MTS-DeepNet for Lane Change Prediction

Xipeng Wang Yi Lu Murphey
Department of Electrical and Computer Engineering
University of Michigan-Dearborn
Dearborn, MI, USA
{xipengw,yilu}@umich.edu

Dev S. Kochhar
Vehicle Design & Infotronics
Ford Motor Company
Dearborn, MI, USA
dkochhar@ford.com

***Abstract*— Time series data are ubiquitous and are of importance in many application problems in engineering, science, medicine, economics and entertainment. Many real world pattern classification problems involve the processing and analysis of multiple variables in the temporal domain. These types of problems are referred to as Multivariate Time Series (MTS) problems. In many real-world applications, an MTS problem can involve a large number of signals, and require algorithms to select signals and extract temporal and spatial features from them. In this paper, we present an innovative convolutional neural network, MTS-DeepNet that is specially designed for MTS pattern classification. The system integrates signal and feature selections with MTS pattern classification in one learning framework. MTS-DeepNet is applied to a real-world problem, namely predicting driver lane departure based on driver's physiological signals. Our experimental results showed that, in comparison to a multi-layer neural network trained with the backpropagation algorithm, MTS-DeepNet gave better prediction accuracy.**

Keywords—Multivariate Time Series; Lane change prediction; MTS-DeepNet.

I. INTRODUCTION

Time series data are of importance in many application problems in engineering, science, medicine, economics and entertainment. In this paper, we consider time series data as unidimensional signals in the temporal domain. Multivariate time series (MTS) data contain multiple time series signals that are synchronized in the time domain and used together to characterize a given problem. MTS data are common in many multimedia, medical, transportation, mobility and financial applications. Many real world pattern classification problems involve the processing and analysis of MTS data. The application problem addressed in this paper, namely driver lane departure prediction based on physiological signals, is one such example. Multiple physiological signals, including heart rate, skin conductance, and breathing rate, can be collected through wearable sensors and used to predict a driver's intention to change lanes before the lane change actually occurs. Timely warnings could be issued if there is imminent or probable danger in making the lane change. Others have proposed that a driver's intention to change lanes can be made on the basis of vehicle data [28, 29]. However, such a prediction may not be timely enough. The use of driver physiological data, in addition to vehicle and other time domain data may provide a better basis for timely and accurate prediction of lane changes. The existence of a large number of variables may cause many feature selection and machine learning algorithms to be computationally heavy and impractical. It is well recognized that signals selection, and feature extraction and selection are extremely important in MTS classification problems. Traditional pattern classification systems have separate processes for signal selection, feature generation and selection. Many practitioners still rely on professional expertise and repeated trial-and-error processes to select signals and features for a given problem.

Recently, Convolutional Neural Networks (CNNs) have become popular in solving computer vision and pattern classification problems. CNNs have been shown to be effective in feature extraction with image classification [1,5]. CNNs can be viewed as an enhancement of conventional neural networks where the main difference is that conventional neural network uses hard coded filters to generate features while CNNs use convolutional layers to learn features. While CNNs have been popular in solving computer vison problems [1,2,5], it has not been studied much for solving MTS problems. The main reason is that MTS signals have different characteristics than images, which makes applying CNNs to MTS signals rather challenging. Besides the spatial correlations between different signals, MTS signals also contain useful temporal information within each individual signal. In this paper, we propose a novel deep learning algorithm, MTS-DeepNet, for learning how to extract underlying spatial and temporal features from raw MTS signals and classify MTS patterns based on the extracted feature vectors.

This paper is organized as follows: in Section II, we present related works in MTS pattern classification. The MTS-DeepNet framework is explained in detail in Section III. Section IV describes drive lane departure prediction using MTS-DeepNet, and Section V concludes the paper.

II. RELATED WORKS

Massive amounts of time series data are generated daily, in areas as diverse as astronomy, industry, sciences. One obvious problem of analyzing time series data concerns its typically massive size—gigabytes or even terabytes are common, with more and more databases reaching the petabyte scale. The intrinsic structural characteristics of time series data, such as the high dimensionality and feature correlation, combined with the measurement-induced noises makes pattern classification of MTS data challenging. MTS pattern classification algorithms usually fall into the usual three categories:

The research is supported, in part, by a grant from the Ford University Research Program

supervised learning (pattern recognition), unsupervised learning (clustering) and semi-supervised learning (outlier/anomaly detection). Examples of supervised algorithms are Artificial Neural Networks (ANNs) [6-9], Support Vector Machines (SVMs) [10-14], Random Forests [15,16], Extreme learning[17] based on different features extraction methods [18-21], such as Principle Component Analysis (PCA) [7], Singular Value Decomposition (SVD) [8], Autoregressive Moving Average (ARMA) [9,12,22], and so on. Unsupervised algorithms are mostly based on clustering [23,24] and k-nearest neighbors [25-27].

Min & Mingming [7] presented a system with conventional neural network as classifier combined with PCA for feature extraction. PCA is popular in multivariate processing data for identifying important combinations of the original variables. In [7], Min & Mingming used PCA to eliminate the redundant information in complicated and high dimensional input data streams so that useful information can be processed in a low-dimensional space. In another paper, Min & Mingming[8] used SVD to extract feature components in the multivariate time series, because, as they argued, the structure of the embedded time series is complex.

Besides ANN, SVM is also a very popular tool for solving MTS pattern classification. Jeong& Raja [13] proposed two support vector-based supervised learning algorithms, one is called multiclass support vector data description with weighted dynamic time warping kernel function (MSVDD-WDTWK) and the other is called multiclass support vector machines with weighted dynamic time warping kernel function (MSVM-WDTWK), which provides a flexible and robust kernel function for time series classification for non-aligned time series data. Their WDTW kernel function provided an optimal match between two time series data, which not only allows a non-linear mapping between two data sequences, but also considers a relative significance measure derived from the phase difference between points on time series data.

Random Forest classification has also been used in MTS analysis. Liu, Xu, Fang et al. [16] presented an algorithm for selecting a compact subset of relevant signals and then used Random Forrest for pattern classification for solving application problems involving MTS data. The algorithms include a statistical causality modeling algorithm to select relevant signals, and a correlation analysis method to remove redundant signals.

All these supervised learning algorithms require user selecting signals and generating signal features as input to the classifiers. As we know, signal and feature selections are important to a classification system. However, it is not easy to develop effective machine learning algorithms for signal or feature selections. In many applications, signal and feature are selected based on researchers' own experience and knowledge. Signals and features may work well on one application problem but may not well on other problems. The proposed MTS learning algorithm, MTS-DeepNet has the capability of learning signal features within the same learning framework of pattern classification. This makes MTS-DeepNet applicable to a broad range of MTS problems. Empirical experiment results show that the driver lane departure system implemented in the MTS-DeepNet framework outperformed the conventional neural network that uses with statistical features generated by signals selected based on statistical causality analysis.

III. MTS-DeepNet Model

A. MTS-DeepNet Structure

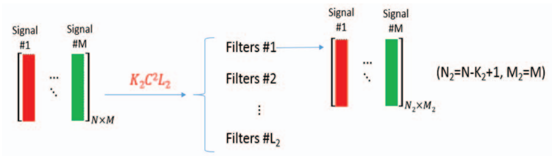
Deep convolutional neural network has been proved to be very useful in dealing with both image classification and speech recognition [3-5]. Because of the largely reduced number of parameters, the training of deep convoluted neural networks are much faster than deep fully-connected networks. The nodes in deep convoluted networks share the weights. For image classification, researchers use 2-D filters to process the image, and then pooling layers are used to down-sampling the convolved image. But this method doesn't work very well on the MTS signals due to the following reasons. First, an image has a unique property, namely, local similarity. The nearby pixels are similar within an area unless they contain different regions of color and/or textures. This property exists in MTS only in the time domain, i.e. in individual signals. Secondly, locally-connected networks cannot extract good spatial domain features because of not knowing what combinations of signals are good features. A better solution is to use a fully-connected network for a group of signals that are acquired simultaneously and at the same sampling rate. The proposed MTS-DeepNet consists of 1-D convolutional filters in the temporal domain and fully-connected networks in the spatial domain.

The network layers are represented as follows:

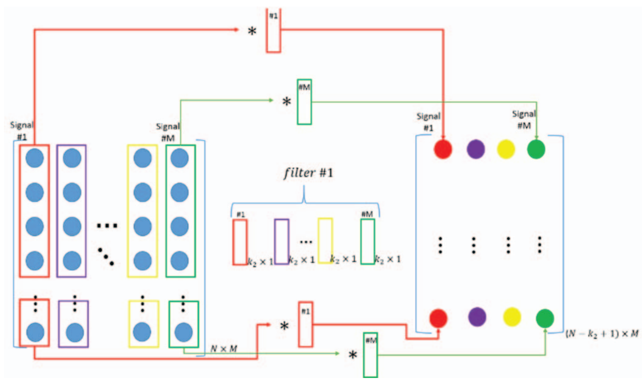
$$\mathrm{N} \times M \rightarrow K_2 C^2 L_2 \rightarrow P_3 S^3 L_3 \rightarrow \cdots \rightarrow K_i C^i L_i$$
$$\rightarrow P_{i+1} S^{i+1} L_{i+1} \rightarrow \cdots \rightarrow K_{n-1} C^{n-1} L_{n-1} \rightarrow P_n S^n L_n \rightarrow h^{n+1}$$

where M is the number of signals and N is the dimension of MTS signals, i.e. the number of samples for each signal in a time period. The network requires all the signals to be synchronized and to have the same sample frequency. C^i means that the layer i is a convolutional layer and that this layer has L_i different filters. Each filter for each signal is a vector in the space $\mathbb{R}^{K_i}$, which means vector size is $K_i * 1$. S^{i+1} means the layer $i+1$ is a down-sampling layer. This layer has P_{i+1} hidden nodes in the auto-encoder network. h^{n+1} is the number of hidden nodes in the final layer, which is a conventional network. L_i in the convolutional layer is always equal to L_{i+1} in the down-sampling layer.

Fig.1 (a) shows the structure of the convolutional layer. In this case, the layer has L_2 different filters for each signal and the filter size is K_2. So after this convolutional layer, the number of features maps will be L_2. Since we do 1-D convolution for each signal in the time domain, the dimension of the feature map will be $N_2 * M_2$, where $N_2 = N - K_2 + 1, M_2 = M$. Fig.1 (b) shows the connection between the input signals and the convolution filters. Filter #1 consists of N different one-dimension filters, which have the same window size $K_2 \times 1$. Each input signal will be convolved with one filter.

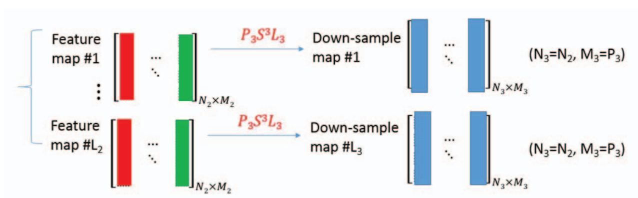


(a) Convolutional Layer General Representation

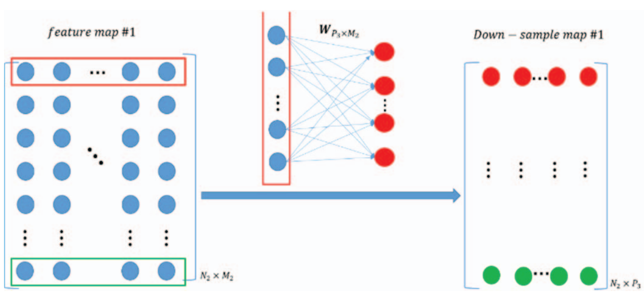


(b) Filter #1 convolution operation

Figure 1 Convolutional Layer Structure



(a) Down-sampling Layer General Representation



(b) Feature Map #1 down-sampling operation

Figure 2 Down-sampling Layer Structure

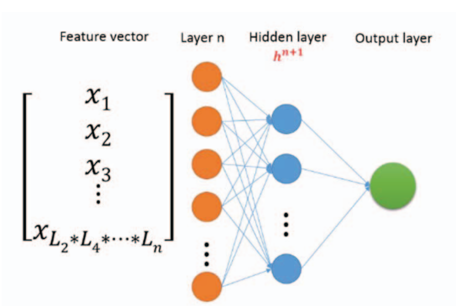


Figure 3 Final Layer Structure

Fig.2 (a) shows the structure of the down-sampling layer. Each filter output has one corresponding down-sampling filter. So after this convolutional layer, the number of features maps will be L_3, where $L_3 = L_2$. Since we use auto-encoder to reduce the dimension of signals, so for each feature map, the dimension of the feature map will be $N_3 * M_3$, where $N_3 = N_2, M_3 = P_3$. The auto-encoder training is only used in the first round of forward propagation in order to get better initial weights of the network. After the first training round, the weights are updated by using the back-propagation algorithm.

Fig.3 shows the structure of the final layer. After several convolutional and down-sampling layers, the dimension of final input feature vector would be in the space $\mathbb{R}^{L_2 * L_4 * \dots * L_n}$. The final hidden layer has h^{n+1} hidden nodes, and the final output layer only has a single node. These three layers form a fully-connected conventional neural network.

B. MTS-DeepNet Forward Propagation

Fig.4 shows the parameters used in the forward-propagation of the convolutional layer. One feature map output of Layer i is represented by a matrix $\boldsymbol{a^i}$ with the dimension $N_i * M_i$. Each column of the matrix $\boldsymbol{a^i}$ can be calculated as in equation (1).

$$\boldsymbol{a_j^i} = f(\boldsymbol{z_j^i}) \quad (1)$$

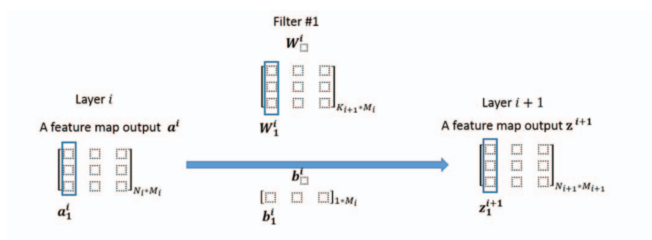


Figure 4 Convolutional Layer Forward Propagation

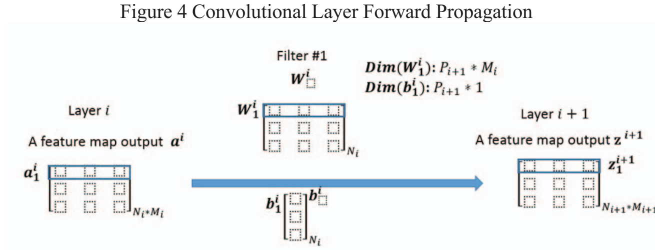


Figure 5 Down-sampling Layer Forward Propagation

The superscript i represents the layer number, and subscript j represents the column number. Function $f()$ is the activation function in this layer. In this study, we are using sigmoid function, as shown in equation (2)

$$\boldsymbol{f(z)} = \frac{\boldsymbol{1}}{\boldsymbol{1+e^{-z}}} \quad (2)$$

The filter weights and bias in layer i are represented by $\boldsymbol{W}$ and $\boldsymbol{b}$. The output of layer $i+1$ is calculated as in equation (3).

$$\boldsymbol{z_j^{i+1}} = \boldsymbol{W_j^i} * \boldsymbol{a_j^i} + \boldsymbol{b_j^i} \quad (3)$$

Operation $*$ is the valid convolution operation. The results of valid convolution are computed without the zero-padded edges. For example, $[a,b,c]*[e,d,f,g]$ is equal to $[a*e+b*d+c*f, a*d+b*f+c*g]$.

Fig.5 shows the parameters used in the forward-propagation of the down-sampling layer. When training this layer, we are using auto-encoder. So this layer is fully-connected of all the signal samples at same time. The output of layer $i+1$ is calculated as equation (4). $\boldsymbol{W_j^i}$ is a matrix with the dimension $P_{i+1} * M_i$, and $\boldsymbol{b_j^i}$ is a matrix with the dimension $\boldsymbol{P_{i+1} * 1}$.

$$\boldsymbol{z_j^{i+1}} = \boldsymbol{W_j^i a_j^i} + \boldsymbol{b_j^i} \ (4)$$

Fig.6 shows the final layer propagation. It is a fully-connected conventional neural network. The output of layer $n+1$ is calculated as in equation (5) (6).

$$\boldsymbol{z^{n+1}} = \boldsymbol{W^n a^n} + \boldsymbol{b^n} \ (5)$$

$$\boldsymbol{a^{n+1}} = \boldsymbol{f}(\boldsymbol{z^{n+1}}) \ (6)$$

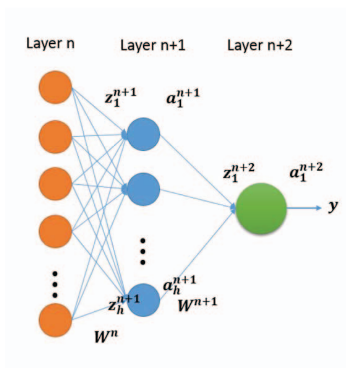


Figure 6 Final Layer Forward Propagation

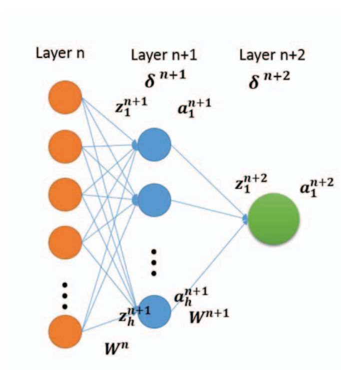


Figure 7 Final Layer Backward Propagation

C. *MTS-DeepNet Backward Propagation*

Suppose we have training samples as $(\boldsymbol{x}_1, \boldsymbol{t}_1), \dots, (\boldsymbol{x}_s, \boldsymbol{t}_s)$, where $\boldsymbol{x}$ is a feature matrix with the dimension $N*M$. The training cost function can be calculated as in equation (7), where $\boldsymbol{W}, \boldsymbol{b}$ are the MTS-DeepNet weights, $\boldsymbol{y}$ is the MTS-DeepNet outputs vector, and $\boldsymbol{t}$ is the training sample truth.

$$J(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{x}, \boldsymbol{t}) = 0.5 * \|\boldsymbol{t} - \boldsymbol{y}\|^2 = 0.5 * \sum_{i=1}^{s} (\boldsymbol{t}_i - \boldsymbol{y}_i)^2 \ (7)$$

The objective of the training is to minimize the cost function. In this paper, we use the gradient decent method to minimize the cost $J(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{x}, \boldsymbol{t})$. Fig.7 shows the parameters used in the final layer propagation. $\boldsymbol{W^i}$ and $\boldsymbol{b^i}$ represent the i layer's weights, and $\boldsymbol{\delta}^{i}$ is the error term at layer i. At the layer n+2, which is also the final output layer, the error term can be calculated as in equation (8).

$$\boldsymbol{\delta^{n+2}} = -(\boldsymbol{t} - \boldsymbol{a^{n+2}}) \cdot \boldsymbol{f'}(\boldsymbol{z^{n+2}}) \ (8)$$

The function f is the activation function in the layer n+2, and the operator $\cdot$ denotes the element-wise product operation.

At the layer n+1, the error term can be calculated as in equation (9)

$$\boldsymbol{\delta^{n+1}} = (\boldsymbol{W^{n+1}})^{\boldsymbol{T}} \boldsymbol{\delta^{n+2}} \cdot \boldsymbol{f'}(\boldsymbol{z^{n+1}}) \ (9)$$

Based on the error term, we can calculate the gradient of the weight by using equations (10) and (11).

$$\triangle \boldsymbol{W^l} = \boldsymbol{\delta^{l+1}} (\boldsymbol{a^l})^{\boldsymbol{T}} (10)$$

$$\triangle \boldsymbol{b^l} = \boldsymbol{\delta^{l+1}} \ (11)$$

Fig.8 shows the parameters used in the down-sampling layer. $\boldsymbol{W^{n-1}}$ and $\boldsymbol{b^{n-1}}$ represent the $n-1$ layer's weights, and $\boldsymbol{\delta}^{\boldsymbol{n-1}}$ is the error term at layer $n-1$. Based on the equation (4), we know that the $\boldsymbol{z_1^n}$ is calculated based on $\boldsymbol{W_1^{n-1}}$, $\boldsymbol{a_1^{n-1}}$ and $\boldsymbol{b_1^{n-1}}$. Also based on the equation (9), the error term can be calculated as in equation (12).

$$\boldsymbol{\delta_i^{n-1}} = (\boldsymbol{W_i^{n-1}})^{\boldsymbol{T}} \ \boldsymbol{\delta_i^n} \cdot \boldsymbol{f'}(\boldsymbol{z_i^{n-1}}) \ (12)$$

Fig.9 shows the parameters used in the convolutional layer. $\boldsymbol{W^{n-2}}$ and $\boldsymbol{b^{n-2}}$ represent the $n-2$ layer's weights, and $\boldsymbol{\delta}^{\boldsymbol{n-2}}$ is the error term at layer $n-2$. Based on the equation (3), we know that the $\boldsymbol{z_1^n}$ is calculated based on $\boldsymbol{W_1^{n-1}}$, $\boldsymbol{a_1^{n-1}}$ and $\boldsymbol{b_1^{n-1}}$. Also based on the equation (9), the error term can be calculated as in equation (13).

$$\boldsymbol{\delta_1^{n-2}} = \boldsymbol{\delta_1^{n-1}} (*) \boldsymbol{W_1^{n-2}} \cdot \boldsymbol{f'}(\boldsymbol{z^{n-2}}) \ (13)$$

The operator $(*)$ represents the fully convolution operation. The results of fully convolution are computed with the zero-padded edges. For example, $[a,b,c]*[e,d]$ is equal to $[a*e, a*d+b*e, b*d+c*e, c*d]$.

Based on the error term, we can calculate the gradient of the weight by using equations (14) and (15).

$$\triangle \boldsymbol{W_1^{n-2}} = \boldsymbol{\delta_1^{n-1}} * \boldsymbol{a_1^{n-2}} \ (14)$$

$$\triangle \boldsymbol{b_1^{n-2}} = \sum \boldsymbol{\delta_1^{n-1}} (15)$$

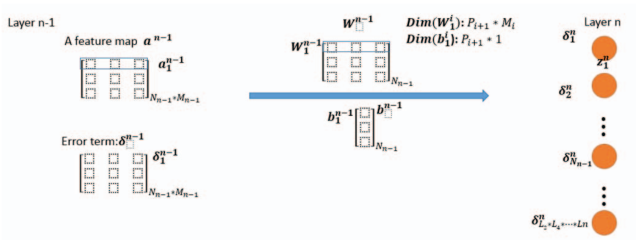


Figure 8 Down-sampling Layer Backward Propagation

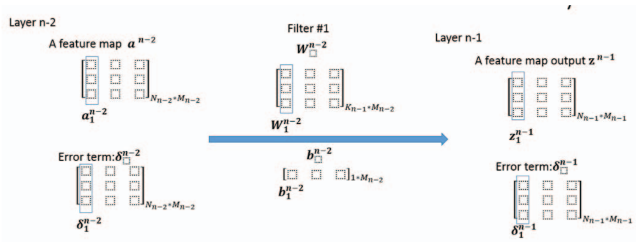


Figure 9 Convolutional Layer Backward Propagation

IV. Lane Change Prediction

Side swipe accidents occur primarily when drivers attempt an improper lane change, drift out of lane, or vehicles lose lateral traction. Past studies[28-29] of lane change detection have relied on vehicular data, such as steering angle, velocity and acceleration. In this paper, we applied MTS-DeepNet to 22 raw physiological signals to predict driver's lane change behavior before driver crosses the lane.

A. Modeling Lane Changes

The lane change detection problem is formulated as follows. At each time t, we begin with a set of signals:

$$\xi(t) = [x_1(t), x_2(t), \ldots, x_n(t)]$$

The task is to classify $\mathrm{x}(t)$ as representing a lane change event (LC) or a non-lane change event (NLC). For this study, the signal set $\mathrm{x}(t)$ consists of a set of physiological signals collected from the driver, as described in the following section. In order to train and test the system, in addition to acquiring the driver physiological data $\mathrm{x}(t)$, we also acquire a video of the driver's view of the road. A schematic of the lane change maneuver is shown in Fig.10. We hand classify each frame of the video as either LC (target 1) or NLC (target 0). To do this, we first identify all lane change events in the video for the duration of the drive. If there are m such events in the video, we identify the time stamp for each $t_{0_i}, i = 1,2, \ldots m$ at which the driver crosses the center lane.

The time t_c is the critical time where we consider a lane change event to be anticipated by the driver and/or is actually in the process of occurring i.e., occurring before the center line is crossed. The target for each time sample in the range $(t_{0_i} - t_c) \leq t \leq t_{0_i}$ is set to 1 to represent a lane change event (LC). For all samples outside this region, the target is set to 0 to represent a non-lane change event (NLC). Here, $\boldsymbol{t_{0_i}}$ represents the crossing time for the ith lane change. All time samples in the video have the target set to 0 except for the regions $\boldsymbol{(t_{0_i} - t_c) \leq t \leq t_{0_i}}$.where the target is set to 1.

During normal driving, for the vast majority of time, the driver is not changing lanes. Hence, the training set will be unbalanced and will contain many more target 0 samples than target 1 samples. Since we have a wealth of non-lane change data, we can be selective about the choice of training data. In particular, we choose NLC training samples that are at least 5 minutes from any given lane change time t_{0_i}. This will allow sufficient time for any physiological signals that were perturbed during a lane change event to return to a normal state.

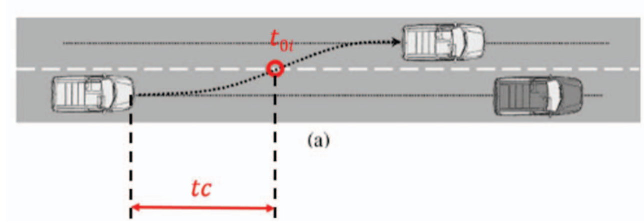


Figure 10 The Lane Change Event Model.

B. Physiological Data Description

Three different types of physiological signals were recorded in real-time while the driver drove: electrocardiogram (ECG) heart signals, galvanic skin response (GSR) signals, and respiration signals (RR). A summary of these raw signals, including sample rate, is given in Table I.

Fig. 11 shows the data acquisition equipment. A chest belt (model: Equivital Chest Belt B2) is used to collect ECG and respiration data; skin conductance sensors are attached to two fingers and connected to the belt, a sensor electronics module (SEM) is connected to the belt and coordinates the data acquisition, and the SEM is connected by USB to a laptop which stores all of the real-time physiological data.

Based on the three raw signals listed in Table I, 19 biologically meaningful signals can be computed, such as heart rate, conductance of the skin, respiratory rate, etc. These signals are typically sampled at different rates. In order to synchronize the data, we pre-process all the signals so that they have a common sample rate, which, in this study, was taken to be 10 Hz. Signals sampled at a rate higher than 10Hz were down sampled and those with a rate smaller than 10Hz were up sampled using linear interpolation.

TABLE I. Table Type Styles

Raw Signal Type	Sample Rate	Description
ECG	256Hz	Electrical activity of the heart
GSR	25.6Hz	Galvanic skin response (GSR) measures the conductance of the skin in response to discrete stimuli
Respiration	25.6Hz	Measures the volume of breath-to-breath variability

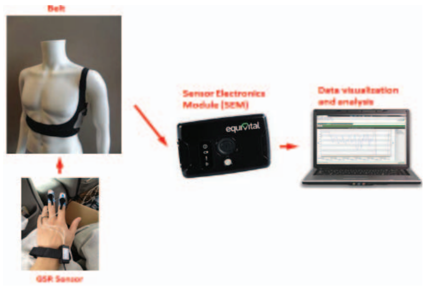


Figure 11 Physiological Signal Acquisition Equipment**.**

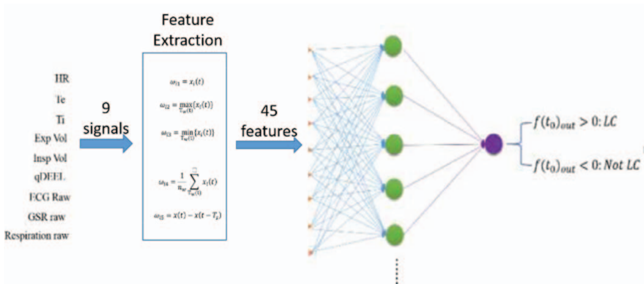


Figure 12 Network Structure of Benchmark System, MTS-Net, for Lane Change Prediction

C. *Performance of a benchmark neural network*

We developed a three layer conventional neural network, MTS-Net, for lane departure prediction. The input to the MTS net were 45 features extracted from 9 signals. The ninesignals are selected from 22 signals by using granger causality and correlation algorithms[16]. The network structure and selected signals are shown in Fig. 12. Five features - instantaneous signal value, minimum value, maximum value, average value, difference value – were extracted from each of the 9 selected signals using the following procedure. First, we applied a sliding window to each signal of length T_w seconds. Suppose that each window contains n_w samples. For the experiments discussed below, we chose $T_w = 2$ seconds, and since the sampling rate was $10Hz$, each feature window contained $n_w = 20$ sample points. In each window and each signal, 5 statistical features were computed.

The MTS-Net system was trained and evaluated on real driving data collected on both city streets and highways. The data consisted of 5 trips taken by a single driver over 5 different days. During these 5 trips, the driver made 60 lane changes in total. Hence, we extracted 60 lane change events, and then randomly extracted 60 non-lane change events, where each event is two minutes long. Since the available data were limited amount of lane change data, we used a 10-fold cross-validation method for testing. In this case, we partitioned the 60 lane change events and 60 non-lane change events into 10 sample data sets using the random stratified sampling method, each data set contained six lane change events and six non-lane change events. At each cross validation fold, nine of the 10 data sets were used for training and the remaining fold for evaluation.

The MTS-Net is a multilayered neural network, in which each hidden unit used a sigmoid activation function and the single output unit used a linear activation function. During a preliminary study, the number of hidden units was varied from 5 to 50 nodes and the network that produced the best results contained 15 nodes. Therefore we fixed the neural net architecture at 15 hidden nodes in subsequent experiments.

The system performance was evaluated by prediction accuracy:

$$Accuracy = \frac{\#\ of\ correct\ prediction}{\#\ of\ data\ samples}.$$

Table II presents a summary of the performance of the MTS-Net measured at each of the ten-fold cross validation. The neural network performed well on training data, which has an average accuracy of 98.36%, but an average performance of 62.12% on all test data.

TABLE II. ACCURACY OF MTS-NET ON THE TESTING DATA SET AT EACH OF THE 10-FOLD CROSS VALIDATION

FOLD INDEX	**Training Results**	**Testing Results**
1	99.78%	57.50%
2	99.11%	65.83%
3	91.61%	66.25%
4	99.51%	57.08%
5	100%	57.08%
6	97.95%	73.75%
7	97.14%	50.40%
8	100%	79.58%
9	98.53%	54.17%
10	100.00%	59.58%
Average	**98.36%**	**62.12%**

D. *MTS-DeepNet System Performance*

The MTS-DeepNet system was evaluated using the same real driving data described in section C. We used all 22 signals as input to the MTS-DeepNet. The same data set used in MTS-Net was used to evaluate the MTS-DeepNet, and the data were partitioned using a random stratified sampling process for a 10-fold cross validation process.

The MTS-DeepNet has a structure of $20 \times 22 \rightarrow 15C^21 \rightarrow 1S^31 \rightarrow 5$. It has an input array of 22 signals, each has 20 samples. It contains one convolutional layer and one down-sampling layer. The major advantage of the MTS-DeepNet compared to the benchmark system is that it does not require to generate features and select signals as preprocessing. The MTS-DeepNet integrates signal selection, feature extraction and pattern classification into one learning process. Signals and features are generated through the filtering instead of generating 5 statistic features, this system generated features automatically through the selection of the filter parameters at the training stage.

Table III shows a summary of the performance results generated by the MTS-DeepNet system. The average prediction accuracy of 10 folds is 71.18%, which is an improvement of 9.06% over the MTS-Net. Although the overall prediction does not appear to be high, the performance is achieved based on physiological signals only. When a system involves physiological signals with vehicle and environmental signals, the total number of signals performance will improve. When all these signals, more than one hundred, are involved in study, it is even more important to use the MTS-DeepNet methodology to build the prediction system, since separate process of signal selection and feature generation may be even less effective due to the fact that more try and error attempts would be needed to select good signals and features.

The experiments were all conducted on a computer with a platform of Intel i7-4810MQ quad-core processor @2.8GHz, 16GB memory, Microsoft Windows 8.1, and Matlab 2015a. It took 20 minutes to conduct the 10-fold cross validation using the MTS-Net, and 20 hours to conduct the 10-fold cross validation using the MTS-DeepNet.

TABLE III. ACCURACY OF MTS-DEEPNET ON THE TESTING DATA AT EACH OF THE 10-FOLD CROSS VALIDATON

Fold Index	Training Results	Testing Results
1	90.45%	72.25%
2	96.43%	83.33%
3	92.81%	79.58%
4	85.18%	66.67%
5	93.88%	57.50%
6	89.20%	72.92%
7	86.96%	81.25%
8	94.46%	64.58%
9	98.26%	58.75%
10	92.46%	75.00%
Average	**92.01%**	**71.18%**

For automotive safety engineers it is also important to know the best time period to predict a driver's intention to change lanes. Table IV shows the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) prediction numbers over the total 60 positive events and 60 negative events at each time instance from $(t_0 - 0.1)$ seconds to $(t_0 - 2)$ seconds. Note that the lane change occurs at t_0 and TP + FN = 60 and FP + TN = 60. Based on the data shown in Table IV, we can conclude that the most reliable window to predict driver lane departure is: $(t-1) \leq t \leq (t-0.4)$.

Figure 13 illustrates the true position and false position prediction rates at each time instance, where the time index i represents the time instance, $t_0 - 2 + i * 0.1$, i = 0 , 1, …, 19, where t_0 was the time when the driver crossed the lane boundary. The highest true position rate occurred around at i = 9, 10, 11, and the lowest false position was close to t_0.

TABLE IV. PREDICTION RESULTS AT EVERY TIME INSTANCE

	t-2.0	t-1.9	t-1.8	t-1.7	t-1.6	t-1.5	t-1.4	t-1.3	t-1.2	t-1.1
TP	36	36	36	36	37	37	39	40	41	41
FP	16	16	15	17	16	16	16	15	15	13
TN	44	44	45	43	44	44	44	45	45	47
FN	24	24	24	24	23	23	21	20	19	19
	t-1.0	t-0.9	t-0.8	t-0.7	t-0.6	t-0.5	t-0.4	t-0.3	t-0.2	t-0.1
TP	41	39	40	40	40	39	39	37	37	36
FP	13	12	13	12	11	10	9	9	8	9
TN	47	48	47	48	49	50	51	51	52	51
FN	19	21	20	20	20	21	21	23	23	24

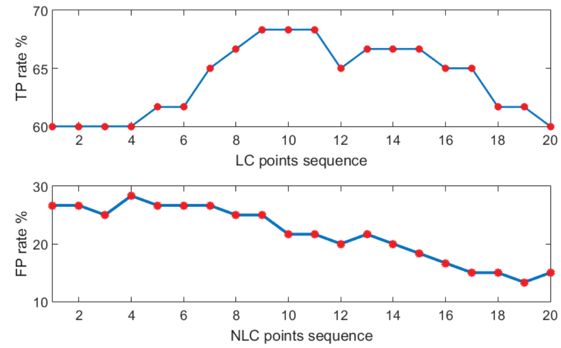


Figure 13 True positive TP and false positive FP rates at each time instance.

V. CONVLUSION

In this paper we presented an innovative machine learning algorithm, MTS-DeepNet, for MTS pattern classification. MTS-DeepNet is an integrated machine learning framework for MTS feature extraction and pattern classification. The MTS-DeepNet training consists of three stages, namely forward propagation, backward propagation and cost function optimization. MTS-DeepNet was implemented in Matlab. We applied the MTS-DeepNet framework to the driver lane departure prediction. Our experimental results showed that the proposed MTS-DeepNet system made a significant improvement, 9.06%, over the conventional neural network, MTS-Net, which requires additional pre-processes, namely, signal selection, and feature extraction and selection.

REFERENCES

[1] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. InAdvances in neural information processing systems 2012 (pp. 1097-1105).

[2] Ji S, Xu W, Yang M, Yu K. 3D convolutional neural networks for human action recognition. Pattern Analysis and Machine Intelligence, IEEE Transactions on. 2013 Jan;35(1):221-31.

[3] Lee, Honglak, Peter Pham, Yan Largman, and Andrew Y. Ng. "Unsupervised feature learning for audio classification using convolutional deep belief networks." In Advances in neural information processing systems, pp. 1096-1104. 2009.

[4] Sainath, Tara N., Brian Kingsbury, George Saon, Hagen Soltau, Abdel-rahman Mohamed, George Dahl, and Bhuvana Ramabhadran. "Deep convolutional neural networks for large-scale speech tasks." Neural Networks 64 (2015): 39-48.

[5] LeCun Y, Bengio Y. Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks. 1995;3361(10)

[6] Wu, Berlin. "Pattern recognition and classification in time series analysis."Applied Mathematics and Computation 62, no. 1 (1994): 29-45.

[7] Han, Min, and Mingming Fan. "Application of neural networks on multivariate time series modeling and prediction." American Control Conference, 2006. IEEE, 2006.

[8] Han, Min, and Mingming Fan. "Multivariate time series prediction by neural network combining SVD." Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on. Vol. 5. IEEE, 2006.

[9] de Lautour, Oliver R., and Piotr Omenzetter. "Damage classification and estimation in experimental structures using time series analysis and pattern recognition." Mechanical Systems and Signal Processing 24.5 (2010): 1556-1569.

[10] Orsenigo, Carlotta, and Carlo Vercellis. "Combining discrete SVM and fixed cardinality warping distances for multivariate time series classification." Pattern Recognition 43.11 (2010): 3787-3794.

[11] Modinos, Gemma "Multivariate pattern classification reveals differential brain activation during emotional processing in individuals with psychosis proneness." Neuroimage 59.3 (2012): 3033-3041.

[12] Kini, B. Venkataramana, and C. Chandra Sekhar. "Large margin mixture of AR models for time series classification." Applied Soft Computing 13.1 (2013): 361-371.

[13] Jeong, Young-Seon, and Raja Jayaraman. "Support vector-based algorithms with weighted dynamic time warping kernel function for time series classification." Knowledge-Based Systems 75 (2015): 184-191.

[14] Lee, Dongha, Changwon Jang, and Hae-Jeong Park. "Multivariate detrending of fMRI signal drifts for real-time multiclass pattern classification." NeuroImage108 (2015): 203-213

[15] Douzal-Chouakria, Ahlame, and Cécile Amblard. "Classification trees for time series." Pattern Recognition 45.3 (2012): 1076-1091

[16] Liu, Ruoqian, Shen Xu, Chen Fang, Yung-wen Liu, Yi L. Murphey, and Dev S. Kochhar. "Statistical modeling and signal selection in multivariate time series pattern classification." In Pattern Recognition (ICPR), 2012 21st International Conference on, pp. 2853-2856. IEEE, 2012

[17] Wang, Xinying, and Min Han. "Improved extreme learning machine for multivariate time series online sequential prediction." Engineering Applications of Artificial Intelligence 40 (2015): 28-36

[18] Hu, Xiao, and Valeriy Nenov. "Multivariate AR modeling of electromyography for the classification of upper arm movements." Clinical neurophysiology 115.6 (2004): 1276-1287

[19] Pitarch, Yoann, Dino Ienco, Elodie Vintrou, Agnès Bégué, Anne Laurent, Pascal Poncelet, Michel Sala, and Maguelonne Teisseire. "Spatio-temporal data classification through multidimensional sequential patterns: Application to crop mapping in complex landscape." Engineering Applications of Artificial Intelligence 37 (2015): 91-102

[20] Smith, Daniel, et al. "Bag of Class Posteriors, a new multivariate time series classifier applied to animal behaviour identification." Expert Systems with Applications 42.7 (2015): 3774-3784

[21] He, Guoliang, Yong Duan, Rong Peng, Xiaoyuan Jing, Tieyun Qian, and Lingling Wang. "Early classification on multivariate time series."Neurocomputing 149 (2015): 777-787

[22] Duchêne, Florence, Catherine Garbay, and Vincent Rialle. "Learning recurrent behaviors from heterogeneous multivariate time-series." Artificial intelligence in medicine 39.1 (2007): 25-47

[23] Vedavathi, K., K. Srinivasa Rao, and K. Nirupama Devi. "Unsupervised learning algorithm for time series using bivariate AR (1) model." Expert Systems with Applications 41.7 (2014): 3402-3408

[24] D'Urso, Pierpaolo, Livia De Giovanni, and Riccardo Massari. "Time series clustering by a robust autoregressive metric with application to air pollution."Chemometrics and Intelligent Laboratory Systems 141 (2015): 107-124

[25] Yang, Kiyoung, and Cyrus Shahabi. "An efficient k nearest neighbor search for multivariate time series." Information and Computation 205.1 (2007): 65-98

[26] Weng, Xiaoqing, and Junyi Shen. "Classification of multivariate time series using locality preserving projections." Knowledge-Based Systems 21.7 (2008): 581-587

[27] Antonucci, Alessandro, Rocco De Rosa, Alessandro Giusti, and Fabio Cuzzolin. "Robust classification of multivariate time series by imprecise hidden markov models." International Journal of Approximate Reasoning 56 (2015): 249-263

[28] Morris, Brendan, Anup Doshi, and Mohan Trivedi. "Lane change intent prediction for driver assistance: On-road design and evaluation." In Intelligent Vehicles Symposium (IV), 2011 IEEE, pp. 895-901. IEEE, 2011

[29] Tomar, Ranjeet Singh, Shekhar Verma, and Geetam Singh Tomar. "Prediction of lane change trajectories through neural network." In Computational Intelligence and Communication Networks (CICN), 2010 International Conference on, pp. 249- 253. IEEE, 2010.