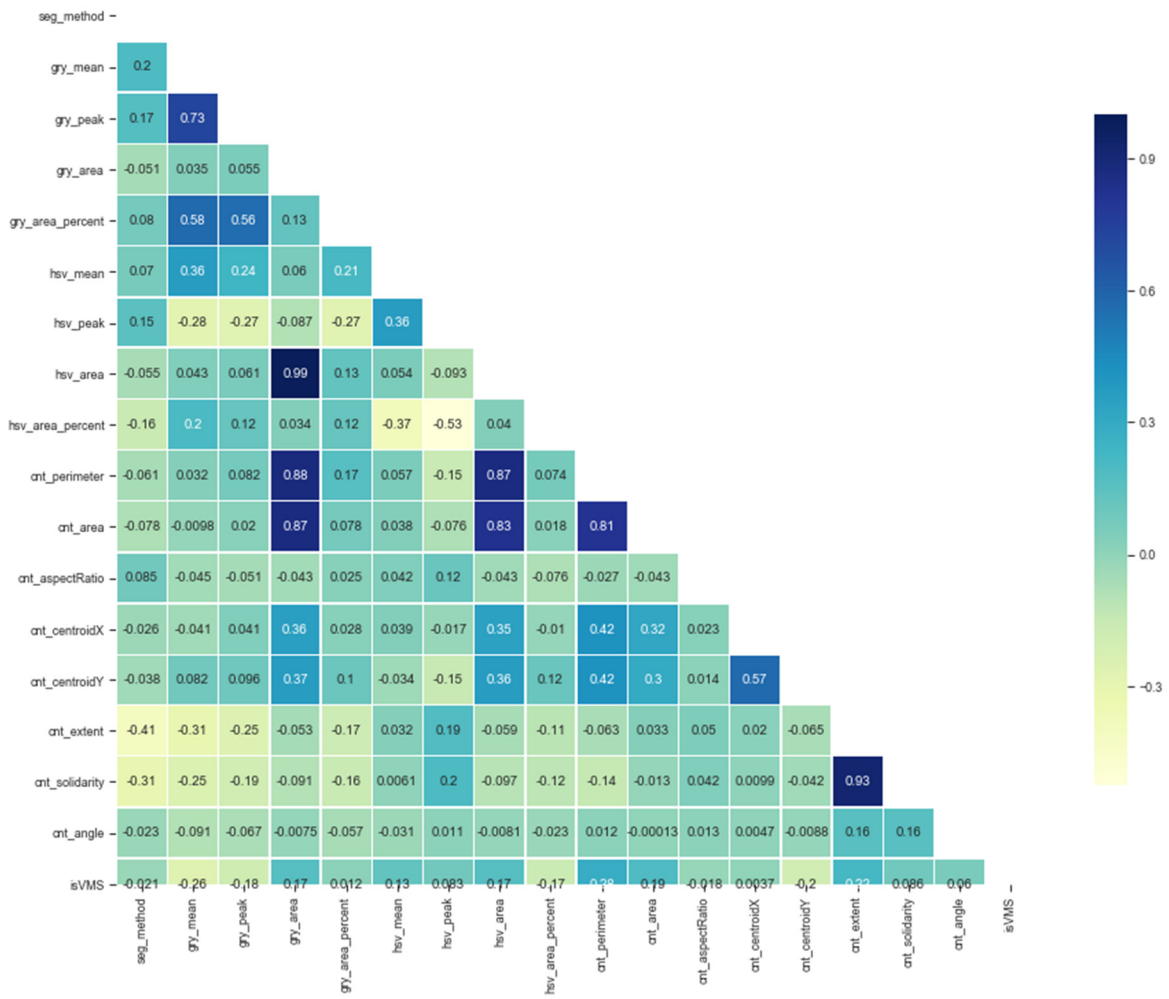


Figure 3: Correlation of All Features

There are several highly correlating features observed in the initial correlation. The idea then is to limit the number of features as much as possible without degrading any model performance. It should be noted that model performance is listed in the following section below; however, for clarity's sake we will cover feature down selection in this section while understanding the two features sets under discussion have nominally the same performance.

There were a few iterations that happened between the initial feature set above and the final feature set below, which involved the tweaking of different parameters in the feature extraction module for more separable data as well as development of a convolution of features to decrease the overall number of features while preserving important data within each feature. The contributions of the top features can be seen in the SHAP plot below – this aided in the down selection as it provided a visual representation of how the model was making correct decisions in positive and negative classification instances.

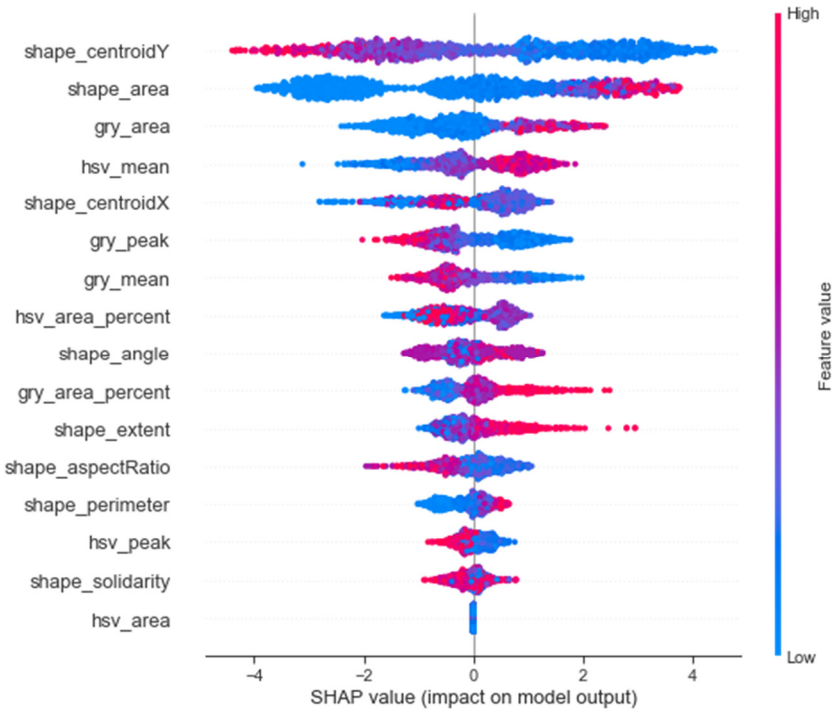


Figure 4: SHAPELY plot of entire Feature set and the impact of model output

It can be seen in the figure above that the Shape based metrics had the largest impact on model outputs. When considering the features to include in the down selection, there were two primary factors of interest that were adhered to. The first and main point was to include features from each extraction method (shape, greyscale, and color). When moving to fewer features, if you only select the strongest feature from a single extraction method, you will be left with a more rigid model prone to failure under a prescribed set of conditions. With a more diverse feature set, one will have a more generalized model capable of performing under a variation of environmental conditions – such as lighting changes, reflections, color shifts, etc. The second item of note was that relating the time required to process each image. With the full set of 22 features, the data pipeline was averaging around 250 milliseconds to extract all features (>90% of entire image pipeline) – approximately 11.4 milliseconds per feature if a linear relationship is assumed. If we have a goal of processing image streams at 10 frames per second and assume a linear relationship between feature extraction time and the number of features collected, then there is a target of generating a model capable of robust classification using only 7 features. So, the question then becomes, can we build a model with similar performance off 1/3rd the original features.

After multiple iterations of feature removal and convolution, the final feature set was compiled as described by the SHAP plot below. This subset of features showed extremely similar performance to the original, larger set of features. It should be noted that there were 3 features from the Shape module (the strongest predictor), 2 features from the greyscale module, and 2 features from the color module (labeled as 'hsv' below). This should allow for a more robust, generalized classification model.

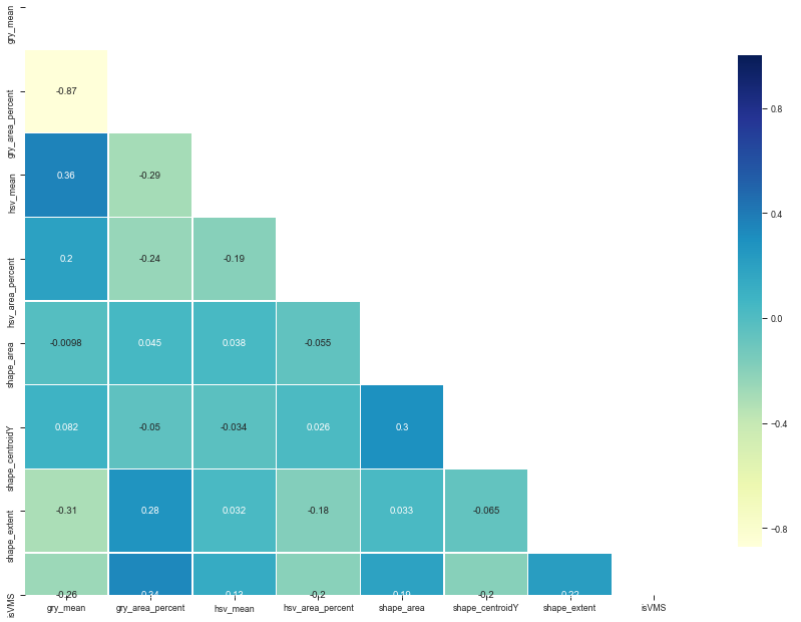


Figure 5: Correlation of Select Feature Set

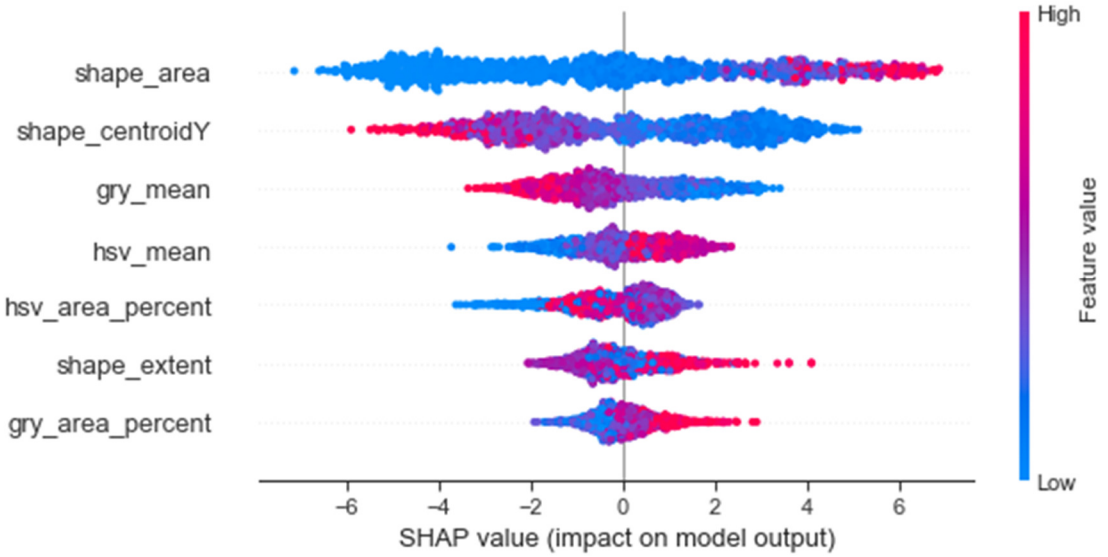


Figure 6: SHAPELY Plot of feature impact of Select Feature Set on XGBoost Output

Experiments and Results

Model Building

This was a binary classification problem to detect whether each ROI extracted from an image (Segmentation module) contained a Variable Message Sign or not. There were a set of 5 models evaluated: Logistic Regression, Naïve Bayes, Decision Tree, Random Forest, and XGBoost. For discussion, Naïve Bayes and Random Forest will be omitted as they performed similarly to Logistic Regression and XGBoost respectively. For the model selection experimentation, a 10-fold cross validation was used for gathering of accuracy, precision, recall, f1, and AUC metrics. Metrics below are reported for data extracted from 'All Features', the full set of 22 features described above, and for 'Select Features', using the subset of 7 features described above. Note, all metric values are displayed as a percentage and a discussion regarding performance is located below.

Table 1: Metrics reported for Models Trained on All Features (22 total)

	Logistic Regression	Naïve Bayes	Decision Tree	Random Forest	XGBoost
accuracy	75.03	68.34	84.96	90.35	90.79
precision	75.62	81.02	82.02	89.30	89.40
recall	63.22	28.66	83.40	88.79	88.71
f1	67.69	39.49	81.51	88.34	88.77
roc_auc	85.53	87.51	84.05	97.39	97.07

Table 2: Metrics reported for Models Trained on Select Features (7 total)

	Logistic Regression	Naïve Bayes	Decision Tree	Random Forest	XGBoost
accuracy	69.64	75.17	87.48	89.28	89.35
precision	67.74	81.46	82.92	89.54	88.04
recall	55.86	57.11	85.46	85.81	86.59
f1	60.21	64.94	83.81	86.79	86.94
roc_auc	78.55	85.69	86.34	96.55	96.40

Interestingly, the Naïve Bayes and Decision Tree models showed improved performance with the smaller feature set when compared to the entire feature set. This could be due to a simplification of features which avoided overfitting and led to a more generalized model. Performance among the other models generally dropped; however, with the ensemble methods (Random Forest and XGBoost) the drop in performance was rather negligible.

Table 3: Comparison of XGBoost Trained with All and Select Features

	All Features	Select Features
accuracy	90.79	89.35
precision	89.40	88.04
recall	88.71	86.59
f1	88.77	86.94
roc_auc	97.07	96.40

XGBoost was the model selected to evaluate further. The Logistic Regression and Naïve Bayes models showed too poor of performance. The Decision Tree model performed well and has the benefit of some explain ability; however, in this scenario explain ability isn't crucial (although helpful for tuning). Both the XGBoost and Random Forest showed similar performance and the decision was made to move forward with the XGBoost model as it tended to be faster during evaluations. There was little difference between the Random Forest Classifier and the XGBoost Classifier.

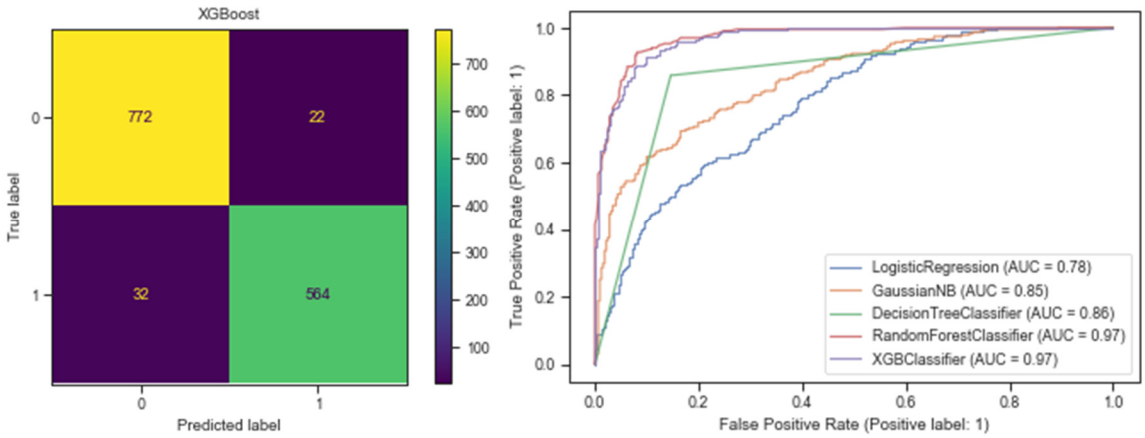


*Figure 7: (left) Correlation Matrix for XGBoost and (right) AUC Plot for Select Feature Set (**Not Cross Validated**)*

Discussion

There was a small set of images used for validation of the models (total of 21) which provided a total of 56 contours to test on. The correlation matrix for the XGBoost model which ran the Validation set is displayed below.

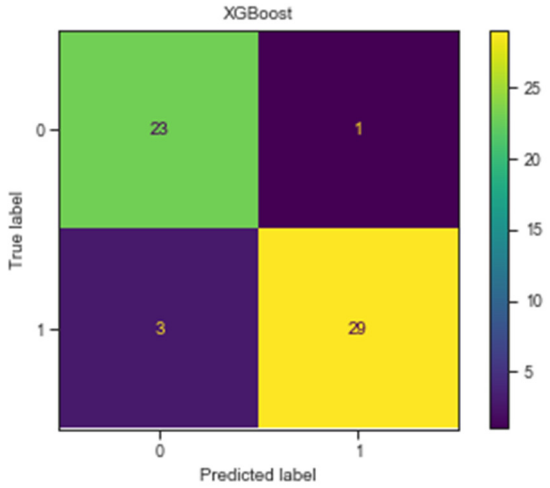


Figure 8: Correlation Matrix of XGBoost model on Validation Set

The Validation Set used was too small of a dataset to draw any meaningful conclusion, it will need to be expanded in future iterations of the project; however, it appears to be performing at a similar level to that of the cross validated test and training sets used. Attached below are some images of output generated by the model selection. Below, the ROI colored 'Green' indicates a positive label by the model, and a ROI with a 'Red' color indicates a negative label placed by the model. Additionally, the probability of classification for each ROI is printed on the image in the same color as the contour.

Figure 9: Successful classification examples by the XGBoost model using Selected Feature set

Generally, the output of the XGBoost model predictions which are correct tend to look similar. In every scenario in which the segmentation method drew a tight contour around the VMS, the XGBoost model was able to accuratley identify a VMS. This showed a tight coupling between the performance of the segmentation method and the output of the model. This is not necissarily ideal behavior.

Additionally, it should be noted the confidence at which the model is predicting each of these cases. In general, the model was much more confident in predicting negative classes when compared to postiive classes. This could be due to a slight imbalance in data or a lack of descriptive features for identifying postive classes. With respect to the images in the database, the Segmentation method was successful around 92% of the time and the model performed well as seen by the metrics reported. However, with real world data, the segmentation method may struggle to perform as well, leading to a significant drop in performance. For a robust classification, the model should be able to detect a VMS with a failed segmentation method. It could also be a result of the metrics chosen not accurately describing a positive class as clearly as it does a negative class.

The failure modes are much more interesting to look into and give a better idea of model improvements possible in future iterations. Below you can see two False Positive cases and one False Negative case (Note: these aren't all from the validation set). The most common error in model classification can be seen by a failed segmentation. This can be seen in both the top images. Additionally, themodel had difficulty classifying larger contours properly. This is likely due to the large influence of contour area on the decision. In future iterations, to further generalize model performance there should be an investigation regarding decoupling or decrease the weight of the shape parameters from the feature set to allow for non-perfect contours to be properly classified.

Figure 10: Classification Errors by XGBoost model using Selected Features set

Conclusion and Future Work

This project has shown two separate feature sets which demonstrate acceptable differentiability between image segments containing VMS and those which do not. The smaller feature set displayed shows a negligible drop in performance when compared to the larger features set, indicating a minimal loss in meaningful data with features being dropped. Finally, a good classification model of images was build using XGBoost with the 7 selected features, showing high performance as seen in Table 4, which may be feasible for adaptation to a real-time scenario. The performance of this pipeline is currently capable of processing approximately 3 to 4 frames per second at the current state. Time limitations relating to the manual classification of each ROI constrained the size of the dataset, in the future this should be expanded to portray differing weather and lighting conditions more accurately for a more generalize model. Other future work may investigate the application of weights to each of the shape, color, and greyscale features with the hope so further decoupling the model performance from the segmentation success while still maintaining performance. Additionally, there may be future work looking to optimize the feature extraction step for further reductions in processing time for increasing the Frames per Second processed for real-time scenarios.

Appendix A: Segmentation Method

The segmentation aspect of the project aims to identify contours withing an image indicating a ROI which may or may not contain a VMS. Two Discrete Processing Pipelines were investigated for feasibility, Adaptive Thresholding and Color Detection. Currently, the two segmentation methods are run independently, each generating a unique output of contours; however, in the future there are plans for investigating the possibility of chaining the two methods together or using some form of ensemble for improved performance.

There are a lot of similarities in both methods listed above and they generally following the following pipeline.

- Image is input and a Gaussian Blur is applied with varying kernel size
- Image is converted to color format of interest (GRAY or HSV)
- Initial Thresholding (Course)
- Secondary Thresholding (Fine)
- Combination of Morphological transformation (Dilation and / or Erosion) on binary image
- Bitwise Masking on various binary images to generate composite mask
- Contour Generation on Masked Image
- Output three largest contours by area which are assumed most likely to contain VMS data

Adaptive Thresholding

In general, this segmentation method performed well in that it was able to identify a ROI containing a VMS in >94% of the input cases. Adaptive thresholding takes advantage of the fact that VMS were designed to stand out in the environment with large rectangular shapes often with a dark background. See code for specific steps regarding Adaptive Thresholding; General advantages and disadvantages are listed below with respect to the alternative Color thresholding

Advantages

Adaptive thresholding works well in high contrast scenarios such as a VMS up against a blue sky or light, well lit backgrounds. This allows the Segmentation method to easily identify and separate out the dark background of the VMS. The benefit in most cases here, the entire sign is detected as a ROI which doesn't crop useful information such as text characters on the edge. Additionally, this method, when successful, returns the entire sign so that all text is preserved and not separated.

Disadvantages

The disadvantages of using this method are generally the opposite of the advantages. This segmentation method performs poorly in dimly lit images or images with darker backgrounds where edges cannot be detected between the VMS and the background. Occasionally, this segmentation methos will also bring in extra, irrelevant data when generating contours. This isn't as much of a problem when considering the goal is to identify text; however, it could have the effect of biasing the Machine Learning model with the inclusion of larger amounts of noise.

Examples

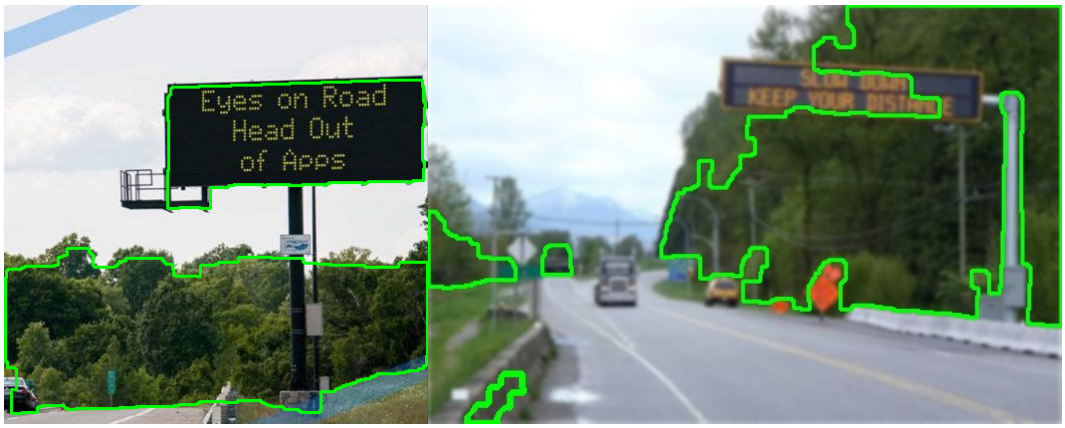


Figure 11: Successful Adaptive Threshold Segmentation (Left) and Unsuccessful Adaptive Threshold Segmentation (Right)

Color Thresholding

The Color Thresholding segmentation method was not as successful as the adaptive thresholding method (only ~88% successful); however, it often succeeds in instances where the adaptive thresholding fails making it a good complementary method. This segmentation method takes advantage generally standardized Orange to Yellow text located on the VMS.

Advantages

Color thresholding works extremely well in low light conditions. Orange colors on the sign show strong and sometimes even have a 'halo' effect which increases the number of pixels colored as orange/yellow. This serves well in complement with the Adaptive Thresholding method which often fails in low-light conditions.

Disadvantages

Color thresholding fails more frequently than the Adaptive Thresholding method. The main failure mode is due to splitting VMS into multiple ROI considering the spaces between each orange character on the VMS. Most cases of this are taken care of with a Morphological Transformation referred to as 'Dilation' which can expand ROI based on a set kernel size; the issue here is there is an optimal kernel size based on the size of the text in the image itself. Too large of a kernel, and one will begin to include additional noise which isn't representative of a VMS. Too small of a kernel and there will be multiple ROI for a single VMS. This parameter is a key candidate for improvement to increase the efficacy of the color thresholding.

Other failure modes include instances where there are color shifts, often time due to glare or saturated images. Additionally, sign orientation with respect to the sun and camera is important as it can result in a color shift where the orange letter is now out of band of the thresholding method. These failure mechanisms are also candidates for improvement if there can be a shift in the color detection range based on the saturation of an image.

Examples

Figure 12: Unsuccessful Color Threshold Segmentation (Left) and Successful Color Threshold Segmentation (Right)

Discussion

The segmentation method of the data pipeline is arguably the most crucial. Considering the sentivity fo the segmentation due to minor parameter changes and the downstream effects of each ROI with respect to the training of the model and ultimately the classification – the better the segmetnation method, the more accurate the ML will be able to classify each ROI. Each of the resuling ROI from the discussed segmentation methods was manually labeled indicating whether or not the region selected contained a VMS or not. Results can be summarized below:

Currently, both segmentation methods are being used to feed two sets of contours to the Feature Analysis Module. In the future, this needs to be simplified and improved; howoever, due to the

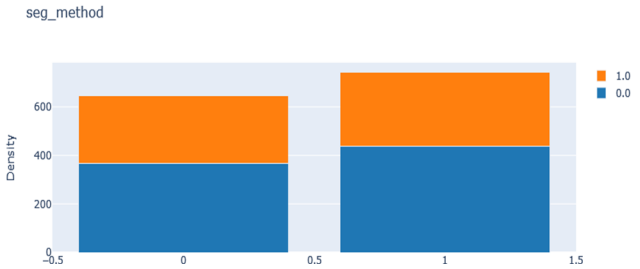


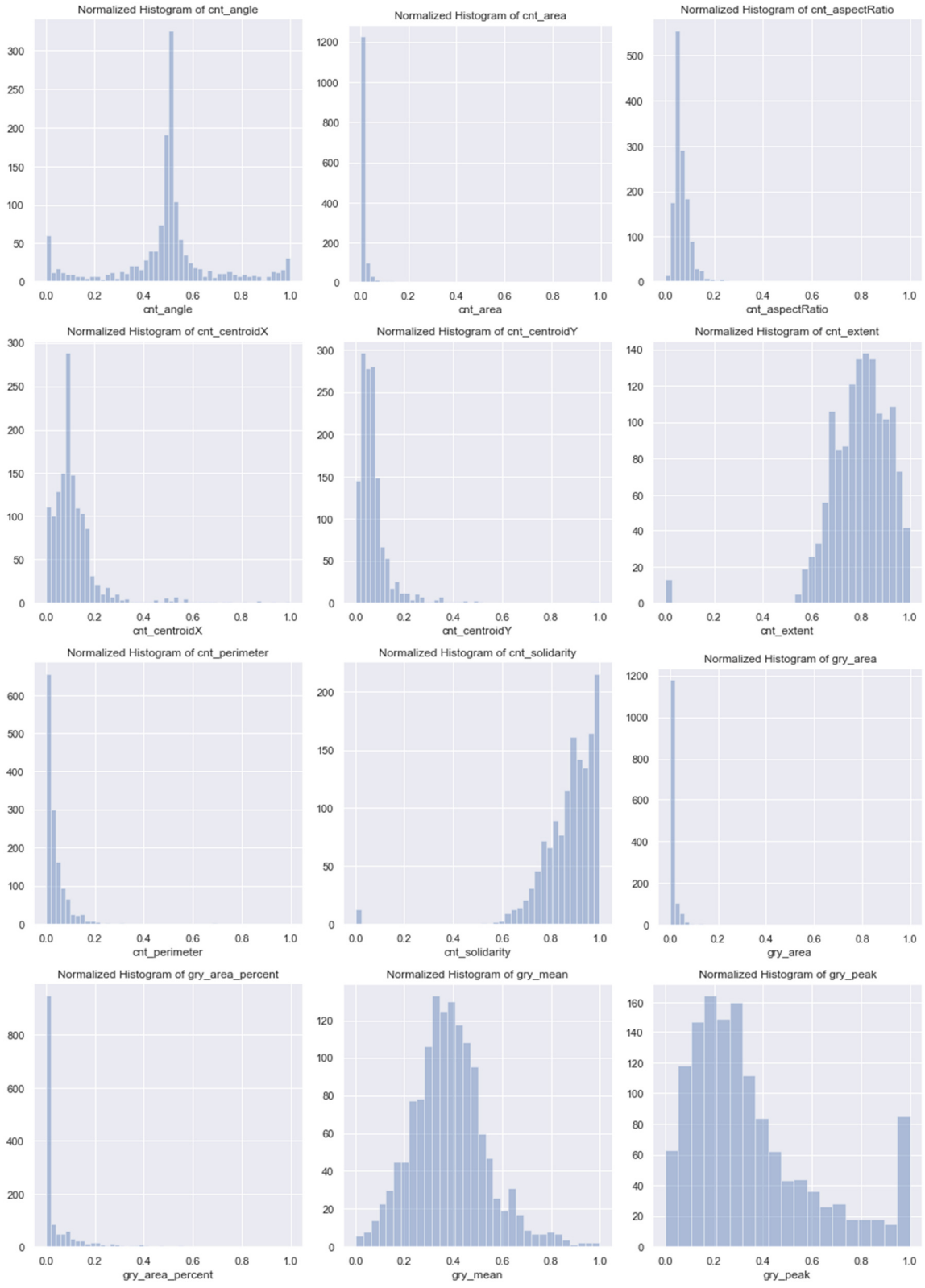
Figure 13: Segmentation Method Breakout { Orange = positive, Blue = negative }. X-axis represents segmentation method used { 0 = Adaptive Thresholding, 1 = Color Detection }

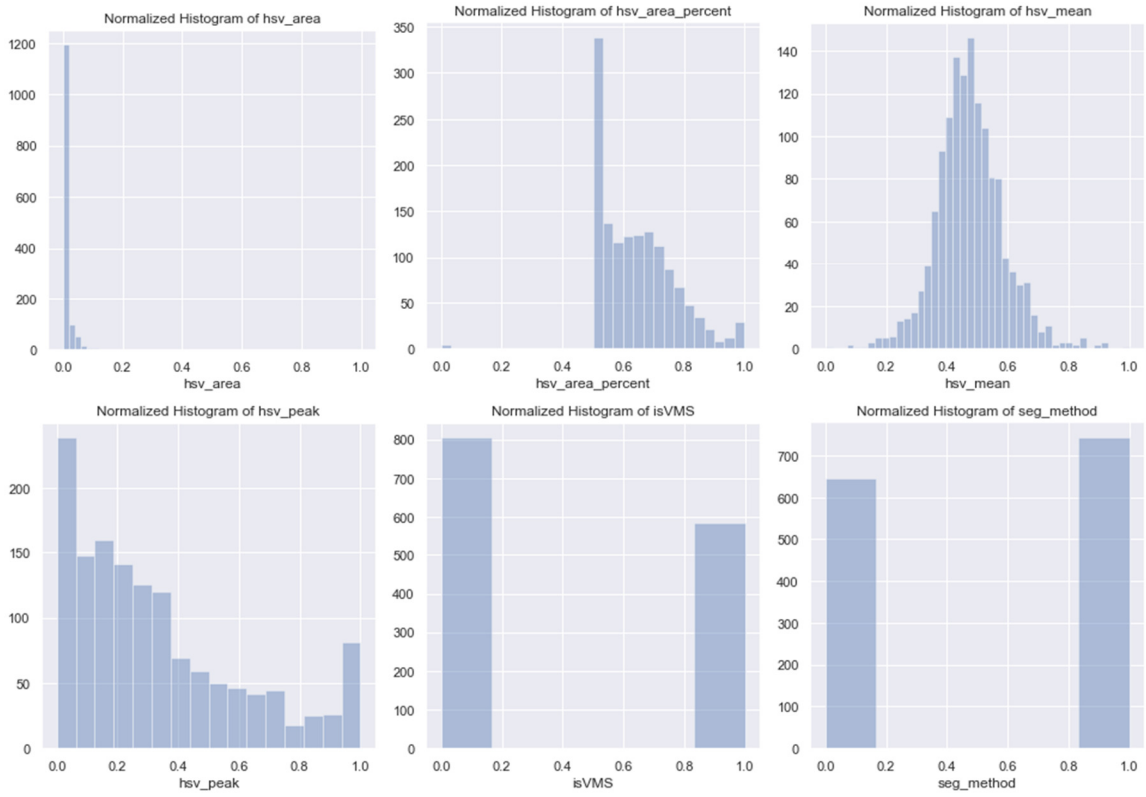
time limit imposed on this project, I am accepting the current status as 'good-enough' and moving forward with Feature extraction and analysis on two sets of contours.

Future Improvements

- Simplify segmentation method to single method to reduce the number of contours being processed and more accurately identify ROI
- Improving performance of Adaptive thresholding by combining with edge detection methods to identify Histogram Oriented Gradients (HOG) around VMS to remove noise.
- Adaptive kernel sizing would help immensely to improve the segmentation of images based on color thresholding

Appendix B: Feature Distribution




Normalized Histogram of hsv_area
hsv_area
Normalized Histogram of hsv_area_percent
hsv_area_percent
Normalized Histogram of hsv_mean
hsv_mean
Normalized Histogram of hsv_peak
hsv_peak
Normalized Histogram of isVMS
isVMS
Normalized Histogram of seg_method
seg_method