Classification of Gaps at Uncontrolled 3-legged Intersection (Day & Night)

Using SVM, ANN and BLM

Videsh Suman (150040095)
Kumar Priyanshu (150040096)

Abstract— **Gap classification and acceptance predictions provide very important inputs for performance evaluation and safety analysis of 3-legged uncontrolled intersections. The focus of this report is on the application of support vector machines (SVMs), Binary logit model (BLM) and Artificial Neural Networks (ANN) in understanding and classifying gaps at these facilities and comparing the results amongst these techniques to find the one with higher accuracy. The SVMs are supervised learning techniques originating from statistical learning theory and are widely used for classification and regression. ANN is a form of connectionism, inspired by biological neural networks, that learns progressively to predict the results of highly complex functions. BLM is a regression model where the dependent variable is categorical and the output is either 0 or 1. In this report, the feasibility of the SVM, ANN and BLM in analyzing gap acceptance is examined by comparing its results with each other. To accomplish our objective, SVM, BLM and ANN models are developed and compared by using data collected at 3-legged uncontrolled intersections at day and night. This technique can be used in advance safety warning systems for vehicles and pedestrians waiting to cross major stream vehicles.**

I. Algorithms

A. Support Vector Machine

A Support Vector Machine constructs a hyper plane or set of hyper planes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyper plane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

The basic linear SVM learning decision rules are given by F (x) = w • x + b where w is the weight vector and b is a bias. F (x) is the discriminant function associated with the hyper plane. Training data D is a set of n points of the form

$$D = \{(x_i, y_i) | x_i \in R^d, y_i \in \{-1, +1\}\}_{i=1}^{n}$$

Where y_i is either 1 or −1, to which the point x_i belongs in a D-dimensional feature space, R^d. The distance between the optimal separating hyper plane and the origin is given by |b|/ ||w||. If the training data are linearly separable, two hyper planes can be selected that separate the data such that there are no data points between them. The region bounded by two hyper planes is called the "margin". These two hyper planes, which are parallel to the optimal separating hyper plane, can be described by the equations w • x − b = +1 and w • x − b = −1. For the data sets that cannot be separated cleanly, Cortes and Vapnik modified the SVM algorithm by adding a soft margin. The soft margin method chooses the hyper plane that splits the mislabeled examples as cleanly as possible. The margin between the two hyper planes is given by 2/||w||, thus minimizing that ||w|| will result in maximizing the margin. The optimal separating hyper plane is calculated by maximizing the margin of the two hyper planes and minimizing the error as given by Equation

$$\min_{w,b,\xi} \left\{ \frac{\|w\|^2}{2} + C \sum_{i=0}^{n} \xi_i \right\}$$

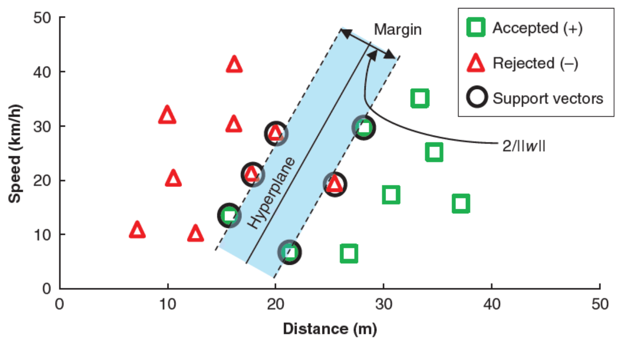

Fig. 1

Subject to

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i \qquad \text{for } i = 1, \ldots, N$$

The factor C and the slack variable ξ i in Equation 2 are introduced to take non separable data into account. The shape of the discriminant function (Equation 3) is controlled by constant C by applying a penalty for the samples that are located on the wrong side of the hyper plane.

The nonnegative slack variable ξ_i, measures the degree of misclassification of the data x_i. Through Lagrange dual optimization, the minimization problem in Equation 2 can be solved. The maximum margin hyper plane can be represented as in Equation 3 in regard to the support vectors.

$$f(x) = \sum_{i \in n} \alpha_i y_i k(x_i, x_j) + b$$

Where $k(x_i, x_j)$ = kernel function,

α_i = Lagrange multipliers, and

n = set of support vectors.

A kernel function is used to transform the data into a high-dimensional space. Various kernel functions are as follows: linear kernel $k(x_i, x_j) = (x_i, x_j)$, polynomial kernel $k(x_i, x_j) = (x_i, x_j + 1)^d$, and Gaussian radial basis function $k(x, y) = \exp(-\| xi - xj\|^2/2\delta^2)$, where d is the degree of the polynomial kernel and δ^2 is the bandwidth of the Gaussian radial basis function kernel. These functions can be used for constructing the OSH for different types of nonlinear input data. In the present study, as data were linearly non separable, the linear kernel function was used.

B. Artificial Neural Network

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurones. This is true of ANNs as well.

An Artificial Neuron is a device with many inputs and one output. The neuron has two modes of operation; the training mode and the using mode. In the training mode, the neuron can be trained to fire (or not), for particular input patterns. In the using mode, when a taught input pattern is detected at the input, its associated output becomes the current output. If the input pattern does not belong in the taught list of input patterns, the firing rule is used to determine whether to fire or not.

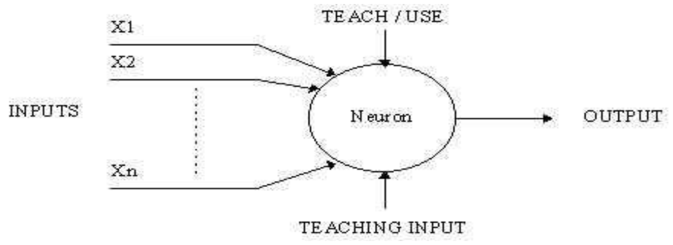

Fig 2.

Feed-forward ANNs allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straight forward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down.

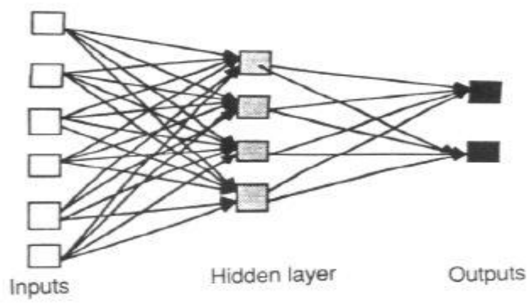

Fig 3.

Network Layers:

The commonest type of artificial neural network consists of three groups, or layers, of units: a layer of "input" units is connected to a layer of "hidden" units, which is connected to a layer of "output" units. The activity of the input units represents the raw information that is fed into the network. The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units. The behavior of the output units depends on the activity of the hidden units and the weights between the hidden and output units. This simple type of network is interesting because the hidden units are free to construct their own representations of the input. The weights between the input and hidden units determine when each hidden unit is active, and so by modifying these weights, a hidden unit can choose what it represents.

We also distinguish single-layer and multi-layer architectures. The single-layer organization, in which all units are connected to one another, constitutes the most general case and is of more potential computational power than hierarchically structured multi-layer organizations. In multi-layer networks, units are often numbered by layer, instead of following a global numbering.

Supervised Learning:

Supervised learning which incorporates an external teacher, so that each output unit is told what its desired response to input signals ought to be. During the learning process global information may be required. Paradigms of supervised learning include error-correction learning, reinforcement learning and stochastic learning.

An important issue concerning supervised learning is the problem of error convergence, i.e. the minimization of error between the desired and computed unit values. The aim is to determine a set of weights which minimizes the error. One well-known method, which is common to many learning paradigms is the least mean square (LMS) convergence.

Transfer Function:

The behavior of an ANN (Artificial Neural Network) depends on both the weights and the input-output function (transfer function) that is specified for the units. This function typically falls into one of three categories:

- **Linear (or ramp)**
- **Threshold**
- **Sigmoid**

For **Linear** units, the output activity is proportional to the total weighted output. For **Threshold** units, the output is set at one of two levels, depending on whether the total input is greater than or less than some threshold value. For **Sigmoid** units, the output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurones than do linear or threshold units, but all three must be considered rough approximations.

To make a neural network that performs some specific task, we must choose how the units are connected to one another, and we must set the weights on the connections appropriately. The connections determine whether it is possible for one unit to influence another. The weights specify the strength of the influence.

We can teach a three-layer network to perform a particular task by using the following procedure:

1. We present the network with training examples, which consist of a pattern of activities for the input units together with the desired pattern of activities for the output units.
2. We determine how closely the actual output of the network matches the desired output.
3. We change the weight of each connection so that the network produces a better approximation of the desired output.

C. Binary Logit Model

In Binary Logit Model **(**BLM**)**, the outcome is usually coded as "0" or "1", as this leads to the most straightforward interpretation. If a particular observed outcome for the dependent variable is the noteworthy possible outcome (referred to as a "success" or a "case") it is usually coded as "1" and the contrary outcome (referred to as a "failure" or a "non-case") as "0". Binary logistic regression is used to predict the odds of being a case based on the values of the independent variables (predictors). The odds are defined as the probability that a particular outcome is a case divided by the probability that it is a non-case.

The logit of success is then fitted to the predictors using linear regression analysis. The predicted value of the logit is converted back into predicted odds via the inverse of the natural logarithm, namely the exponential function. Thus, although the observed dependent variable in binary logistic regression is a zero-or-one variable, the logistic regression estimates the odds, as a continuous variable, that the dependent variable is a success (a case). In some applications the odds are all that is needed. In others, a specific yes-or-no prediction is needed for whether the dependent variable is or is not a case; this categorical prediction can be based on the computed odds of a success, with predicted odds above some chosen cutoff value being translated into a prediction of a success.

The logistic regression can be understood simply as finding the 'β' parameters that best fit:

$$y = \begin{cases} 1 & \beta_0 + \beta_1 x + \varepsilon > 0 \\ 0 & \text{else} \end{cases}$$

where 'ε' is an error distributed by the standard logistic distribution. (If the standard normal distribution is used instead, it is a probit regression.)The associated latent variable is $y' = \beta_0 + \beta_1 x + \varepsilon$. The error term 'ε' is not observed, and so the ***y'*** is also an unobservable, hence termed "latent" (the observed data are values of y and x). Unlike ordinary regression, however, the y and x. parameters cannot be expressed by any direct formula of the y and x values in the observed data. Instead they are to be found by an iterative search process, usually implemented by a software program, that finds the maximum of a complicated "likelihood expression" that is a function of all of the observed y and x values. The estimation approach is explained below. The logistic function is useful because it can take any real input 't', (t ∈ R) , whereas the output always takes values between zero and one and hence is interpretable as a probability. The logistic function σ (t) is defined as follows:

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

A graph of the logistic function on the t-interval (-6,6) is shown in

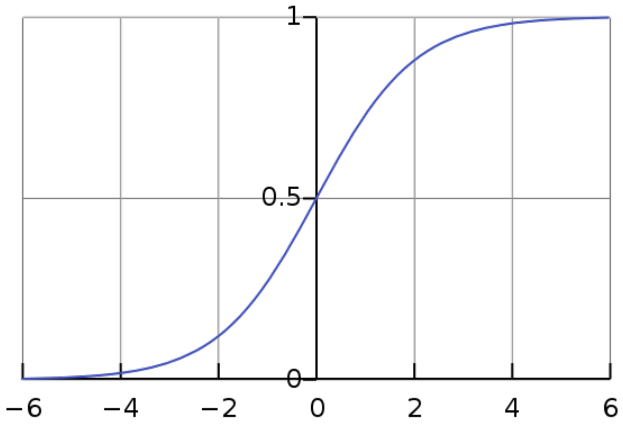

Fig 4. Sigmoid Function

Let us assume that 't' is a linear function of a single explanatory variable 'x' (the case where 't' is a linear combination of multiple explanatory variables is treated similarly). We can then express 't' as follows:

$t = \beta_0 + \beta_1 x$

And the logistic function can now be written as:

$$F(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

Note that F(x) is interpreted as the probability of the dependent variable equaling a "success" or "case" rather than a failure or non-case. It's clear that the response variables Y_i are not identically distributed P (Y_i=1|X) differs from one data point X_i to another, though they are independent given design matrix X and shared parameters 'β'.

Model Fitting:
Logistic regression is an important machine learning algorithm. The goal is to model the probability of a random variable 'Y' being 0 or 1 given experimental data. Consider a generalized linear model function parameterized by Θ,

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

If we attempt to model the probability 'y' is 0 or 1 with the function

$$Pr(y|x;\theta) = h_\theta(x)^y(1-h_\theta(x))^{(1-y)},$$

we take our likelihood function assuming that all the samples are independent,

$$\begin{aligned} L(\theta|x) &= Pr(Y|X;\theta) \\ &= \prod_i Pr(y_i|x_i;\theta) \\ &= \prod_i h_\theta(x_i)^{y_i}(1-h_\theta(x_i))^{(1-y_i)} \end{aligned}$$

Typically, the log likelihood is maximized with a normalizing factor N^{-1}

$$N^{-1}\log L(\theta|x) = N^{-1}\sum_{i=0}^{N}\log Pr(y_i|x_i;\theta)$$

which is maximized using something like gradient descent.

Assuming the (x,y) pairs are drawn uniformly from the underlying distribution, then in the limit of large N,

$$\begin{aligned} &\lim_{N\to+\infty} N^{-1}\sum_{i=0}^{N}\log Pr(y_i|x_i;\theta) \\ &= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} Pr(X=x,Y=y)\log Pr(Y=y|X=x;\theta) \\ &= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} Pr(X=x,Y=y)\left(-\log\frac{Pr(Y=y|X=x)}{Pr(Y=y|X=x;\theta)}+\log Pr(Y=y|X=x)\right) \\ &= -D_{KL}(Y\|Y_\theta) - H(Y|X) \end{aligned}$$

where H (X|Y) is the conditional entropy and D_{KL} is the Kullback-Leibler divergence. This leads to the intuition that by maximizing the log-likelihood of a model, you are minimizing the KL divergence of your model from the maximal entropy distribution. Intuitively searching for the model that makes the least number of assumptions in its parameters.

II. Skill Scores

To measure the performance of a gap acceptance prediction algorithm, it is necessary to compute categorical statistics and scalar skill scores according to a "confusion matrix" or "contingency matrix"

Contingency Matrix
The contingency matrix is a two-dimensional "square" table that displays the discrete joint distribution of predictions and observations in regard to frequencies or relative frequencies. For dichotomous categorical predictions having only two possible outcomes (yes or no), a 2 × 2 contingency matrix can be defined. In Table 1, h is the number of "yes" (accepted gap) predictions that matches with the actual "yes" (accepted gap) observations, or the number of hits; f is the number of "yes" (accepted gap) predictions when the observation is "no" (rejected gap), or the number of false alarms; m is the number of "no" (rejected gap) predictions when observations are "yes" (accepted gap), or the number of misses; and z is the number of "no" (rejected gap) predictions that matches with the observed "no" (rejected gap), or the number of correct negatives.

Contingency Matrix

	Observed		
Prediction	No	Yes	Row Total
No	z	m	z + m
Yes	f	h	f + h
Total	z + f	m + h	

z = correct negatives;
m = misses;
f = false alarms;
h = correct positives;

Bias
The bias is the ratio of the number of accepted gaps predicted by the SVM to the total number of observed accepted gaps. The bias value indicates whether the SVM underestimates (when bias is less than 1) or overestimates (when bias is greater than 1) the number of accepted gaps. A bias is calculated with

$$\text{Bias} = \frac{f+h}{m+h}$$

Precision
Precision is the ratio of the number of accepted gaps correctly predicted by SVM to the total number of predicted accepted gaps. The precision value gives the percent of selected gap events that are correct. The formula used for calculating the precision is

$$\text{Precision} = \frac{h}{h+f}$$

Probability of Detection
POD, also known as recall, is the ratio of the number of accepted gaps correctly predicted by the SVM to the total number of observed accepted gaps (see Equation 6). The POD gives the fraction of observed gap events that are correctly forecast. The

value of POD ranges from 0 to 1, and POD = 1 indicates that the SVM correctly detects all accepted gaps.

$$\text{POD or Recall} = \frac{h}{h+m}$$

F-Measure

The F-measure is a weighted harmonic mean of the precision and recall (see Equation 7). An F-measure close to 1 indicates the best score; an F-measure close to 0 indicates the worst score.

$$\text{F-measure} = 2*\frac{Precision*Recall}{Precision+Recall}$$

False Alarm Ratio

The FAR is the ratio of the number of incorrect gaps predicted by the SVM to the total number of gaps predicted by the SVM. The FAR value indicates the fraction that the SVM detects accepted gaps that were not detected in the observed accepted gap data. The FAR has a best score of 0 with a range of 0 to 1.

$$\text{FAR} = \frac{f}{f+h}$$

Heidke's Skill Score

The HSS is popularly used in forecasting since all elements from the contingency matrix are considered. Perfect prediction receives an HSS of 1, a prediction equivalent to the reference prediction receives a score of zero, and the predictions worse than the reference prediction receive negative scores (28).

$$\text{HSS} = 2*\frac{zh-fm}{(z+f)(f+h)+(m+h)(z+m)}$$

III. Data Modelling

Data were collected at one three-legged signalized intersection and was collected during the day and night. Different gap acceptance behavior by drivers were observed during the day and the night; hence, day- and nighttime data were analyzed differently.

Site Details	No. of Obs.	Training Data (75%)	Testing Data (25%)
3-legged Int.(Day)	1469	1102	367
3-legged Int.(Night)	1103	827	276

Using R-Studio and its library 'Carat', 3 models were prepared, the classification efficiencies of which were compared in terms of Skill Scores and Accuracy. For each of these models, 10-fold, 3 times repeated cross-validation step was used. The SVM model was tuned, and achieved its maximum accuracy at C = 0.01. The ANN (1 hidden layer) model was tuned to 6 neurons in the hidden layer, along with a decay of 0.05.

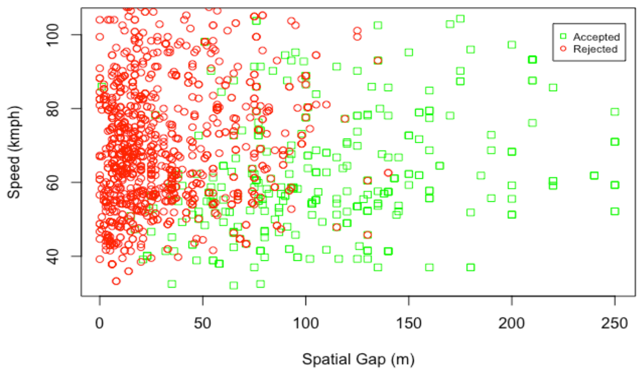

Fig 5. Day time data at 3-Legged Int.

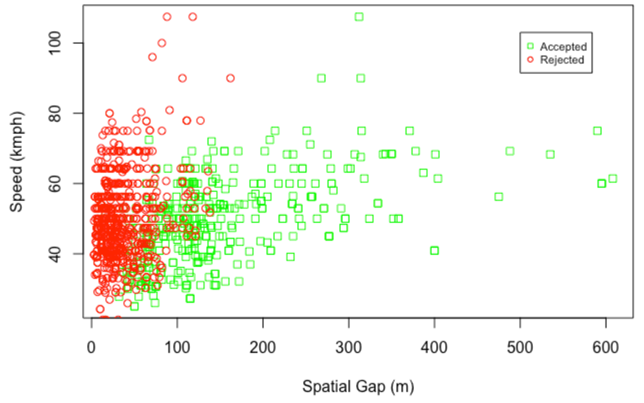

Fig 6. Night time data at 3-Legged Int.

III. Prediction Results

A. Support Vector Machine

3-Legged Intersection (Day)

Prediction	Observed			
	Rejected	Accepted	Row Total	Observed (%)
Rejected	56	2	58	15.80
Accepted	22	287	309	84.20
Total	78	289	367	100
Predicted (%)	21.25	78.75	100	

3-Legged Intersection (Day)

Prediction	Observed			
	Rejected	Accepted	Row Total	Observed (%)
Rejected	81	11	92	33.33
Accepted	15	169	184	66.67
Total	96	180	276	100
Predicted (%)	34.78	65.22	100	

B. Artificial Neural Networks

3-Legged Intersection (Day)

Prediction	Observed			
	Rejected	Accepted	Row Total	Observed (%)
Rejected	59	4	63	17.16
Accepted	19	285	304	82.84
Total	78	289	367	100
Predicted (%)	21.25	78.75	100	

3-Legged Intersection (Night)

Prediction	Observed			
	Rejected	Accepted	Row Total	Observed (%)
Rejected	87	13	100	36.23
Accepted	9	167	176	63.77
Total	96	180	276	100
Predicted (%)	34.78	65.22	100	

C. Binary Logit Model

3-Legged Intersection (Day)

Prediction	Observed			
	Rejected	Accepted	Row Total	Observed (%)
Rejected	50	6	56	15.26
Accepted	23	288	311	84.74
Total	73	294	367	100
Predicted (%)	19.89	80.11	100	

3-Legged Intersection (Night)

Prediction	Observed			
	Rejected	Accepted	Row Total	Observed (%)
Rejected	84	8	92	33.33
Accepted	23	161	184	66.67
Total	107	169	276	100
Predicted (%)	38.76	61.23	100	

IV. Comparison of SVM, ANN & BLM

On the basis of the observed and predicted gap acceptance decision by SVM, ANN and BLM, it is clearly evident that all models perform reasonably well. Another observation from the prediction table is that the ANN appears best, while SVM predicts the gap acceptance values slightly less accurately than those predicted by the ANN; whereas the BLM seems to be the least accurate. Moreover, the results by these methods differ by a marginal amount only. The final table shows the skill scores and accuracy for SVM, ANN and BLM. The bias scores for SVM and ANN are quite close to each other, which indicates that the SVM and ANN both are reasonably well able to predict gap acceptance and rejection. The recall values show that the ANN and SVM have a higher score compared with the BLMs, but the SVM has a higher FAR than the BLM whereas ANN has a lower FAR, which indicates that the SVM predicts more "accepted" events that are observed to be rejected events whereas ANN is better for both acceptance and rejection of gaps. The difference between the skill scores for the methods is not remarkable, which can be considered as a good indication that the SVM and ANN perform quite well.

Skill Scores & Accuracy

	3-Legged Int. (DAY)			3-Legged Int. (NIGHT)		
	BLM	SVM	ANN	BLM	SVM	ANN
Bias	1.058	1.065	1.052	1.089	1.022	0.978
Precision	0.926	0.929	0.938	0.875	0.918	0.949
Recall	0.980	0.993	0.986	0.953	0.939	0.928
FAR	0.074	0.071	0.063	0.125	0.082	0.051
F-measure	0.952	0.960	0.961	0.912	0.929	0.938
HSS	0.728	0.784	0.799	0.757	0.790	0.826
Accuracy (%)	92.10	93.46	93.73	88.77	90.58	92.03

V. Conclusion

This study uses SVM and ANN to classify and predict gap acceptance and rejection for uncontrolled intersections and compares the two with a widely used BLM. Two types of data sets from different times of the day were obtained and used to develop the models. The prediction success tables and skill scores, such as bias, precision, recall, F-measure, FAR, and HSS, are used for performance evaluation. Study results are encouraging; SVM and ANN techniques outperform the BLM in most of the skill scores as well as accuracy.

The prediction of gap acceptance at uncontrolled road sections can be important in developing real-time applications such as the advance safety warning system and advanced traffic management systems. These systems will help drivers and pedestrians to make an appropriate choice of action during crossing at intersections and midblock crossings. Future studies should apply SVM and ANN techniques to data from different cities and check the applicability of the models developed. The study can also be extended by adding a third dimension to the input data, such as vehicle type and transverse position of a conflicting vehicle.

VI. Bibliography

[1] “Classification of Gaps at Uncontrolled Intersections and Midblock Crossings Using Support Vector Machines” by Digvijay S. Pawar, Gopal R. Patil, Anita Chandrasekharan, and Shruti Upadhyaya

[2] “Machine Learning Course” by Andrew Ng on Coursera

[3] Tutorials on RStudio

[4] Wikipedia