A Data Driven Method of Optimizing Feedforward Compensator for Autonomous Vehicle

Tianyu Shi\textsuperscript{1}, Pin Wang\textsuperscript{2}, Ching-Yao Chan\textsuperscript{3}, Chonghao Zou\textsuperscript{1}

\textbf{Abstract}—A reliable controller is critical and essential for the execution of safe and smooth maneuvers of an autonomous vehicle. The controller must be robust to external disturbances, such as road surface, weather, and wind conditions, and so on. It also needs to deal with the internal parametric variations of vehicle sub-systems, including powertrain efficiency, measurement errors, time delay, etc. Moreover, as in most production vehicles, the low-control commands for the engine, brake, and steering systems are delivered through separate electronic control units. These aforementioned factors introduce intransparency and ineffectiveness issues in controller performance. In this paper, we design a feedforward compensator via a data-driven method to model and further optimize the controller’s performance. We apply the Principal Component Analysis (PCA) to the extraction of most influential features. Subsequently, we adopt a Time Delay Neural Network and include the accuracy of the predicted error in a future time horizon. Utilizing the predicted error, we then design a feedforward compensator to improve the control performance. Finally, we demonstrate the effectiveness of the proposed feedforward compensator in simulation scenarios.

\textbf{Keywords} – Controller Performance Analysis, Error Estimation, Neural network

\textbf{1. INTRODUCTION}

In recent years, Autonomous Vehicle (AV) has attracted much attention. A typical autonomous system can feature several main functional layers: perception, decision-making, path planning and control. The control layer guarantees the safety of the vehicle, tracking the desired command input of velocity and yaw rate pairs \((v, \omega)\) precisely. The measured velocity and yaw rate pairs \((v, \omega)\) will also affect the upper layers (e.g. decision and planning layers) and further influence the command inputs \([1]\). As bucket effect reveals, even if we had properly functioned upper layers, the whole performance of the autonomous vehicle will not be optimal without solid low-level control layers. Therefore, learning and modeling the vehicle low level controller’s performance based on real-world driving scenarios is a vitally important issue in the development of an autonomous vehicle system. Currently, most research institutions adopt the autonomous testing platforms with a drive-by-wire system by controlling throttle paddle \(T_t\), brake paddle \(T_b\) and steering wheel angle \(\delta\) to track the desirable waypoints once a trajectory to be followed is defined. The first challenge is that in most production vehicles, the access to the engine, brake, and steering systems is through several electronic control units in low-level control, which means a problem of opaqueness will raise. The second challenge is that it is hard to obtain a precise dynamic model due to the inherent coupling of dynamics of multiple sub-systems and highly non-linearity of the vehicle \([2, 3, 4]\). These difficulties of modeling the vehicle systems in a wholistic manner make it difficult to address safety hazards when the controller encounters uncertainties in operating conditions.

Data driven methods have a great potential for nonlinear system prediction. For instance, Neural Network (NN) has an excellent capacity of mapping complex inputs and outputs. By training a network with a large amount of off-line data, we can obtain the model of a complicated system or process. Some researchers used neural network to optimize the existent controller’s parameter in autonomous vehicle \([5, 6]\). The basic idea of adaptive PID control was based on a neural network and generally used supervised learning to optimize the parameters, however, sometimes the application was limited. For instance, the labeling signal of supervised learning was hard to obtain due to environmental noise. In addition, the optimized result was unstable, which rendered the result inadequate and inappropriate for implementation in the low level controller. Some researchers used neural network in vehicle dynamic modeling. Alexa, O, et al \([7]\), they obtained the data by measuring the dynamic parameters of vehicle, then set the mathematical model to predict the engine torque and vehicle speed using non-linear neural network. However, they did not make it explicit how and why they chose these data for the input nor further validate whether their method could optimize the controller’s performance. Wang, P., et al proposed a lane change control model under continuous action space based on reinforcement learning algorithm, but how the output action was converted to a low-level controller was not addressed in the paper. \([8]\) Unlike previous research, we mainly focus on developing an error mapping model between the designated input command \((v, \omega)\) and actually measured output of vehicle dynamics \((v, \omega)\), based on a data-driven method. By building the input-output model, we predicate errors between input and output, which can be applied to the feedforward compensate optimization. The compensator based on the prediction for future errors will be more reliable and safer for optimizing the motion control of autonomous vehicles since we do not change the basic structure of the drive-by-wire controller.

To summarize, we investigate a data-driven method for modeling the controller’s behavior. The training data were collected from on-board sensors of the vehicle. The methodology is constructed in section II, Principle Components Analysis (PCA) is used to extract key input features from sensor measurements. A Time delay neural network (TDNN) based model was implemented to construct

\textsuperscript{1}Tianyu Shi and Chonghao Zou are with Beijing Institute of Technology, Zhongguancun Street, Haidian District, Beijing, 100081, tianyu.s@outlook.com

\textsuperscript{2}Pin Wang (corresponding author), Ching-Yao Chan are with California PATH, University of California, Berkeley, 1357 South 46th Street, Richmond, CA. 94804, US. {pin_wang, cychan} @berkeley.edu

PATH, University of California, Berkeley, 1357 South 46th Street, Richmond, CA. 94804, US. {pin_wang, cychan} @berkeley.edu
the prediction model. The predicting results were compared with that of BP network, NARX network and LSTM network, based on our analysis for the characteristic of the control system, which is detailed in section III. Then, we applied feedforward compensator based on predicted error in the ‘double lane change scenario’. The simulation and validation results of our model are given in Section IV.

II. METHODOLOGY

Our goal is to derive a mathematical model of a system using observed data. The structure and parameters of our model and simulation scenario are defined based on our prior knowledge from the experiment. Then we will test the predicting and optimizing performance of our model. Finally, the optimal model will be generated, which will be good enough for our purpose.

The architecture of the methodology is depicted in Figure 1.

![Methodology architecture](image)

Figure 1. Methodology architecture

A. Key factors extraction based on PCA

The control errors are sensitive to various variables, such as steering wheel angle, vehicle speed, speed feedback of wheels and so on, which lead to a complicated situation, thus increasing the difficulty of designing a good error compensation model.

PCA uses orthogonal transforms to convert a set of observations of potentially related variables into a set of linear uncorrelated variable values called principal components to extract main influential factors [9]. The goal of PCA is to reduce the number of input features to the model and preserve the amount of information in the data. We describe the core calculation process of PCA in our system below.

In our training data, the number of samples collected from on-board sensors is m (m=5989), the number of features is n (n=16). The input features include steering wheel angle, steering wheel torque, velocity (x, y, z axis), angular velocity (x, y, z axis), acceleration (x, y, z axis), wheel speed (front left, front right, rear left, rear right). The data matrix $X$ is established as follows:

$$X = \begin{bmatrix} x_{11} & \ldots & x_{1n} \\ \vdots & & \vdots \\ x_{n1} & \ldots & x_{mn} \end{bmatrix}$$

(1)

where each row represents a set of experimentally obtained data, and the number of rows is equal to the number of samples we get. The principle is to represent m vectors as points in an n-dimensional space.

To continue, we can perform feature centering by calculating the mean and covariance matrix of the sample of each dimension feature.

The principal component is determined by calculating the contribution rate of different components. Based on principal component analysis, the contribution rates are sorted from high to low, and the principal components corresponding to the top 3 eigenvalues that satisfy our required contribution rate are selected. The formula of the contribution rate $Cr$ is as follows:

$$Cr = \frac{\lambda_i}{\sum_{i=1}^{n} \lambda_i}$$

(2)

The eigenvectors $v = (v_1, v_2, \ldots, v_n)$ corresponding to the eigenvalues $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_n]^T$. Finally, we select the steering wheel angle, steering wheel torque and longitudinal velocity as our main input features to the network. (Detailed computation result can be seen in data analysis of section III)

B. Error estimation based on neural network

The control system for autonomous vehicle is a highly complex hysteresis nonlinear dynamic system. We will elaborate on this aspect further in Section III. By considering the previous and current features of main influential factors into our training inputs, we choose to adopt TDNN [14], which can ensure that the output of our network has previous information. As a result, we can model the hysteresis characteristic of the dynamic system.

![Time delay network structure](image)

Figure 2. Time delay network structure

The prediction model can be established in the following process. The structure of the neural network is represented in Figure 2. The input vectors $[x(t), x(t-T), \ldots, x(t-(m-1)T)]$ are the current and previous features of main influential factors, and each vector includes steering wheel angle, steering wheel torque and longitudinal velocity, which are selected by PCA in the following Data analysis part in section III. The
output \( y(t + \tau) \) represents the predicted error \( \hat{e}(t + \tau) \) between the command input and the actual output in the future. The network has two hidden layers with 8 nodes and 6 nodes, for the first hidden layer, we use the following functions:

\[
x'_j = \sum_{i=1}^{m} w_{ij} x(t - (i - 1)T) - b_j
\]

where \( j = (1,2, \ldots , 8) \) is the number of nodes in the first hidden layer, \( w_{ij} \) is the weight between cell \( i \) and cell \( j \), \( b_j \) is the bias. For the activation function, we use a \( \tanh \) function as follows:

\[
y_j = \tanh(x'_j) = \frac{2}{1 + e^{-2x'_j}} - 1
\]

since the second hidden layer has a similar structure with the first one, we will not repeat it here.

After constructing the TDNN network, we use back propagation learning algorithm to update the weights and biases.

We use the square error function as our loss function:

\[
e(w, b) = \frac{1}{2n} \sum_{i=1}^{n} (y_{mea}(x_i) - y_{(t+\tau)}(x_i))^2
\]

where \( e(w, b) \) is the loss function, \( x_i \) is the sample, \( n \) is the total number of samples, \( y_{mea}(x_i) \) is the measured steering wheel angle error between the input and output, while \( y_{(t+\tau)}(x_i) \) is the predicted error based on neural network. The objective function is:

\[
w^*, b^* = \arg \min_{w,b} e(w, b)
\]

where \( w^* \) is the final weight and \( b^* \) is the final bias.

During the implementation of the algorithm, the learning rate is 0.001 and the final output is the predicted steering wheel angle error between command input and measured output.

C. Feedforward compensator based on learned error model

The compensate process is depicted in Figure 3. The reference trajectory for path tracking is given as the input to the system. Then, the path tracking control (PTC) approach will generate several time series \( u_2(t) \) as command input to the controlling plant (P). As the unavoidable error between the command input \( u(t) \) and the actual output \( \theta(t) \), the TDNN network can learn from the current and previous knowledge of the data acquired from on-board sensors to forecast the future compensation error \( \hat{e}(t + \tau) \) in the next \( (t + \tau) \) time series.

To optimize the control performance of the autonomous vehicle, we have designed a PI controller and a PD controller for different situations. The relationship between \( \hat{e}(t + \tau) \) and \( u_2(t) \) is illustrated in the following equation (7). In the equation, \( T \) is the sampling period, \( T = 0.05s \). \( w_0 \) is the threshold, \( w_0 = 2^\circ/s \). Output \( u_2(t) \) is added into the command input \( u_1(t) \) to generate the compensated input \( u(t) \).

\[
\begin{cases}
u_1(t) = (k_p + k_i) T \frac{T}{z} \hat{e}(t + \tau), & |\gamma| < w_0 \vspace{1mm} \\
u_1(t) = (k_p + k_d) \frac{z - 1}{Tz} \hat{e}(t + \tau), & |\gamma| > w_0
\end{cases}
\]

When the desired yaw rate \( \gamma \) is within the threshold, we can assume the vehicle drive on a straight road and the predicted error \( \hat{e}(t + \tau) \) goes through the PI controller to generate the processed signal \( u_1(t) \). The output of the PI controller will be reset to zero when the predicted error \( \hat{e}(t + \tau) \) crosses zero. This can make the steering control on a straight road more stable and mitigate steering angle oscillation when proceeding on a straight road. When the yaw rate is beyond the threshold, we can assume the vehicle drives on a curved road. The predicted error \( \hat{e}(t + \tau) \) will go through the PD controller, generating the processed signal \( u_1(t) \) to compensate the \( u_2(t) \) to optimize the control stability of the plant. With the help of the PD controller on a curved road, the compensator can generate quick response and enhance the control performance.

III. DATA COLLECTION AND ANALYSIS

A. Data collection

We use Lincoln MKZ, equipped with different sensors and software, to collect real-world driving data, as shown in Figure 4.

![Figure 4. Sensors and computer implemented on testing platform](image)

In the vehicle platform, the drive-by-wire system receives the sensor data over the CAN bus, and transmit the data to the planning layer. The planning layer will generate the desired commands to actuators of the vehicle. The drive-by-wire system contains a steering controller to execute the command of yaw rate. The steering control is implemented with a feedforward proportional controller, and the yaw rate and current speed measurement are used to compute a nominal steering angle based on a kinematic bicycle model. The steering control work flow is shown in Figure 5.
The test was conducted in Richmond Field Station, UC Berkeley. The driving scenarios include straight road, U-turn, and sloped road surface, which correspond to the essential real-world driving conditions typically seen in suburban. With the use of the preview target waypoints for path tracking, the upper layer can generate desired input velocity and yaw rate pairs \((v_d, w_d)\) into the lower level controller.

### B. Data analysis

The experimental vehicle is built on the robot operation system (ROS) [10], with which we can collect data of the vehicle controller performance. We basically record the data from the IMU, GPS and CAN bus. The data elements include: orientation of x, y, z, w, linear acceleration of x, y, z, steering wheel angle command, steering wheel angle output, steering wheel torque and so on. After preprocessing the data collected from sensors, we use MATLAB2018b to analyze the performance of the control system during autonomous driving.

First, we calculated the contribution rate based on PCA as shown in table I. We can select ‘steering wheel angle, steering wheel torque, longitudinal velocity’ as our main input features.

| Type of data read by Sensors | Eigenvalues | Cr  |
|-----------------------------|-------------|-----|
| Steering wheel angle        | 117.383     | 47.80% |
| Velocity (x axis)           | 116.385     | 47.39% |
| Steering wheel torque       | 11.674      | 4.75%  |
| Turning radius              | 0.033       | 0.01%  |
| Linear acceleration (x,y,z axis) | 0.032 | 0.01% |
| Velocity (y,z axis)         | 0.03        | 0.01%  |
| Angular velocity (x,y,z)    | 0.027       | 0.01%  |
| Front right wheel speed     | 0.021       | 0.01%  |
| Front left wheel speed      | 0.004       | 0.00%  |
| Rear right wheel speed      | 0.004       | 0.00%  |
| Rear left wheel speed       | 0.001       | 0.00%  |

As is shown in Figure 6, during the drive on a straight road (steering wheel angle between -0.2 rad to 0.2 rad), the \(RMSE_{straight} = 0.0581\). As for the curved road (absolute value of steering wheel angle more than 0.2 rad) the \(RMSE_{curve} = 0.3218\). We can conclude that the error will be relatively larger in the curved route. In the simulating validation of our algorithm, we then choose to command our test vehicle driving in the ‘double lane change’ scenario [11] to analyze the steering stability of our testing autonomous vehicle on a curved road.

Meanwhile, we find that there are errors between the expected output in the control system and the measurement output of the system. Take the steering wheel angle as an example, as shown in Figure 6, the error (here we use RSME to evaluate) between the steering angle during the whole test is 0.254. It can be seen that the output response always falls behind the command input because of the time delay \(\tau\). To quantify the response time delay \(\tau\), we translated the command input \(\Delta t\) (\(\Delta t = 0.02s, 0.04s, 0.06s, \ldots 0.40s\)) to the right, then measure the RMSE between the command input and measured output, the result is shown in Figure 7. Observing the trend of RMSE in the Figure 7, we can conclude that in \(\Delta t = 0.2s\), the smallest RMSE=0.0713 (28.7% of the RMSE when \(\Delta t = 0s\)) which indicates that the error between the command input and the measured output of the steering wheel angle is the minimum. According to the above analysis, we can quantify the delay time as 0.2 s. After compensating the delay time, we find that 71.3% of the error is due to the steering control time delay.

As is shown in Figure 5, steering control work flow
IV. MODEL TRAINING AND VALIDATION

A. Error estimation model based on naturalistic autonomous driving data

The training process with the neural networks is conducted in matlabR2018b. In the error estimation process, we tested four typical networks, BP network [15], TDNN network, NARX network [16], and LSTM network [17]. BP network is one of the simplest and most basic network which can learn and store a large number of input-output features, and map relationships without prior disclosure of the mathematical equation. The nonlinear autoregressive network with exogenous inputs (NARX) is a kind of recurrent dynamic network, with feedback connections enclosing several layers of the network. The NARX model is based on the linear ARX model, which is commonly used in time-series modeling. The Long short-term memory network are units of the linear ARX model, which is commonly used in time-series modeling. Several layers of the network. The NARX model is based on the linear ARX model, which is commonly used in time-series modeling. The LSTM network has one layer with 150 hidden units. We have trained different types of networks both for ten times, and every time the network starts from random parameters. So that we can get ten different sets of network parameters from the ten trainings and average the prediction results of them. By this way, it can reduce the randomness of the prediction performance in training and get more realistic reflection of network performance. The average predicted results are shown in Figure 9.

![Figure 8. Predicting comparison between different networks](image)

As shown in the Figure 8, each network was trained based on the same input variables: steering wheel angle, steering wheel torque, longitudinal velocity (with 5989 samples range). The other three networks have the same learning rate $\eta = 0.001$. The NARX and BP network both have two hidden layers with 8 nodes and 6 nodes. The LSTM network has one layer with 150 hidden units. We have trained different types of networks both for ten times, and every time the network starts from random parameters. So that we can get ten different sets of network parameters from the ten trainings and average the prediction results of them. By this way, it can reduce the randomness of the prediction performance in training and get more realistic reflection of network performance. The average predicted results are shown in Figure 9.

![Figure 9. Scatter plot of predicted obtained from different models in training and testing phases](image)

As we can see from Figures 8 and 9 the results show that BP network has a weak ability for predicting nonlinear system. As the changing speed of the input signal gets larger, the network’s prediction gets less accurate. The LSTM network will be easily overfitting because the complex structure of the network. Besides, the LSTM network has the longest training time, which is nearly three times longer than the other networks. Therefore, it is not an ideal network to implement in the compensator of the vehicle. NARX and TDNN network have similar predicting results. To further compare the performance of these two networks, we consider two coefficients to evaluate these two networks’ predicting accuracy [12]. First, the correlation coefficient (CC) for evaluating accuracy is defined as:

$$CC = \frac{\sum_{t=1}^{N}[H_{\text{get}}(t) - \bar{H}_{\text{get}}][H_{\text{pre}}(t) - \bar{H}_{\text{pre}}]}{\sqrt{\sum_{t=1}^{N}[H_{\text{pre}}(t) - \bar{H}_{\text{pre}}]^2} \sqrt{\sum_{t=1}^{N}[H_{\text{get}}(t) - \bar{H}_{\text{get}}]^2}}$$  \hspace{1cm} (8)

In the equation 8, the $N$ is the number of the samples, $H_{\text{get}}(t)$ is the get by measuring steering wheel angle error in time $t$, and $H_{\text{pre}}(t)$ is the predicted steering wheel angle error in time $t$. The $\bar{H}_{\text{get}}$ and $\bar{H}_{\text{pre}}$ are the mean values of the measured and predicted data.

Second, the coefficient of efficiency (CE) for evaluating efficiency is defined as:

$$CE = 1 - \frac{\sum_{t=1}^{N}[H_{\text{pre}}(t) - H_{\text{get}}]^2}{\sum_{t=1}^{N}[H_{\text{get}}(t) - \bar{H}_{\text{get}}]^2}$$  \hspace{1cm} (9)

We also extracted the straight and curved road scenario with different number of samples to increase the uncertainty of the training data. The modeling results with the evaluating indexes are shown in the following table II.

| Case       | Samples number | Network | CC   | CE  |
|------------|----------------|---------|------|-----|
| Straight    | 5989           | TDNN    | 0.895 | 0.874 |
|            |                | NARX    | 0.924 | 0.901 |
| Curve      | 5989           | TDNN    | 0.981 | 0.961 |
|            |                | NARX    | 0.921 | 0.912 |
According to our data analysis in section III, the total error in straight-road cases is less than those in curved road. The main reason for the error may be the inaccuracy of the sensors or the disturbance from the environment. NARX network shows good modeling ability for the irregular data. As in curve road, the main contributing factor may be the time delay of the steering control system, the TDNN has good modeling ability for the apparent regular data. Furthermore, with the decrease of training data quantity, the predicting ability of TDNN network decreases. However, the NARX network’s predicting ability remains almost the same.

Given the fact that the low-level control system has the problem of inaccuracy in U-turn control and the TDNN network is good at optimizing U-turn, the TDNN network, compared with NARX, is more suitable in our analysis. The existent control system (without compensated) usually has more steering wheel angle errors during U-turn than during straight route. To solve this problem, we can collect necessary data to ensure the training accuracy by modeling and compensating. In conclusion, the TDNN will be more stable and accurate, which is better for our purpose. So we chose TDNN as the predicting network in our compensator.

B. Model validation in simulation

To validate the data-driven method’s predicting ability, we test our algorithm in simulation. We adopted CarSim as our simulation environment, which can be co-simulated with MATLAB SIMULINK. CarSim is an object-oriented parameter modeling software, which integrate the traditional vehicle dynamics and modern multi-rigid-body dynamics modeling to generate a close-to-reality vehicle model. [13]

As shown in Figure 11 and 12, the steering wheel angle performance becomes more and more smooth and stable, which demonstrates the improved performance of our controller.

|        | TDNN | NARX |
|--------|------|------|
| 0.872  | 0.893| 0.901|

The following Figure 11 and 12 depicted the optimized results compared with the original results in ‘double lane change scenario’.

V. CONCLUSION

In this paper, we propose a data-driven method for modeling and optimizing the controller of autonomous vehicles. From the analysis of the naturalistic driving data, we found out that 71.3% of the steering error is due to the time delay in the steering controller. After comparing the predicting performances of the four typical networks (BP, NARX, LSTM, TDNN), we found that the TDNN network is the most accurate and stable network. With the help of the error compensation from the TDNN model, the overall path tracking performance is improved. The maximum path
tracking error after optimization is improved by 44.4% compared with the original one. The steering wheel angle oscillation is 73.3% of the original result. In conclusion, our method shows its capability in improving the steering stability and control accuracy.

Currently, we have mainly analyzed the improvement for the control algorithm based on a data driven method. For the future work, we will investigate whether this data-driven approach can be applied to the decision-making layer in autonomous driving.

VI. ACKNOWLEDGMENT

The author would thank the support of Berkeley Deep Drive. We are also grateful for the discussion with PhD long Xin, Dr. Chen-Yu Chan, and Prof. Chaoyang Jiang.

REFERENCES

[1] Badue, C., Guidolini, R., Cameiro, R. V., Azevedo, P., Cardoso, V. B, Forecki, A., ... and Oliveira-Santos, T, “Self-Driving Cars: A Survey,” arXiv:1901.04407, 2019.

[2] Silva, L. I., Magallan, G. A., De Angelo, C. H., and Garcia, G. O, “Vehicle dynamics using multi-bond graphs: Four wheel electric vehicle modeling,” An Introduction to Signal Detection and Estimation. New York: Springer-Verlag, pp.2846-2851, 2008.

[3] Will, A., and Zieak, S, “Modelling and control of an automated vehicle,” Vehicle System Dynamics, vol. 27, no. 3, pp.131-155, 1997.

[4] Smith, D. E., Starkey, J. M., and Benton, R. E, “Nonlinear-gain-optimized controller development and evaluation for automated emergency vehicle steering,” In Proceedings of the 2000 IEEE American Control Conference, pp.3586-3591, 2000.

[5] Zhijun Li, Yang Li, Chenguang Yang, and Nan Ding, “Motion control of an autonomous vehicle based on wheeled inverted pendulum using neural-adaptive implicit control,” in IEEE/RSJ International Conference on Intelligent Robots and Systems on. IEEE, 2010, vol.25, pp.133-138.

[6] Alexa, O., Ilie, C. O., VilášU, R., Marinescu, M., and Truta, M, “Using neural networks to modeling vehicle dynamics,” Applied Mechanics and Materials, pp.133-138, 2014.

[7] P. Wang, C. Chan, “A Reinforcement Learning Based Approach for Automated Lane Change Maneuvers,” in 2018 IEEE 29th Intelligent Vehicles Symposium(IV), pp.1379-1384.

[8] Jolliffe, I. T, “Principal Component Analysis,” second edition Springer-Verlag, 2002, pp.10-20.

[9] Double lane-change ISO:https://www.iso.org/obp/ui/#iso:std:67973:en

[10] Yen-Ming Chiang, Ruanan Hao , Nannan Li, and Jian Wang, “Water Level Forecasting by Static and Dynamic Neural Networks,” Journal of Tianjin University (Science and Technology), vol.3, pp. 25-34, 2017

[11] Johansson, R., Williams, D., Berglund, A., and Nuguas, P, “Carsim: a system to visualize written road accident reports as animated 3D scenes,” in Text Meaning and Interpretation. Association for Computational Linguistics workshop. pp.57-64, 2004.

[12] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. J, “Phoneme recognition using time-delay neural networks,” In Readings in speech recognition. pp. 393-404,1990

[13] Hecht-Nielsen, “Theory of the backpropagation neural network.” International Joint Conference on Neural Networks, 2002

[14] R. J. M. P. M., and Barreto, G. A, “Long-term time series prediction with the narx network: an empirical evaluation,” Neurocomputing, vol.16, pp.3335-3343, 2008

[15] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J, “Lstm: a search space odyssey,” IEEE Transactions on Neural Networks and Learning Systems, vol.16, pp. 2222-2232, 2015.