Robust Predictive Cruise Control for Commercial Vehicles[*]

Jaime Junell
Oregon State University
junellja@gmail.com

Kagan Tumer
Oregon State University
kagan.tumer@oregonstate.edu

Abstract

In this paper we explore learning based predictive cruise control and the impact of this technology on increasing fuel efficiency for commercial trucks. Traditional cruise control is wasteful when maintaining a constant velocity over rolling hills. Predictive cruise control is able to look ahead at future road conditions and solve for a cost effective course of action. Model based controllers have been implemented in this field but cannot accommodate the many complexities of a dynamic environment which includes changing road and vehicle conditions. In this work, we focus on incorporating a learner into an already successful model based predictive cruise controller in order to improve its performance. We explore back propagating neural networks to predict future errors then take actions to prevent said errors from occurring. The results show that this approach improves the model based predictive cruise control by up to 60% under certain conditions. In addition, we explore the benefits of classifier ensembles to further improve the gains due to intelligent cruise control.

1 Introduction

Improving the fuel efficiency of commercial vehicles not only provides an environmental benefit, but also an economic one. Most work in this area focuses on directly designing vehicles with better fuel efficiency. But a secondary approach is to improve the fuel efficiency of a given vehicle by simply *driving it more efficiently*. Driving at lower speeds on the highway, braking smoothly and accelerating slowly are all known methods for saving fuel. However, it is impractical for a driver to determine the optimal driving speed given a schedule, terrain and road incline. Providing a learning algorithm to predict the best speed at any given time would significantly improve fuel consumption in commercial vehicles.

Cruise control, now standard or optional in most cars, helps improve fuel efficiency on long flat stretches, but when the vehicle approaches an incline, the amount of fuel needed to maintain a constant speed is much greater. In this scenario, conventional cruise control can be very wasteful. One way to improve upon conventional cruise control is to use knowledge of the vehicles environment to adjust the vehicles speed for optimal fuel efficiency. If an incline is approaching then the vehicle can speed up before the hill so that it does not have to put so much gas into ascending that incline. This kind of technology is called Predictive Cruise Control (PCC) [16]. Predictive cruise control utilizes Global Positioning Systems (GPS) to obtain the changes in road elevation and curvature along the intended path of travel. Using this knowledge, it is possible to calculate the best velocity curve to optimize factors such as fuel economy and cost of time.

[*]Appears in the *International Journal of General Systems*, 42(7):776–792, 2013.

One of the largest industries to feel the fuel price increase is the trucking industry. Semi-Trucks have notoriously poor fuel efficiency due to their large size and weight. Although fuel rates are a big factor to trucking companies and independent truckers alike, a driver still wants to get to his/her destination in a timely manner. One of the challenges in optimizing fuel efficiency is finding a good balance between fuel consumption, and scheduling constraints. Given a good model of the vehicle, its environment, and an objective function, an optimal solution can be found [16]. However, obtaining an accurate model that includes both truck dynamics, full path trajectory, detailed terrain information road conditions is difficult at best, and impossible most of the time.

Traditional control methods are not often robust to inaccuracies in the model or unaccounted for sources of noise [29]. In the semi-truck example, such inaccuracies could be a result of miscalculated weight of the truck, slightly flat tires, wet or icy road conditions, high winds, and many more uncontrollable or unaccounted for factors. Adaptive control methods are much more robust to unknown functions [23]. As long as these unknown factors produce a continuous function, then a learning algorithm can find this trend despite not knowing the origin of the error [1]. This quality makes learning controls ideal for a scenario in the variable highway environment.

The contribution of this paper is to improve upon a model based control system by integrating a learning control method into the system. The adaptive part of the control improves the performance of the controller by accounting for imperfections in the model that cause error. This improvement will ultimately reduce errors in PCC, resulting in:

1. An increase in the controllers ability to reduce fuel consumption
2. A potential increase in the recalculation distance, thus reducing the processing power required.

In this paper, we explore the use of neural network learning as applied to predictive cruise control. Section 2 provides background information on recent works related to PCC, optimal fuel control, and the challenges in creating accurate models in environments of high uncertainty. Section 3 defines the problem and introduces the learning approach. Section 4, presents the "single concept" approaches, where only information from either the desired velocity curve or elevation data is used. Section 5 presents multi-concept approaches where information from different sources is combined. Finally, Section 6 provides a conclusion and discussed possible future directions.

2 Related Research

Many of the advances in vehicle technologies are to improve the comfort of driving or energy efficiency of the vehicle while maintaining or enhancing safety features. One of the more significant inventions is Adaptive Cruise Control (ACC). Traditional cruise control is a very common feature in modern vehicles and gives the driver the ability to take his/her foot off the pedal and still move at a constant speed that is inputted by the driver. Adaptive Cruise Control extends this concept to automatically slow down when approaching a car directly in front of it. The vehicle with ACC will then maintain a safe following distance from the preceding vehicle [15, 22, 34]. Adaptive Cruise Control started with just control over the accelerator, but more current models can also control the brakes, therefore being able to avoid rear end collisions without any input from the driver [15, 25]. In addition, studies show that ACC can reduce traffic congestion [4].

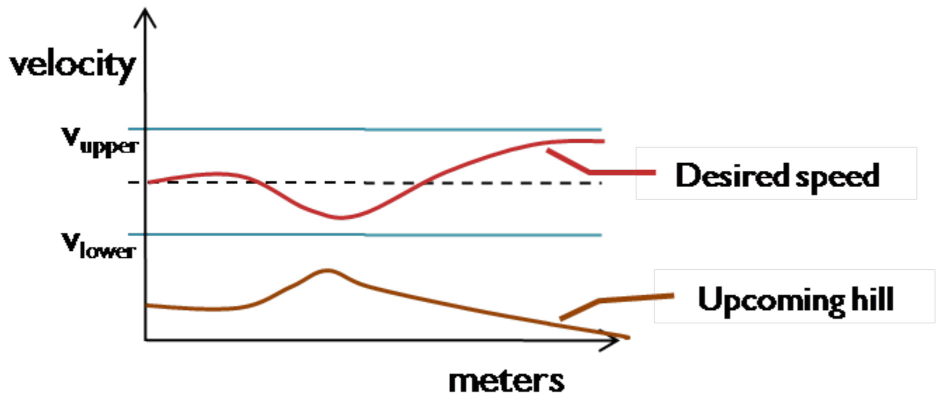

Figure 1: Predictive Cruise Control main concept: Given specified speed constraints, PCC system will solve for the optimal velocity profile for several meters ahead of the vehicle's current position.

The next most noteworthy technology being researched is predictive cruise control (PCC). Traditional cruise control is able to increase gas efficiency on long flat highways, while ACC saves fuel in higher traffic scenarios. Adding PCC to the control system of the vehicle will help increase fuel efficiency by saving gas while driving through variable road conditions. Predictive cruise control is still in the early years of research [16]. A study into "Look Ahead Control to minimize fuel consumption" was conducted in Sweden in 2007 and utilizes a complex vehicle and road model within a control system using several numerical methods. Results of the study found a successful decrease in fuel usage [9].

Predictive Cruise Control uses knowledge of road conditions further along the intended path so that the vehicle can take actions that will decrease the total fuel needed for not only the present time, but also for several moments in the future. In traditional cruise control, a hill can be approaching but the vehicle will continue to go the same speed. In PCC, the fuel optimal action to take may be to increase speed before the hill and decrease speed during the climb up knowing that once the peak of the incline is reached, gravity may help to alleviate the energy required from the engine. Figure 1, shows the basic concept of PCC. In the figure, lower and upper bounds for the velocity are shown. Having these boundaries prevent the vehicle from driving too fast and getting a speeding ticket, and driving too slow to where it doesn't meet the time restraints.

Although the models used in the current PCC algorithms are complex and proprietary, the basic concepts are simple. Figure 2 shows the basic forces in play, which are: drag force, force associated with the incline of the road, the force applied by the brake, and the force that results from the energy spent by the engine. Given these forces, the acceleration of the vehicle can be determined by Equation 1.

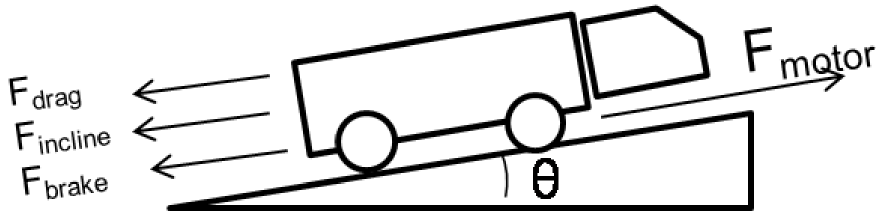

Figure 2: Forces on a vehicle: Several forces affect the vehicle as it moves

$$\sum F = m\frac{dv}{dt} = F_{motor} - F_{drag} - F_{incline} - F_{brake} \tag{1}$$

The force required by the motor increases as the drag force, incline, brake force and acceleration of the vehicle increase. Brake force is assumed zero due to a built-in safety condition that turns off cruise control when the driver taps the service brakes. Optimizing this equation, heavily depends on approximations, as each term has uncertainty that cannot be eliminated (drag depends on the geometry of the truck, mass varies with load, engine efficiency depends on many terms). Therefore, though good models can be found, no model captures the full physics of the problem.

3 Problem Description and Approach

Although predictive cruise control has already had fairly successful results, we believe that applying a learning algorithm to the control system will make the results more accurate in highly variable conditions. A learning environment will account for several variables not easily modeled in traditional control systems. This could result in a greater increase in fuel efficiency.

The problem at hand is that there already exists a model based predictive cruise control algorithm that outputs a desired velocity which is not obtainable by the truck. This problem is demonstrated in Figure 3. The solid light grey line is the desired velocity, v_{des}, that represents the velocity at which PCC wants the truck to travel. The dashed dark grey line is the actual velocity, v_{act}, that the truck is obtaining.

Rather than learn the entire objective function with the learning controller, we can implement the learning into the system *with* the PCC algorithm. If the offset between the PCC desired velocity and the actual realized velocity by the vehicle were known, then it would be possible to correct the desired velocity to values that the truck can reach. This would reduce the frequency at which the desired velocity curve needs to be recalculated, thus saving computational time and fuel. Therefore, we want the learner to learn the offset in order to make a correction to the system. This proposed solution is demonstrated in the block diagram in Figure 4, where the learner can use information from both the GPS and the desired velocity curve outputted from the PCC. Using that information, it will have to accurately output an offset for each time step.

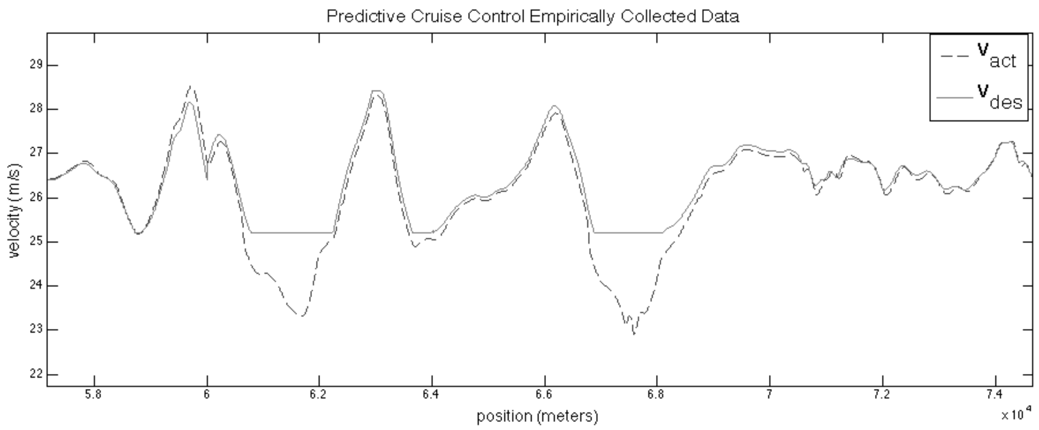

Figure 3: PCC error offset: The solid light grey line labeled v_{des}, is the desired velocity curve that is outputted from the PCC algorithm. The dashed dark grey line labeled v_{act}, is the actual velocity that the truck travels. The difference between the two is the *offset*.

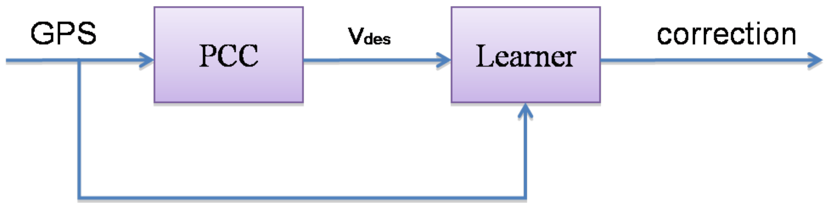

Figure 4: Proposed Solution to the PCC problem: Given the model based PCC, we propose to implement a learning based control method in order to predict the offset between the desired velocity from PCC and the actual velocity the vehicle travels. If we know what the error is going to be ahead of time, then we may be able to correct the problem or adjust the desired velocity in order to minimize the number of times PCC needs to recalculate.

3.1 State Representation and Input/Output Mapping

In this work, we use a one-hidden layer feed forward neural network with sigmoidal activation functions to perform the input/output mapping to improve PCC [1, 20, 23]. The data used in training is split into a training set and a test set using two different methods. As displayed in Figure 5, the first method is a segregated approach where the training set consists of the first third of the data and the test set is the other two thirds of the data. This method is used first because it is all around easy to implement in the code. However, as seen in the figure, the training set does not seem to represent the total data set very well. For this approach to work, the training and test sets need to come from the same distribution. The second method shown in Figure 6 is an integrated approach. One third of the data is still used for training but it is evenly spaced throughout the entire data set, so that the training is more representative of the test set.

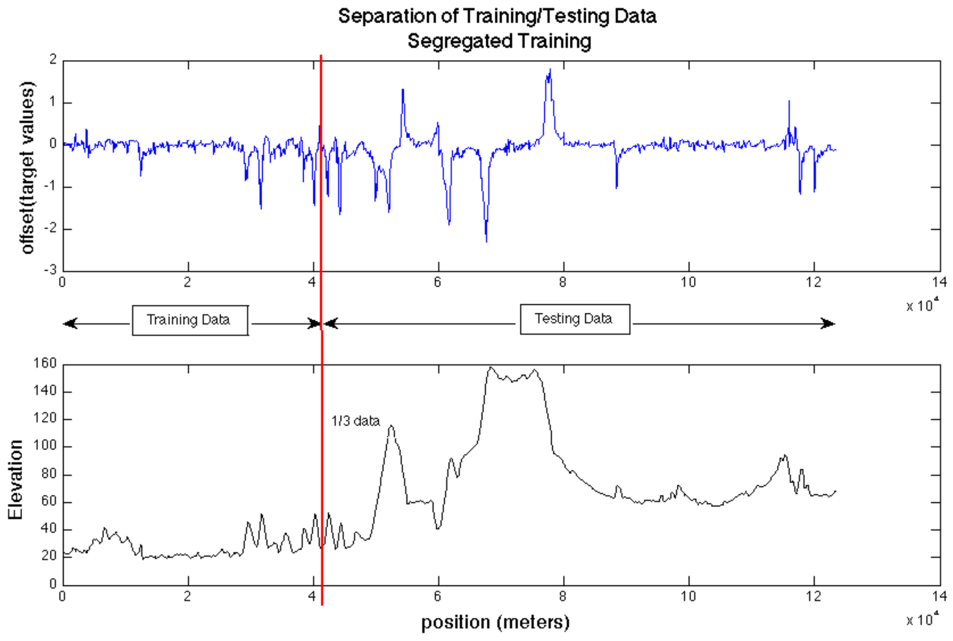

Figure 5: Separation of Training and Testing set using segregation: The first method of separating training and testing data is to take the first third of the data for training and the rest will be reserved for testing. This method is very straight forward and easy to implement but may cause large error if terrain in the test set is not similar to anything from the training set.

When an unknown mapping needs to be performed, it is vital to find the most relevant inputs to fit the problem. For example, in this study we are trying to learn the offset between the PCC desired velocity and the actual velocity the vehicle realizes. Factors that we suspect play a major role in the offset are the velocity of the vehicle, the PCC desired velocity, and the grade of the incoming terrain.

Given this prediction and a prior knowledge of how neural networks perform, we chose to train and test the networks based on the 7 different input configurations shown in Figure 8. These are all variants of the desired velocity and elevation curves. The left column shows the elevation based input configurations and the right column shows the desired velocity based

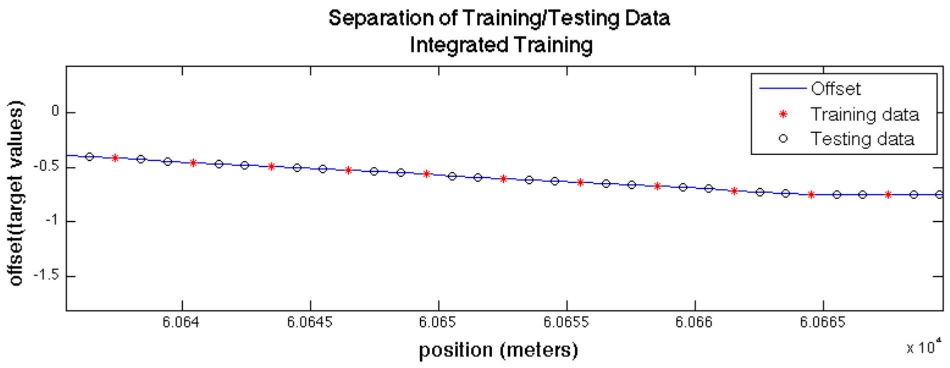

Figure 6: Separation of Training and Testing set using an integrated approach: This method of separating the training data will better represent the entire data set since it is inter-dispersed throughout the data. Every third data point is used as a training point and the rest is used for testing.

input configurations.

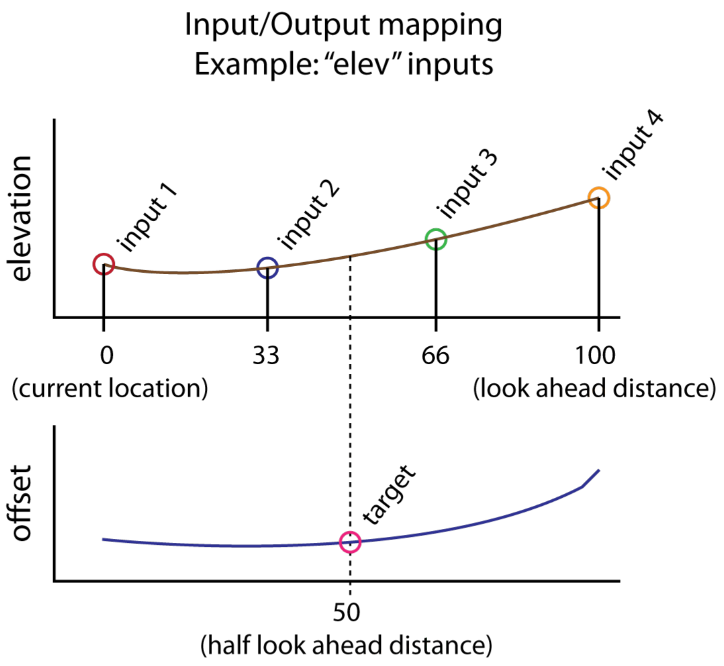

Figure 7: Input/Output mapping for a 100 meters look ahead distance using "elev" input configuration: These plots show that the inputs used in the neural network come from evenly spaced points of the elevation or desired velocity curve. The output used for each sample is the offset at 1/2 the look ahead distance. In this example, the look ahead distance is 100 meters, therefore the inputs will be the elevation at a distance of 0, 33, 66, and 100 meters, while the output will be the offset at a distance of 50m.

Figure 7 shows the setup of the learning problem. In this example, the inputs are from the "elev" configuration. There are four elevations from equally spaced points within the "look ahead window" (i.e. the furthest distance that the neural network is gathering information from). The "look ahead" distance helps to find how much distance is relevant. For example, the elevation curve contains relevant information; however, an elevation 50 miles ahead of the vehicle does not affect the next 50 meters. In this example, there is a look ahead distance of 100 meters. Therefore, the 4 inputs are placed at approximately 0, 33, 66, and 100 meters away from the current vehicle location. The offset we are trying to learn will be half of the look ahead distance. In this case, it is 50 meters away. Now that we know some terminology and the basic set up of the learning problem, we can discuss in more detail the plots in Figure 8.

Starting on the top left and moving down, the "elev" configuration inputs elevations at equally spaced points within the look ahead window. The "grade" configuration inputs the road grade of the road at these same points. In the "slopes" and "bridge" configurations, one more equally spaced point is placed in the look ahead window in order to get the same number of inputs. The inputs for "slopes" are the slopes of the lines that connect the current elevation to the four future elevations. The inputs for "bridge" are the slopes between each neighboring elevation point, like the slopes of a connect-the-dots puzzle. On the right column, the "vdes" configuration inputs the four equally spaced desired velocities within the look ahead window. Do not confuse the "vdes" configuration with the desired velocity curve, v_{des} which is the actual curve while "vdes" is referring to the input configuration mapping. The "vdesslopes" and "vdesbridge" configurations follow the same rules as "slopes" and "bridge" respectively, however they use the desired velocity curve instead of the elevation curve. Note that the desired velocity based inputs underwent a preprocess smoothing procedure. In the data, the PCC model based control system recalculates the desired velocity every 4000 meters. This results in discontinuities in the data where it will abruptly change to meet the actual velocity curve. Since neural networks cannot learn discontinuous functions, it was vital to smooth this data out. To smooth the data, each point in the vdes curve is replaced with an average of the values of itself and the nearest 20 points.

This study starts with 4 inputs with a look ahead distance of 100 meters in examples but will also see the affects of 6, 8, and 10 inputs as well as look ahead distances of 400 and 800 meters. Also to begin with, one type of input for each configuration was used. That is, the inputs are in the same form and derive from either the desired velocity curve or the elevation curve. The results for these "single-concept" neural networks are presented in section 4. Section 5 shows the effects of using multi-concept inputs. That is, the input space is enriched by using both elevation based and desired velocity based inputs in the neural network to see if the addition of information is beneficial to the system. Number of hidden units was set to 20 hidden units for the 4 and 6 input variations and 40 hidden units for the 8 and 10 input variations. The learning rate parameter was optimized within a toolbox.

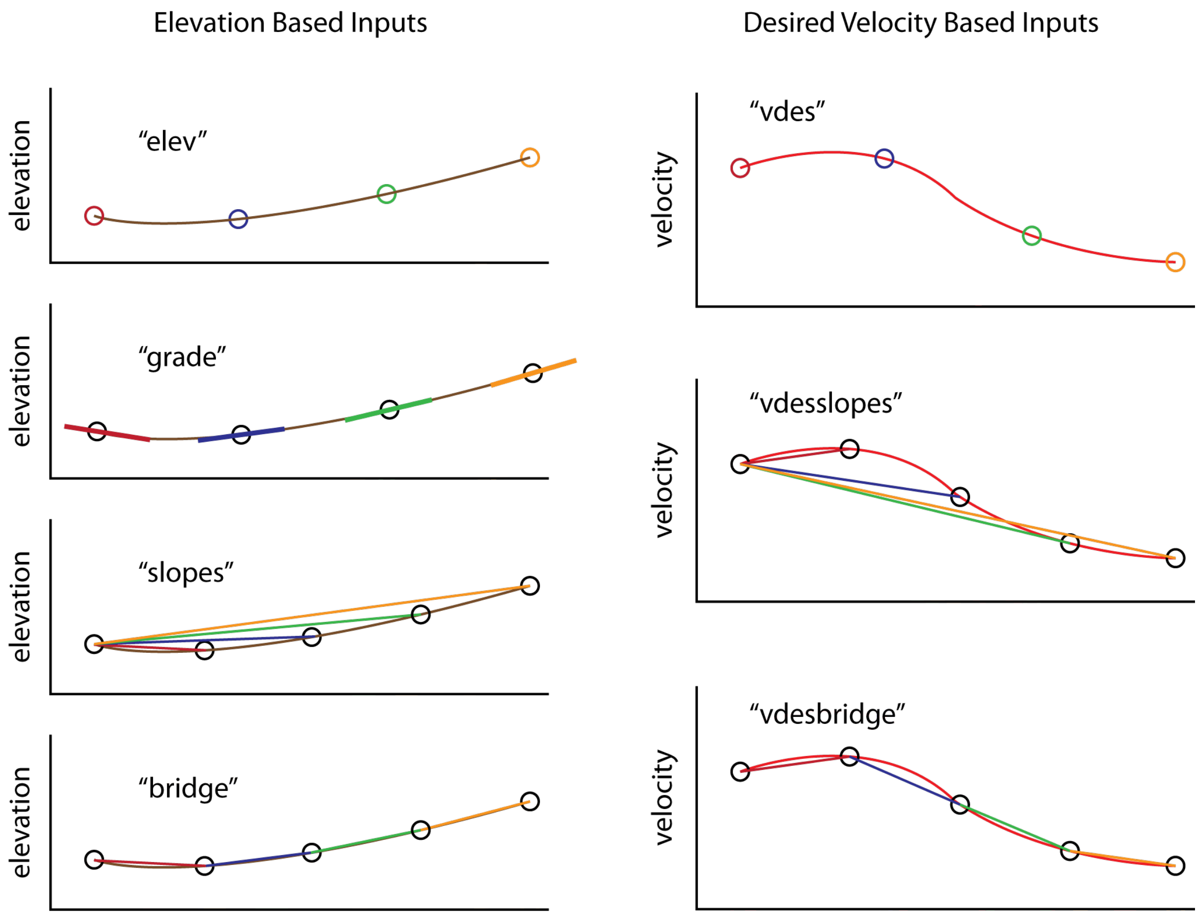

Figure 8: Configurations for neural network inputs: The neural network is exposed to several different configurations in order to see what information is most relevant to our desired target. Configuration **"elev"** inputs elevation values at equally spaced points within the look ahead window. Configuration **"grade"**inputs the road grade at equally spaced points within the look ahead window. Configuration **"slopes"** inputs the slopes from the current elevation to n evenly spaced elevations in the look ahead window. For this example, n, the number of inputs, is 4. Configuration **"bridge"** inputs the slopes between each neighboring equally spaced elevation, like the slopes of a connect-the-dots puzzle. Configuration **"vdes"** inputs the desired velocity values at equally spaced points within the look ahead window. Configurations **"vdesslopes"** and **"vdesbridge"** are calculated in the same way as their elevation based counter parts, except the desired velocity curve is used to extract the input values.

4 Single Concept Inputs

The first results to discuss come from the single concept input configurations that were presented in Figure 8. Over three hundred different single concept variations of neural networks were run to find out which inputs and training methods would perform the best. The varied parameters included:

1. **Configuration:** Seven configurations as described in Figure 8 were used. Representing the data in different ways will give an idea of what aspects of the environment most affect the PCC error.

2. **Number of Inputs:** The number of inputs were varied to see how many inputs were needed to sufficiently represent the system. Fewer inputs are desirable because greater number of inputs will be computationally taxing. We experimented with 4, 6, 8, and 10 inputs.

3. **Look Ahead Distance:** How much of the future terrain affects the offset? Not looking far enough will yield inadequate information, but looking too far will result in extraneous data. For this study, 100, 400, and 800 meter look ahead distances were sampled.

4. **Training Method:** Two training methods were used: Segregated and Integrated training as demonstrated in Figure 5 and Figure 6 respectively.

5. **"No limits" analysis:** "No limits" analysis is another variation of training and testing that has not yet been introduced. This analysis only uses a subset of the data. Results for "no limits" analysis will be presented in Section 4.2.

The single concept results section presents how changing different parameters of the neural network can affect the overall outcome. We discuss how number of inputs and look ahead distances impact the results. Then "no limits" analysis is presented and the neural network performance evaluated.

4.1 Parameter Impact on Performance

Parameters including number of inputs, look ahead distance, configuration, and training method define the system and ultimately determine how the learners will perform. The focus is to find the trends that form from all the parameter variations.

Table 1 shows the results. Each cell represents the average result of 30 runs, training and testing, through the neural network using 250 maximum number of epochs and the corresponding configuration (rows) and number of inputs (columns). Therefore, each table will display 28 different variations (7 configurations $\times$ 4 inputs). During each run, every testing point simulates a prediction of what the actual velocity will be. The offset between the simulated v_{act} and the data v_{act} is calculated for every testing point and averaged. The offset distance without learning is simply the difference at every point between the desired velocity and the actual velocity. These offsets will have units of meters/second since they are velocities. In graphical form it is desired that the simulated v_{act} (black line) be closer to the data's v_{act} (dashed dark grey line) than the PCC's desired velocity, v_{des} (solid light grey line). In the quantifiable form shown in the tables, it is desired for the learners offset to be smaller than PCC's offset.

Table 1: Results for both segregated and integrated training: Each cell represents the percent improvement that the learner has obtained over the "no learning" scenario.

100 look ahead window

	Segregated Training				Integrated Training			
Configuration	4inputs	6inputs	8inputs	10inputs	4inputs	6inputs	8inputs	10inputs
no learning	0.225				0.195			
elev	-353 ± 27	-195 ± 17	-246 ± 24	-355 ± 25	**27.8** ± 0.4	**29.9** ± 0.4	**30.5** ± 0.3	**30.0** ± 0.2
grade	**23.1** ± 0.2	4.8 ± 10	18.3 ± 0.3	17.7 ± 0.4	**28.4** ± 0.2	**28.1** ± 0.2	**30.9** ± 0.2	**30.5** ± 0.2
slopes	**22.0** ± 0.2	15.6 ± 6.1	**22.3** ± 0.2	**21.7** ± 0.2	**23.7** ± 0.1	**24.5** ± 0.2	**25.1** ± 0.3	**25.1** ± 0.3
bridge	-20.3 ± 14.3	16.7 ± 6.1	**22.4** ± 0.2	**22.5** ± 0.2	**24.7** ± 0.1	**24.4** ± 0.2	**25.0** ± 0.2	**24.2** ± 0.5
vdes	**21.8** ± 0.2	**21.6** ± 0.2	**21.2** ± 0.2	14.9 ± 6.0	**23.7** ± 0.1	**23.2** ± 0.1	**24.9** ± 0.1	**24.3** ± 0.1
vdesslopes	4.7 ± 5.7	11.1 ± 0.2	7.0 ± 0.1	7.4 ± 0.1	13.6 ± 0.4	13.5 ± 0.4	12.2 ± 0.2	12.2 ± 0.2
vdesbridge	10.4 ± 0.2	10.2 ± 0.3	6.7 ± 0.3	4.8 ± 0.2	10.2 ± 0.7	5.7 ± 1.0	-1.5 ± 1.0	-92 ± 48

400 look ahead window

	Segregated Training				Integrated Training			
Configuration	4inputs	6inputs	8inputs	10inputs	4inputs	6inputs	8inputs	10inputs
no learning	0.225				0.195			
elev	-222 ± 27	-337 ± 27	-342 ± 25	-281 ± 27	**25.3** ± 0.1	**25.3** ± 0.2	**28.2** ± 0.1	**28.0** ± 0.2
grade	**22.6** ± 0.4	**22.4** ± 0.5	16.5 ± 0.7	11.2 ± 0.6	17.4 ± 0.1	17.7 ± 0.2	**20.7** ± 0.3	**23.9** ± 0.1
slopes	17.4 ± 0.6	12.9 ± 6.0	4.7 ± 0.9	5.5 ± 1.0	15.7 ± 0.1	17.6 ± 0.2	**21.3** ± 0.1	**21.9** ± 0.1
bridge	16.9 ± 0.7	15.2 ± 0.6	7.3 ± 0.5	6.6 ± 0.8	13.5 ± 0.6	17.9 ± 0.2	**21.7** ± 0.2	**22.2** ± 0.2
vdes	19.9 ± 0.3	**20.7** ± 0.4	5.4 ± 0.5	6.7 ± 0.7	16.3 ± 0.2	16.1 ± 0.3	**20.3** ± 0.1	**21.6** ± 0.1
vdesslopes	6.6 ± 0.4	3.7 ± 0.4	-4.5 ± 0.4	-5.5 ± 0.4	6.4 ± 0.3	4.5 ± 0.2	5.4 ± 0.3	5.3 ± 0.2
vdesbridge	5.5 ± 0.4	7.8 ± 0.3	1.6 ± 0.5	-1.6 ± 0.6	4.5 ± 0.3	6.0 ± 0.2	2.0 ± 0.8	3.0 ± 0.3

800 look ahead window

	Segregated Training				Integrated Training			
Configuration	4inputs	6inputs	8inputs	10inputs	4inputs	6inputs	8inputs	10inputs
no learning	0.226				0.195			
elev	-378 ± 25	-215 ± 24	-4.4 ± 16	-340 ± 24	8.4 ± 0.1	7.3 ± 0.2	10.5 ± 0.1	9.3 ± 0.1
grade	**20.3** ± 1.3	**20.9** ± 0.9	7.3 ± 1.5	5.1 ± 1.1	1.6 ± 0.2	3.3 ± 0.1	6.7 ± 0.1	7.0 ± 0.1
slopes	13.6 ± 1.2	13.1 ± 1.4	-6.5 ± 1.4	-10.4 ± 1.3	3.5 ± 0.1	3.7 ± 0.1	6.2 ± 0.1	6.6 ± 0.1
bridge	15.3 ± 1.2	17.3 ± 1.0	-1.6 ± 1.3	-7.0 ± 1.2	2.8 ± 0.1	3.1 ± 0.1	6.2 ± 0.1	6.9 ± 0.1
vdes	3.6 ± 1.5	5.6 ± 2.3	-4.6 ± 2.2	-39.4 ± 1.4	3.0 ± 0.1	3.2 ± 0.1	5.3 ± 0.1	5.4 ± 0.1
vdesslopes	2.6 ± 0.3	-1.4 ± 0.4	-11.5 ± 0.5	-13.6 ± 0.5	-11.8 ± 0.7	-12.1 ± 0.3	-10.7 ± 0.2	-10.3 ± 0.2
vdesbridge	0.3 ± 0.3	-3.3 ± 0.5	-9.8 ± 5.3	-17.5 ± 0.5	-13.7 ± 0.8	-11.7 ± 0.2	-12.1 ± 0.4	-12.3 ± 0.4

The value in each cell is the percent improvement that the learner has obtained over the "no learning" scenario, and is given in percent improvement. All results that exceed 20% improvement are bolded for quick reference. . For Table 1, the "no learning" scenario for a 100 meter look ahead distance and segregated training resulted in an average of 0.225 meters/second over the entire testing set. Any value lower than that means that the learner has improved the system by some positive percentage. Any value greater than that means that the learner made the system worse and will result in a negative percent improvement. The error in the mean, E, is calculated using the standard deviation(σ) and the number of runs(N). Where $E = \frac{\sigma}{\sqrt{N}}$.

The results indicated that a look ahead distance of 400 meters or greater presents the neural network with extraneous information that is not needed to predict the offset and will even hurt the system performance. In addition, the "grade" configuration generally produces the largest improvements, and "vdes" performs the best of the desired velocity based inputs while "vdesslopes" and "vdesbridge" typically do the worse of all the other configurations. Similarly, "slopes" and "bridge" maintain about the same improvement values. As can be predicted elevation alone ("elev") provides good results when the training and test sets are similar (integrated training) and very poor results when they're not (segregated training).

In general though, an input mapping that performs well using segregated training should be preferred over one that only does well on integrated training because that implies it adapts better to new terrain and is therefore a more robust mapping.

4.2 Removing Limit Areas

We also investigated system performance when certain areas of the data were excluded from the training and testing procedure. The areas where the desired velocity curve reaches its minimum allowable speed and flattens out are called lower velocity limits. Much less of an issue are the upper velocity limits where the vehicle has reached its maximum allowable speed. The largest offsets between the PCC desired velocity and the actual velocity occur at the lower velocity limits. These are also the areas where our learners made the largest improvements to the PCC system.

Figure 9 displays several areas where the desired velocity has reached its lower limit. The star markers on the plot represent each position that corresponds to a data point that will be excluded from the training and testing set. Data starts to be excluded before it reaches the limit area due to the look ahead window. The larger the look ahead distance is, the more data will be excluded.

During a "no limits" scenario, it is likely that the vehicle has reached a long incline in the road and is called to decelerate until it reaches the minimum allowed velocity and then hold at that speed. However, the truck is so massive that it is difficult to simply stop decelerating, so it overshoots the target speed. This overshoot cannot be prevented and therefore learning the offset will not lead to better performance on a practical level. This leads to the "no limits" rationale that it will be more beneficial to focus on improving the performance of the other areas more effectively. These other areas are mostly flat, mid-range, and downhill terrain.

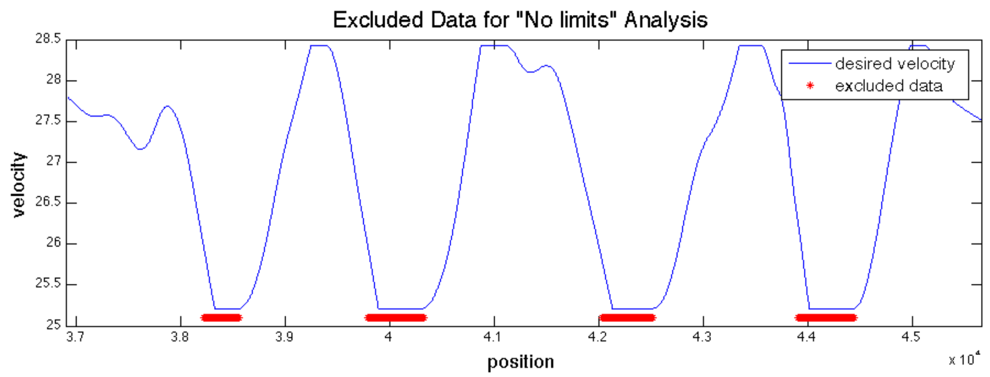

Figure 9: Excluded data from "no limits" analysis: All data involving a desired velocity that has reached its lower velocity limits will be excluded from the training and testing samples.

Table 2 shows the results and leads to the following conclusions:

- Most of the segregated data produces negative improvement which means the function was not learned at all.
- Most of the integrated data produces positive improvement.
- Segregated training generally yields worse results with greater input quantity.

- Integrated training generally yields better results with greater input quantity.
- Segregated training generally yields worse results with greater look ahead distance.
- Integrated training generally yields better results with greater look ahead distance.

Table 2: Results for "no limits" analysis: Each cell represents the percent improvement that the learner has obtained over the "no learning" scenario

100 look ahead window

	Segregated Training				Integrated Training			
Configuration	4inputs	6inputs	8inputs	10inputs	4inputs	6inputs	8inputs	10inputs
no learning	0.147				0.127			
elev	-131 ± 10	-150 ± 13	-146 ± 9.2	-121 ± 7.1	$\mathbf{20.5} \pm 0.6$	$\mathbf{22.6} \pm 0.4$	$\mathbf{25.5} \pm 0.3$	$\mathbf{26.1} \pm 0.5$
grade	-25.1 ± 7.2	-11.8 ± 0.2	-15.7 ± 0.2	-15.7 ± 0.3	16.0 ± 0.2	17.4 ± 0.2	19.8 ± 0.4	$\mathbf{20.0} \pm 0.3$
slopes	-31.4 ± 8.5	-8.8 ± 0.2	-11.0 ± 0.2	-10.2 ± 0.2	10.6 ± 0.1	11.1 ± 0.2	12.0 ± 0.2	11.7 ± 0.2
bridge	-9.0 ± 0.1	-8.1 ± 0.1	-23.8 ± 7.2	-9.2 ± 0.1	10.8 ± 0.2	10.6 ± 0.2	11.0 ± 0.3	10.0 ± 0.5
vdes	-38.2 ± 9.4	-8.5 ± 0.2	-11.1 ± 0.2	-10.9 ± 0.2	3.8 ± 0.1	4.1 ± 0.2	6.9 ± 0.2	6.3 ± 0.2
vdesslopes	-5.7 ± 0.1	-5.4 ± 0.1	-6.9 ± 0.2	-6.9 ± 0.1	1.6 ± 0.1	1.5 ± 0.1	1.4 ± 0.1	1.4 ± 0.1
vdesbridge	-5.3 ± 0.1	-4.9 ± 0.1	-10.8 ± 3.9	-7.1 ± 0.1	1.5 ± 0.1	1.4 ± 0.1	1.5 ± 0.1	1.5 ± 0.1

400 look ahead window

	Segregated Training				Integrated Training			
Configuration	4inputs	6inputs	8inputs	10inputs	4inputs	6inputs	8inputs	10inputs
no learning	0.144				0.124			
elev	-93.5 ± 5.8	-106 ± 4.8	-101 ± 2.1	-123 ± 4.8	$\mathbf{35.7} \pm 0.3$	$\mathbf{37.7} \pm 0.3$	$\mathbf{41.7} \pm 0.2$	$\mathbf{43.5} \pm 0.2$
grade	-7.8 ± 0.3	-7.3 ± 0.2	-14.7 ± 0.3	-13.1 ± 0.6	$\mathbf{25.0} \pm 0.4$	$\mathbf{30.3} \pm 0.3$	$\mathbf{38.8} \pm 0.2$	$\mathbf{39.8} \pm 0.1$
slopes	-10.8 ± 0.2	-35.1 ± 8.1	-16.6 ± 0.4	-17.7 ± 0.4	$\mathbf{20.2} \pm 0.3$	$\mathbf{24.3} \pm 0.3$	$\mathbf{30.9} \pm 0.2$	$\mathbf{32.1} \pm 0.3$
bridge	-111 ± 3.6	-11.7 ± 0.3	-16.9 ± 0.3	-17.5 ± 0.3	$\mathbf{21.2} \pm 0.2$	$\mathbf{24.3} \pm 0.3$	$\mathbf{30.8} \pm 0.3$	$\mathbf{31.1} \pm 0.3$
vdes	-9.6 ± 0.2	-13.6 ± 0.3	-20.9 ± 0.4	-21.6 ± 0.4	14.6 ± 0.3	18.0 ± 0.2	$\mathbf{29.7} \pm 0.2$	$\mathbf{30.7} \pm 0.2$
vdesslopes	-5.5 ± 0.2	-6.1 ± 0.1	-11.7 ± 0.2	-12.3 ± 0.2	3.2 ± 0.1	2.8 ± 0.2	6.6 ± 0.3	6.9 ± 0.2
vdesbridge	-5.7 ± 0.1	-7.0 ± 0.1	-12.1 ± 0.2	-12.2 ± 0.2	3.3 ± 0.1	3.2 ± 0.1	5.8 ± 0.2	6.2 ± 0.2

800 look ahead window

	Segregated Training				Integrated Training			
Configuration	4inputs	6inputs	8inputs	10inputs	4inputs	6inputs	8inputs	10inputs
no learning	0.142				0.122			
elev	-155 ± 11	-179 ± 11	-146 ± 11	-161 ± 11	$\mathbf{44.5} \pm 0.1$	$\mathbf{44.5} \pm 0.2$	$\mathbf{53.1} \pm 0.4$	$\mathbf{51.9} \pm 0.3$
grade	-13.5 ± 0.6	-15.4 ± 0.7	-23.2 ± 0.7	-21.4 ± 0.6	$\mathbf{25.2} \pm 0.4$	$\mathbf{27.9} \pm 0.6$	$\mathbf{49.9} \pm 0.2$	$\mathbf{51.2} \pm 0.1$
slopes	-19.4 ± 0.6	-21.6 ± 0.8	-31.1 ± 1.2	-32.3 ± 1.0	$\mathbf{30.1} \pm 0.2$	$\mathbf{34.7} \pm 0.3$	-23.9 ± 34.1	$\mathbf{45.6} \pm 0.3$
bridge	-19.3 ± 3.4	-21.0 ± 0.5	-36.8 ± 0.7	-43.4 ± 4.6	$\mathbf{29.5} \pm 0.2$	$\mathbf{34.0} \pm 0.2$	$\mathbf{42.5} \pm 0.4$	$\mathbf{44.8} \pm 0.5$
vdes	-13.3 ± 4.5	-10.9 ± 0.5	-19.9 ± 4.4	-32.9 ± 0.7	$\mathbf{29.0} \pm 0.2$	$\mathbf{35.3} \pm 0.2$	$\mathbf{52.1} \pm 0.2$	$\mathbf{54.4} \pm 0.2$
vdesslopes	-11.6 ± 4.7	-7.7 ± 0.3	-21.2 ± 0.5	-22.8 ± 0.6	6.1 ± 0.3	9.9 ± 0.5	-5.6 ± 20	-37.2 ± 33
vdesbridge	-14.3 ± 5.7	-8.0 ± 0.4	-22.0 ± 0.5	-22.4 ± 0.5	4.2 ± 0.4	6.3 ± 0.3	15.7 ± 0.3	$\mathbf{20.5} \pm 0.3$

Segregated training and integrated training provide opposite trends with respect to how number of inputs and look ahead distance affects the performance. These events can be understood using the same explanations as in the previous section. Segregated training only gives a subset of the possible terrains to train on. If that data is learned too well, then it will not test well on new data. However, if it doesn't learn enough, it will also perform poorly. We can tell which extreme is occurring by the trends. By adding more inputs, more information about the elevation or desired velocity curve is presented to the neural network. It is possible that greater amounts of inputs defines too specific of a curve and fewer inputs generalize the terrain better, therefore making it easier to implement on new data. However, in the integrated training examples. More specific is better because it will be tested on data that is very similar to the training data. The same argument can be made for look ahead distance.

5 Multi-Concept Inputs

Although single concept inputs produce good results, in many cases a multi-concept approach can further improve the performance of a learner. In this work, we consider two types of multi-concept inputs:

1. **Enriched Input Space:** This method enriches the input space by including inputs from both the elevation based configurations and the desired velocity based configurations.

2. **Ensemble:** This method combines the results from several of the single-concept neural networks. In this method, not just one input from each input space will be used, but up to 28 different input representations will be incorporated into one solution.

5.1 Enrich Input Space

To enrich the input space we simply add more types of inputs to the neural network. For example, in our single concept 4 input configurations, 1 piece of information is given at 4 points in the future. Now in our new enriched input space, we input 2 pieces of information each at 4 points in the future for a total of 8 inputs. For this study, two types of information are readily available in the form of the GPS acquired elevation curve and the PCC acquired desired velocity curve. Using the 4 input configurations from before, the two types of information are inputted into the neural network together.

Three different input pairings were chosen for testing. Each of these pair an elevation based configuration input with a desired velocity based configuration input. Following are the input pairings and the reason behind its selection:

1. **"vdes & grade":** This pairing was chosen based on their superior performance in the single concept results.

2. **"vdesslopes & slopes":** Although these inputs come from different data, they are in the *same form*, which may help the neural network make the correct connections for learning.

3. **"vdesbridge & bridge":** This pairing was chosen as a variation of a *same form* input type

Table 3 shows results for networks with 40 hidden units. Four inputs of each type were used for a total of 8 inputs to the system. Columns are divided into 100, 400 and 800 look ahead window data. For comparison, the best single concept improvement from a single run is tabulated under the multi-concept configurations. Values are in bold if they exceed the improvement of the best single concept result.

In both basic and "no limits" analysis the multi-concept solutions for segregated training exceeded the best single concept solution only once and by an inconsequential amount. The integrated training runs show that each configuration with 100 or 400 look ahead windows and every combination in "no limits" analysis exceeded the best single concept result. The configuration "vdes & grade" consistently performs the best. This result is worth following, however multi-concept systems are more complex and if it were to be utilized, a cost analysis would need to be conducted to see if it really does perform better than the single concept networks.

Table 3: Enriched input space results with various multi-concept configurations: Each cell represents the percent improvement that the learner has obtained over the "no learning" scenario

Basic analysis

	Segregated Training			Integrated Training		
Configuration	100 look	400 look	800 look	100 look	400 look	800 look
no learning	$0.225 m/s$	0.225	0.225	0.195	0.195	0.195
"vdes & grade"	12.3 ± 1.1	12.8 ± 0.9	5.1 ± 1.1	**55.6** ± 0.1	**34.3** ± 0.04	9.6 ± 0.2
"vdesslopes & slopes"	22.5 ± 0.5	0.2 ± 1.9	-4.5 ± 6.1	**42.2** ± 0.4	**29.3** ± 0.1	9.1 ± 0.1
"vdesbridge & bridge"	**25.9** ± 0.4	9.3 ± 1.4	-7.7 ± 2.0	**40.1** ± 0.4	**29.2** ± 0.1	9.1 ± 0.05
best single concept	23.1 ± 0.2	22.6 ± 0.4	20.9 ± 0.9	30.9 ± 0.2	28.2 ± 0.1	10.5 ± 0.1

"No limits" analysis

	Segregated Training			Integrated Training		
Configuration	100 look	400 look	800 look	100 look	400 look	800 look
no learning	0.147	0.144	0.142	0.127	0.124	0.121
"vdes & grade"	-32.1 ± 0.6	-23.4 ± 0.5	-25.5 ± 0.8	**49.1** ± 0.2	**60.0** ± 0.1	**63.4** ± 0.1
"vdesslopes & slopes"	-23.5 ± 0.3	-26.3 ± 0.5	-47.8 ± 1.2	**32.2** ± 0.3	**47.2** ± 0.6	**57.2** ± 0.5
"vdesbridge & bridge"	-22.7 ± 0.3	-27.6 ± 0.5	-107 ± 0.1	**31.4** ± 0.2	**47.4** ± 0.3	**56.5** ± 0.5
best single concept	-4.9 ± 0.1	-5.5 ± 0.2	-7.7 ± 0.3	26.1 ± 0.5	43.5 ± 0.2	54.4 ± 0.2

5.2 Ensemble Methods

The second approach uses ensembles of neural network. Applied to this study, the ensemble will average the solutions of each system so as to let each system contribute to the ensemble solution [1, 2, 11, 18, 21, 26, 32]. The total number of systems that can be used for the ensemble is 28, given 7 configurations and 4 choices of input quantity. Runs of different training method, and look ahead distance cannot be effectively combined because they do not share common testing samples. For this study, a simple approach was chosen and 16 systems were used in the ensemble. Four configurations, "grade", "slopes", "vdes", and "vdesslopes" were used so that elevation and desired velocity based inputs were represented equally. Consider that each configuration contributes 4 systems corresponding to that configuration analyzed with 4, 6, 8,and 10 inputs.

The ensembles tested are as follows:

- For 100, 400 and 800 look ahead distances
  - A 16 system ensemble using:
    - "grade", "slopes", "vdes", and "vdesslopes" configurations
    - 4, 6, 8, and 10 inputs

Table 4 shows the results for these tests using the basic analysis and "no limits" analysis. The first column of each training method is labeled "no learning" and is the average meters/second offset between the actual velocity and PCC's desired velocity. All other cells represent the percent improvement(%) over the no learning offset. The value is an average over 10 runs and once again the error in the mean is displayed for statistical significance. The 16 system ensemble is shown in comparison to the best run from the single concept networks. These values can be extracted from Table 1 for the basic analysis or Table 2 for "no limits" analysis.

Ensemble methods using segregated training performs the best at 400 meters look ahead distance where the single concept shows a different trend of better performance with smaller

Table 4: Ensemble results with a 16 system ensemble: Each cell represents the percent improvement that the learner has obtained over the "no learning" scenario

Basic analysis

	Segregated Training			Integrated Training		
Configuration	100 look	400 look	800 look	100 look	400 look	800 look
no learning	0.225	0.225	0.225	0.195	0.195	0.195
16 system ensemble	**25.3** $\pm$ 1.7	**31.4** $\pm$ 0.2	18.9 $\pm$ 0.5	**32.9** $\pm$ 0.1	24.4 $\pm$ 0.1	8.8 $\pm$ 0.1
best single concept	23.1 $\pm$ 0.2	22.6 $\pm$ 0.4	20.9 $\pm$ 0.9	30.9 $\pm$ 0.2	28.2 $\pm$ 0.1	10.5 $\pm$ 0.1

"No limits" analysis

	Segregated Training			Integrated Training		
Configuration	100 look	400 look	800 look	100 look	400 look	800 look
no learning	0.147	0.144	0.142	0.127	0.124	0.121
16 system ensemble	**-4.3** $\pm$ 0.2	**-2.0** $\pm$ 0.2	**-3.5** $\pm$ 0.7	17.6 $\pm$ 0.2	36.8 $\pm$ 0.1	46.7 $\pm$ 1.9
best single concept	-4.9 ± 0.1	-5.5 ± 0.2	-7.7 ± 0.3	26.1 $\pm$ 0.5	43.5 $\pm$ 0.2	54.4 $\pm$ 0.2

look ahead distances. Also, for the 400 look ahead window, the ensemble improves upon the best single concept by a substantial amount. Integrated training shows minimal or no improvement upon the single concept results.

In the "no limits" analysis ensemble reaches higher improvement percentages than the single concepts using segregated training but not integrated training. However, these improvements in the segregated training are still worse than no training at all, therefore the result is inconsequential. Comparing the two multi-concept approaches, ensembles perform better with segregated training and enriched input space performs much better than ensemble using integrated training.

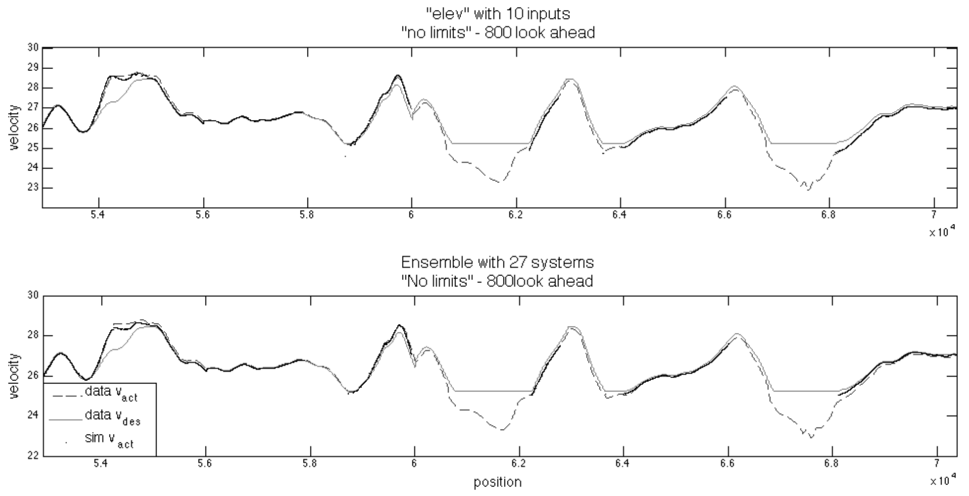

Figure 10: **top:** The best solution found for a single concept neural network. **bottom:** The best ensemble solution found for "no limits" analysis. A comparison between the best solutions from single concept and ensemble methods.

6 Conclusion

In this work, we aimed to determine the input representations that improve the performance of a pre-existing model based controller using predictive cruise control technologies. The biggest challenge in this study was to find the input mapping that best predicts the PCC desired velocity offset from the actual vehicle velocity. We analyzed hundreds of different cases where input parameters such as input quantity, look ahead distance, and configuration as well as training and testing methods such as segregated training, integrated training, and "no limits" analysis.

In applying neural network learning in combination with the PCC model based learner, we showed that the learner can benefit the system by up to 60% improvement. Success is contingent upon several trends that were seen from experimentation with different networks. Perhaps the most interesting results come from the differences between segregated and integrated type training methods. Every trend seems to be opposite for integrated training than it is for segregated training. This makes it difficult to tell what parameters are beneficial. Ultimately, the performance of the parameter is affected by the training method. Integrated training, as expected, is able to get better results because the training data is very similar to the testing data. The segregated training method will identify the more robust input mappings. Overall, the training data that is available and the terrain that the vehicle will be subject to, will determine what network will be chosen.

Furthermore, we also show what does **not** work. Several of the configurations including "elev", "vdesslopes", and "vdesbridge" do not provide useful performance to reach an adequate performance. Multi-Concept solutions were most often not worth the added complexity and computational time. However, there were a few instances that easily bested even the highest single concept performance within the same learning method and look ahead distance categories.

The main contributions of this paper are to:

1. **Improve PCC by implementing learning control:** Improvement in the total system of up to 60% can be achieved by implementing a learner into the Predictive Cruise Control algorithm. This improvement will ultimately reduce errors in PCC, resulting in: 1) An increase in the controllers ability to reduce fuel consumption and 2) A potential increase in the recalculation distance, thus reducing processing power required.

2. **Extract relevant input configurations:** Results reveal which environmental attributes contribute most to the PCC model offset. For example, "grade" and the other elevation based input mappings performed the best in the single concept results. For the multi concept results, "grade" paired with "vdes" configurations performed the best. Knowing which configurations perform well will lead to the development of a more effective predictive cruise controller.

3. **Determine training method and data set best suited for practical implementation:** In analyzing results with different training methods, we show the important role that the training set will play during the practical implementation of this controller. Given what data is available to train on, we can predict how much the learner can benefit the PCC system on a given road. For example, if the training data is very representative of the road the truck will be driving on, then we can predict that the right learner will benefit the system by up to 60%.

Having demonstrated that learning methods are a good fit for this problem, there are several options open for future work in this area. For example, improving the "no limits" analysis will be a major focus, since that's the case where the data is trimmed by the physical capabilities of the vehicles. In addition, analyzing the pay-off/cost of multi-concept learning methods will be valuable. Finally, hierarchical decision making approaches can be used to better used known terrain information.

Acknowledgements

We'd like to thank Derek Rotz at Daimler Trucks NA for his help throughout this project, and in particular for providing the data on which this work is based.

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[2] H.-J. Chang, J. Ghosh, and K. Liano. A macroscopic model of neural ensembles: Learning induced oscillations in a cell assembly. *International Journal of Neural Systems*, 3(2):179–198, 1992.

[3] F.-C. Chen. Back-propagation neural network for nonlinear self-tuning adaptive control. *Intelligent Control, 1989. Proceedings., IEEE International Symposium*, pages 274–279, Sep 1989.

[4] L. C. Davis. Effect of adaptive cruise control systems on traffic flow. *Phys. Rev. E*, 69(6):066110, Jun 2004.

[5] H. Drucker, C. Cortes, L. D. Jackel, Y. LeCun, and V. Vapnik. Boosting and other ensemble methods. *Neural Computation*, 6(6):1289–1301, 1994.

[6] E. Frazzoli, M.A. Dahleh, and E. Feron. Robust hybrid control for autonomous vehicle motion planning. In *Decision and Control, 2000.Proceedings of the 39th IEEE Conference*, volume 1, pages 821–826, Sydney, NSW, Australia, Dec 2000.

[7] P. Gaudiano, E. Zalama, and J.L. Coronado. An unsupervised neural network for low-level control of a wheeled mobile robot: noise resistance, stability, and hardware implementation. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions*, 26(3):485–496, Jun 1996.

[8] S. Hashem. Effects of collinearity on combining neural ensembles. *Connection Science, Special Issue on Combining Artificial Neural Networks: Ensemble Approaches*, 8(3 & 4):315–336, 1996.

[9] Erik Hellström, Maria Ivarsson, Jan Åslund, and Lars Nielsen. Look-ahead control for heavy trucks to minimize trip time and fuel consumption. In *Fifth IFAC Symposium on Advances in Automotive Control*, Monterey, CA, USA, 2007.

[10] Ronaldo R. Torres Jr., Jose' L. Silvino, Paulo F. M. Falmeira, and Julio C. D. de Melo. Control of autonomous robots using genetic algorithms and neural networks. In *IEEE*

International Conference on Emerging Technologies and Factory Automation, pages 151–157, October 1999.

[11] A. Krogh and J. Vedelsby. Neural network ensembles, cross validation and active learning. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems-7*, pages 231–238. M.I.T. Press, 1995.

[12] Kristof Van Laerhoven, Kofi A. Aidoo, and Steven Lowette. Real-time analysis of data from many sensors with neural networks. In *IEEE Fifth International Symposium on Wearable Computers*, page 115, October 2001.

[13] Julien Laumônier, Charles Desjardins, and Brahim Chaib-draa. Cooperative adaptive cruise control: A reinforcement learning approach. In *4th Workshop on Agents in Traffic And Transportation, AAMAS 06*, Hakodate, Japan, May 2006.

[14] Tsang-Wei Lin, Sheue-Ling Hwang, Jau-Ming Su, and Wan-Hui Chen. The effects of in-vehicle tasks and time-gap selection while reclaiming control from adaptive cruise control(acc) with bus simulator. In *Accident Analysis and Prevention*, pages 1164–1170, May 2008.

[15] Greg Marsden, Mike McDonald, and Mark Brackstone. Toward and understanding of adaptive cruise control. *Transportation Research Part C: Emerging Technologies*, 9(1):33–51, 2001.

[16] Konstantin Neiss, Thomas Connolly, and Stephan Terwen. Predictive speed control for a motor vehicle. *US Patent no. 6990401*, January 2006.

[17] Hiroshi Ohno, Toshihiko Suzuki, Keiji Aoki, Arata Takahasi, and Gunji Sugimoto. Neural network control for automatic braking control system. *Neural Netw.*, 7(8):1303–1312, 1994.

[18] D. W. Opitz and J. W. Shavlik. Actively searching for an effective neural network ensemble. *Connection Science, Special Issue on Combining Artificial Neural Networks: Ensemble Approaches*, 8(3 & 4):337–354, 1996.

[19] D. W. Opitz and J. W. Shavlik. Generating accurate and diverse members of a neural-network ensemble. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems-8*, pages 535–541. M.I.T. Press, 1996.

[20] Kevin M. Passino. *Biomimicry for Optimization, Control, and Automation.* Springer-Verlag, London, 2005.

[21] M.P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In R. J. Mammone, editor, *Neural Networks for Speech and Image Processing*, chapter 10. Chapmann-Hall, 1993.

[22] R. Rajamani and C. Zhu. Semi-autonomous adaptive cruise control systems. *IEEE Transactions on Vehicular Technology*, 51(5):1186–1192, 2002.

[23] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Pearson Education, Inc., New Jersey, 2003.

[24] Roland Schulz and Kay Fürstenberg. *Advanced Microsystems for Automotive Applications 2006*. Springer Berlin Heidelbergh, 2006.

[25] Bobbie D. Seppelt and John D. Lee. Making adaptive cruise control (acc) limits visible. *International Journal of Human-Computer Studies*, 65(3):192–205, 2007.

[26] A. J. J. Sharkey. (editor). *Connection Science: Special Issue on Combining Artificial Neural Networks: Ensemble Approaches*, 8(3 & 4), 1996.

[27] P. Sollich and A. Krogh. Learning with ensembles: How overfitting can be useful. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems-8*, pages 190–196. M.I.T. Press, 1996.

[28] N.A. Stanton, M. Young, and B. McCaulder. Drive-by-wire: The case of driver workload and reclaiming control with adaptive cruise control. *Safety Science*, 27(2):149–159, 1997.

[29] K. Tumer and A. Agogino. Robust coordination of a large set of simple rovers. In *Proceedings of the IEEE Aerospace Conference*, Big Sky, MO, March 2006.

[30] K. Tumer, K. D. Bollacker, and J. Ghosh. A mutual information based ensemble method to estimate the bayes error. In C. *et al.* Dagli, editor, *Intelligent Engineering Systems through Artificial Neural Networks*, volume 8, pages 17–22. ASME Press, 1998.

[31] K. Tumer and J. Ghosh. Limits to performance gains in combined neural classifiers. In *Intelligent Engineering Systems through Artificial Neural Networks*, volume 7, pages 419–424, St. Louis, 1995.

[32] K. Tumer and J. Ghosh. Linear and order statistics combiners for pattern classification. In A. J. C. Sharkey, editor, *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*, pages 127–162. Springer-Verlag, London, 1999.

[33] K. Tumer and J. Ghosh. Robust order statistics based ensembles for distributed data mining. In H. Kargupta and P. Chan, editors, *Advances in Distributed and Parallel Knowledge Discovery*, pages 185–210. AAAI/MIT Press, 2000.

[34] A. Vahidi and A. Eskandarian. Research advances in intelligent collision avoidance and adaptive cruise control. *IEEE Transactions on Intelligent Transportation Systems*, 4(3):143–153, 2003.

[35] D. H. Wolpert, K. Wheeler, and K. Tumer. General principles of learning-based multi-agent systems. In *Proceedings of the Third International Conference of Autonomous Agents*, pages 77–83, 1999.

[36] D. H. Wolpert, K. Wheeler, and K. Tumer. Collective intelligence for control of distributed dynamical systems. *Europhysics Letters*, 49(6), March 2000.