Iowa State University

Crash Identification Based on Waze Reports

Exploring Shallow and Deep Models

5-4-2017

Table of Contents

1. Introduction

Traffic crash identification is one of the hottest research fields in transportation safety. Traffic engineers aim at analyzing reasons for crashes and figuring out effective ways to predict and prevent possible crashes in the future. Iowa Department of Transportation (DOT) and Iowa State University are taking joint efforts to develop a data-driven methodology and framework for crash identification on freeways of Iowa, in order to improve road safety and decrease the probability that crashes happen. This project mainly focuses on the crash identification of Interstate 235 (both directions) in Des Moines, IA. This freeway segment is a 13.78-mile long corridor that is frequently used in the Des Moines metropolitan area.

The word "big-data era" has been gaining increasing popularity in recent years. The huge amount of data is transforming the way of conducting research, especially in transportation science and engineering. The data used in this project come from different sources—Waze traffic issue reports, Wavetronix traffic condition data, Traffic Management Centers accident reports, and Road Weather Information System data. These data spread across 4 months from September 2016 to December 2016. The data description and data processing can be found in Section 2.

This project applies machine learning algorithms, including 5 shallow models (random forests, decision trees, boosted decision trees, logistic regression, and linear support vector machine) and 2 deep models (fully connected neural networks and convolutional neural networks) to tentatively recognize and identify the traffic crashes on I-235 based on Waze traffic issue reports. Different learning models are compared and the results show that the random forest model is the best performance shallow model and CNN is better than FC neural networks.

Machine learning algorithms have wide applications in transportation engineering, especially in traffic safety and accident prediction and prevention. Nassiri et al. (2014) used Negative Binomial Regression, Zero Inflated Negative Binomial Regression, Support Vector Machine and Back-Propagation Neural Network to predict accident frequencies on highways based on traffic and roadway characteristics, including Volume to Capacity ratio (V/C), Vehicle Kilometers Travelled (VKT) and roadway width. Pereira et al. (2013) built several models, such as linear regression, support vector regression, artificial neural networks, decision trees and radial basis functions, to predict traffic incident duration (the period between incident reporting and road clearance) using 2 years of accident cases, so that drivers can be better informed of possible road blockage time. Koesdwiady et al. (2016) took weather parameters into account when they predicted traffic conditions at San Francisco Bay Area. The method they proposed incorporated deep belief networks for traffic flow prediction with weather data. In short, these papers focus on using machine learning models to recognize traffic incident features and make predictions. Other research focuses on improving traffic safety from the perspective of drivers. Yan et al. (2015) applied convolutional neural networks to recognize and detect driver fatigue and inattention from the face movement images. By processing and classifying face images with CNN, a warning signal is then sent to the driver if he/she does not focus on driving.

This project has two main contributions. First, the accidents' features in terms of traffic and weather conditions on I-235 have been learnt by machine learning models. With these models, Iowa DOT can predict a possible accident based on traffic data and weather data, and therefore remind drivers to drive cautiously. Second, this work can help Waze evaluate whether its users are correctly reporting traffic issues and score the users based on real traffic and weather conditions.

2. Data description and processing

2.1. Waze reports data

2.1.1. The Waze app

Waze is a GPS-based navigation app working on smartphones that help drivers plan travel routes and estimate travel time. Its user interface is seen in Figure 2. Different from other navigation software such as Google Maps and Apple Maps, Waze is community-driven, gathering complementary map data and traffic information from its users (Waze, 2017). Users can report traffic (moderate, heavy, or standstill), police, crash (minor, major, or other side), hazard (on-road, shoulder, or weather), gas prices, traffic cameras, road closure, etc. (seen in Figure 1). This project focuses on crash, hazard, and traffic reports. The reporting function let other users be informed of nearby events and make navigation interacted, social and interesting.

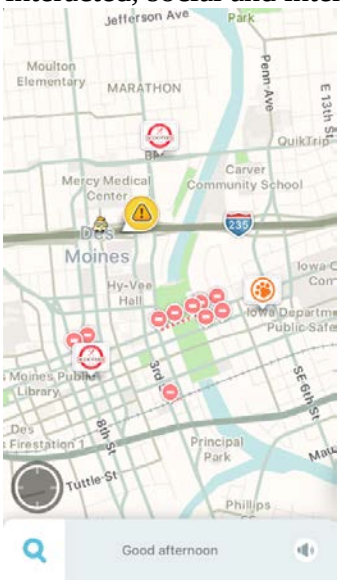
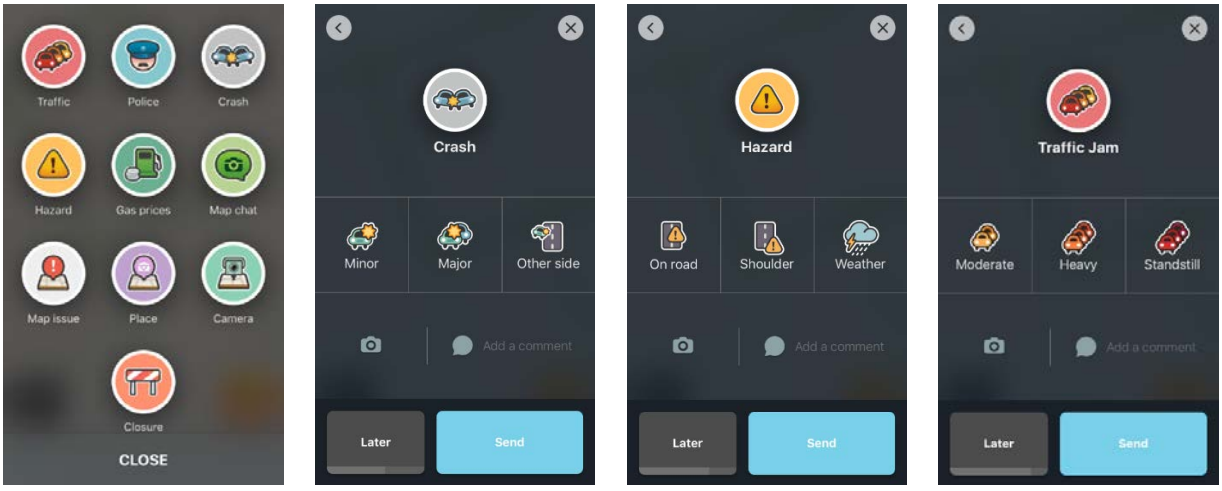


Figure 1. User interface of Waze app.



(a) (b) (c) (d)

Figure 2. Waze report user interface.

2.1.2. The reports data

For each incident report, Waze records the start timestamp, GPS coordinates of incident location, corresponding road segment, and event type. A sample data is given in Table 1. The Waze report data have been used in many research fields of transportation, such as planning of urban public transportation networks (Baloian et al. 2015), traffic jam prediction (Heiskala et al., 2016), etc.. During the 4 months from September 2016 to December 2016, 2840 incidents in total reported by Waze are available, among which 254 are accidents, 1,233 are jam, and 1,353 are hazard (seen in Figure 3).

Table 1. A sample data of Waze incident reports

ID	Start Timestamp	Latitude	Longitude	Road	Direction	Event Type
d3985182	9/14/2016 4:47:00 PM	41.595106	-93.610275	I235	E	JAM_HEAVY_TRAFFIC
75ae6e9a	9/14/2016 4:53:00 PM	41.595691	-93.62608	I235	E	ACCIDENT_MAJOR
184274cf	9/19/2016 5:27:00 PM	41.595013	-93.648489	I235	E	ACCIDENT_MAJOR
d881f69d	12/7/2016 8:40:00 AM	41.593247	-93.713616	I235	E	HAZARD_ON_SHOULDER_CAR_STOPPED

Figure 3. Number of available Waze reports by type.

2.2. Traffic Management Centers reports

Traffic Management Centers (TMCs) of Iowa DOT are responsible for monitoring traffic signals, traffic flow and special events for Iowa's major streets and highway networks. They deploy traffic management strategies to reduce congestion and coordinated state and local authorities during special events (TAMU, 2017). Operators in the TMC monitoring room watch real-time traffic conditions sent from a closed circuit television (CCTV) system. TMCs may not always be the first detector of crashes and traffic jams, but they are always an important and accurate information source. Table 2 gives a sample data of TMC reports, which include incident ID, start timestamp, GPS coordinates of incident location, corresponding road segment, and event type. During the 4-month study period, there are 239 verified crash reports available.

Table 2. A sample data of TMC reports.

ID	Start Timestamp	Latitude	Longitude	Road	Direction	Event Type
49530	1/4/2016 7:31:14 AM	41.592605	-93.742094	I235	E	JAM
49644	1/5/2016 5:32:58 PM	41.592764	-93.696594	I235	E	3+ VEHICLE COLLISION

49725	1/7/2016 5:39:19 PM	41.592657	-93.693913	I235	W	2 VEHICLE COLLISION
50037	1/13/2016 8:03:45 AM	41.59268	-93.694736	I235	W	1 VEHICLE COLLISION

2.3. Wavetronix traffic condition data

2.3.1. Wavetronix sensors

Iowa DOT installed traffic sensors made by Wavetronix along Iowa freeways. Figure 4 displays the locations of the traffic sensors that are collecting data on Interstate 235. There are 15 sensors on east bound and 14 on west bound. These traffic sensors collect real-time traffic information including traffic volume (veh/h), traffic speed (mi/h), and sensor occupancy (%), at an interval of 20 seconds (Wavetronix, 2017). The three data fields are the basic traffic characteristics of a freeway segment. Traffic volume is defined as the number of vehicles passing a point on a highway during a specific time interval (20 seconds in this project). Traffic speed is defined as the average speed of all vehicles passing a point on a highway over some specified time period (20 seconds). Occupancy is defined as the proportion of time that a detector is occupied by a vehicle in a defined time period (20 seconds). This project uses 1-minute traffic volume, speed and occupancy by aggregating the 20-second interval raw data. A sample raw data is given in Table 3.

Figure 4. Locations of Wavetronix sensors along Interstate 235.

Table 3. A sample data of Wavetronix traffic conditions.

Detector	Timestamp	Volume	Speed	Occupancy
I-235 at 42nd STREET EB-WB	1/1/2017 7:21:00 AM	2	65	6.66
I-235 at 42nd STREET EB-WB	1/1/2017 7:21:20 AM	3	65	4
I-235 at 42nd STREET EB-WB	1/1/2017 7:21:40 AM	1	64	7.33
I-235 at 42nd STREET EB-WB	1/1/2017 7:22:00 AM	0	67	5
I-235 at 42nd STREET EB-WB	1/1/2017 7:22:20 AM	5	64	4.33
I-235 at 42nd STREET EB-WB	1/1/2017 7:22:40 AM	2	66	4.66
I-235 at 42nd STREET EB-WB	1/1/2017 7:23:00 AM	3	69	5.66

2.3.2. Data interpolation

In 1-minute traffic data, there are still some missing values due to sensor malfunction or no vehicle presenting at night. Figure 5 shows a raw 1-minute speed heat map by using sensor's mile marker as y-axis and time of day as x-axis. One day example is plotted.

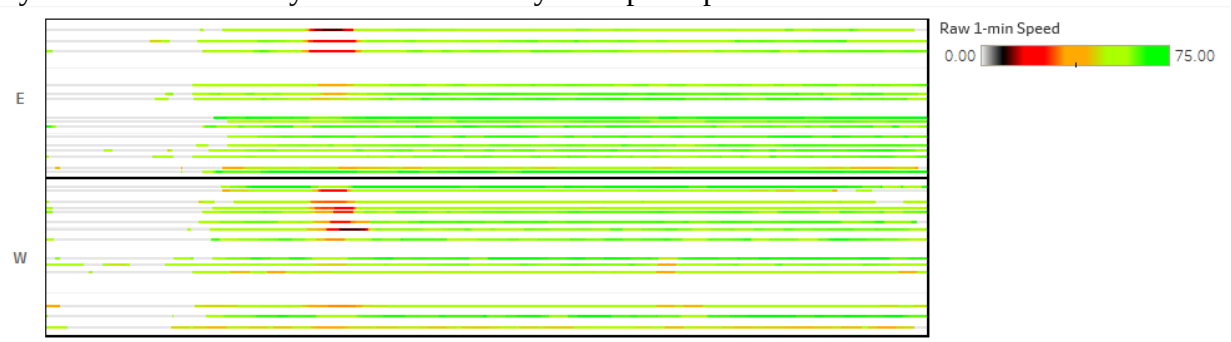


Figure 5. Raw 1-min Speed Heat Map

As we can see, in Figure 5, the left part (night time) has lots of grey blocks, which means there are not many vehicles in that period. To fill the empty value for all timestamps, we linearly interpolated the traffic data (here we use speed data as an example) by every 1 minute. The temporal-interpolated speed are shown in Figure 6.

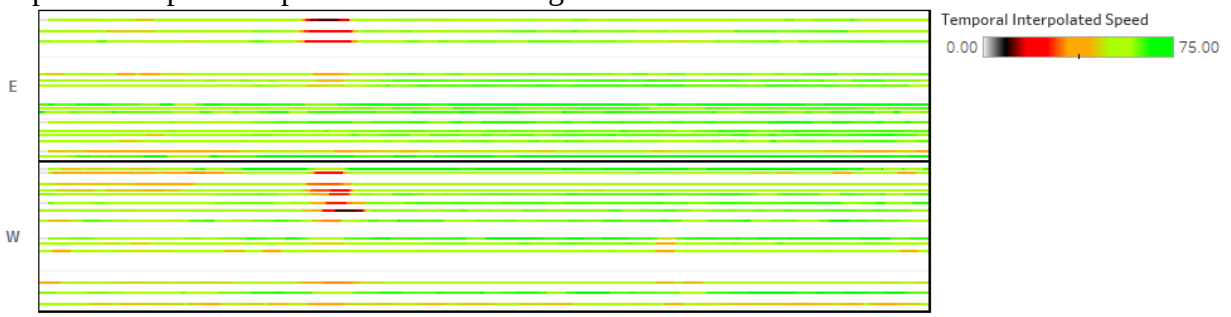


Figure 6. Temporal Interpolated Speed Heat Map

After temporal interpolation, we are still lack of data for any locations between sensors. Thus, a spatial interpolation is also needed. Since we have the mile markers for each sensors, which means we know the traveling distance between each sensor pairs, thus, the spatial interpolation is implemented based on the distance between sensors. Here we assume the changing is gradual in terms of distance. The final spatial and temporal interpolated data are illustrated in Figure 7.

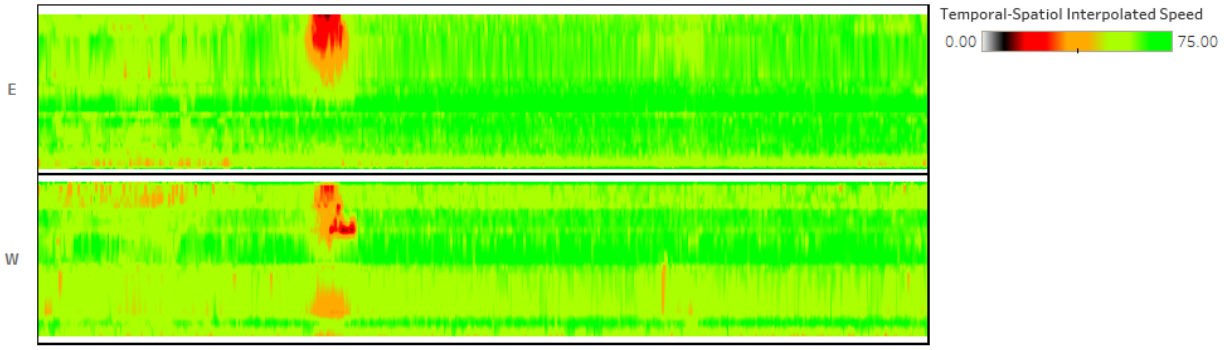


Figure 7. Spatial and Temporal Interpolated Speed Heat Map

2.4. Road Weather Information System data

The weather data of Interstate 235 come from the Road Weather Information System (RWIS) of Iowa Environmental Mesonet (IEM, 2017). RWIS collects road temperatures and pavement conditions of Iowa highways. There is a weather information detector locating in the middle of I-235 (see Figure 8.). The data-collecting area covers the entire road segment, so the weather conditions for the entire I-235 at a certain time point are the same. The data fields collected are listed in Table 4.

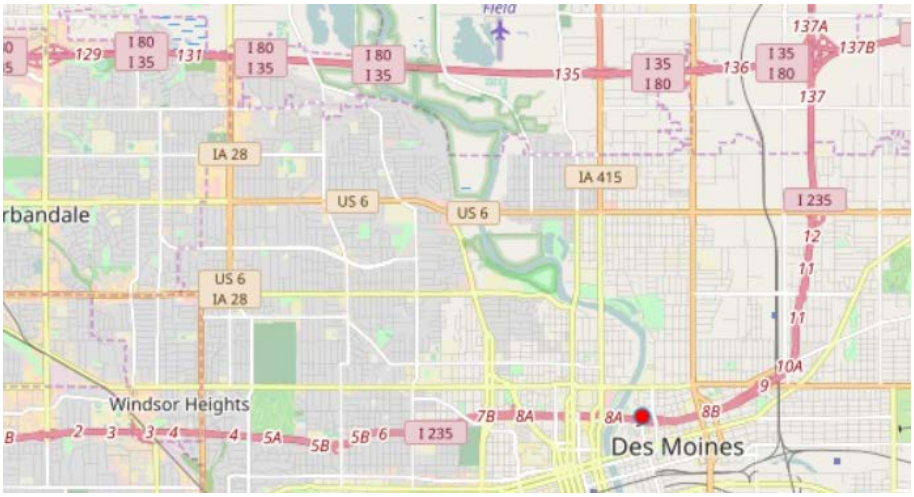


Figure 8. Location of weather information detector on Interstate 235.

Table 4. RWIS weather data fields

station	Name of weather observation station
obtime	Timestamp when weather data were collected. Timezone is America/Chicago CST/CDT
longitude	Longitude of weather observation station
latitude	Latitude of weather observation station
tmpf	Air temperature [F]
dwpf	Dew point temperature [F]
drct	Wind direction [degree N]
gust	Wind gust [knots]
tfs0	Pavement sensor 0 temperature [F]
tfs0_text	Pavement sensor 0 Condition
subf	Subsurface temperature [F]

There are 6 pavement conditions—dry, chemically wet, trace moisture, wet, ice watch, and ice warning. In majority of cases the road condition is dry (see Figure 9), while there is very little rain and ice. Therefore the road condition is classified as dry (label 0) and not-dry (label 1).

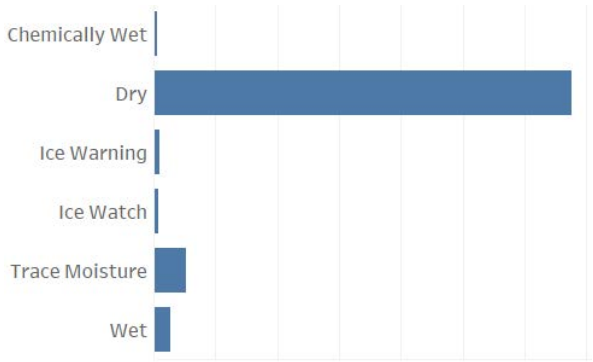


Figure 9. Pavement conditions.

2.5. Create labels

Waze reports are not 100% accurate because there might be lags between the real beginning of an accident and the reporting on the app and users might mistakenly report an accident that does not even happen. Therefore we use TMC crash report as the ground truth and make comparison with Waze incident reports. Figure 10 shows how to use the start and end time of crash in TMC to relabel all the Waze incidents.

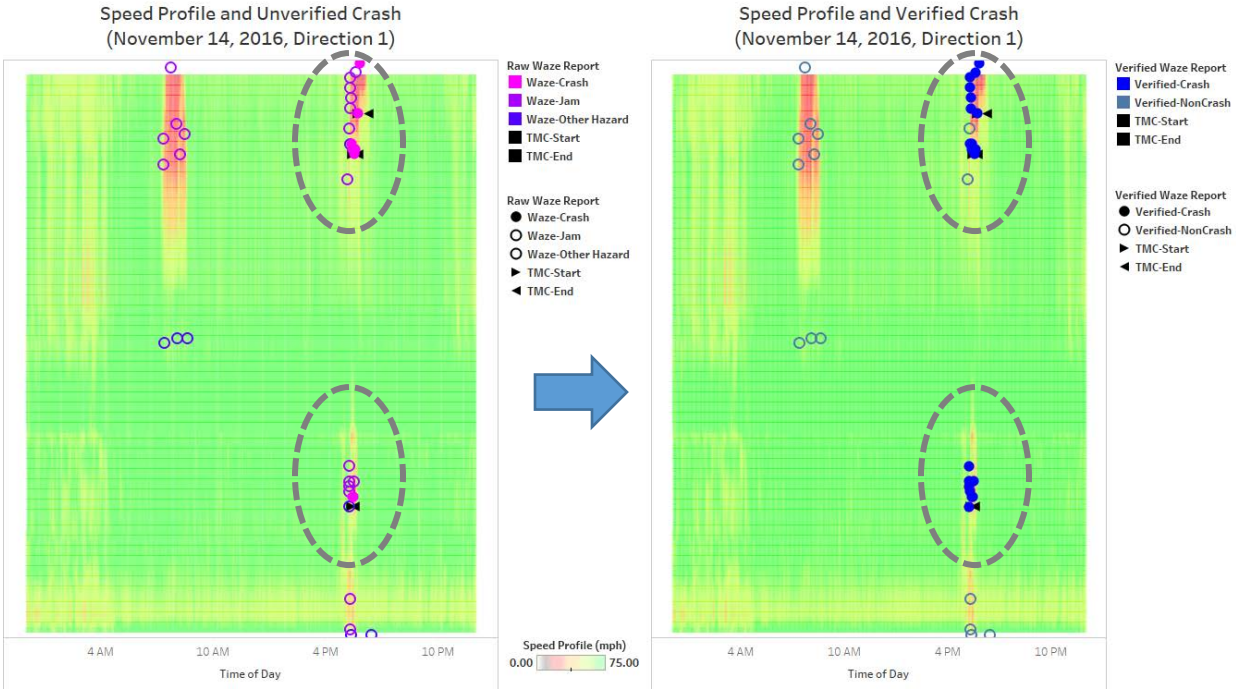


Figure 10. Speed profile and verified crash.

The left half of Figure 10 is the unverified Waze incidents. They are colored by incident type and the crashes reported by Waze have been emphasized by using filled dots. However, in some areas (circled by dashed line), when a TMC crash presents, not all the Waze incidents are reported as crash. The reason could be that the users did not see the crash happen but experienced the jam caused by the crash. Thus, we decide to associate all the Waze incidents with their corresponding TMC crash within a certain spatial and temporal threshold.

The threshold is determined to be: a) 10 minutes before TMC crash start time and 5 minutes after TMC crash end time; b) 2.5 miles upstream and 0.5 miles downstream from TMC crash location. Thus, as shown in the right half of Figure 10, those non-crash Waze incidents that are very close to a TMC crash have been labeled as crash. And those Waze incidents far away from a TMC crash have been labeled as non-crash.

After verifying all the Waze reports, a summary of verification rate is shown in Figure 11 and Table 5. In Waze crashes, there are 61.8% were verified as true crash in TMC. And only 19.4% of jam, 4.9% of hazard were verified as crash in TMC.

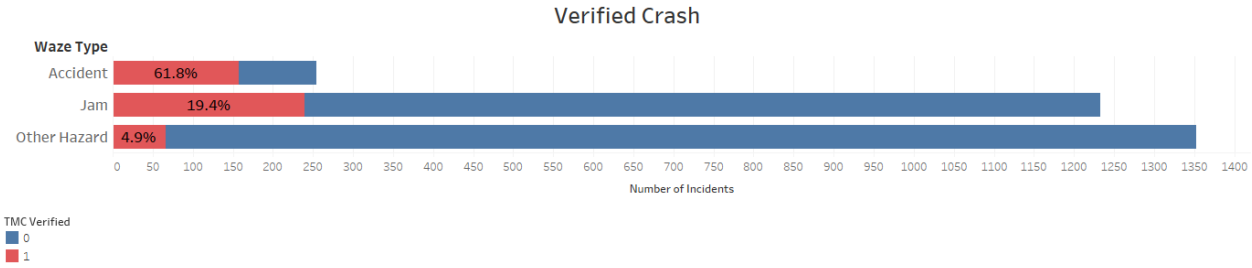


Figure 11. Percentage of Waze reports confirmed as crash by TMC reports.

Table 5. Number of Waze reports confirmed as crash by TMC reports.

Waze Type	Accident	Jam	Hazard	Total
Verified as Crash	157	239	66	462
Verified as Non-crash	97	994	1287	2378
Total	254	1233	1353	2840

2.6. Create samples

We transformed the raw data into 9-channel full mesh-grids with one pixel represents 0.1 miles in distance and 1 minute in time. Since we want to identify the traffic and weather characteristics before an accident happens, we set the area of interest as a large window from 0.6 miles upstream to 1.6 miles downstream in distance and 5 minutes back in time of one target incident (see Figure 12). In the area of interest, we used small window to generate samples. The window was set as 5-minute (5 pixels) wide and 1-mile (10 pixels) high (see Figure 12). For each area of interest, we selected maximum 42 windows using temporal stride 1 minute and spatial stride 0.2 miles, and each window had 9 features (so the sample shape is 10×5×9). The 9 features are volume, speed, sensor occupancy, air temperature, dew point temperature, pavement temperature, subsurface temperature, and pavement condition.

In summary, 462 out of 2,840 Waze reports are labeled as *Crash*. In deep model part, we reserved 1 day data for test to see if model could give fair advice when Waze are wrongly reported. The remaining were all used for modelling. Thus, a total of 83,803 samples were generated, and we down-sampled the non-crashes to balance the data. Finally we had 28,752 crashes vs. 28,752 non-crashes. Further, we used 70% of the data for training, and 30% for validation.

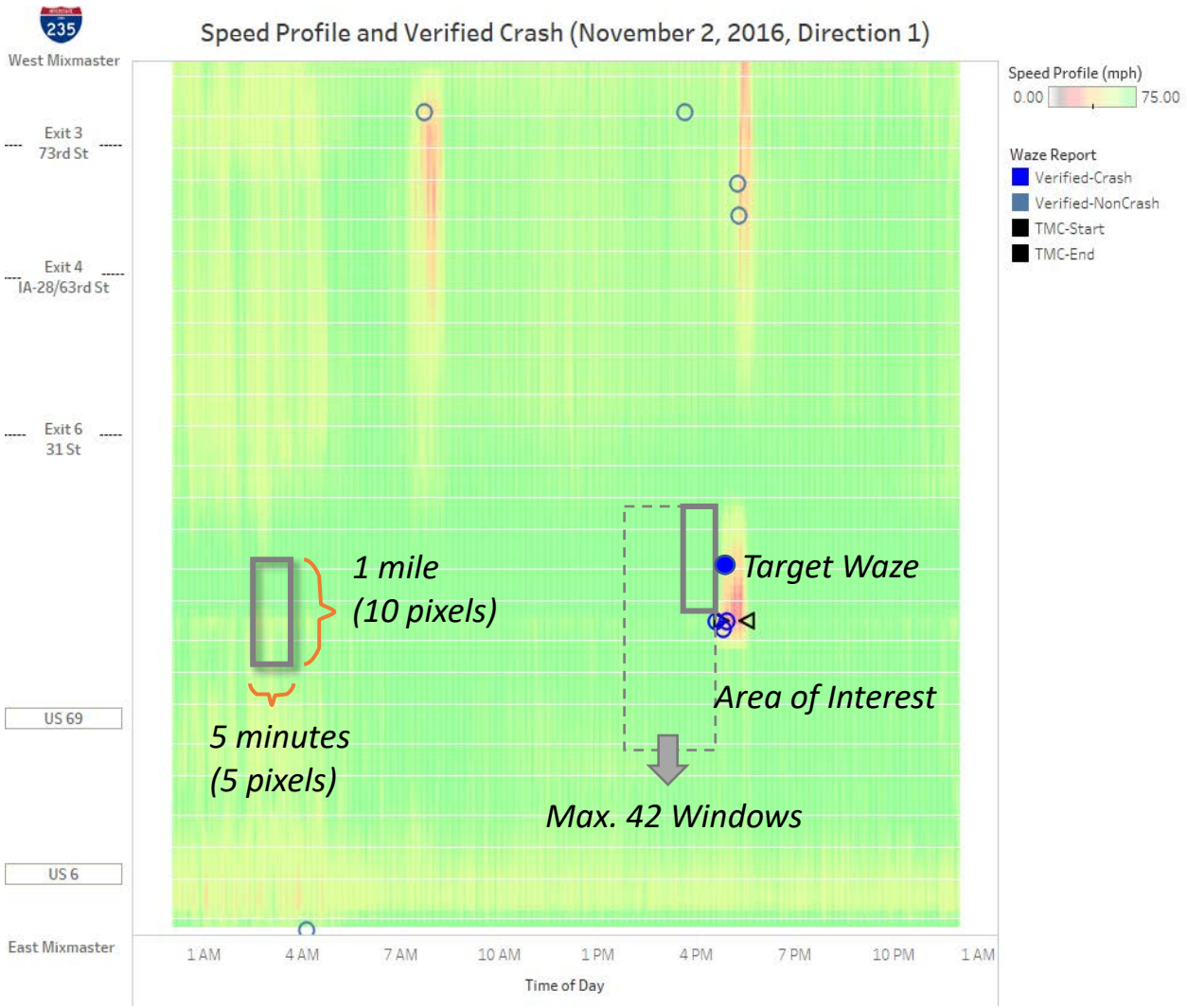


Figure 12. Sample generation.

3. Model results

This project tentatively uses 5 different shallow models (random forests, decision tree, boosted decision tree, logistic regression, and linear support vector machine) and 2 deep models (ANN and CNN) to identify the crashes on Interstate 235.

3.1. Shallow models

Several traditional shallow models were fitted with variation of model hyper-parameters.

3.1.1. Single Decision Tree

Decision tree algorithm was evaluated first, where several experiments were conducted with respect to leaf aggregation level. Tree with leaf size 1 is prone to over-fit, therefore, as aggregation level goes up, the model accuracy is decreasing. Overall, the single decision tree model yields an accuracy around 85~90%, so, can we do better by using bagging method to improve the prediction power?

Figure 13. Performance of decision tree with different levels of leaf aggregation.

3.1.2. Random Forest

Random forest is a bagging method, which combines multiple random decision trees. By averaging (voting) results from multiple trees, it can reduce error and overcome over-fitting issues. Figure 13 below illustrates that, the modeling performance is getting better as the number of trees increasing. After certain point (20 trees or higher), the performance stays the same and overall, random forest performs better than a single determinist tree.

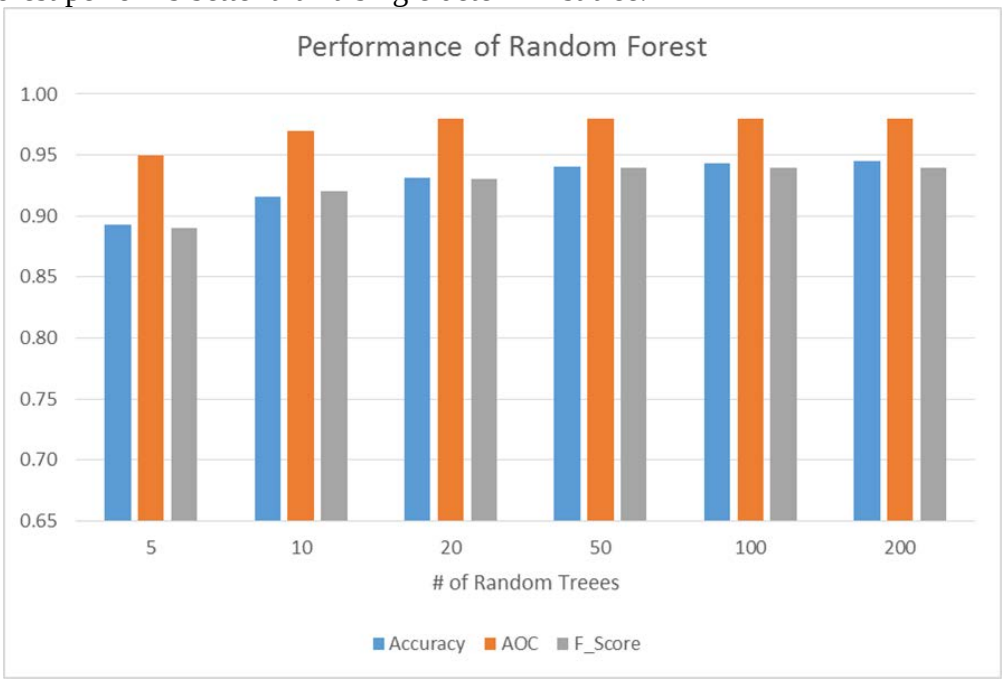


Figure 14. Performance of random forest model with different number of trees.

3.1.3. Other shallow models

Besides the trees' family, other shallow models were evaluated as well for modeling comparison, including logistic regression, linear Support Vector Machine, and AdaBoost (with decision tree). Trees' family models (forest, single tree, and boosted trees) stand out, with an accuracy of as high as 94%. However, logistic regression and linear SVM do not have satisfying performance, with accuracy of lower than 70%, so more hyper-parameters need to be tuned to increase their prediction power.

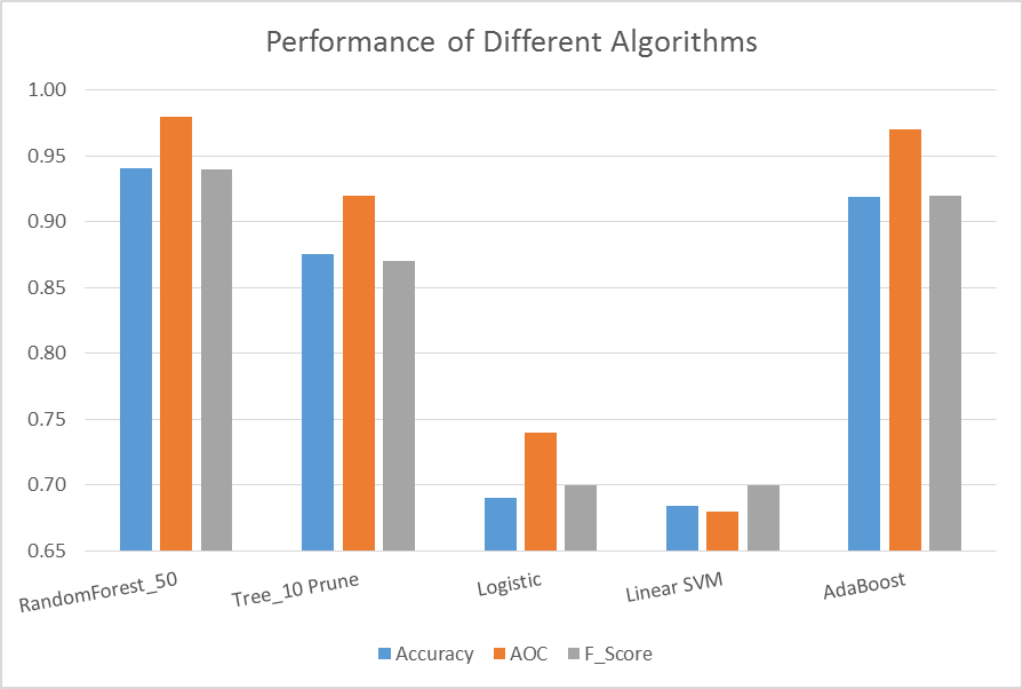


Figure 15. Performance of different shallow models.

3.2. Deep models

3.2.1. Model Structures

Two types of model structure are considered in this study: fully-connected neural network (FNN) and convolutional neural network (CNN).

FNN is the regular artificial neural network in which neurons between two adjacent layers are fully pairwise connected, but neurons within a single layer share no connections. The structure of the FNN model we have developed is shown in Figure 16. Since all layers in FNN are fully connected, thus it tends to have a large number of parameters. In our model, we used 3 hidden layers with 128, 64 and 32 nodes respectively. This results in 67,904 parameters in total.

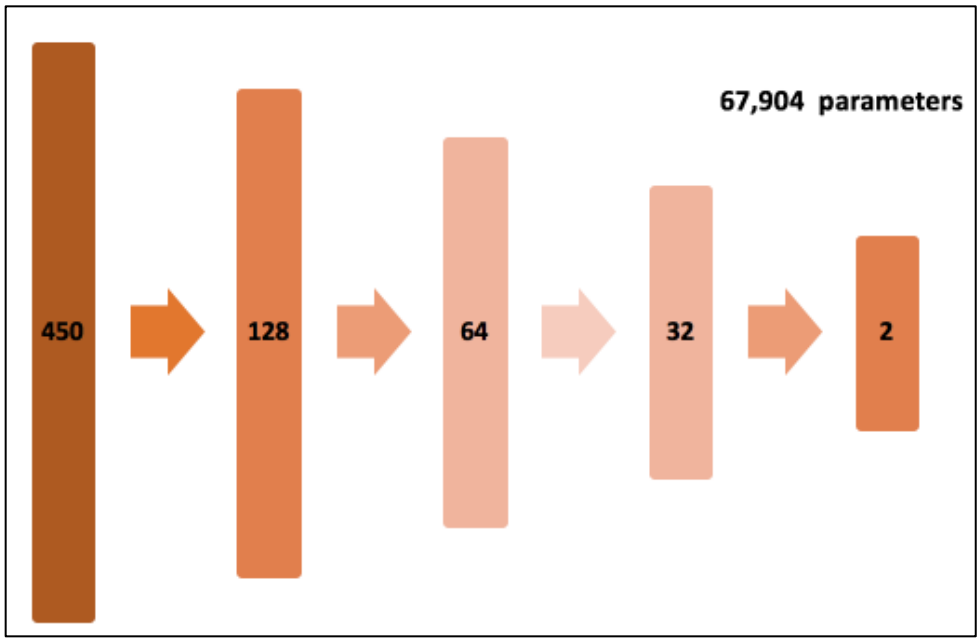


Figure 16. ANN model structure.

CNN is inspired by that the organization of animal visual cortex can extract local structural features that are shift-invariant. The certain spatial and temporal variations of traffic and weather information can be the precursor of traffic incidents and are expected to be better captured by CNN. In order to better investigate CNN, one base CNN structure and four variant CNN structures are studied. The structure of the base CNN model we have developed is shown in Figure 17. The model has three hidden layers including two convolutional layers using 3 by 3 filters and 1 fully connected layer. The base CNN contains 12,848 parameters. The other four CNN structure variants are shown in Figure 18:

a. A smaller CNN. Add one max pooling layer after the second convolutional hidden layer and decrease the number of neurons in the fully connected hidden layer to 10. The total number of parameters is 1,412.
b. A bigger CNN. Increase the number of filters in the two convolutional hidden layers to 32 and 64, respectively. The total number of parameters is 25,568.
c. A deeper CNN. Add one more fully connected hidden layer with 64 neurons. The total number of parameters is 16,944.
d. A bigger and deeper CNN. Increase the number of filters in the two convolutional hidden layers to 32 and 64, respectively, and add one more fully connected hidden layer with 64 neurons. The total number of parameters is 29,664.

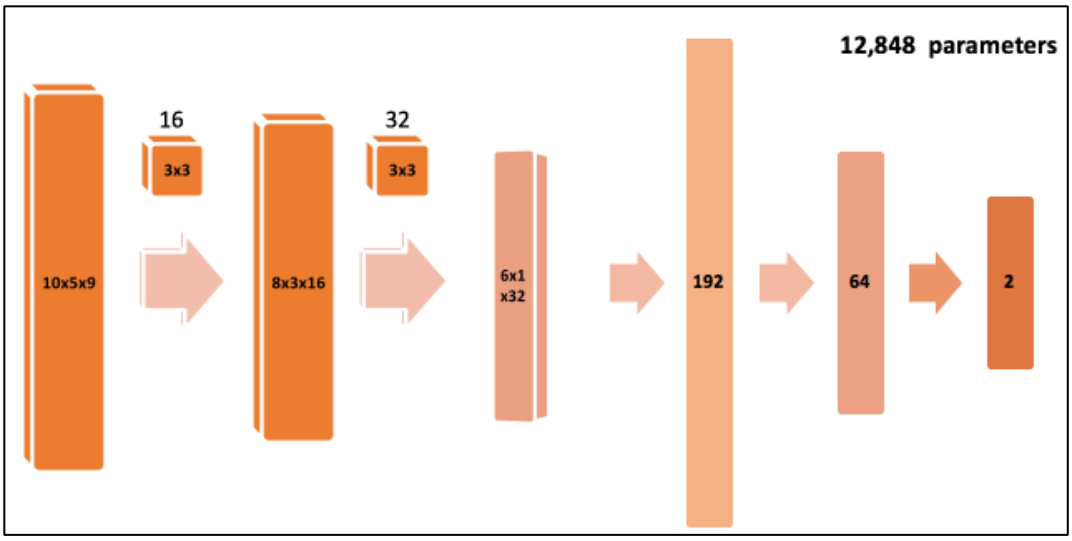
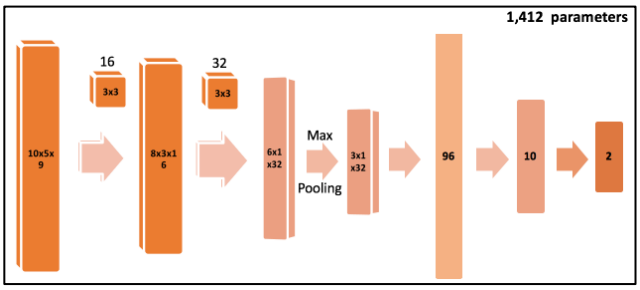
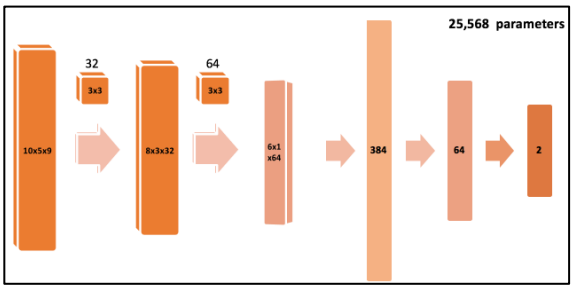


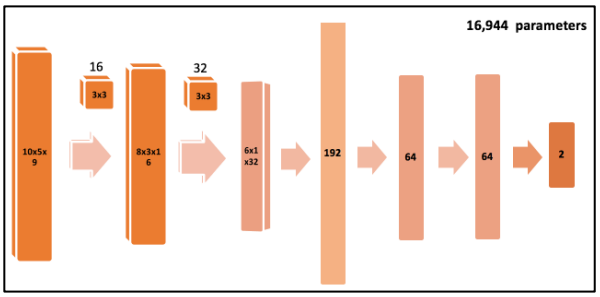
Figure 17. Model structure of the base CNN.



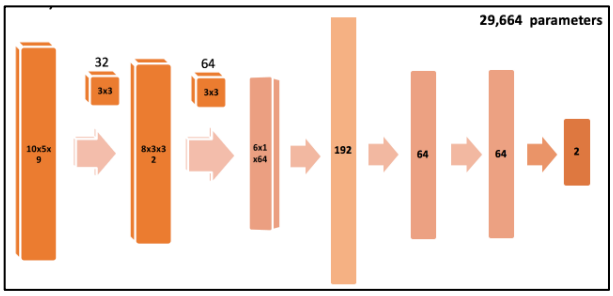
(a) Model structure of the smaller CNN.



(b) Model structure of the bigger CNN.



(c) Model structure of the deeper CNN.



(d) Model structure of the bigger & deeper CNN.

Figure 18. Other tested CNN model structures.

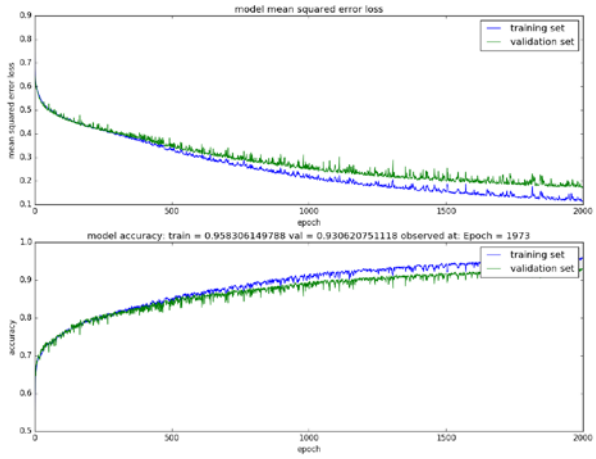
3.2.2. Model Results

In order to find the optimal model configuration for our problem, a greedy search is applied to find the best hyper-parameter setting. Model structure, activation function, batch normalization and dropout are tuned in our searching process. Every time when one hyper-parameter is under investigation, all other higher parameters are fixed. The models are evaluated based on the validation accuracy.

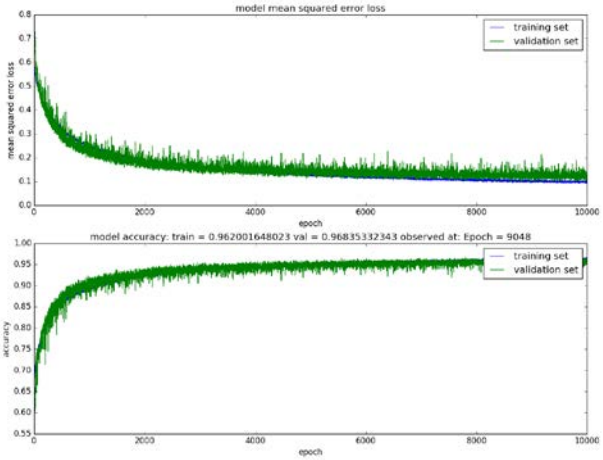
Table 6 and Figure 19 show the comparison of FNN and the base CNN under different settings of other hyper-parameters. From Table 6, there is no obvious evidence that CNN is superior to FNN or the other way around in terms of model validation accuracy. By comparing subplot (b) and (d) in Figure 19, FNN looks more stable than CNN during the training process. Considering that there is no significant accuracy difference between FNN and CNN and FNN uses significantly more parameters than CNN. CNN is selected for further study.

Table 6. Result comparison between FNN and base CNN

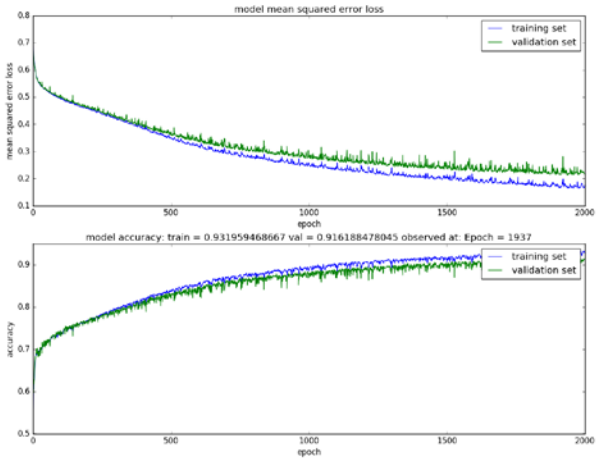
Val-Accuracy					
FNN	**CNN**	**Activation**	**BatchNormalization**	**dropout**	**Epoch**
92.4%	89.9%	ReLU	no	no	2,000
93.1%	91.6%	ELU	no	no	2,000
88.3%	90.3%	ELU	yes	all_0.5	2,000
92.1%	93.6%	ELU	yes	all_0.5	10,000
94.6%	95.4%	ELU	yes	all_0.25	2,000
96.8%	97.4%	ELU	yes	all_0.25	10,000
96.4%	95.3%	ELU	yes	no	2,000



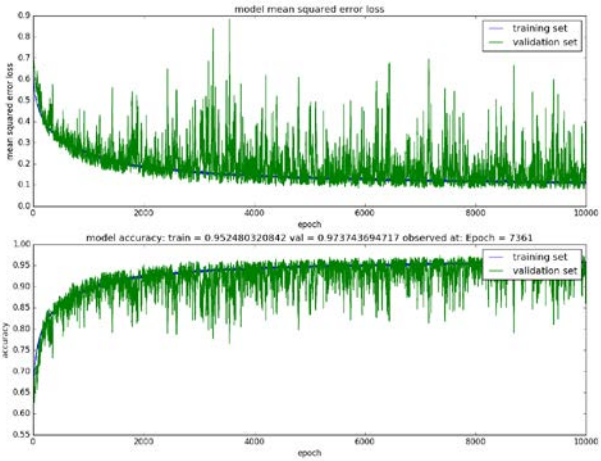
(a) FNN trained in 2,000 epochs using ELU activation, no batch normalization and no dropout.



(b) FNN trained in 10,000 epochs using ELU activation, batch normalization and dropout at all hidden layers with a rate of 0.25.



(c) The base CNN trained in 2,000 epochs using ELU activation, no batch normalization and no dropout.



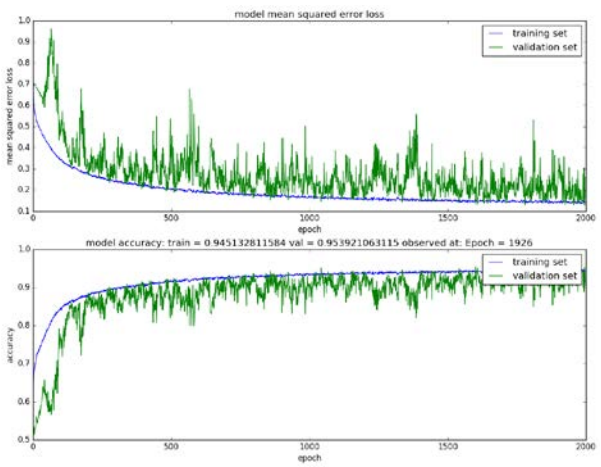
(d) The base CNN trained in 10,000 epochs using ELU activation, batch normalization and dropout at all hidden layers with a rate of 0.25.

Figure 19. Result comparison between FNN and CNN.

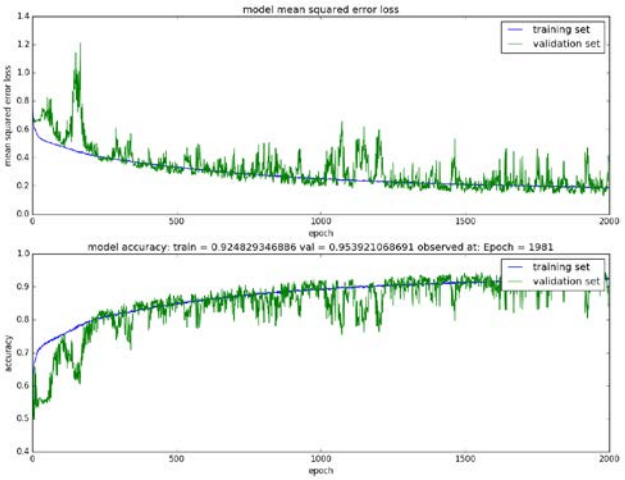
Table 7 and Figure 20 show the comparison of ReLU and the ELU as the activation function using the base CNN under different settings of other hyper-parameters. From Table 7, there is no obvious evidence that ELU is superior to ReLU or the other way around in terms of model validation accuracy. By comparing subplot (a) and (b) in Figure 20, ELU has smaller fluctuation than ReLU during the training process. In addition, the fact that ELU better captures the impact of negative activations than ReLU. ELU is selected in the further study.

Table 7. Result comparison between ReLU and ELU

Val-Accuracy					
ReLU	**ELU**	**Model**	**BatchNormalization**	**dropout**	**Epoch**
89.9%	91.6%	CNN	no	no	2,000
95.4%	95.4%	CNN	yes	all_0.25	2,000
96.1%	95.3%	CNN	yes	no	2,000
97.1%	97.4%	CNN	yes	all_0.25	10,000



(a) The base CNN trained in 2,000 epochs using ReLU activation, batch normalization and dropout at all hidden layers with a rate of 0.25.



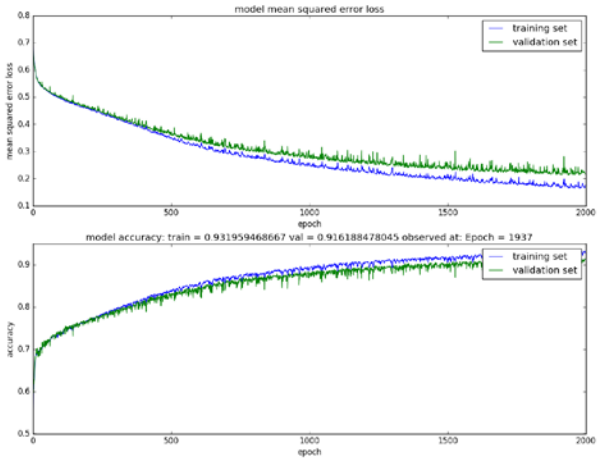
(b) The base CNN trained in 2,000 epochs using ELU activation, batch normalization and dropout at all hidden layers with a rate of 0.25.

Figure 20. Result comparison between ReLU and ELU.

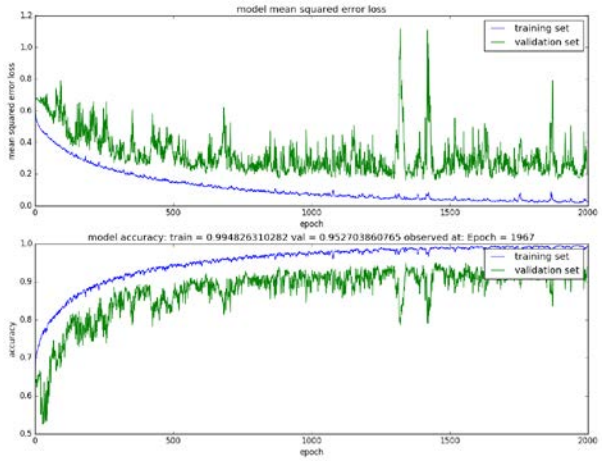
Table 8 and Figure 21 show the effect of batch normalization using the base CNN under different settings of other hyper-parameters. From Table 8, batch normalization significantly improves the model validation accuracy. From the subplot (b) in Figure 20, batch normalization helps the model to reach high training accuracy and some hints of overfitting can be observed.

Table 8. Result comparison between batch normalization and no batch normalization

Val-Accuracy					
w/o BN	**w/ BN**	**Model**	**Activation**	**dropout**	**Epoch**
91.6%	95.3%	CNN	ELU	no	2,000



(a) The base CNN trained in 2,000 epochs using ELU activation, no batch normalization and no dropout.



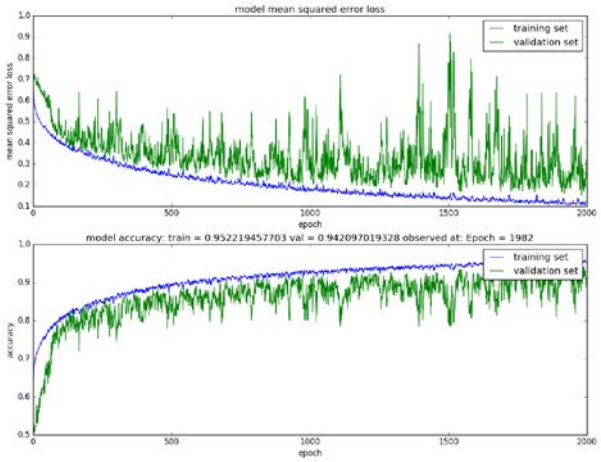
(b) The base CNN trained in 2,000 epochs using ELU activation, batch normalization and no dropout.

Figure 21. Result comparison between batch normalization and no batch normalization.

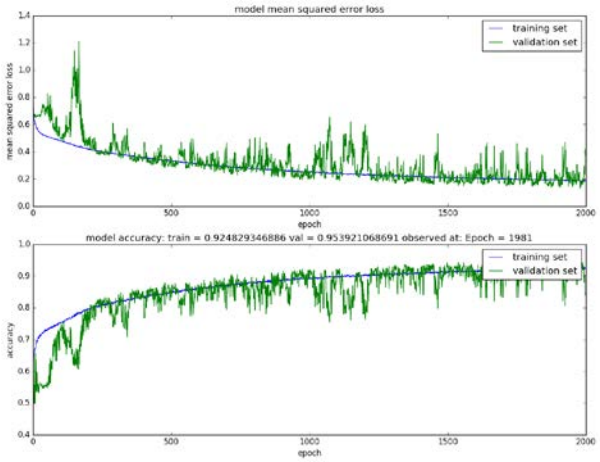
Table 9 and Figure 22 show the comparison of different dropout strategies using the base CNN under the same setting of other hyper-parameters. Combining the information from both Figure 22 and Table 9, applying dropout to all hidden layers is better than applying dropout to only the last hidden layer in terms of both model accuracy and training stability.

Table 9. Result comparison between different dropout strategies

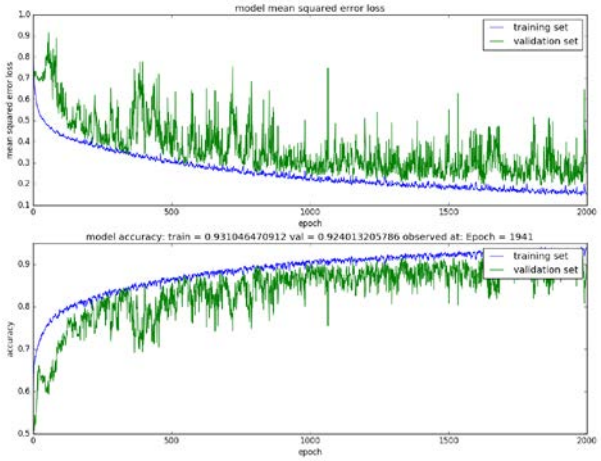
dropout	Val-Accuracy	Model	Activation	BatchNormalization	Epoch
all_0.5	93.6%	CNN	ELU	yes	2,000
all_0.25	97.4%	CNN	ELU	yes	2,000
last_0.5	92.4%	CNN	ELU	yes	2,000
last_0.25	94.2%	CNN	ELU	yes	2,000



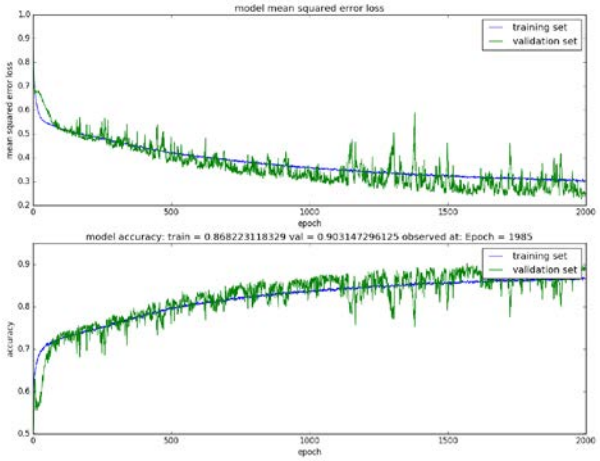
(a) The base CNN trained in 2,000 epochs using ELU activation, batch normalization and dropout at the last hidden layer with a rate of 0.25.



(b) The base CNN trained in 2,000 epochs using ELU activation, batch normalization and dropout at all hidden layers with a rate of 0.25.



(c) The base CNN trained in 2,000 epochs using ELU activation, batch normalization and dropout at the last hidden layer with a rate of 0.5.



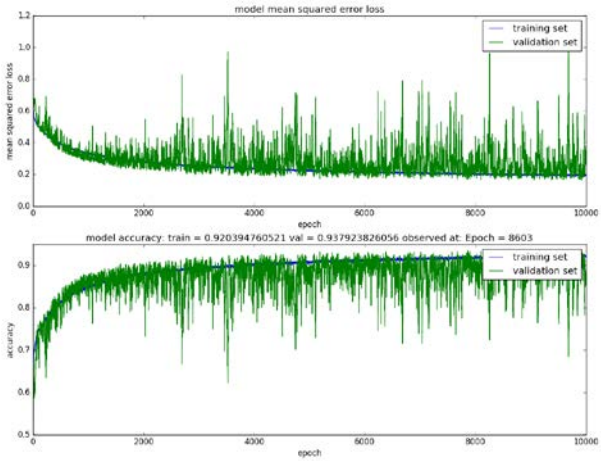
(d) The base CNN trained in 2,000 epochs using ELU activation, batch normalization and dropout at all hidden layers with a rate of 0.5.

Figure 22. Result comparison between different dropout strategies.

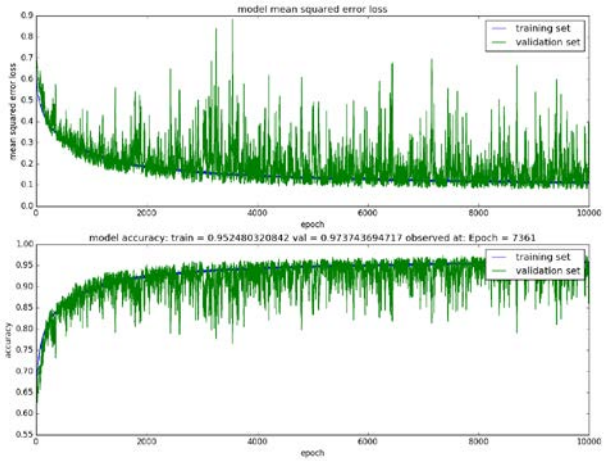
Table 10 and Figure 23 show the comparison of the base CNN and the smaller CNN under different settings of hyper-parameters. Similar training behavior of the two CNN structures are observed in Figure 23 and the base CNN outperforms the smaller CNN with respect to model validation accuracy under the same configurations.

Table 10. Result comparison between the based CNN and the smaller CNN

Val-Accuracy					
Smaller CNN	**Base CNN**	**Activation**	**BatchNormalization**	**dropout**	**Epoch**
89.6%	93.6%	ELU	yes	all_0.5	10,000
93.8%	97.4%	ELU	yes	all_0.25	10,000
92.6%	97.1%	ReLU	yes	all_0.25	10,000



(a) The smaller CNN trained in 2,000 epochs using ELU activation, batch normalization and dropout at all hidden layers with a rate of 0.25.



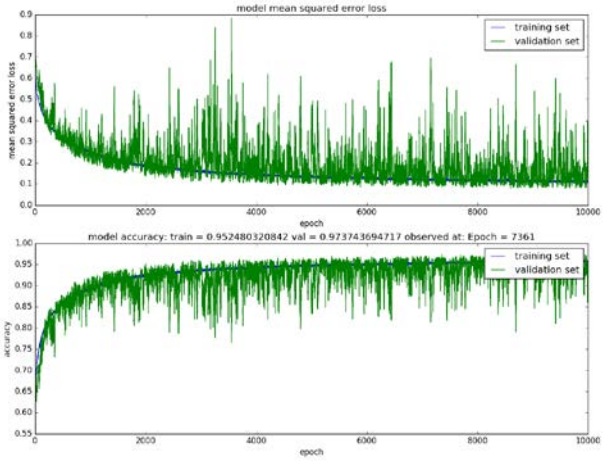
(b) The base CNN trained in 2,000 epochs using ELU activation, batch normalization and dropout at all hidden layers with a rate of 0.25.

Figure 23. Result comparison between the based CNN and the smaller CNN.

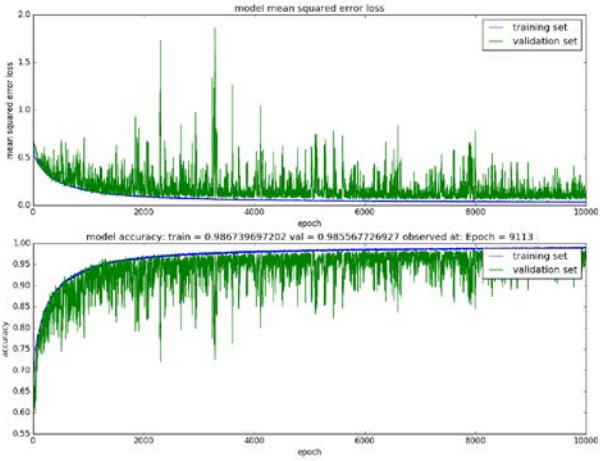
Table 11 and Figure 24 show the comparison of the base CNN and bigger CNNs under the same setting of hyper-parameters. From Figure 24, extra fully connected hidden layer helps to stabilize the training process and more filters in the convolutional hidden layers helps to increase the model validation accuracy.

Table 11. Result comparison between the based CNN and bigger CNNs

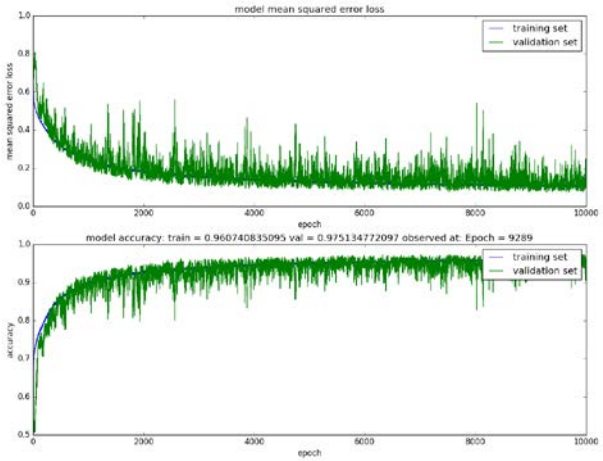
Model	**Val-Accuracy**	**Acti-vation**	**Batch Norm**	**dropout**	**Epoch**	**Model parameter**
base CNN	97.4%	ELU	yes	all_0.25	10k	12,848
bigger CNN	98.6%	ELU	yes	all_0.25	10k	25,568
deeper CNN	97.5%	ELU	yes	all_0.25	10k	16,944
bigger & deeper CNN	98.7%	ELU	yes	all_0.25	10k	29,664



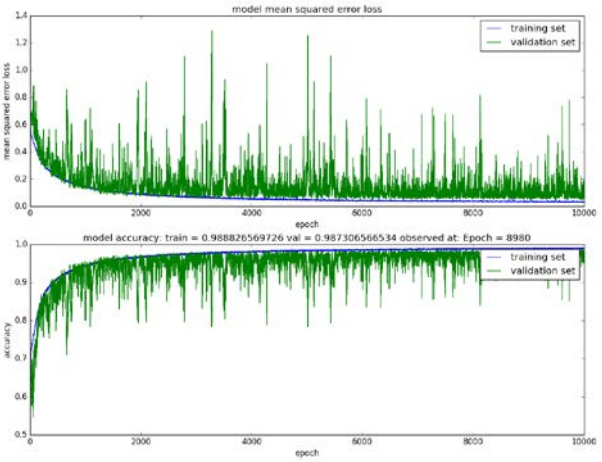
(a) The base CNN trained in 10,000 epochs using ELU activation, batch normalization and dropout at all hidden layers with a rate of 0.25.



(b) The bigger CNN trained in 10,000 epochs using ELU activation, batch normalization and dropout at all hidden layers with a rate of 0.25.



(c) The deeper CNN trained in 10,000 epochs using ELU activation, batch normalization and dropout at all hidden layers with a rate of 0.25.



(d) The bigger & deeper CNN trained in 10,000 epochs using ELU activation, batch normalization and dropout at all hidden layers with a rate of 0.25.

Figure 24. Result comparison between the based CNN and bigger CNNs.

At the end of the greedy search, taking the number of parameters into consideration we find that the best deep model configuration for our case is the bigger CNN structure shown as subplot (b) in Figure18 with ELU activation, batch normalization and dropout at all hidden layers with a rate of 0.25. This model has a training accuracy of 98.7% and validation accuracy of 98.6% after 10,000 epochs of training.

3.2.3. Model Prediction and Waze Report Comparison

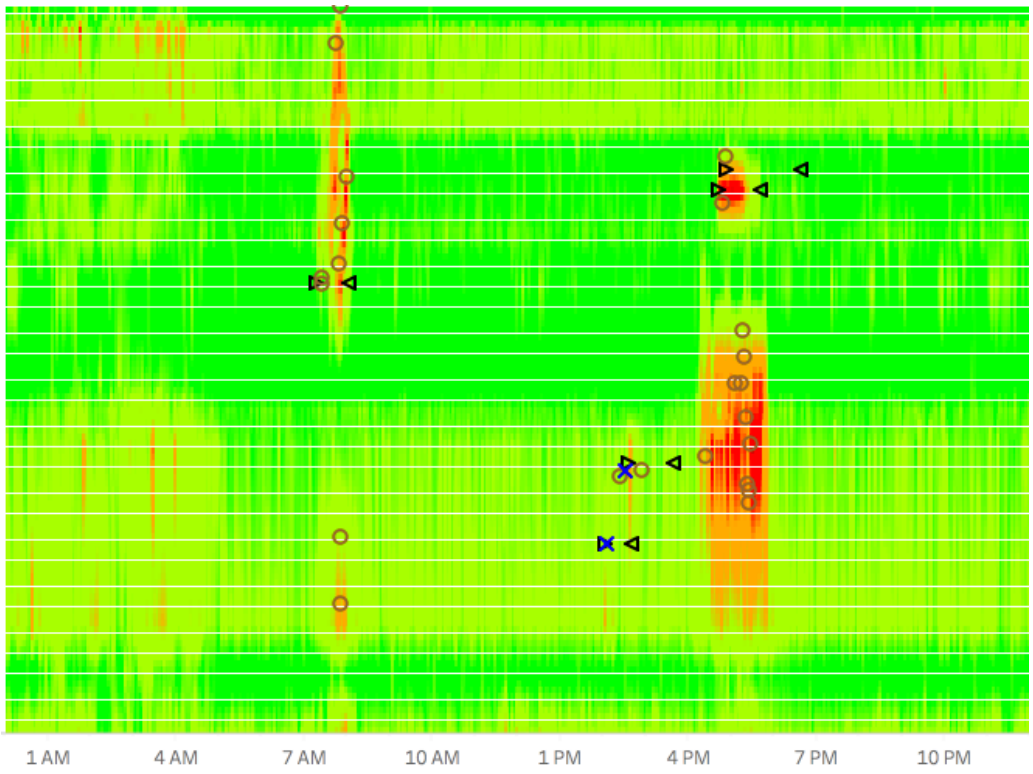
The trained model takes one patch of features as input and outputs a value ranging from 0 to 1 that indicate the likelihood of being a crash. To test the practical value of the selected model, the model is applied to classify each Waze report and the accuracy of the predicted labels are compared with the accuracy of Waze-reported event type. To test the model on Waze report: first the same patching method as described in section 2.6 was used to sample up to 42 patches per Waze report;

And then the model predicts crash likelihood for each patch; Finally each Waze reported was predicted as crash if the average crash likelihood of all related patches was greater than 0.5 and was predicted as non-crash otherwise. Using the label created in section 2.5 as the group truth, Table 12 compares the crash detection accuracy of our model and Waze-reported event type on all Waze reports. From Table 12, with respect to crash detection, using our model alone achieves much higher recall and precision than using Waze-reported event type alone.

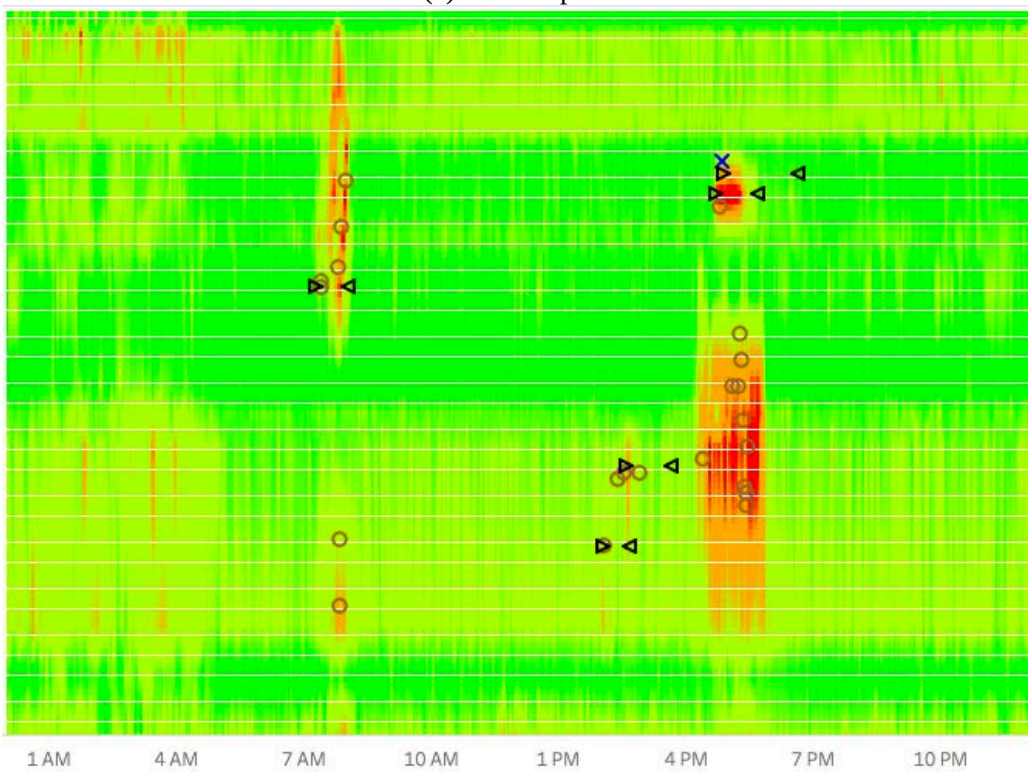
Table 12. Comparison model predicted accuracy and Waze reported accuracy on all Waze reports used in training and validation

	model predicted			Waze reported		
	non-crash	crash	total	non-crash	crash	total
real non-crash	1448	20	1468	1406	62	1468
real non-crash	0	305	305	186	119	305
total	1448	325	1773	1592	181	1773
	recall =	100.0%		recall =	39.0%	
	precision =	93.8%		precision =	65.7%	
	accuracy =	98.9%		accuracy =	86.0%	

To test the transferability of the model we also test it on a test set that the model has never seen during training. The test set contains the Waze reports on I-235 westbound on Nov 9th 2016. Figure 25 visualizes the results with black triangles indicating the start and end of a real crash, brown circle indicating the classified non-crash, blue cross indicating the classified crash and back group color indicating the traffic speed. From Figure 25, Waze correctly reported two crashes while missing the other three and our model correctly identified one different crash that has bigger impact on traffic conditions than the two crashes Waze reported. This example shows that our model tends to pick up the system impact better and human reported Waze can reflect the sense from an on-site view. Our model can add the values of systematic view into crash detection and a better crash detection can be done by combining human reported Waze and our model.



(a) Waze report



(b) Model prediction

Figure 25. Model Prediction and Waze Report Comparison on I-235 westbound, Nov 9th 2016.

4. Conclusions

This project applies machine learning algorithms including 5 shallow models and 2 deep models to solve a traffic safety problem—crash identification on Interstate 235. Waze reports adjusted by TMC accident reports are used as the labels. For each crash label, we extract 42 windows of 5-minute long and 1-mile high around the crash location. Each pixel within the window has 9 features—traffic volume, speed, sensor occupancy, air temperature, dew point temperature, pavement temperature, subsurface temperature, and pavement condition. Among the 5 shallow models only random forests with 50 trees can compete with deep models in terms of accuracy. For the deep models, we have tried different combinations of activation function, number of epoch, batch normalization, drop-out position and drop-out rate. It is found that (a) ELU activation is more stable than ReLU in training performance; (b) batch normalization could significantly increase training accuracy; (c) drop-out to all hidden layers could reduce overfitting in a better way; (d) CNN has more satisfying performance than FC neural networks; and (e) the best performance model is the bigger CNN structure shown as the subplot (b) in Figure18 with ELU activation, batch normalization and dropout at all hidden layers with a rate of 0.25. This model has a training accuracy of 98.7% and validation accuracy of 98.6% after 10,000 epochs of training. This best CNN model can be used by Iowa DOT for crash prediction and prevention on Interstate 235.

Reference

Baloian, N., Frez, J., Pino, J.A. and Zurita, G., 2015, December. Efficient planning of urban public transportation networks. In International Conference on Ubiquitous Computing and Ambient Intelligence (pp. 439-448). Springer International Publishing.

Heiskala, M., Jokinen, J.P. and Tinnilä, M., 2016. Crowdsensing-based transportation services—An analysis from business model and sustainability viewpoints. Research in Transportation Business & Management, 18, pp.38-48.

Iowa Environmental Mesonet, 2017. https://mesonet.agron.iastate.edu/request/rwis/fe.phtml

Koesdwiady, A., Soua, R. and Karray, F., 2016. Improving Traffic Flow Prediction With Weather Information in Connected Cars: A Deep Learning Approach. IEEE Transactions on Vehicular Technology, 65(12), pp.9508-9517.

Nassiri, H., Najaf, P. and Amiri, A.M., 2014. Prediction of roadway accident frequencies:: Count regressions versus machine learning models. Scientia Iranica. Transaction A, Civil Engineering, 21(2), p.263.

Pereira, F.C., Rodrigues, F. and Ben-Akiva, M., 2013. Text analysis in incident duration prediction. Transportation Research Part C: Emerging Technologies, 37, pp.177-192.

TAMU, 2017. https://mobility.tamu.edu/mip/strategies-pdfs/traffic-management/technical-summary/traffic-management-centers-4-pg.pdf

Wavetronix, 2017. https://www.wavetronix.com/en

Waze, 2017. https://www.waze.com/

Yan, C., Coenen, F. and Zhang, B., 2016. Driving posture recognition by convolutional neural networks. IET Computer Vision, 10(2), pp.103-114.