Vehicle Speed Prediction using Deep Learning

Joe Lemieux
Department of Electrical and Computer Engineering
University of Michigan
Dearborn, Mi USA
jjlemieu@umich.edu

Yuan Ma
Department of Electrical and Computer Engineering
University of Michigan
Dearborn, Mi USA
myuan@umich.edu

***Abstract*— Global optimization of the energy consumption of dual power source vehicles such as hybrid electric vehicles, plug-in hybrid electric vehicles, and plug in fuel cell electric vehicles requires knowledge of the complete route characteristics at the beginning of the trip. One of the main characteristics is the vehicle speed profile across the route. The profile will translate directly into energy requirements for a given vehicle. However, the vehicle speed that a given driver chooses will vary from driver to driver and from time to time, and may be slower, equal to, or faster than the average traffic flow. If the specific driver speed profile can be predicted, the energy usage can be optimized across the route chosen. The purpose of this paper is to research the application of Deep Learning techniques to this problem to identify at the beginning of a drive cycle the driver specific vehicle speed profile for an individual driver repeated drive cycle, which can be used in an optimization algorithm to minimize the amount of fossil fuel energy used during the trip.**

Keywords—Deep Learning, Stacked Auto Encoders, Neural Networks, Traffic Prediction

I. INTRODUCTION

As concerns over global climate change, natural resource depletion, and urban pollution levels increase, governments are legislating lower and lower levels of CO_2 emissions per 100 km. This translates directly into reduced consumption / higher fuel efficiency of fossil fuel burning vehicles. One way to accomplish this goal in a way that meets all drivers' needs (e.g. ability to travel over 100 miles between charges) is to use a dual energy source vehicle, where one source is "clean" such as a battery, and the other is "dirty" such as an internal combustion engine. However, once the "clean" energy is depleted, the "dirty" energy source becomes primary resulting in overall generation of CO_2 that is typically lower than the global optimum minimum CO_2 generation for the specific drive cycle.

If the vehicle control system is able to predict the energy used during the complete trip, and also predict energy usage in subsections of the trip, it can optimize the balance between "clean" and "dirty" energy sources to approach or equal the global optimal minimum CO_2 generation. During a trip, the primary vehicle characteristic that drives energy consumption is vehicle speed. If vehicle speed at small time increments is known (i.e. 1-sec intervals), the energy consumption for a given vehicle can be calculated. However, predicting vehicle speed at the beginning of a trip is difficult as it can be affected by road conditions and driver behavior. Using data such as average speed across the trip route from public data (i.e. TMC broadcast data) could result in a sub-optimal solution if the conditions are changing or the driver does not drive at the same speed the traffic is flowing.

The purpose of this study is to investigate if a deep learning network based on Stacked Autoencoders(SAE) can learn features of a freeway section such that, when these features are used as the input to a tradition Neural Network that learns a particular driver's behavior, can accurately predict the vehicle speed at each point over the drive route.

II. RELATED WORK

Much of the research on Deep Learning Networks such as Stacked Autoencoders (SAE) and Deep Belief Networks (DBN) have focused on image processing. However, some work has been performed on traffic flow prediction. In [1], SAEs were used to predict the flow of traffic using the Caltrans Performance Measurement System (PeMS) database. Although the paper focused on traffic flow, extension to average speed is not difficult. The performance results of SAEs shows good performance on short term prediction, but increasing errors on longer periods. However, this system measured traffic flow on freeway sections, or the average across the sections vs. a profile across the freeway. Information was lost and could not be used for any type of vehicle control.

Another example of an SAE used to define high level features is shown in [3], where unlabeled images are used to learn high level features that could then be input into a classifier, although this step was not performed. This research showed that an SAE can learn features based upon a large set of unlabeled data that can successfully identify images. For example, it developed a feature identifier (top level neuron) that could identify that there was a face in the image with 80.7% accuracy. It could be expected that adding a classifier layer could provide more accurate identification of images with faces.

The DBN is the most common and effective approach among all deep learning models. It is a stack of Restricted Boltzmann Machines (RBM), each having only one hidden layer. The learned units' activations of one RBM are used as the "data" for the next RBM in the stack. Hinton et al. proposed a way to perform fast greedy learning of a DBN, which learns one layer at a time [4].

Huang *et al.* used DBN for traffic flow prediction combined with Multitask Learning [2]. The work of predicting

978-1-4673-7637-2/15/$31.00 ©2015 IEEE

traffic flow contains two steps - feature learning and model learning. The DBN on the bottom of their Deep Architecture was used as the feature learning model. History traffic flow data was fed into the DBN to learn the features. Each layer in the DBN is a process of nonlinear feature transformation. Features learned in the top layer of the DBN are the most representative feature for the modeling the data. Moreover, the output of the DBN was fed into a regression layer to do the prediction. The proposed method was used on two different datasets, PeMS and EESH, to do prediction of the traffic flow on highway and stations. The result of Huang *et al.* shows that their method performed better that traditional methods, such as ARIMA model, NN, and so on, especially for long term and high value traffic flow prediction.

This project uses some of the same concepts in current research – big data input, multiple layer deep learning networks, and unsupervised learning of historical data. However, the goal of current research is to predict average traffic flow, not the vehicle speed of an individual vehicle. In order to use the results of these predictions to develop an optimal powertrain control strategy, a high-fidelity profile of vehicle speed is required. Our project attempts to predict this high-fidelity profile of vehicle speed so an optimal powertrain energy management strategy can be developed that is customer specific.

III. Data Generation

For this project, two kinds of data are needed, one is the historic driver's data, which show the driver's speed profile along in the route. The other is the historical Traffic Message Channel (TMC) data, a technology for delivering traffic and travel information to motor vehicle drivers, are downloaded from a database, which records all the current flow and freeway flow of all TMC sections in Michigan. Based on the historic TMC data and driver, we want to predict the driver's speed profile along one route at the beginning of the trip. To create data, two steps were required: extraction of TMC data from the historical TMC database, and generation of real-world drive cycle data across one route by a specific driver.

A. TMC Data Extraction

A **TMC_data_Query** system was implemented, which can be used to query the complete history traffic flow data in the data repository given a specific route. This system is shown in figure 1. The system contains two subsystems: **Route_TMC_mapping** and **TMC_data_Extraction**. The **Route_TMC_mapping** subsystem is executed first to map the route data to the TMC sections, i.e., to extract a minimum sequence of TMC sections that cover the given route. The output of the first part is fed into the second subsystem **TMC_data_Extraction**. Given a list of TMC codes, **TMC_data_Extraction** is capable of finding all corresponding history data in the data repository and saves these data into output files, which will be used to generate training and testing dataset for the Deep Learning Network.

C++ is the main programming language used here, which gives a huge improvement in the speed of the program over the interpreted Matlab environment. Two open C++ libraries *pugixml* and *boost filesystem* were used in this system to ensure the robustness and speed of program. The processing speed of **TMC_data_Query** is 2 seconds per file.

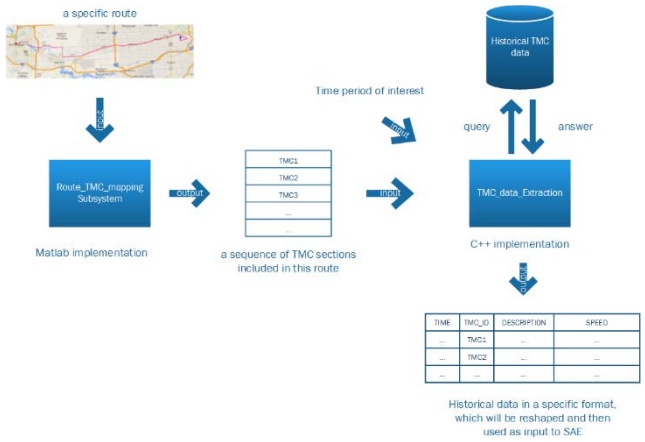


Fig. 1. – TMC Data Extraction

B. Drive Cycle Generation and Extraction

To generate the driver specific data, vehicles were instrumented with GPS data logging systems and information was logged for every trip the driver took. Data collected included instantaneous latitude, longitude, speed, altitude, heading, time since beginning of trip, and date and time at start of trip. This research used the latitude, longitude, speed, and date and time at start of trip information only. Altitude information was not used for this study, but could be used in the future. Over 700 trips were logged for one driver. Processing of the trip data indicated that this driver had multiple repeated routes during this time. The route we investigated is shown in figure 2.

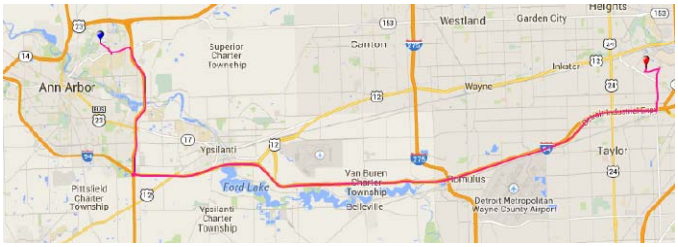


Fig. 2. Route used: Ann Arbor to Dearborn.

For this study, we chose roadway sections with TMC data as inputs to our network. Sections of this trip that did not include TMC data such as neighborhood and private roads were not included.

To obtain the necessary speed profile resolution, the route R was broken into a set of point called Standard Points SP as described in [5]. The trip is therefore defined as:

$$R \triangleq \{SP_0, SP_1, SP_2, \ldots SP_l\} \quad (1)$$

where l is the number of standard points on the route. The velocity profile V is defined as:

$$V \triangleq \{DV_0, DV_1, DV_2 \ldots DV_l\} \quad (2)$$

Where DV_i is the velocity of the individual driver at the standard point i. The vectors V were extracted from the raw route data and used as the target data for teaching the networks. From the data we were provided, there were 21 trips on this route.

IV. Network Definition and Development

Our target Deep Learning Network is constructed of two parts as shown in Figure 3.

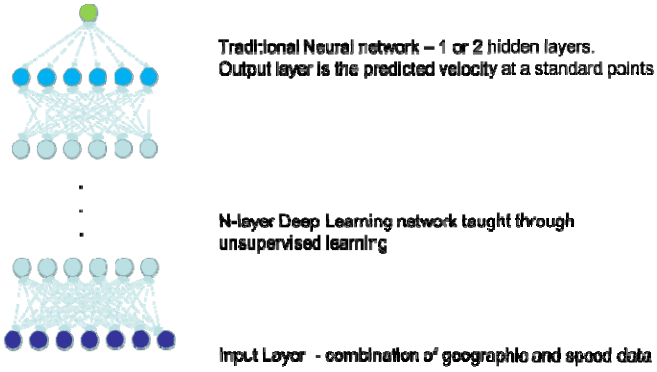


Fig. 3. Network Architecture

This network was chosen instead of using a traditional neural network based upon previous work that showed that deep learning networks can learn features that traditional neural networks cannot. For a repeated drive cycle, our hypothesis is that features such as average speed, slowing traffic, etc. can be predicted and used to predict the speeds at the standard points for a given driver.

The first part is a deep learning network which includes the input layer and the N layers above it. In our work, this will be implemented as an SAE with the number of layers and number of units per layer varied through our experiments. The Input Layer consists of geographic data and speed data for a given Standard Point SP_n. The speed data is a combination of TMC speed data and driver specific speed data.

The second part of the network is a prediction layer for a given standard point SP_n, which is implemented with a neural network with one hidden layer that uses the output of the Stacked Autoencoder as the input. The output is the predicted driver specific speed at the standard point.

The input to the SAE network consists of three components:

1. Road Specific Geometric Data
2. Temporal and Spatial TMC Data
3. Driver Specific Speed Data

A. Road Specific Geometric Data

The Road Specific Geometric Data consists of data that is specific to the roadway at the standard point SP_n. The data is based upon the data at the closest Shape Points. Shape points are geometric locations that are differentiated based on the change of road curvature. The data at each SP_n is:

- Relative distance from standard point to upstream shape point - $D_{SP}(SP_n)$
- Curvature at the standard point - $\delta(SP_n)$
- Altitude at the standard point - $Alt(SP_n)$
- Number of lanes at the standard point - $l(SP_n)$
- Speed limit at the standard point - $V_{lim}(SP_n)$

For our research, we used the Geometric Data for the current Standard Point, and "looked forward" to the upcoming n standard points, where n varied from 0 to 5.

B. Temporal and Spatial TMC Data

The SAE uses Traffic Message Channel (TMC) data that is based upon the instantaneous speed along the drive route at the start of the trip. For each Standard Point n, we used a window of k TMC points before and after the current point. In addition, we used the m previous samples of TMC data in the same window. The SAE input for TMC data is therefore:

$$Input \triangleq \{TM_{-k}^{0}, \dots TM_{-1}^{0}, TM_{0}^{0}, TM_{1}^{0}, \dots TM_{k}^{0}, TM_{-k}^{-1}, \dots TM_{-1}^{-1}, TM_{0}^{-1}, TM_{1}^{-1}, , \dots TM_{k}^{-m}\}$$

C. Driver Specific Speed Data

The final set of data that is input to the SAE is the actual driver data for the r previous Standard Points before the current Standard Point. The current Standard Point is not used as an input to the SAE – it is the target value. The final SAE Input is then:

$$Input \triangleq \{V_{SP-1}, V_{SP-2}, V_{SP-3}, \dots V_{SP-r}\}$$

V. Data Evaluation Method

The data for the route above was used as input to a Stacked Auto Encoder deep learning network. In the data obtained, there were 16 instances of the driver using the above route. Depending upon the values for the above parameters chosen, the number of data points per instance of the route ranged from 1600 to 2000. For this experiment, we used 13 routes to teach the SAE and 3 routes for evaluation.

The below experiments were performed and a comparison of results was made.

- Look forward j standard points, from 0 to 5, for Geometric Data.
- Ramp k from 1 to 5 and m from 0 to 10 for TMC input data.
- Ramp r from 1 to 10 for the previous driver speed data.
- Vary the number of hidden nodes in both the SAE and the Neural Network.

Results were compared using the Root Mean Squared Error calculation:

$$RMSE = \sqrt{\frac{\sum_{l=1}^{Q}(V(l) - \hat{V}(l))^2}{Q}}$$

This was compared to the RMSE of the trip versus using the following baseline, non-learned data as the prediction of the trip:

1. TMC data along the chosen route at the start of the trip, used directly.
2. Posted speed along the route.

VI. Evaluation Results

After analyzing the results of multiple networks, it was determined that the network with the following parameters resulted in the lowest RMSE for the neural network.

- j = 5
- k =3
- m = 7
- r = 4
- The number of inputs resulted in 90 inputs
- SAE had 3 hidden layers of 81,72,63 nodes respectively
- Prediction Layer had one layer of 10 hidden nodes.

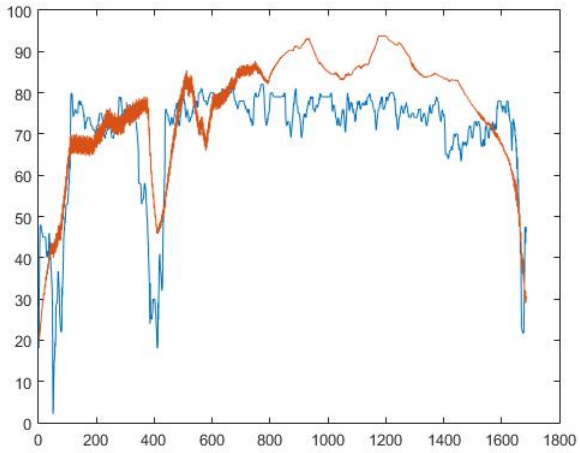
To provide proper prediction, only a subsection of the route was used. The neural network required history data at the beginning of the route, and look forward data at the end of the route. Consequently, of a route of 2040 standard points, 1686 points were predicted. Although not analyzed here, a full route prediction could use posted speed limit, TMC data, or particular driver average speed for the start and ending route speeds.

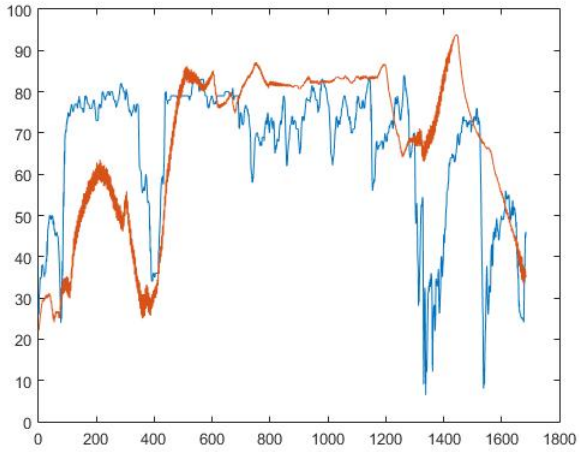
Using the 1686 standard points, the SAE was taught using 13 routes of 1686 points for a total of 21918 training data points. Each layer of the SAE was taught using the Matlab Neural Network Toolbox in the following manner:

1. Create a single hidden layer network of 81 nodes. Input layer and output layer were set to the input data set.
2. Run the neural network using Scaled Conjugate Gradient training for a set number of epochs. Both 25000 and 50000 epochs were used. There was a slight improvement with 50000 epochs, but not much.
3. To train the second layer, the input and output target for training was the activation values of the first hidden layer (the output layer of the first NN was discarded) using the training set.
4. The third hidden layer was trained in the same manner, using as input the activation values of the second hidden layer after the second output layer was discarded.
5. Finally, the prediction layer was trained using the activation values of the third hidden layer using the training set successfully applied through the three layers of the SAE. The target output was the driver's actual velocity at that standard point.

After the weights and biases for all 4 layers of the network were calculated, these were saved and used to analyze the 3 routes set aside for testing. These three routes result in 5058 test points. Testing was performed by applying each route to the saved network, predicting a velocity for the route, applying a 5^{th} order averaging filter to smooth out the prediction noise, and comparing it to the actual driver velocity. The RMSE for each route was then calculated, and the combined RMSE was also calculated.

In the three figures below, the driver actual speed and the predicted speed are graphed for comparison.





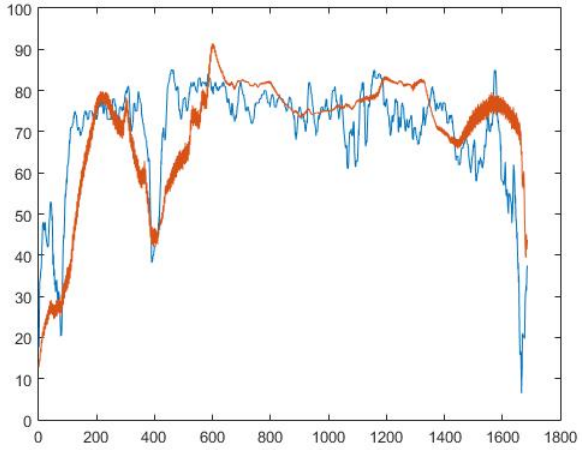


Figure 4 – Actual vs Predicted speeds

The RMSE values calculated and compared to the raw TMC data and the posted vehicle speed are shown in table 1 below.

	NN	TMC	Posted Speed Limit
Sample 1	11.7	14.2	9.4
Sample 2	19.3	17.7	14.7
Sample 3	11.7	13.5	8.6
Combined	14.7	15.2	11.2

Table 1 – RMSE Results

VII. Conclusions and Future Research

Based on the RMSE values shown above, for this driver the best Neural Network improved vehicle speed prediction by about 4% over using raw TMC data for the route. However, if the posted speed was used instead, the result would have been an increased error of about 30% to use the Neural Network. Looking at the plots of the data above, it can be seen that the Neural Network does a fairly good job of following the driver profile, however, tends to overshoot occasionally, especially after the driver accelerates from a traffic slowdown as seen in the second sample – which also results in the highest RMSE value.

This Neural Network, which uses real time route data at the start of the trip, improves accuracy over using the real time data directly, but does not improve on the accuracy if the posted speed limit was used directly. However, due to the limited amount of data and the single driver, the projection of these results for a larger set of drivers over a larger number of instances cannot be made at this time.

This research established a baseline for using deep learning networks to predict individual driver drive profiles over a given route, and eventually to generate a more generalized network for generalized route prediction. Learnings during this research resulted in the creation of future areas of research to create a more accurate prediction of the driver speed, including:

- Analysis using a Deep Belief Network instead of a Stacked Autoencoder.
- Rolling update of route prediction using real time TMC data and driver actual speed as the route is driven.
- Implementation of incident data from TMC as an input to the deep learning network.

References

[1] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic Flow Prediction With Big Data: A Deep Learning Approach," IEEE Trans. Intell. Transp. Syst., vol. PP, no. 99, pp. 1–9, 2014.

[2] W. Huang, G. Song, H. Hong, and K. Xie, "Deep Architecture for Traffic Flow Prediction: Deep Belief Networks With Multitask Learning," IEEE Trans. Intell. Transp. Syst., vol. PP, no. 99, pp. 1–11, 2014.

[3] Q. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Doan, and A. Ng, "Building High-level Features Using Large Scale Unsupervised Learning", Proceedings of the 29th International Conference on Machine Learning, Edinburgh, Scotland, UK 2012.

[4] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," Neural Comput., vol. 18, no. 7, pp. 1527–1554, May, 2006.

[5] J. Park, Y. Murphey, R. McGee, J. Kristinsson, M Kuang, and A. Phillips, "Intelligent Trip Modeling for the Prediction of an Origin-Destination Traveling Speed Profile" , IEEE Transactions on Intelligent Transportation Systems, publication TBD

[6] Palm, Rasmus Berg. "Prediction as a candidate for learning deep hierarchical models of data." Technical University of Denmark, Palm (2012).