Validation of a Machine-Learning Based Vehicle Trajectory Prediction Model with Improved Data and Data Management

Daniel Hamalainen
Department of Resource Analysis, Saint Mary's University of Minnesota, Minneapolis MN 55404

Keywords: Neural-Networks, Convolutional Social Pooling, Automated Vehicles, Long Short-Term Memory, Trajectory Prediction, NGSIM, highD

Abstract

The use of neural-networks and machine learning to create and train trajectory prediction models has shown promising results for automated driving technologies. In this paper, the results of such a model are validated with improved data and preprocessing techniques. The model explores Deo and Trivedi's (2018) Convolutional Social Pooling model. The original model is recreated, trained, and evaluated with the original NGSIM dataset, then trained and evaluated with the newer highD dataset. Results are measured and compared with root-mean-squared error. The cleaner and more processed highD dataset shows a lower average error, but results are inconclusive as to why this is so. Discussion of these results illuminates both future possibilities and concerns in the fields of trajectory prediction and automated vehicle technology.

Introduction

In 2016, the United States lost over 37,000 lives to traffic accidents on highways (National Center for Statistics and Analysis, 2018), and between 94 and 96 percent of those accidents were due to human error (NCSA, 2017). These statistics are some of the main motivators for autonomous vehicles, which eliminate human error altogether. Before governments allow fully autonomous vehicles (AVs) to be used on a mass scale, they must prove their worth by showing the ability to avoid collisions and make roads safer. A sensible tactic would be to predict dangerous scenarios before they occur. In the long-term, predicting may be unnecessary since AVs can signal intentions to their counterparts and coordinate. In the near term, however, such will only be partially possible due to the presence of conventionally driven vehicles, whose movements are dictated by human motor skills, cognition, and distractibility. AVs able to predict CVs movements will therefore be most prized not only for their ability to avoid dangerous situations, which would have external benefits to other road users, but also for their ability to drive efficiently by avoiding potential slowdowns both locally - in a particular lane - and more globally - on a particular road.

These prospects have generated a recent boon in the field of vehicle trajectory prediction research, particularly via the use of machine learning on neural networks. One such study is that of Deo and Trivedi (2018). Though the two are not the first to leverage machine learning to predict vehicle movement, their study is novel in that it combines both recent travel patterns, specifically the last three seconds of motion, and traffic context to predict the next five seconds of a vehicle's trajectory. Their model does so relatively accurately, outperforming all other models at the time of publication (Deo and Trivedi, 2018).

Hamalainen, D. 2020. Validation of a Machine-Learning Based Vehicle Trajectory Prediction Model with Improved Data and Data Management. Volume 23, Papers in Resource Analysis. 24 pp. Saint Mary's University of Minnesota. University Central Services Press. Winona, MN.

The Trajectory Prediction Model: Using Neural Networks to Predict Vehicle Motion

Deo and Trivedi's (2018) model is a neural network, or a series of interconnected models where data is passed from model to model until a final output is produced. The number of models featured varies from network to network: some feature a one or two while others feature dozens. The exact design of the network depends heavily on the ends it serves, and since trajectory prediction is a type of sequence prediction, numerous models exist for such an end. One of such models is that of a specific type of recurrent neural network called long short-term memory (LSTM), which uses the relationship between elements in a sequence to predict an output. Since vehicle motion is a time-series of interdependent positions, an LSTM fits the purpose of trajectory prediction quite well.

Another critical component of their network is the use of convolutional social pooling to provide a proper traffic context for the prediction of future trajectories. Since the trajectory history of neighboring vehicles are also included as inputs, they, too, are processed through an LSTM, but for the model to decipher the meaning of these outputs, Deo and Trivedi (2018) process them through a series of layers they call convolutional social pooling. These layers generalize the meaning of the various spatial configurations of neighboring vehicles so that when the model is presented with a configuration not yet witnessed, the model will be able to relate it to patterns seen previously.

The final feature of note in their model is the incorporation of encoded driving maneuvers, or well-defined, logical adjustments to a driver's path. Two of such adjustments include lane changes and breaking, which are not only greatly impactful on a vehicle's trajectory, but are also greatly dependent upon the traffic scenario in which a driver finds her or himself in. These two aspects make maneuvers particularly useful to a model that uses neighboring trajectories to predict a vehicle's future path, since the maneuver itself can be predicted and then incorporated into the prediction of the exact trajectory.

These aspects together combine to create an estimation of a vehicle's future path given the conditions of its former path and those of its neighboring vehicles. A more fully fleshed outline of their model can be found in their 2018 paper, "Convolutional Social Pooling for Vehicle Trajectory Prediction," but for the most detailed description, the code for their work is posted on GitHub (Deo and Trivedi, 2018).

Deo and Trivedi's (2018) model has since been built upon by numerous published studies; however, the data in that model did not contain vehicle trajectory data. Because of such, the two researchers developed their model with the Federal Highway Administration's (FHWA) Next Generation Simulation dataset (NGSIM), which tracks the motion of thousands of vehicles from four locations, two of which are on US freeways. The data from those two freeways, Oakland's Interstate 80 and Los Angeles's US 101, were the two collections used by Deo and Trivedi (2018). The NGSIM dataset is one of the only massive trajectory datasets available, yet it is riddled with anomalies.

Original Model Data Challenges

Issues with the NGSIM dataset were first pointed out by a few studies in 2008 that found noise in the vehicle motion data, yet it was not until 2011 that an exhaustive search pointed out flaws that were beyond the scope of correction (Coifman and Li, 2017). In that year, one study noted the irregularity

of not only the velocities and accelerations but also of the relative positioning between vehicles (Punzo, Borzacchiello, and Ciuffo, 2011). Most notably, the study found that numerous vehicles overtook the position of their leading vehicles: a clear indication of a collision. Though it is possible for there to be a collision or two, the data collected from Interstate 80 alone features 747 vehicle tracks like this.

Along with the collision issue, they point out two other pervasive issues: abnormally high acceleration values over 3.05 m/s^2 and abnormally steady yet slow speeds. The latter can be found by looking for acceleration values of zero, which indicate constant velocity. This can be expected if vehicles are stopped, but these acceleration values are found at non-zero speeds. Constant speeds can also be found in free-flow traffic, especially with the aid of adaptive cruise control, but these speeds are at or below 1.52 m/s which is almost never seen (Coifman and Li, 2017). They explain these abnormalities as being due to the nature of NGSIM's data collection procedure, which utilizes multiple cameras to record vehicle motion then imposes an automated tracking algorithm to extract the trajectories at a time when such automation was in its infancy. Coifman and Li then show that the errors can only be repaired by re-extracting vehicle positions from the original recordings since the positions themselves are erroneous. The two did such but only for one of the six freeway subsets and only from one of the multiple cameras used in the original data collection.

A Possible Solution

Past studies suggesting the NGSIM dataset is unreliable have not deterred researchers from using the data. These errors may introduce question into models since the efficacy of those models and applications are proven using samples outside the scope of reality. Fortunately, new vehicle trajectory data - the highD dataset - was released in 2018 and could provide a reliable way to evaluate models like Deo and Trivedi's (2018). Derived from German freeway traffic, the highD dataset features all the elements of the NGSIM dataset and may hold several advantages over the older collection. Its engineers report a positional accuracy within 10 centimeters: an improvement that may be due to the advances in digital image processing since 2005 or because the vehicles are recorded from a single camera flying overhead instead of multiple cameras recorded at an angle (Krajewski, Bock, Kloeker, and Eckstein, 2018).

The arrival of the highD dataset presents an opportunity to validate Deo and Trivedi's (2018) model by training and evaluating it with improved data. Because the highD and NGSIM datasets differ in important ways, the two's methods, scopes, and general features will be compared, as well as their traffic properties and anomalies. This is followed by a detailed outline of the preprocessing steps Deo and Trivedi followed to prepare the data along with rationale for the changes made to both improve data integrity and address the new dataset's differences from the old. Finally, an overview of the training and evaluating processes is presented coupled with the reasoning for the minor adjustments made to them followed by a discussion of the results.

Methods

Dataset Comparison

Both the NGSIM and highD data come from video recordings taken on days with clear visibility and no precipitation. Both come in the form of comma separated value tables

(CSVs) where each row corresponds to a specific vehicle at a specific frame in the recording. The CSVs are made up of entries that each correspond to a vehicle-recording frame pair. They provide local vehicle positions in the form of an x-value, or the distance from the starting-line of the study zone, and a y-value, or the distance from the boundary on one of the sides of the road, as well as the vehicle's lane, velocity, and acceleration at that frame.

The NGSIM collection comes from six different CSVs: the three that come from Oakland's Interstate 80 (westbound) were gathered on April 13, 2005 for three 15-minute periods in the afternoon while the three that came from Los Angeles's US Highway 101 (southbound) were collected on June 15, 2005 for three 15-minute periods in the morning. Synchronized video cameras perched atop 30-plus story buildings adjacent to the freeways, recorded traffic across Interstate 80's six main lanes and on-ramp and US 101's five main lanes and on-off-auxiliary-lane (Figure 1).

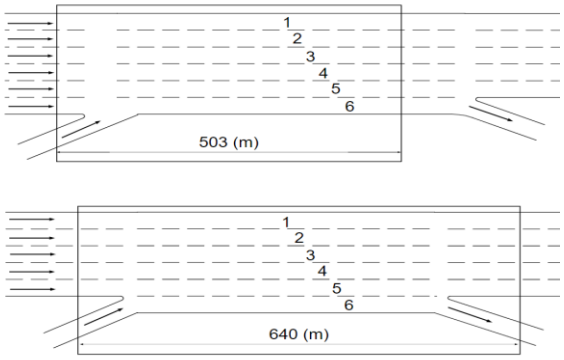

Figure 1. Schematic diagrams of NGSIM's study sites on Oakland's Interstate 80 (top) and Los Angeles's US Highway 101 (bottom).

Software created by Cambridge Systems extracted the trajectories from the video, providing positions every one-tenth of a second. Though the exact resolution of the cameras is unknown, the researchers down sampled the resolution to 640 x 480 pixels (Coifman and Li, 2017). These six subsets combine to provide the trajectories of 11,779 vehicles across 8,665,320 total positions. The distribution of these trajectories across the subsets along with the recording times can be examined in Appendix A.

While the NGSIM data are spread across six subsets, the highD data come from 60 different subsets, with each coming from one of six different 420-meter freeway sections near Cologne, Germany. The data features vehicles traveling in both directions of traffic. The time-duration of these subsets vary from six and half minutes to just over 20 minutes, with the majority spanning around 18 minutes. A drone (DJI Phantom 4 Pro Plus) flying 100 meters over each roadway recorded traffic in 4096 x 2160-pixel resolution, collecting vehicle positions every 25th of a second. An adapted U-Net neural network architecture classified each pixel as belonging to a vehicle or the background. A tracking algorithm extracted the vehicle trajectories by comparing the image classifications between frames and the objects' relative distances. After smoothing positions and speeds the final highD data features a total of 110,516 vehicles across 39,725,708 positions. When down-sampled to the NGSIM frame rate of one-tenth of a second, the number of positions totals to 15,890,283 entries. The exact distribution of these trajectories can be examined in Appendix B.

Comparison of Traffic Patterns

One thing that may affect the model's performance aside from the accuracy of the data is the difference in traffic patterns between the two collections. The fact sheets on the NGSIM dataset (Colyar and Halkais, 2007a; 2007b) posit the time periods featured represent the transition between uncongested and congested periods, but the dataset itself can reveal exactly what traffic was like.

The distribution of vehicle velocities is a good indicator for the purposes of this study for two reasons. For one, roadways with sufficiently high congestion will have a breakdown in the flow of traffic due to vehicles slowing down, but the distribution itself may be more important than the exact cause of such speeds because it shows the variety of trajectories that the model will be exposed to. Though an average alone may be misleading due to outlier-skewing, when supplemented by the median, range, and standard deviation, velocity can give good insight into the types of trajectories featured in each dataset.

Since the highD dataset has a specificity of 25 positions per second compared to NGSIM's 10, the velocities are averaged by the second. Arrays of velocities are created using Python and the open-source data processing package NumPy.

The resulting velocities somewhat confirm NGSIM's claims, with a low average velocity of 7.50 m/s (meters per second) typical of congested conditions, and a maximum velocity of 28.99 m/s indicating that at least some vehicles drove at free-flow speeds. However, the percentiles show that most of the vehicles travel at a slow velocity. The median velocity for all the NGSIM velocities is 7.31 m/s. With a standard deviation of 4.48 m/s, and with 95% of the velocities falling between zero and 17.23 m/s, the majority of the NGSIM data can be deemed as coming from some form of congestion.

While the NGSIM data is primarily concentrated during periods of congestion, the velocities found in the highD vary wildly by comparison. With an average of 28.14 and a median of 28.73 m/s (about 63 and 64 mi/h), the highD velocities are much higher. Though the standard deviation (6.91 m/s or about 15.50 mi/h) is greater than NGSIM's, it is not fully indicative of the highD's variance: 95 percent of the velocities are found between 9.63 and 39.38 m/s (22 - 88 mi/h), a range far greater than NGSIM's. This can be seen most clearly in Figure 2, which compares the ranges of velocities. The breakdown by subset can be seen in Appendix C.

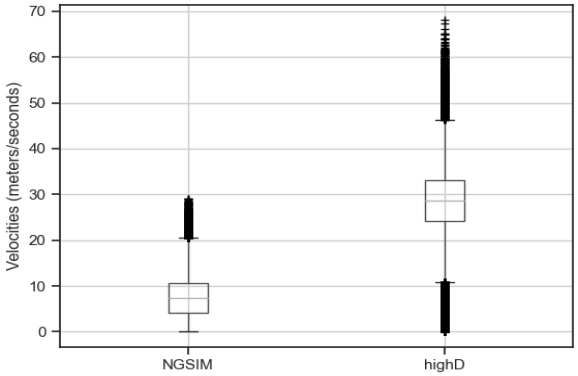

Figure 2. The distributions of speeds found in the NGSIM and highD datasets, with the latter displaying greater diversity than the former.

Comparison of Errors

This study seeks to validate Coifman and Li's (2017) findings through conducting an error search on both the NGSIM and the highD dataset. While the former features a distance headway (DHW) for each entry, it is measured by the distance between the front of the vehicle and the front of its leading vehicle instead of the leading vehicle's back. They are therefore modified by subtracting the length of each lead-vehicle from the DHW originally provided. As for the velocities and accelerations, the originals may be a product of how they were measured rather than a product of the changes in positions, so to validate consistency, these, too, are calculated via

$$v_i := \frac{x_{i+1} - x_{i-1}}{t_{i+1} - t_{i-1}}, \text{ and}$$

$$a_i := \frac{v_{i+1} - v_{i-1}}{t_{i+1} - t_{i-1}},$$

where x_{i+1} and x_{i-1} are the positions before and after the frame, t, of entry i. These measures differ very little to not at all from the originals, showing that they are consistent with the change in positions and are therefore representative of the trajectories themselves.

Across all subsets, 10.51% of vehicles feature negative distance headways. This is normally an indication of a vehicle crash, and for 10% of these vehicles to have an accident would be a newsworthy incident and one the FHWA would most likely note if true. Nearly all vehicles accelerate over 3.05 m/s^2 at one position or another. 3.30% of vehicles feature low, constant velocities, which are indicated by an acceleration of zero coupled with a velocity at or below 1.52 m/s. The exact distribution of these errors can be examined in Appendix D.

In addition to the erroneous DHWs, velocities, and accelerations, the earliest vehicles featured in each subset do not have a lead-vehicle. This is a problem because the prediction model uses the trajectories of both the vehicle in question and its neighbors. Though only a limited number of vehicles suffer from this lack of context, it may still impact the model's ability to learn. A similar yet more pervasive issue is the degree of variation in starting positions. It would be one thing if these starting positions varied only between subsets or between the mainline and the on-ramps, but the variance can be found within any given lane of any given subset. This is clearly an error since vehicles were obviously not dropped from the sky onto the road while others drove into view conventionally, and it exacerbates the missing-neighbors problem since some vehicles will be without all their neighbors for a stretch of time.

The first problem can be easily fixed by simply omitting the early vehicles from the sample-feed while keeping their track history for the purpose of providing neighboring samples their full traffic context. The second issue could be fixed if the start positions are all beyond a certain threshold, which could be as a cutoff for whether to train or evaluate with any given sample. Such is the case for most of the subsets: for these, all lanes in the mainline solely feature starting positions within 50 meters or so of the start-boundary for the study, while the starting positions for on-ramp vehicles are consistently within a different but specific range. However, starting positions in the fifth lane of two US 101 subsets range from 0.61 to 155.48 meters away from the start of the study region. Though there are only five vehicles starting beyond 150 meters, they still pose a challenge since by that distance, the on-off-auxiliary lane is fully connected to the mainline, making such a threshold infeasible to use as a criterion for inclusion as it would omit an important section of the freeway.

By contrast, the highD data contains little to none of the abnormalities found in NGSIM. The former has no vehicles with negative DHWs nor do any feature low velocities with no acceleration. Some trajectories accelerate just slightly over 3.05 m/s^2, but such are sparsely populated throughout the 60 subsets.

The highD dataset does feature errors, though. The most glaring of these occurs in three subsets of the same location, where trajectories from one traffic direction contains a good portion of vehicles going the wrong direction. These trajectories occur during what appears to be heavy congestion: the velocities continually slow till they reach 0 m/s and then below it. Since these entries are the only where vehicles slow down to a standstill throughout the entire dataset, it is likely that the tracking algorithm breaks down when faced with such behavior. These trajectories cannot be rectified since it is uncertain whether the vehicles are moving

slowly or not at all and therefore could not be used to train or evaluate the model.

Because the backwards vehicles are contained to three of the 60 subsets and in only one direction, the issue is relatively minor compared to the far more pervasive issue of varying start positions. Like with NGSIM, the trajectories in the highD dataset do not consistently begin at the same point, yet unlike NGSIM, the distribution of these initial positions is far wider. Most of the vehicles in the highD set begin their trajectory at or before the starting point of 0, but 556 vehicles start their tracks at marks greater than 25 meters, and that excludes vehicles arriving at the study area when recording began and vehicles belonging to the three subsets already omitted. Because the span of each study area for the highD dataset is about a fifth shorter than the shortest study area featured in NGSIM, the starting threshold solution proposed earlier would sever off a significant number of entries simply to preserve the neighboring context for a handful of vehicles. This dilemma is exacerbated further by the fact that across the 57 preserved subsets, 25 vehicles start at positions over 100 meters into the study section, three at 150 meters in, and two at the 200-meter mark.

Though the highD dataset is clean of the NGSIM errors, its own inconsistencies present a challenge for trajectory prediction. Therefore, these errors must be overcome before training and evaluating the model.

Error Processing the highD Dataset

Two remedies are implemented to solve the issue of various start positions. Since most of the vehicles begin their tracks within a small range of lateral positions, an initial cutoff is created by setting such to zero then incrementally increasing it by two meters until doing so does not add any new start positions to the cut-off region.

As for the vehicles with starting positions beyond the cut-off, the absence of their starting trajectories poses an issue for their would-be neighbors. A remedy would be to remove the entries of would-be neighbors at the frames in which an adjacent vehicle is missing. However, since the exact number of missing frames is unknown, the entries of their would-be-neighbors cannot be removed without an estimation of how long it took for the vehicle to get from the cutoff to its first recorded position. Such an estimation can be made by leveraging the range of times it took for other vehicles in the same subset to get from the cutoff to the recorded starting point of the vehicle missing the beginning of its track. Because this duration is correlated with the velocity at the fake starting position, a linear model is created to predict how long it took for the vehicle in question to get from the cutoff to its first recorded position. This linear model is defined as

$$\widehat{fd_{ix}} = \beta_0 + \beta_1 v_{ix},$$

where $\widehat{fd_{ix}}$ represents the number of frames for a vehicle of subset *i* to get from the cutoff to position *x*, $\beta_{0,1}$ represent the coefficients, and v_{ix} represents the velocity of the vehicle at position *x*. For each given fake starting position, a model is created by providing a NumPy array containing the velocities of each fully tracked vehicle in the subset at or near position *x* and the number of frames it took for that vehicle to get from the cutoff to *x*. This array is then fed to a linear model creator imported from the Python package scikit-learn. Using the frame-duration generated by the linear model, a final safe-estimate is made via

$$mf_{0,vehicle} = f_{0,vehicle} - \widehat{fd_{i,x}} - 2\sigma_{i,x},$$

where *vehicle* is the vehicle missing the beginning of its trajectory, $f_{0,\,vehicle}$ is its first recorded frame, and $\sigma_{i,x}$ is the standard deviation in the number of frames it takes

for vehicles to get from the cutoff to *x*. The earliest missing frame combined with the frame prior to $f_{0,\ vehicle}$ form the safe estimate of missing frames for *vehicle*. All missing frame estimates are stored to be used for filtering the entries of any would-be neighbors at those frames.

Deo and Trivedi Preprocessing Procedure

Both the NGSIM and the highD datasets require certain preprocessing steps before feeding the samples to the model for training. This includes extracting relevant data fields, generating new ones, and partitioning the data into two sections: one for training and the other for evaluation. Deo and Trivedi's (2018) steps for doing such are presented for further discussion. These steps, along with the error processing mentioned earlier, are all performed in Python by using NumPy arrays to store and transform the data prior to training and evaluating the model.

To predict a vehicle's trajectory, the model takes three seconds worth of a vehicle's positions as input, which equates to 30 individual entries since the frame rate for the NGSIM dataset is 10 frames per second. At each of these entries, the model takes the following: the x and y-coordinates, the lane identification number, the encodings for driving maneuvers, and the neighboring vehicles' identification numbers and positions relative to the vehicle in question.

To encode the lane change maneuver at a given time (or entry), the vehicle's lanes four seconds before and after are compared. If the lane stays the same, the maneuver field is encoded with "1" (no lane change), "2" if the lane increases in that span (change from left to right), and "3" if the lane decreases in that span (change from right to left).

As for the braking maneuver, a similar method is followed, but instead of looking four seconds forward and back, the positions corresponding to three seconds before and five seconds after are examined. The change in x positions between the earlier and current frame is compared with the change in x between the current and later frame. If dx_{future}/dx_{past} is less than 0.8, then the braking maneuver for that entry is encoded with "2" to represent that braking did occur; in any other case, the entry is encoded with "1" to represent no braking maneuver.

Because the neighboring vehicles must be readily accessible for the model, a 13 x 3 spatial grid is defined for each entry. Each of the three columns represents a lane: the first corresponds to the neighboring left lane, the second to the entry's own lane, and the third to the right lane. Within each lane's columns, the rows are separated by 4.57 meters, or about one car length (Deo and Trivedi, 2018). Therefore, neighbors are defined as being in the same lane or one adjacent to the target vehicle and within 27.42 meters lengthwise. The exact row the neighbor belongs to in its lane's column is determined by the formula

$$row = round\left(\frac{x_{neighbor} - x_i}{4.57}\right),$$

where *round* is to the nearest integer. The resulting row is then populated with the neighbor's identification number in the column corresponding to the neighbor's lane.

Once the arrays of each subset feature all relevant data fields, they are ready to be partitioned into their training and evaluation sets. Deo and Trivedi's (2018) partitioning method divides each subset by vehicle identification number. Vehicle ID's at the 80th percentile or below are placed in the training set while the rest are put in the evaluation set. Each entry is then marked

with an identification number for the subset it originally belonged to.

Before merging the partitioned subsets into their train or evaluate set, each vehicle's track has the first three and last fifth of a second filtered out so to allow for predictions on any of the used samples. Finally, the filtered tracks are merged into a single array that will function as the set of samples from which predictions will be made while the full tracks are placed in a dictionary data structure where the full track can be found with its vehicle and subset identification numbers when needed.

Missing Neighbors and Biased Partitioning

The preprocessing steps outlined above may be improved with respect to the creation of the neighbor grid. This is because the method for filling the grids has no recourse for instances where two or more vehicles are assigned to the same index: if a vehicle is already encoded into an index, its identification number is overwritten when a new vehicle is assigned to it. Though the highD dataset never suffers from such, across all entries in the NGSIM dataset, over 116,000 instances of shared-grid spaces occur. The issue is amplified even further by the partitioning method. Since partitioning is based on identification number, which is primarily a product of when the vehicle was recorded, the splits correlate to time; vehicles in the training set will mostly feature earlier frames than those of the evaluation set. Some frames, however, will be shared by both sides of the split, thereby separating vehicles from their neighbors.

Another consequence of the partitioning method are biases towards earlier frames in the training set than the evaluation set. This is an issue because earlier frames typically feature less congestion than later ones, meaning the types of trajectories featured in one will favor certain traffic conditions different than those in the other.

A more complete neighbor grid can be generated by using smaller grid spacing or by redirecting vehicles with the same index to different indices within the grid, and a framebuffer would ensure that vehicles are not separated from their neighbors during partitioning and that the two sets are not exposed to the same entries. However, these tactics have their own consequences. Redirecting vehicles to different indices or decreasing the distance between indices in the neighbor-grid may distort the data somewhat by presenting vehicles to be at different grid locations than they really are. Partitioning with a frame buffer would omit samples to be used for training or evaluation, and the frame buffer would not solve the issue of biased partitioning. The nature of partitioning a subset will always face an inevitable trade-off. On the one hand, splitting the data by frame, positioning, or any other data field will lead to some sort of bias; on the other, a randomly partitioned subset will sever vehicles from their neighbors. A way to partition unbiasedly while maintaining the connection between neighboring vehicles would be to allow the model to view the any neighboring track regardless of its assigned set. Such would destroy the integrity of the train-evaluate split altogether, since the model would have tangential access to the evaluation set's data: a process tantamount to cheating on an exam.

Because of the spatial-temporal nature of traffic, there is no clean way to split a single subset. Both the partitioning issue and the index-collisions are therefore left unresolved, and the NGSIM data are used only to verify that the model itself produces similar-enough results to those of Deo and Trivedi's (2018) work.

Preprocessing the highD Dataset

The highD dataset differs from the NGSIM dataset in ways that completely avoid the preprocessing complications that plague the latter. The spacing between vehicles in the highD data is sufficiently large enough to avoid a single instance of index collision when generating an entry's neighbor grid, and the multitude of subsets from the same locations allows the collection to be partitioned without dividing a single subset. Before partitioning or conducting most of the preprocessing steps outlined above, the subsets required additional steps to ensure consistency both between the trajectories and between the highD dataset and the NGSIM dataset.

While each vehicle's position comes from its front and center in the NGSIM data, the positions in the highD data come from the upper-left corner of the bounding box generated during object tracing. For vehicles in the upper-lanes of traffic, this corresponds to the right-hand side of the front bumper but corresponds to the back-left for vehicles in the lower lanes which are heading in the opposite direction. Furthermore, because the positions within a subset are based on the coordinates within the study site regardless of driving direction, vehicles in the upper-lanes featured decreasing positions (from 420 meters down to zero) while vehicles in the lower-lanes featured increasing coordinates (from zero to 420).

Because the model is primarily concerned with the change in position over time, the fact that positions are not based on the front and center of the vehicle is not of great importance so long as such is consistent. It does, however, greatly affect lane changes. Vehicles in the upper-lanes are often found to switch to the right-hand lane before quickly moving back to their original lane, while vehicles in the lower-lanes can be found to move from their lane to the lane on their left before quickly moving back. These are not real lane changes but rather vehicles swaying slightly out of their lane then adjusting back to it. Positions therefore needed to be adjusted to represent their front-and-center as opposed to the upper-left-corner positions.

After splitting each subset into two based on direction of travel, adjusting the positions to correspond to the front-and-center of each vehicle requires the vehicle's dimensions and the angle of its trajectory. The highD dataset does not explicitly give the trajectory angle for each entry but does so implicitly by providing the velocities along the x and y axes. The angle of travel, θ, is derived by taking the arctangent of the two velocities. For vehicles going from left to right, the front-center position for each vehicle is derived by

$$x_{front} = x + 0.5 * w * sin\theta \ + l * cos\theta,$$

$$y_{center} = y - 0.5 * w * cos\theta + l * sin\theta,$$

where *w* and *l* correspond to the width and length of the vehicle in question, and *x* and *y* correspond to those values for the entry in question. The equivalent for vehicles moving from right to left are

$$x_{front} = x - 0.5 * w * sin\theta, \text{ and}$$

$$y_{center} = y + 0.5 * w * cos\theta.$$

After these translations, to avoid any other possible confusion for the model, the direction of travel for vehicles in the upper lanes needs to be reversed so their x-coordinates ascend from zero to 420 rather than descend from 420 to zero. To do so, the x-coordinates must be reflected about the y-axis. Likewise, both the y-coordinates and the lane identification numbers need to be reflected about the x-axis so that lane changes would not be distorted. Reflections X_R, Y_R, and L_R are expressed by

$$X_R = \ -(X - max(X)),$$

$$Y_R = -(Y - max(Y)),$$

$$L_R = -(L - max(L)),$$

where *X*, *Y*, and *L* represent the entire array of values for x, y, and lane ID that belong to the trajectories in the upper-lanes of a subset, and *max* represents the maximum-value function. These equations have the effect of taking the largest value and making it the smallest and vice-versa: if the max value for a field is 20, then all values of 20 become zero, all values of 19 become one, values of 18 become two, and so on.

The final additional step is to reduce the amount of memory required to store the data so the model can process it more quickly. In their original state, the arrays store the data with 32-bits for each datum. Only the positional coordinates, which are given in meters to the thousandths, require more than 16 bits and are barely over its limit. Because the highD engineers state its accuracy to be within 10 centimeters of error, the centimeter unit can be discarded. From there, the units are converted to decimeters, at which point the data can be stored in NumPy arrays as unsigned-16-bit integers. However, the machine learning environment used for training and evaluating the model cannot process such a datatype: it can only process signed-16-bit integers, and in such a form, the largest of the x-coordinates would be over the capacity limit and lose its value in storage. To work around this, the x-coordinates of each subset are "centered" at zero by subtracting the median-x-value from the entirety of the subset's x-coordinates. In this state, none of the x-values exceed the lower or upper limits for NumPy's 16-bit integer datatype.

After this conversion, the original preprocessing steps are conducted in the same way as outlined earlier. Filtering also occurs, but the missing frames kept earlier are used to filter out any would-be neighbors of vehicles as well. To determine if a vehicle would be a neighbor, an *x* position for the vehicle is required across its missing frames. The *x* for each missing frame is estimated by assuming a constant velocity from the cutoff to the first-known position. Because this is an estimation, the neighbor-radius is increased to 40 meters to account for the likely error.

Finally, the subsets are assigned to train and evaluate sets. To ensure proportionality, the allocation of subsets is selected so that the total number of vehicles belonging to the training set is four times as many as the number in the evaluation set, or the same proportionality as in Deo and Trivedi's (2018) study. To ensure exposure to locations, subsets are allocated so that the evaluation set has at least one subset from each location in both directions of travel. All possible allocations are produced that meet these criteria; from these, the allocation where the training set's traffic conditions differ the least from that of the evaluation set is chosen to be the final.

To best compare the overall variety of traffic conditions featured in the allocation possibilities, traffic flow per lane is used. The traffic flow per lane for a given subset is determined by the equation

$$flow = \frac{|vehicles|}{|lanes| * |seconds|},$$

where |*vehicles*| is the number of vehicles in the subset, |*lanes*| is the number of lanes, and |*seconds*| is the duration of time for the subset (Hall, 1992). The composition of each possible training and evaluation set can be determined by using the weighted average and weighted standard deviation of traffic flow. The weights for such are determined not by the number entries originally in each subset but rather by the number of kept entries after all preprocessing, since the model will gain exposure to these traffic conditions only via those kept. The weighted average of one of

the splits can be derived by

$$\mu_N = \sum_{i=1}^{|N|} w_i \, flow_i \, ,$$

where $|N|$ represents the number of subsets in N, and w_i represents the assigned weight for subset i, which is derived in turn by

$$w_i = \frac{|entries|_i}{|entries|},$$

where $|entries|_i$ is the number of kept-entries for the subset i, and $|entries|$ is the number of kept-entries in total for the assortment of subsets in question. This average gives the average traffic condition faced by a given entry in one of the splits. However, to account for outliers, the weighted standard deviation is used as well. The weights are used so to account for the uneven distribution of kept entries across the different subsets. This can be derived by

$$\sigma = \sqrt{\sum_{i=1}^{|N|} w_i \, (flow_i - \mu)^2}.$$

The optimal sorting of subsets renders a weighted average traffic flow of 0.349 vehicles per lane and second for the training set and 0.351 for the evaluation set while the weighted standard deviations came in at 0.076 and 0.075. Both of metrics differ by just over a thousandth of a vehicle per lane and second. Though a higher weighted standard deviation for the evaluation set would be preferred, all other possibilities compromise proportionality or similarity.

Training and Evaluating the Model

When training a neural network or any other model with machine learning, the goal is to calibrate the weights contained within the model so that it produces the best-possible output, where best-possible is usually defined as being the most accurate. To calibrate in such a way, a machine learning environment measures the error of a given guess and recalibrates the model's weights given this error. if its previous recalibration resulted in a decrease in error, the training-environment figures that it made a good adjustment, and adjusts further based on this feedback, and then has the model guesses on another sample. This feedback-loop continues until all samples have been processed.

This learning-process is defined by a set of learning parameters. Because the learning parameters will vary from model to model, engineers often adjust these parameters after a learning-cycle, or after all the samples have been processed through. Such parameters include the error-function, the number of samples provided between weight-calibration, or batch-size, and the optimization function, which determines how error measures will impact changes to the model's weights.

Deo and Trivedi (2018) use the open-source machine learning library PyTorch for the entirety of their training and evaluation. They measure error with both root-mean-square error (RMSE) and negative log-likelihood (NLL), feature 128 samples per batch, and use the optimization algorithm ADAM to calibrate model weights. They switch between the error measures by training five cycles with RMSE then three with NLL. Because the purpose of this project is to evaluate theirs with different data, all learning parameters are kept except for the method of measurement for NLL. In their code, Deo and Trivedi inconsistently measure NLL (Deo and Trivedi, 2018), so the negative log-loss function is adjusted so it is both consistent with the probability density function of a bivariate normal distribution and consistent throughout the code (Weisstein, 2002).

Results

Because of the adjustment to the calculation for NLL, the comparison of errors from Deo

and Trivedi's (2018) study and that of this paper is kept strictly to the average RMSE as presented in Table 1.

Table 1. Comparison between Deo and Trivedi's (2018) and this project's evaluative root mean square errors (RMSE) at each second across the five second prediction horizon when using the NGSIM dataset; RMSE equates to the average distance between the predicted and actual positions at a given time along the prediction horizon.

Time (s)	Deo & Trivedi, NGSIM (m)	Hamalainen, NGSIM (m)
1	0.62	0.59
2	1.29	1.29
3	2.13	2.14
4	3.20	3.18
5	4.52	4.48

The difference in error between Deo and Trivedi's (2018) and this study's is minor enough to deem the replica a fair representation of their model. The comparison between this study's NGSIM and highD results is expanded to include maneuver accuracies as seen in Table 2.

Table 2. Comparison between the model's evaluative root-mean-square error (RMSE) when using the NGSIM and highD dataset at each second across the five second prediction horizon; RMSE equates to the average distance between the predicted and actual positions at a given time along the prediction horizon.

Error Measure	Time (s)	NGSIM	highD
RMSE (m)	1	0.59	0.26
	2	1.29	0.78
	3	2.14	1.54
	4	3.18	2.69
	5	4.48	7.85
Lane Change Accuracy		97.96 %	97.94%
Braking Accuracy		89.29 %	99.94%

Discussion

Across the first four seconds of a vehicle's trajectory, the model and machine learning algorithm perform better on the highD dataset than on the NGSIM dataset, but the model is exceedingly worse at predicting the fifth second. Though the lane change accuracies are similar, the model learns the breaking tendencies of the highD data far better than it learns those of the NGSIM data.

The differences between Deo and Trivedi's (2018) work and this project's NGSIM results should be addressed before other discussion. Though the preprocessing steps are modified for the highD dataset, the only modification made for the NGSIM set is that made to the negative log-likelihood function. Such a change is a probable explanation for the difference in RMSE: the model learned from a more consistent error function. However, the model samples from the dataset randomly, the order of these samples simply may have been somehow better for the model.

As for the differences between this paper's model's attempts with NGSIM and highD, the exact explanations are less clear. A likely candidate for the difference in braking maneuver accuracy is the inaccurate vehicle positions in the NGSIM data, whose velocities and accelerations are so erratic that they may not conform to a consistent pattern as does those of the highD dataset. However, the traffic patterns themselves may also play a role in this difference. Braking tendencies on an open road are somewhat more logical than those in congested conditions. In the former, when approaching a slower lead vehicle, the driver can either change lanes or slow down; when the adjacent lanes are occupied in this situation, braking becomes the only option. Driving tendencies also may make it easier for the model, too, since European drivers

are more likely to reserve the left lanes for passing, making maneuvers generally more predictable (Treiber and Kesting, 2009).

This last point would be expected to translate into an advantage for the lane changing maneuvers as well. The fact that it does not may be explained by the roadways themselves: the NGSIM highways feature some sort of on-ramp on the right side of their roadways. The geometries are even more similar when accounting for the fact that while Interstate 80 does not feature an off-ramp in its study area like US 101 does, there is an off-ramp located just outside of the former's region. Such lends itself to a clear pattern in lane changing for the right-most lanes, giving NGSIM a sort of counter-advantage to the highD dataset, where only three of the subsets feature a sole on-ramp.

As for the change in RMSE over time, explanations are even less cle
ar and somewhat compete. On the one hand, because of the greater diversity of speeds in the highD data, one may expect the model to have more difficulty learning its tendencies than those of the NGSIM data, whose velocities are more uniform. On the other hand, the erroneousness of the latter may be difficult to translate into any consistent and definable pattern. Furthermore, dense traffic conditions may simply be less stable: a possibility that would explain both the errors in the NGSIM data and the errors found in the highD data's congested subsets. The preprocessing errors may be a cause, too, since the neighboring traffic context is not fully presented in the final NGSIM dataset.

Though the highD iterations win or tie in almost all regards, their one shortcoming – the great leap in error after four seconds – is better understood through a more detailed view. This can be seen in in Figure 3, where the errors for every one-fifth of a second are given and plotted alongside the change in error at each step. At the fourth second, the change in error steeply increases, indicating a greater degree of difficulty with predictions at that moment than earlier. Though the variety of speeds explains why the highD iterations fall short of NGSIM after a certain speed, it may not explain this sudden growth in error. If vehicle A travels at 30 m/s and B at 40 from the same starting position, their difference in position after one second is less than it will be after four, but such adjusts linearly. This can be seen with the NGSIM change in error, and with a greater degree of variety, one would expect the same, just steeper. Therefore, this increase goes beyond just the greater diversity of traffic flow.

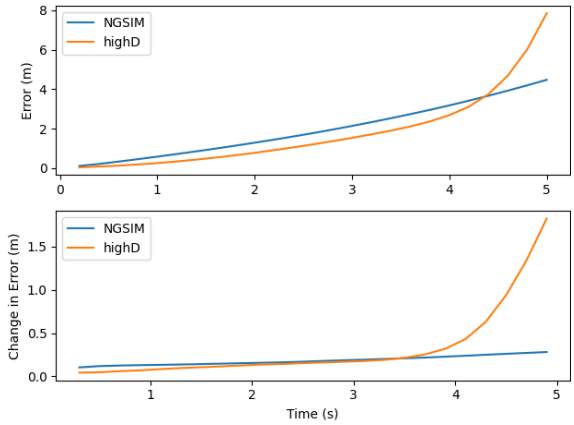

Figure 3. Comparison between the model's evaluative root-mean-square error (RMSE) when using the NGSIM and highD dataset; depicted are comparisons of both the RMSE and change in RMSE at each time-step (one fifth of a second) across the prediction horizon of five seconds; RMSE is best thought of as the average distance between the predicted and actual positions at a given time-step along the prediction horizon.

Conclusion

The results shown are by no means proof that autonomous vehicles will have the ability to avoid all possible accidents, but they do show promise when considering the limited samples provided. Furthermore, because the model generally improved when given less erroneous data, perhaps the improvement can continue as more accurate datasets become available.

It should be reminded that both the NGSIM and highD datasets come from aerial recordings: a far different perspective than that of a vehicle. For an AV to view its surrounding environment without any outside-help, it would need to be through an on-board sensor. Though this would generate similar inputs, in a dense traffic environment, they would be limited to just the vehicles immediately surrounding it, since they would block the view of other vehicles. Because of the interdependence that vehicles have on each other, omitting those others would likely impact the model's performance in dense traffic scenarios.
A way around would be the use of an openly available trajectory feed generated by overhead cameras or by mandating that vehicles communicate their positions using inter-vehicle connectivity systems. The latter would be especially promising as vehicles could broadcast velocities, accelerations, and signals in addition to the positions themselves. However, such methods raise concerns over privacy: should drivers be required by law to broadcast their positions or driving intentions to the public? Many do exactly this when they use navigation applications like Google maps, but drivers are not required by law to do so. Such dilemmas show that the deployment effective autonomous vehicles may require ethical compromises.

Above all things, this study should highlight the need for more accurate and abundant driving data. Hundreds and thousands of studies have used the NGSIM dataset in one way or another, and only a handful do so to examine its flaws (Coifman and Li, 2017). Without more accurate data, human driving tendencies may forever be a mystery. A lack of such knowledge may slow the approach of fully autonomous vehicles and their benefits or require governments and other stakeholders to segregate conventionally driven vehicles from the autonomously driven ones.

Acknowledgements

Special thanks are owed to Saint Mary's University of Minnesota's Dr. John Ebert, and Greta Poser, who helped guide me through both this capstone and the rest of my academic postgraduate journey. Thanks are also due to Nachiket Deo, who responded promptly to my questions and comments on his GitHub repository.

References

Coifman, B., and Li, L. 2017. A Critical Evaluation of the Next Generation Simulation (NGSIM) Vehicle Trajectory Dataset. *Transportation Research Part B, 105*, 362-377. Retrieved April 17, 2020 from http://www2.ece.ohio-state.edu/~coifman/documents/Coifman_and_Li_2017.pdf.

Colyar, J., and Halkias, J. 2007a. US Highway 101 Dataset. Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT07-030.

Colyar, J., and Halkias, J. 2007b. Interstate-80 Dataset. Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT07-030.

Deo, N., and Trivedi, M. 2018. Convolutional Social Pooling for Vehicle Trajectory Prediction. *Conference on Computer Vision and Pattern Recognition Workshops*, 1468-1476. Retrieved June 13, 2020 from https://openaccess.thecvf.com/content_cvpr_2018_workshops/papers/w29/Deo_Convolutional_Social_Pooling_CVPR_2018_paper.pdf and from https://github.com/nachiket92/conv-social-pooling.

Hall, F. 1992. Traffic Stream Characteristics, 7. Retrieved November 20, 2020 from https://www.fhwa.dot.gov

/publications/research/operations/tft/chap2.pdf.

National Center for Statistics and Analysis (NCSA). 2017. 2016 Fatal Motor Vehicle Crashes: Overview. *Traffic Safety Facts Research Note. Report No. DOT HS 812 456*, Washington, DC: National Highway Traffic Safety Administration. Retrieved August 20, 2020 from https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812456.

National Center for Statistics and Analysis (NCSA). 2018. Summary of motor vehicle crashes: 2016 data. *Traffic Safety Facts. Report No. DOT HS 812 580*, Washington, DC: National Highway Traffic Safety Administration. Retrieved August 20, 2020 from https://www.nhtsa.gov/press-releases/usdot-releases-2016-fatal-traffic-crash-data.

Punzo, V., Borzacchiello, M., and Ciuffo, B. 2011. On the Assessment of Vehicle Trajectory Data Accuracy and Application to the Next Generation SIMulation (NGSIM) Program Data. *Transportation Research Part C, 19*(6), 1243-1262.

Krajewski, R., Bock, J., Kloeker, L., and Eckstein, L. 2018. The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, HI, 2118-2125. Retrieved April 22, 2020 from https://axriv.org/abs/ 1810.05642.

Treiber M., and Kesting, A. 2009. Modeling Lane-Changing Decisions with MOBIL, *Traffic and Granular Flow*. Springer, Berlin, Heidelberg. Retrieved September 24, 2019 from https://www.researchgate.net/publication/227128200_Modeling_lane-changing_decisions_with_MOBIL.

Weisstein, E. 2002. Bivariate Normal Distribution. *MathWorld*--A Wolfram Web Resource. Retrieved August 29, 2020 from https://mathworld.wolfram.com/BivariateNormalDistribution.html.

Appendix A. NGSIM Dataset Location Information. Vehicles represents the total number of unique vehicles featured, and entries represents the number of unique vehicle positions.

Location	Date & Time	Lanes	Vehicles	Entries
I-80 Oakland, CA	04/13/2005 16:00 - 16:15	6 + on-ramp	2,052	1,262,678
I-80 Oakland, CA	04/13/2005 17:00 - 17:15	6 + on-ramp	1,836	1,549,918
I-80 Oakland, CA	04/13/2005 17:15 - 17:30	6 + on-ramp	1,790	1,753,791
US 101 Los Angeles, CA	06/15/2005 07:50 - 08:05	5 + auxiliary on-off-ramp	2,169	1,180,598
US 101 Los Angeles, CA	06/15/2005 08:05 - 08:20	5 + auxiliary on-off-ramp	2,017	1,403,095
US 101 Los Angeles, CA	06/15/2005 08:20 - 08:35	5 + auxiliary on-off-ramp	1,915	1,515,240

Appendix B. HighD Dataset Location Information. All data from the highlighted (yellow) subsets were omitted due to an abundance of backward trajectories.

Location 1 (Bundesbahn 4) Subsets					
ID	**Day of Week, Month Year (Time)**	**L: 3 lanes**		**R: 3 lanes**	
		Vehicles	Entries	Vehicles	Entries
1	Fri., 09 2017 (08:21 - 08:30)	466	137,811	389	119,220
2	Fri., 09 2017 (08:37 - 08:55)	823	245,296	797	236,956
3	Fri., 09 2017 (09:24 - 09:42)	723	219,985	697	216,649
4	Fri., 09 2017 (10:36 - 10:46)	413	120,487	443	137,825

Location 2 (Bundesbahn 61) Subsets					
ID	**Day of Week, Month Year (Time)**	**L: 2 lanes**		**R: 2 lanes**	
		Vehicles	Entries	Vehicles	Entries
5	Fri., 09 2017 (08:49 - 09:01)	424	143,807	498	167,336
6	Fri., 09 2017 (09:11 - 09:27)	418	141,663	580	196,389
7	Fri., 09 2017 (09:35 - 09:50)	404	131,988	573	191,662
8	Fri., 09 2017 (10:07 - 10:17)	249	77,615	358	113,280
9	Fri., 09 2017 (10:24 - 10:40)	449	149,464	550	178,830
10	Fri., 09 2017 (10:47 - 10:54)	577	197,332	622	208,307
11	Fri., 09 2017 (11:10 - 11:27)	484	157,428	593	196,002
12	Fri., 09 2017 (11:44 - 11:53)	289	100,028	400	136,176
13	Fri., 09 2017 (12:06 - 12:22)	726	267,060	588	200,005
14	Fri., 09 2017 (12:27 - 12:43)	683	230,997	614	210,015

Location 3 (Bundesbahn 4) Subsets					
ID	**Day of Week, Month Year (Time)**	**L: 3 lanes & on-ramp**		**R: 3 lanes**	
		Vehicles	Entries	Vehicles	Entries
15	Wed., 07 2018 (09:15 - 09:21)	373	117,738	341	115,501
16	Wed., 07 2018 (09:23 - 09:31)	376	119,904	346	111,316
17	Wed., 07 2018 (09:37 - 09:53)	733	240,571	734	259,923

Location 4 (Bundesbahn 61) Subsets					
ID	**Day of Week, Month Year (Time)**	**L: 3 lanes**		**R: 3 lanes**	
		Vehicles	Entries	Vehicles	Entries
18	Thu., 09 2017 (16:18 - 16:28)	1,023	399,790	753	258,981
19	Thu., 09 2017 (17:21 - 17:36)	1,451	683,865	1,277	449,016
20	Thu., 09 2017 (18:04 - 18:22)	1,632	607,596	1,317	437,394
21	Thu., 09 2017 (18:28 - 10:46)	1,573	557,865	1,271	423,744
22	Mon., 10 2017 (08:55 - 09:15)	1,232	1,306,832	1,618	712,920
23	Mon., 10 2017 (09:20 - 09:39)	1,123	819,787	1,585	766,244
24	Mon., 10 2017 (09:46 - 10:06)	1,131	390,694	1,468	561,946
25	Mon., 10 2017 (10:12- 10:33)	1,116	383,280	1,256	435,371
26	Mon., 10 2017 (10:39 - 10:58)	1,014	352,317	1,138	404,511
27	Mon., 10 2017 (11:03 - 11:23)	1,178	421,427	1,301	461,603
28	Mon., 10 2017 (11:28 - 11:47)	1,038	364,327	1,216	444,564
29	Mon., 10 2017 (12:20 - 12:33)	685	237,673	725	252,863
30	Mon., 10 2017 (12:41 - 12:59)	1,062	379,386	1,190	423,048
31	Mon., 10 2017 (13:34 - 13:48)	862	310,693	911	325,468
32	Wed., 10 2017 (11:26 - 11:44)	984	340,423	1,008	351,042
33	Wed., 10 2017 (11:09 - 11:30)	1,237	490,654	1,306	461,482
34	Wed., 10 2017 (11:55 - 12:13)	1,011	358,283	1,077	349,711
35	Wed., 10 2017 (12:20 - 12:40)	1,160	406,202	1,220	425,244
36	Mon., 10 2017 (09:04 - 09:24)	1,191	391,015	1,208	405,144
37	Mon., 10 2017 (09:30 - 09:50)	1,272	420,737	1,091	357,923
38	Mon., 10 2017 (10:41 - 11:00)	1,306	443,153	1,138	379,700
39	Mon., 10 2017 (11:05 - 11:23)	1,194	405,917	1,077	349,711
40	Mon., 10 2017 (11:31 - 11:48)	1,312	440,163	1,237	411,762
41	Mon., 10 2017 (11:54 - 12:14)	1,261	425,093	1,118	390,600
42	Mon., 10 2017 (12:23 - 12:41)	1,319	436,460	1,502	654,508
43	Wed., 11 2017 (08:47 - 09:07)	1,253	441,615	1,209	413,570
44	Wed., 11 2017 (09:15 - 09:37)	1,280	448,130	1,264	421,874

45	Wed., 11 2017 (09:38 - 09:57)	1,033	351,733	1,063	356,664
46	Wed., 11 2017 (10:02 - 10:19)	1,141	389,865	1,093	365,876
47	Wed., 11 2017 (11:38 - 11:57)	1,163	390,570	1,121	378,828
48	Wed., 11 2017 (12:05 - 12:23)	1,121	375,327	1,000	337,049
49	Wed., 11 2017 (12:30 - 12:47)	1,235	427,657	1,207	411,374
50	Wed., 11 2017 (13:15 - 13:32)	1,096	360,541	1,247	413,932
51	Thu., 01 2018 (09:16 - 09:34)	1,050	352,439	1,140	384,900
52	Thu., 01 2018 (09:39 - 09:57)	822	276,301	973	333,718
53	Thu., 01 2018 (10:04 - 10:20)	953	320,787	966	319,183
54	Thu., 01 2018 (10:26 - 10:44)	373	117,738	341	115,501

Location 5 (Bundesbahn 61) Subsets					
ID	**Day of Week, Month Year (Time)**	**L: 3 lanes**		**R: 3 lanes**	
		Vehicles	Entries	Vehicles	Entries
55	Thu., 09 2017 (11:16 - 11:34)	575	197,312	588	193,375
56	Thu., 09 2017 (11:41 - 12:00)	599	203,652	617	198,886
57	Thu., 09 2017 (12:06 - 12:26)	670	229,536	698	226,287

Location 6 (Bundesbahn 46) Subsets					
ID	**Day of Week, Month Year (Time)**	**L: 2 lanes**		**R: 2 lanes**	
		Vehicles	**Entries**	**Vehicles**	**Entries**
58	Tue., 09 2017 (08:38 - 08:53)	594	199483	453	149267
59	Tue., 09 2017 (09:04 - 09:21)	689	237723	424	140392
60	Tue., 09 2017 (09:54 - 09:11)	497	163884	417	139604

Appendix C. Velocity Distributions for NGSIM and highD. These tables show the minimum and maximum velocities (MIN and MAX) along with the 2.5th, 50th, and 97.5th percentile speeds as well as the average (AVG) and standard deviation (STD). The velocity distributions are indicative of the type of traffic scenarios featured in each subset, with the highD data being far more diverse than the NGSIM data. All data from the highlighted (yellow) subsets were omitted due to an abundance of backward trajectories.

NGSIM VELOCITY DISTRIBUTIONS

ID	MIN	2.5%	50%	97.5%	MAX	AVG	STD
I80A	0.0	0.89	7.48	18.48	29.05	7.72	4.06
I80B	0.0	0.0	4.88	15.24	29.05	5.62	3.89
I80C	0.0	0.0	4.23	13.74	29.05	4.83	3.62
US101A	0.0	1.33	11.96	19.56	29.05	11.43	4.53
US101B	0.0	0.22	9.15	15.91	29.05	8.94	4.09
US101C	0.0	0.49	8.03	14.68	29.05	7.84	3.67

highD VELOCITY DISTRIBUTIONS

ID	MIN	2.5%	50%	97.5%	MAX	AVG	STD
L1	21.54	23.29	33.79	44.6	49.24	33.1	6.21
R1	21.83	23.46	33.73	47.6	61.17	33.22	7.09
L2	21.03	23.43	33.67	46.81	57.65	33.49	6.74
R2	21.53	23.33	33.76	48.13	61.72	33.55	7.39
L3	21.17	23.05	33.18	47.36	56.37	32.93	7.17
R3	17.25	22.99	32.23	45.89	68.28	32.21	6.8z
L4	21.89	23.4	34.25	50.27	59.45	34.08	7.55
R4	21.6	23.4	32.91	46.25	58.64	32.53	6.56
L5	19.53	22.53	28.76	40.99	54.1	29.54	5.57
R5	21.08	22.54	29.6	39.74	44.62	29.82	5.07
L6	19.62	22.39	29.22	42.3	54.23	30.03	6.06
R6	20.73	22.28	29.28	42.46	56.4	29.92	5.83
L7	19.66	22.05	31.03	43.26	64.0	30.6	6.23
R7	19.11	22.54	30.26	40.81	50.09	30.15	5.11
L8	21.62	22.91	31.95	45.87	54.75	31.92	6.63
R8	19.99	22.85	32.09	45.33	61.07	31.86	5.76
L9	20.22	22.18	30.38	42.13	50.6	30.35	5.70
R9	20.67	22.92	31.99	41.03	46.75	31.41	5.22
L10	19.75	22.56	29.43	39.92	50.2	29.66	5.20
R10	19.26	22.29	30.51	40.55	51.54	30.16	5.16
L11	20.86	22.57	31.14	42.74	52.92	30.84	5.83
R11	20.15	22.98	31.05	40.64	55.11	30.73	5.03
L12	19.69	21.31	27.98	40.9	50.31	28.96	5.48
R12	21.46	22.77	29.45	38.95	46.17	29.34	4.41
L13	16.66	20.39	26.32	36.68	46.66	27.15	4.40
R13	20.98	22.63	30.26	39.74	48.9	30.05	4.82
L14	20.68	22.13	29.38	40.74	47.64	29.62	5.23
R14	19.77	22.46	29.33	40.58	53.92	29.72	5.19
L15	12.83	20.81	31.22	41.95	55.29	30.97	5.93
R15	-1.87	19.87	28.08	39.15	52.0	28.08	5.46
L16	14.76	21.77	31.41	42.35	53.47	31.12	5.90
R16	12.55	20.32	29.64	42.91	61.28	29.83	6.37
L17	11.48	21.28	31.01	42.01	51.43	30.51	5.96
R17	16.4	18.84	28.56	40.14	62.71	28.31	5.90
L18	17.41	20.05	25.42	32.54	40.65	25.6	3.10
R18	19.37	22.42	30.06	36.7	40.4	29.46	4.17
L19	4.07	12.03	20.88	30.67	36.57	21.41	4.53
R19	19.1	22.31	28.65	36.83	42.25	28.71	4.05

L20	17.49	21.28	26.61	35.77	41.86	27.15	4.02
R20	19.45	22.82	31.11	38.21	43.8	30.53	4.32
L21	19.54	21.91	28.11	36.81	42.16	28.42	4.19
R21	16.0	22.5	30.78	38.54	44.59	30.31	4.43
L22	-0.43	1.98	9.62	15.95	20.4	9.23	3.87
R22	2.73	6.97	23.4	29.64	34.43	22.45	5.20
L23	-2.51	3.16	12.51	31.89	43.49	13.48	6.79
R23	5.37	13.58	21.26	26.76	33.16	20.73	3.523
L24	16.73	21.26	30.14	38.11	54.25	29.44	4.93
R24	11.79	18.85	25.14	35.81	41.88	26.27	4.77
L25	17.03	21.78	30.5	39.0	50.14	29.62	5.26
R25	18.62	22.17	29.93	37.19	45.33	29.32	4.33
L26	14.96	21.78	29.95	37.74	46.09	29.27	4.67
R26	17.39	21.02	29.24	36.94	49.31	28.66	4.69
L27	18.58	21.38	28.75	36.63	43.95	28.44	4.57
R27	16.98	21.52	29.07	36.29	44.04	28.64	4.28
L28	19.24	21.22	29.82	37.63	45.34	29.09	4.95
R28	16.66	20.99	27.88	35.85	45.07	27.87	4.41
L29	18.88	21.88	29.7	37.87	44.64	29.0	4.79
R29	19.05	21.88	29.88	36.84	51.63	29.11	4.56
L30	18.38	21.41	28.85	36.81	47.61	28.42	4.73
R30	17.84	21.49	28.94	36.58	42.7	28.52	4.35
L31	19.29	21.73	28.62	35.92	42.11	28.19	4.38
R31	14.5	21.28	28.85	35.65	42.08	28.31	4.21
L32	11.51	21.93	30.03	38.17	51.73	29.38	4.93
R32	11.29	22.1	29.4	37.71	43.4	29.1	4.64
L33	-0.1	5.27	25.53	35.7	42.08	25.26	7.27
R33	17.28	21.97	29.24	36.92	42.32	28.75	4.43
L34	18.99	21.71	28.58	36.71	45.0	28.47	4.54
R34	18.53	21.83	30.09	39.21	53.1	29.44	5.04
L35	18.61	21.4	29.57	37.38	46.17	28.9	4.88
R35	20.27	22.36	29.76	37.83	45.29	29.18	4.73
L36	16.49	21.7	30.48	39.53	49.08	29.96	5.135
R36	18.58	22.23	31.48	40.32	52.27	30.8	5.09
L37	17.63	22.22	31.44	39.62	50.64	30.8	5.00
R37	19.81	22.22	30.34	39.35	56.64	30.19	4.82
L38	16.86	21.77	31.12	38.88	57.08	30.45	4.99
R38	17.95	22.77	31.7	39.24	50.2	30.94	4.67
L39	18.23	22.0	30.24	38.01	45.47	29.79	4.62
R39	16.04	22.65	30.87	38.71	46.93	30.33	4.61
L40	15.32	21.62	30.34	38.46	48.58	29.87	4.87
R40	20.89	23.05	31.94	40.09	51.09	31.36	4.78
L41	19.39	22.41	30.56	38.73	56.71	30.08	4.82
R41	16.3	22.85	31.02	39.5	46.66	30.49	4.84
L42	19.95	22.13	30.48	38.28	47.49	30.01	4.67
R42	20.67	22.71	31.46	39.83	46.95	30.81	4.88
L43	19.62	22.53	30.99	38.67	46.41	30.44	4.79
R43	2.28	6.56	24.08	35.36	44.99	22.95	6.96
L44	17.99	21.09	28.45	37.6	45.59	28.69	4.63
R44	17.03	22.18	29.69	38.21	43.48	29.66	4.57
L45	17.47	21.49	29.43	37.2	45.19	29.09	4.53
R45	20.35	23.07	30.86	38.12	46.72	30.27	4.54
L46	19.1	22.1	30.15	38.94	45.73	29.83	4.94
R46	19.0	22.83	30.7	38.03	43.85	30.12	4.53
L47	20.14	22.44	29.43	39.18	57.3	29.6	5.07
R47	20.69	23.01	30.8	39.06	46.02	30.25	4.74
L48	20.41	22.89	31.01	39.12	45.33	30.24	4.99
R48	20.01	22.7	30.07	38.76	49.1	29.91	4.77
L49	20.43	22.59	30.41	38.99	48.23	30.04	4.98
R49	18.11	22.7	30.3	38.49	45.97	30.0	4.75
L50	19.15	22.59	29.52	38.18	46.86	29.45	4.51

R50	21.01	23.27	30.27	38.3	45.19	29.87	4.54
L51	19.94	22.5	31.73	39.87	55.73	30.82	5.23
R51	20.05	23.08	31.28	38.63	46.94	30.57	4.66
L52	19.73	22.23	30.63	39.96	48.23	30.17	5.15
R52	19.62	22.8	30.71	38.16	51.32	30.13	4.56
L53	20.84	22.32	30.56	40.26	53.15	30.05	5.33
R53	19.33	22.21	29.95	38.24	46.87	29.61	4.72
L54	19.93	22.48	31.04	39.65	47.59	30.17	5.20
R54	21.28	23.09	31.42	39.7	55.3	30.63	5.10
L55	20.7	22.2	28.32	41.12	49.11	29.35	5.66
R55	20.35	22.98	31.14	40.43	49.18	30.74	5.59
L56	20.08	22.55	28.5	40.33	51.17	29.49	5.54
R56	21.28	23.14	31.58	41.72	57.97	31.21	5.99
L57	20.33	22.51	28.36	40.5	51.32	29.53	5.53
R57	22.01	23.3	31.82	41.82	49.76	31.29	5.80
L58	21.17	22.4	30.11	40.48	48.87	29.83	5.18
R58	20.13	21.36	30.87	42.18	50.26	30.49	6.27
L59	17.22	21.7	29.52	39.23	51.82	29.04	5.09
R59	19.28	20.71	30.73	44.47	62.93	30.45	6.78
L60	20.67	22.41	30.89	44.11	50.98	30.57	6.20
R60	20.93	22.07	30.78	43.85	65.2	30.27	6.59

Appendix D. NGSIM Error Distribution. This shows the prevalence of errors in each of the six subsets. Negative distance headway means a vehicle overtook its leading vehicle's position (an indication of a crash or, more likely, erroneous positioning). Any acceleration over 3.05 m/s^2 is considered abnormally high and therefore counted as a high acceleration. Low, constant velocities count as the number of vehicles featuring a non-zero velocity at or below 1.52 m/s coupled with an acceleration of zero.

Subset	**Negative Distance Headway**		**High Acceleration (Entries)**	**Low, Constant Velocity (Vehicles)**
	Vehicles	**Entries**		
I-80A	205 (9.990%)	3,474 (0.275%)	105,293 (8.340%)	8 (0.390%)
I-80B	319 (17.357%)	10,632 (0.686%)	69,244 (4.468%)	93 (5.065%)
I-80C	353 (19.721%)	11,650 (0.664%)	78,189 (4.458%)	194 (10.838%)
US-101A	120 (5.533%)	1,912 (0.162%)	70,785 (5.996%)	13 (0.599%)
US-101B	139 (6.891%)	1,800 (0.128%)	56,347 (4.016%)	37 (1.834%)
US-101C	102 (5.326%)	1,968 (0.130%)	60,783 (4.011%)	44 (2.300%)