2024 IEEE International Conference on Mobility, Operations, Services and Technologies (MOST) | 979-8-3503-0773-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/MOST60774.2024.00011

Enhancing Driving Behavior Analysis in Autonomous Systems: A Reservoir Computing and Temporal-Aware Machine Learning Approach

Fabiha Nowshin[1], Sanchit Sethi[1], Zheng Dong[2], Yang Yi[1]
[1]Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg VA, USA
[2]Computer Science Department, Wayne State University, Detroit MI, USA

***Abstract*—In the rapidly evolving domain of autonomous vehicles, ensuring safety and reliability through advanced anomaly detection is paramount. Reservoir Computing, a novel approach for processing time-series data in dynamic systems, stands out for its ability to capture complex temporal patterns efficiently. This paper introduces an innovative method that integrates Reservoir Computing with temporal-aware data analysis to enhance driver behavior assessment. Our approach employs a unique combination of autoencoder-based feature extraction and Reservoir Computing to analyze driving metrics from vehicle sensors. The autoencoder compresses and encodes these temporal features, which are then processed by a reservoir computing model, adept at processing intricate temporal dependencies in the data. To evaluate the effectiveness of our model we simulate with a GPS dataset of 10,000 taxis to identify various driving dynamics of speed, acceleration, and state changes providing a comprehensive view of driver behavior. We further categorize drivers into different sets based on their driving performance using a support vector machine (SVM) algorithm. Our algorithm marks a significant step forward in anomaly detection for autonomous vehicles, offering a route to safer driving experiences and advancing vehicle safety technologies.**

***Index Terms*—machine learning, autonomous driving, reservoir computing, autoencoder, support vector machine**

I. Introduction

Autonomous driving (AD) and Intelligent Vehicles (IV) have gained substantial attention in academia, industries, government bodies, and the general populace largely due to the transformative impact they promise in the transportation sector, propelled by breakthroughs in artificial intelligence algorithms [1]. This aligns with the recent advancements in vehicular sensor technologies that have expanded their range of functionalities, including activity recognition, object detection, localization, and tracking, all of which improve the sensing and computational capabilities essential for the operation of autonomous driving systems [2]–[4]. Furthermore, the implementation of IVs is anticipated to significantly reduce road accidents and ease traffic congestion, contributing to enhanced mobility, especially in densely populated urban regions [5].

The advancement of IV technology is a pivotal development in modern transportation, promising enhanced road safety and a transformation in the driving experience. As IVs progress towards higher levels of autonomy, ensuring their reliability and safety becomes increasingly crucial. This evolution is further emphasized by complex sensor technologies and machine learning algorithms that are essential for navigating and interpreting diverse driving environments [6]. For instance, the integration of principle sensor technologies of artificial vision, radar and LiDAR allows exteroceptive perception in the field of AD [7]. These systems, capable of processing vast amounts of environmental data, allow IVs to make informed decisions in complex driving scenarios.

However, despite these advancements, ensuring the fail-safe operation remains a significant challenge [8], [9]. Real-world driving conditions, characterized by unpredictable traffic behavior and diverse environmental factors, pose complex scenarios that can stretch the capabilities of even the most advanced systems. This challenge is particularly pronounced in interpreting unexpected or ambiguous situations, a task at which human drivers excel due to their intuitive understanding and experience [9]. Examining driving behavior as shown in Fig. 1 is pivotal for evaluating driver performance, bolstering traffic safety, and fostering the growth of intelligent and robust transportation infrastructures. This analysis supports a range of vital applications, including surveillance of drivers, vehicles, and road conditions, delivering preemptive alerts and driving assistance, as well as improving overall driving comfort and promoting energy efficiency [10].

In response to these challenges, there is a growing emphasis on employing machine learning, especially deep learning techniques, to develop more adaptive and nuanced driving algorithms. The inherent unpredictability of traffic behavior and environmental factors makes it imperative to develop systems capable of dealing with ambiguity and unexpected scenarios as highlighted in several recent surveys on trajectory prediction methods [11]–[13]. While there are different types of trajectory prediction methods including physics-based

This work was supported in part by the U.S. National Science Foundation (NSF) under Grant CCF-1750450, Grant ECCS-1731928, Grant ECCS-2128594, Grant ECCS- 2314813, Grant CCF-1937487, Grant CNS-2103604 and Grant CNS-2231523.

979-8-3503-0773-3/24/$31.00 ©2024 IEEE
DOI 10.1109/MOST60774.2024.00011

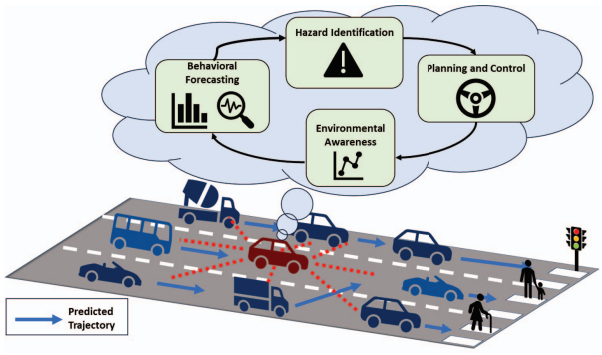


Fig. 1: Trajectory Prediction for Intelligent Vehicles.

methods like Kalman Filtering and Monte Carlo and classical machine learning models like Gaussian Process and Support Vector Machines (SVM), they are mostly geared towards simple prediction situations and short termed tasks [14]–[18].

Deep learning-based approaches for trajectory prediction have recently gained traction due to their comprehensive analysis capabilities. These methods excel not just in accounting for physics and road-related factors, but also in integrating interaction-related factors which allows them to effectively handle complex scenarios in trajectory forecasting [11]. Underdeep learning, there have been several implementations of Convolutional Neural Networks (CNN) for trajectory prediction and driving behavior analysis, taking the historical trajectory as the input to the system while achieving the temporal aspect by stacking the convolutional layers [19], [20]. Despite having a faster runtime, CNNs and traditional machine learning models can only process spatial information. In order to handle the temporal information which is specially crucial to trajectory predictions and driving behavior analysis, Recurrent Neural Networks (RNN)s are much more suited.

RNNs, are designed to process temporal data, store information from previous time steps, and use these along with current inputs to determine outputs [21]. While there have been previous implementations of trajectory prediction and driving behavior analysis using subsets of RNNs including Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM)s, these approaches suffer from increased training complexity and high latency issues [22], [23]. To address these issues Reservoir Computing stands to be a more suited model which simplifies the training complexity by only training the output layer of the network. Reservoir computing's ability to handle spatiotemporal information has enabled its application in diverse tasks such as time-series prediction, classification, segmentation, noise reduction, as well as channel equalization and control [24].

In this work, we address the issues of slow runtimes and high computational demands associated with traditional RNNs by developing the first reservoir computing model specifically tailored for trajectory prediction and driving behavior analysis. This novel approach incorporates an autoencoder preprocessing layer, which effectively compresses and encodes the input data, reducing the dimensionality before feeding it into the reservoir. This integration not only enhances the efficiency of the model but also improves its accuracy in capturing the intricate dynamics of driving behavior. The model's architecture is further augmented by the inclusion of a Support Vector Machine (SVM) classifier in the final stage. This classifier is adept at handling the high-dimensional feature space created by the reservoir, making it a powerful tool for classifying driving behaviors into predefined categories such as safe, risky, or anomalous. By leveraging the strengths of both autoencoder and reservoir computing, along with the robust classification capabilities of the SVM, our model achieves a delicate balance between computational efficiency and predictive performance. The main contributions of our work are summarized below:

- We present the first-ever reservoir computing model specifically designed for trajectory detection and driving behavior analysis. This model represents a significant breakthrough in the field of autonomous vehicle technology, as it effectively captures the complex dynamics and temporal patterns inherent in driving data.
- An innovative preprocessing layer using an autoencoder is integrated into our model. This layer efficiently compresses and encodes the input data, effectively reducing its dimensionality. This step not only streamlines the data processing pipeline but also enhances the model's ability to discern and learn from the subtle nuances in driving behaviors.
- A unique scoring system is developed to evaluate driving behaviors. This system amalgamates various key metrics such as speed consistency, acceleration patterns, and state changes in driving, to compute an average score for drivers. This scoring method is applied to a robust GPS dataset, enabling a nuanced and detailed analysis of driving performance.
- By incorporating a Support Vector Machine (SVM) classifier in the final stage of our model, we have achieved an exceptional accuracy rate of 99%. This classifier excels in managing the high-dimensional feature space outputted by the reservoir and is crucial in accurately classifying driving behaviors into categories like safe, risky, or anomalous. This high level of accuracy underscores the effectiveness of our model in real-world applications, setting a new benchmark in the realm of autonomous vehicle technology.

II. Background and Related Works

A. Dimensionality Reduction Techniques

In the complex realm of transportation systems, dimensionality reduction techniques are essential for extracting meaningful insights. Principal Component Analysis (PCA) and Autoencoders stand as prominent methodologies, each offering distinctive advantages. This section undertakes a comprehensive comparative analysis, shedding light on their unique contributions to transportation research.

Principle Component Analysis (PCA) has proven to be an invaluable tool for researchers navigating the intricate data

landscape of transportation systems. Its capability to distill high-dimensional data into meaningful patterns unlocks opportunities for improving safety, efficiency, and user experience. Driving amidst a multitude of data points, PCA organizes chaos by identifying underlying dimensions, simplifying analysis, and revealing hidden relationships. Notable applications include pre-processing physiological data for effective drowsiness detection [25].

Beyond driver monitoring, PCA empowers intelligent systems. Its application in eco-driving assistance systems models driver behavior, predicting delays and promoting fuel efficiency [26]. PCA also contributes to secure transportation infrastructure by identifying anomalies in network traffic data [27]. PCA's versatility extends to personalizing user experiences in advanced driving technologies. Leveraging PCA in an inverse reinforcement learning approach customizes automated lane change systems based on individual driving styles [28].

While PCA excels at identifying key dimensions, Autoencoders provide another potent technique for understanding complex data in transportation. Autoencoders, such as the Echo State Network (ESN) autoencoder, prove valuable in industrial IoT systems for anomaly detection, enabling proactive maintenance and enhancing operational safety [29].

Autoencoders transcend traditional machine learning limitations by learning directly from data, adapting to diverse data types in transportation, from traffic patterns to vehicle sensor readings [30]. The ability of autoencoders to reconstruct original data unveils valuable insights. This property is leveraged for anomaly detection in time-series signals from traffic or environmental sensors, enabling real-time monitoring and response [31].

Recent advancements in autoencoder architecture, exemplified by the B-Detection framework, combine LSTM autoencoders with boosting algorithms for runtime reliability anomaly detection in mobile edge computing services, leading to more reliable and efficient service delivery in transportation systems [32].

In the context of our project on scoring driver behavior, the integration of PCA and Autoencoders presents a promising avenue. PCA's ability to unravel hidden patterns and optimize algorithms complements Autoencoders' prowess in anomaly detection and handling complex signals. By combining these techniques, our model gains the ability to not only assess driving patterns but also identify anomalies and deviations from expected behaviors. This comprehensive approach ensures a nuanced and accurate scoring system, fostering the development of safer, smarter, and more reliable transportation systems for the future. The synergy between PCA and Autoencoders enhances the depth and precision of our driver behavior analysis, providing valuable insights for proactive decision-making and system optimization.

B. Reservoir Computing

Deep neural networks (DNNs) are primarily categorized into feedforward neural networks (FNNs), which process static input data, and recurrent neural networks (RNNs) which handle both temporal and spatial data. Unlike FNNs, RNNs exhibit dynamic characteristics due to their recurrent connections within the hidden layer, enabling the retention of information over time [33]. Despite their biological nervous system resemblance, RNNs are known for their complex and intensive training procedures. To address these challenges, reservoir computing has been introduced as a simplified alternative, focusing primarily on training the output layer [34]. This approach requires less computational effort due to its reliance on smaller datasets and linear optimization.

In reservoir computing, the neural network comprises three interconnected layers: the input layer, the reservoir, and the output layer. The neuron activations in these layers at any given time step t can be described as $u(t) = (u_1(t), \ldots, u_N(t))$, $v(t) = (v_1(t), \ldots, v_N(t))$ and $z(t) = (z_1(t), \ldots, z_N(t))$, respectively. The activations among these units are defined by the following equations, where $\sigma = (\sigma_1, \ldots, \sigma_M)$ represents the activation function within the reservoir:

$$v(t+1) = \sigma W^{uv}\{u(t+1) + Wv(t) + W^{zv}z(t)\} \quad (1)$$

$$z(t+1) = \gamma W^{vz}\{u(t+1), v(t+1), z(t)\} \quad (2)$$

Here, W^{uv} is the weight matrix connecting the input to the internal units, while W represents the weights within the reservoir. The weight matrix connecting the output to the internal reservoir is W^{zv}, and the weight matrix from the reservoir to the output is denoted as W^{vz} in Equation (2). The training process is made more efficient by randomly initializing connections between the input and the reservoir, and training connections in the output layer using a regularized linear least-squares optimization method, effectively mitigating the vanishing gradient problem [35], [36].

Echo State Networks (ESNs) and Liquid State Machines (LSMs) represent two variations of reservoir computing. Both ESNs and LSMs maintain fixed and random connections from the input to the reservoir and within the reservoir. Despite similar training procedures for the output layer, ESNs and LSMs differ in their core structures: ESNs are rate-based approximations, whereas LSMs are modeled after biologically inspired spiking neural networks (SNNs) [37]. One of the compelling reasons for the preference of ESNs over Liquid State Machines (LSMs) in certain applications, including ours, stems from their computational efficiency and ease of implementation. Unlike LSMs, which are based on biologically inspired spiking neural networks and require intricate mechanisms to handle spike timings and interactions, ESNs operate on continuous values, making them more straightforward to implement and integrate with standard machine learning workflows [38]. Moreover, ESNs are often favored for their robustness in dealing with noisy and non-stationary data, an attribute essential for analyzing complex driving behaviors where data can vary significantly over time.

In the context of our work, ESNs are particularly advantageous due to their ability to model complex temporal dynamics with relatively simple architectures. This simplicity allows for

faster computations and more efficient training, crucial for real-time applications like autonomous driving where quick decision-making based on accurate trajectory predictions and behavior analysis is paramount. Furthermore, the architecture of ESNs facilitates capturing long-term dependencies in data, a critical requirement for understanding and predicting driving patterns and behaviors over extended periods [39]. Therefore, while LSMs offer biologically realistic modeling, ESNs provide a more practical and computationally efficient approach for our application, aligning with the need for real-time processing and analysis in autonomous vehicle systems.

C. Navigating Complexity in Classification

Accurately classifying data serves as a cornerstone of machine learning, with various algorithms tailored to diverse problem settings. Random Forest Classifiers leverage their ensemble-based approach for robustness and versatility, while Linear Classifiers offer interpretability and efficiency for linearly separable data. However, both approaches encounter limitations when faced with complex, non-linear data or the presence of outliers.

For such challenging classification tasks, characterized by intricate patterns and outlier data, Support Vector Machines (SVMs) emerge as a compelling alternative. Their defining characteristic lies in maximizing the margin between classes, creating a clear decision boundary that effectively separates even complex data. This wider "margin of safety" translates to enhanced classification accuracy and reduced misclassification rates [40].

Several key advantages elevate SVMs to a premier choice for these scenarios. Their inherent resilience to outliers makes them robust to data noise and anomalies, as demonstrated in [41] work on personal driving style-based ADAS customization using SVMs. Additionally, their ability to handle non-linear data through kernel functions makes them versatile across diverse problem domains, including autonomous driving applications where fault tolerance is crucial [40]. Furthermore, SVMs offer a degree of interpretability through kernel analysis, enabling insights into their decision-making process.

While classifiers are context-dependent, for complex, non-linear scenarios, SVMs demonstrably offer significant advantages, making them a powerful tool in the classification landscape. Their resilience, versatility, and interpretability position them as valuable contenders for a range of challenging tasks, particularly in fields like autonomous driving and driver behavior analysis.

III. GPS Dataset from Vehicles

Localization, a crucial component of autonomous driving systems, heavily relies on technologies like GPS, IMU, and GNSS. GNSS, encompassing various global navigation satellite systems such as Europe's Galileo and the U.S's GPS, offers variable accuracy, ranging from centimeters to meters based on different observational data and processing methods [42]. GPS, known for its affordability and consistency over time, does not accumulate errors. This section discusses the utilization of GPS data in autonomous vehicles, outlining the nature of the dataset, the process of constructing a relevant data frame, and scoring drivers' performance based on this data [43], [44].

A. Nature of Dataset

The GPS dataset utilized in this paper is denoted as $D = \{a_n, t_n, l_{1n}, l_{2n}\}$, where:

- a_n represents the vehicle identifier,
- t_n signifies the timestamp of data collection,
- l_{1n} and l_{2n} are the latitude and longitude coordinates, respectively.

This study leverages the T-Drive trajectory dataset, encompassing extensive GPS data gathered from a fleet of taxis. This dataset is pivotal for analyzing urban driving patterns, offering granular insights into vehicular movements and behaviors. It allows for a comprehensive examination of mobility patterns and driver behavior in urban settings, aiding in the understanding of complex traffic dynamics.

B. Building the Dataframe

To construct a comprehensive data frame from GPS data, we analyze and preprocess the data to extract key features related to vehicle trajectories and driving behaviors. This process involves calculating distances, bearings, and identifying state changes in driving.

1) Distance and Bearing Calculation: Distance between GPS points is computed using the Haversine formula:

$$a = \sin^2\left(\frac{\Delta\text{lat}}{2}\right) + \cos(\text{lat}_1) \cdot \cos(\text{lat}_2) \cdot \sin^2\left(\frac{\Delta\text{lon}}{2}\right) \quad (3)$$

$$c = 2 \cdot \text{atan2}\left(\sqrt{a}, \sqrt{1-a}\right) \quad (4)$$

$$d = R \cdot c \quad (5)$$

where R is the Earth's radius (6371 km), $\Delta\text{lat} = \text{lat}_2 - \text{lat}_1$, and $\Delta\text{lon} = \text{lon}_2 - \text{lon}_1$.

The bearing between points, denoted as θ, is calculated to determine the direction of travel:

$$\theta = \text{atan2}\left(\sin(\Delta\text{lon}) \cdot \cos(\text{lat}_2), \cos(\text{lat}_1) \cdot \sin(\text{lat}_2) - \sin(\text{lat}_1) \cdot \cos(\text{lat}_2) \cdot \cos(\Delta\text{lon})\right) \quad (6)$$

2) Speed Calculation: Speed at each timestamp is calculated to understand the vehicle's motion dynamics. The speed $v(t)$ at time t is calculated using the distance d between consecutive GPS points and the time difference Δt between these points:

$$v(t) = \frac{d}{\Delta t} \quad (7)$$

The distance d is obtained using the Haversine formula, and Δt is the time interval between successive GPS readings. This speed calculation is crucial for identifying variations in driving patterns and subsequent state changes.

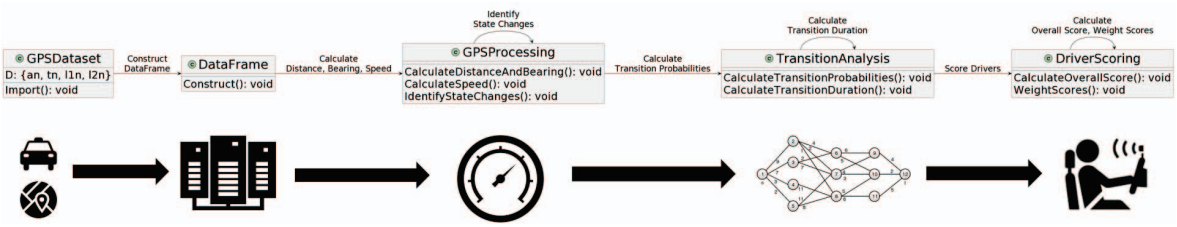


Fig. 2: Overview of GPS Dataset Processing.

3) Identifying State Changes: State changes in driving are determined by analyzing speed and bearing variations over time, allowing for categorization into different driving states. We denote speed at time t as $v(t)$ and bearing as $\theta(t)$. The driving state is categorized based on the changes in $v(t)$ and $\theta(t)$ as follows:

- **Acceleration** ($\Delta v(t) > 0$): An increase in speed over time.
- **Deceleration** ($\Delta v(t) < 0$): A decrease in speed over time.
- **Constant Speed** ($\Delta v(t) \approx 0$): Negligible or no change in speed over time.

Directional changes are identified by analyzing $\Delta\theta(t)$, the change in bearing:

- **Turning Right** ($\Delta\theta(t) > 0$): A positive change in bearing angle.
- **Turning Left** ($\Delta\theta(t) < 0$): A negative change in bearing angle.
- **Moving Straight** ($\Delta\theta(t) \approx 0$): Negligible or no change in bearing angle.

These states provide insights into driving behaviors, contributing to a comprehensive understanding of vehicle movement patterns and potential anomalies in driving.

4) Transition Probability Calculation: After identifying the state changes, we calculate the transition probabilities. For each taxi, the probability P_{ij} of transitioning from state i to state j is determined by:

$$P_{ij} = \frac{\text{Number of transitions from } i \text{ to } j}{\text{Total transitions from } i} \tag{8}$$

5) Transition Duration Calculation: The duration of each transition is also vital for behavior analysis. For each transition from state i to state j, the duration ΔT_{ij} is computed as the difference between timestamps:

$$\Delta T_{ij} = T_{\text{end}} - T_{\text{start}} \tag{9}$$

where T_{end} and T_{start} are the timestamps at the end and the start of the transition.

C. Scoring Drivers Based on Performance

To evaluate driving performance, we introduce a scoring system that considers various aspects of driving behavior. The overall score S for each driver is calculated using speed consistency, state change, and acceleration scores:

$$S = \frac{1}{N}\sum_{i=1}^{N}(w_1 \cdot S_{\text{speed},i} + w_2 \cdot S_{\text{state},i} + w_3 \cdot S_{\text{accel},i}) \tag{10}$$

Here, $S_{\text{speed},i}$, $S_{\text{state},i}$, and $S_{\text{accel},i}$ represent the speed consistency, state change, and acceleration scores for the i-th transition. The weights w_1, w_2, and w_3 are used to balance the importance of each aspect in the overall score.

- Speed Consistency Score (S_{speed}): This score reflects the consistency of the driver's speed, calculated as the normalized inverse of the standard deviation of speed.
- State Change Score (S_{state}): This score is derived from the frequency and nature of state changes, indicating the driver's adaptability and responsiveness.
- Acceleration Score (S_{accel}): Represents the driver's control over the vehicle's acceleration and deceleration, calculated based on the frequency of acceleration-related state changes.

A higher overall score S indicates a safer and more consistent driving pattern, whereas a lower score points to potential areas of improvement in driving behavior. Additionally, these calculated scores S serve as ground truths for our model. They provide a basis for assessing and validating the predictive capabilities of the autonomous driving system. By comparing the predicted driving behaviors against these ground truth scores, we can gauge the accuracy and reliability of the model in real-world scenarios, contributing to the overall safety and efficiency of autonomous vehicles.

IV. The Reservoir Computing Model

The developed reservoir model is depicted in Fig. 3 where the extracted features are used as input to the system and is passed through the autoencoder, reservoir layer and the classifier. This section discusses each layer in depth and our complete model is detailed in Algorithm 1.

A. The Autoencoder Input Layer

Given the extracted features from GPS data, such as 'Transition Duration', 'Transition Probability', 'Speed Consistency Score', 'State Change Score', and 'Acceleration Score', we utilize an autoencoder for effective dimensionality reduction and feature transformation. The encoder compresses the high-dimensional feature vector into a lower-dimensional latent representation. Mathematically, this is expressed as:

$$Z = \text{ReLU}(W_e \times X + b_e) \tag{11}$$

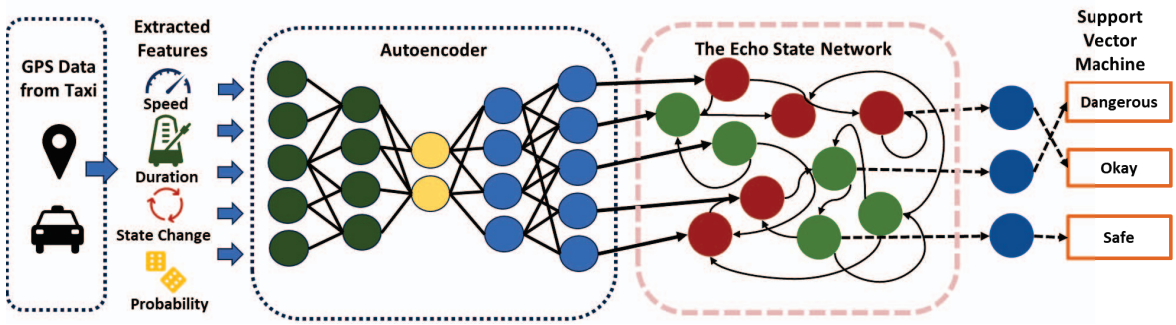


Fig. 3: Overview of the Reservoir Computing Model.

Algorithm 1 Analysis of Taxi Driving Patterns

Initialize: Text files list $inputFiles$
Initialize: Output file path $outputFile$
Initialize: Headers list $headers$
Initialize: DataFrame $dataFrame$
Initialize: Feature vectors $featureVectors$
Initialize: Time-aware vectors $timeVectors$
Initialize: Transitions $transitions$
Merge files into CSV with $outputFile$, $headers$
Data Preprocessing: Read and process data from $inputFiles$
for each row in $dataFrame$ **do**
  Calculate distance and bearing for each point
end for
Create $featureVectors$ from $dataFrame$
Haversine Formula:
for each pair $(lat1, lon1, lat2, lon2)$ **do**
  $d \leftarrow R \cdot \arccos(\cos(lat1)\cos(lat2)\cos(lon2 - lon1) + \sin(lat1)\sin(lat2))$
end for
Autoencoder Training:
Initialize W_{enc}, b_{enc}, W_{dec}, b_{dec}
for each feature in X_{train} **do**
  $encoded \leftarrow \sigma(W_{enc} \cdot feature + b_{enc})$
  $decoded \leftarrow \sigma(W_{dec} \cdot encoded + b_{dec})$
end for
Reservoir Computing:
Initialize reservoir weights W_{in}, W_{res}, state x
for each encoded feature $u(t)$ in $encodedFeatures$ **do**
  $x(t) \leftarrow \tanh(W_{in} \cdot u(t) + W_{res} \cdot x(t-1))$
end for
Train output weights W_{out}
Classifier Predictions:
Initialize classifiers RF, SVM, LR
for each sample s in X **do**
  $RF_{score} \leftarrow$ RF.predict(s)
  $SVM_{score} \leftarrow$ SVM.predict(s)
  $LR_{score} \leftarrow$ LR.predict(s)
end for

where W_e and b_e denote the weights and biases of the encoder, respectively, and X represents the input feature vector.

The decoder phase aims to reconstruct the input data from its compressed form. This process is defined by the equation:

$$X_{rec} = \sigma(W_d \times Z + b_d) \tag{12}$$

Here, W_d and b_d are the weights and biases of the decoder, and σ represents the sigmoid activation function.

The training objective of the autoencoder is to minimize the reconstruction error, which ensures the preservation of essential characteristics of the driving behavior in the compressed feature space. The output from the autoencoder is then utilized as input for the reservoir computing layer for advanced analysis and modeling.

B. The Echo State Network

The Echo State Network (ESN) layer serves as a crucial component for processing GPS data transformed by the autoencoder. This layer is pivotal for capturing temporal dependencies inherent in GPS-based vehicle trajectories. The reservoir within the ESN is initialized with random weights to introduce variability in the system:

$$W_{res} = randn(size, size) - 0.5 \tag{13}$$

where `size` is the predefined reservoir size. This random initialization plays a vital role in determining the unique dynamic characteristics of the reservoir.

To enhance the input's diversity and effectively manage the complexities of GPS data, a masking layer is employed:

$$M = W_{mask} \odot X \tag{14}$$

Here, W_{mask} is a matrix with randomly generated values, and $\odot$ denotes the element-wise multiplication, ensuring varied input propagation through the network. The state of the reservoir R is updated by integrating both the current state and the masked input:

$$R(t+1) = \tanh(W_{res} \cdot R(t) + M(t)) \tag{15}$$

The non-linear activation function $\tanh$ introduces necessary non-linearity into the system, facilitating the capture of complex temporal patterns. This ESN setup, with its dynamic reservoir, is exceptionally suited for analyzing GPS data, capturing nuanced vehicle movement patterns and intricate behavioral dynamics over time.

C. Support Vector Machine Classifier

The final layer in our model architecture utilizes a Support Vector Machine (SVM) classifier. This layer is responsible for classifying the processed GPS data into distinct driving behavior categories. The SVM classifier operates on the high-dimensional features output by the Echo State Network. It's designed to find the optimal hyperplane that separates the different classes of driving behavior in the feature space. The SVM formulation is given by:

$$\text{minimize} \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i \tag{16}$$

$$\text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \tag{17}$$

where w and b are the parameters of the hyperplane, C is the regularization parameter, and ξ_i are the slack variables allowing for misclassification.

The SVM classifier is instrumental in classifying driving behaviors, leveraging the temporal features distilled by the ESN. Its robustness to high-dimensional data and effectiveness in handling non-linear separations make it an ideal choice for classifying complex driving behaviors extracted from GPS data.

V. Evaluation of the Model

A. Loss Function Evaluation

In the realm of reservoir computing (RC) models, the choice of an apt loss function plays a pivotal role in optimizing performance. Different functions prioritize specific aspects of prediction errors, influencing the model's behavior. For instance, Mean Squared Error (MSE) accentuates penalization of larger errors, while Mean Absolute Error (MAE) exhibits resilience to outliers. In a tailored approach, researchers have successfully employed a combination of MSE and Kullback-Leibler divergence to enhance spectrum sensing accuracy in quantized RC systems [45].

In our RC model, a meticulous evaluation identified Median Absolute Error (MAE) as the most effective loss function. Its reduced sensitivity to outliers proved beneficial for handling noisy data, while its ability to provide a robust measure of central tendency surpassed Mean Squared Error. The bar plot in Fig. 4 reveals median absolute error as the superior loss function, followed by root mean squared error and mean absolute error. Mean squared error yielded the highest error, suggesting its relative unsuitability for this model. The simplicity of MAE facilitated swift training, critical for real-time processing and resource-constrained applications [46]. This strategic choice contributed to superior performance, underscoring the significance of tailored loss functions in advancing the efficacy of RC models.

B. Comparison with State-of-the-Art Models

RNNs are adept at processing sequences by storing information from previous time steps and integrating it with current hidden states. This capability is essential for tasks that involve temporal dependencies. However, RNNs are notoriously challenged by the vanishing gradient problem, where the gradients used in training diminish across layers or time steps, hindering the learning of long-range dependencies.

To overcome these limitations, GRUs and LSTMs have been developed. These architectures, being advanced variants of RNNs, incorporate gating mechanisms to better regulate the flow of information. This design enables them to retain long-term dependencies within sequences more effectively than standard RNNs. However, the added complexity of these gating mechanisms leads to increased computational demands, resulting in longer training durations.

Conversely, CNNs, though traditionally associated with spatial data processing, can also be adapted for temporal data. By leveraging their capability to capture spatio-temporal

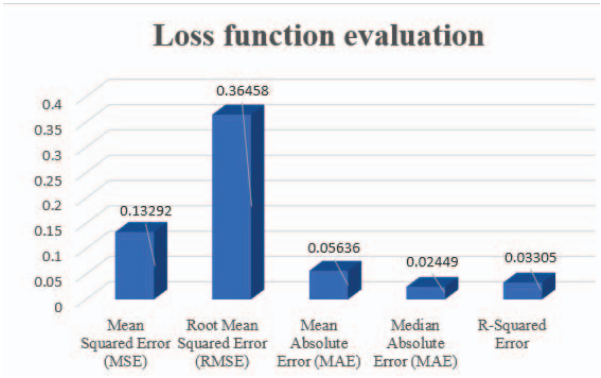
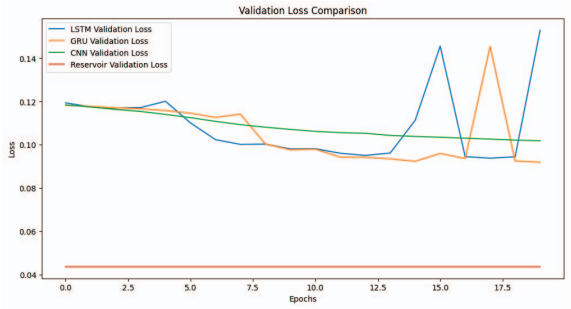


Fig. 4: Loss function evaluation of RC model

(a)



(b)

Fig. 5: Comparison of training and validation loss

TABLE I: Comparison of the State-of-the-art Models

	[47]	[48]	[49]	[50]	[51]	[52]	This Work
Network	LSTM+GRU	CRNN	R-CNN	GRU	Conditional Variance Autoencoder	GSAN + RNN/LSTM	AutoEncoder RC + SVM
Data Type	Radar Signal Sequences	LiDar Data	Images	Images, GPS	LiDar Data	Image + LiDar Data	GPS + Timestamp
Accuracy (%)	96	81	96	N/A	69.2	92.71	99
Loss	N/A	MAE = 0.05173	0.06	MAE = 0.03	0.45	RMSE = 0.43	MAE = 0.02
Prediction Focus	Lane Change Interference	Dynamic Occupancy Grid Mapping	Traffic Signal Image Recognition	Trajectory Prediction	Trajectory Prediction	Lane Change Classification	Trajectory Prediction

Fig. 6: Comparison of training times with different models

continuities, CNNs can effectively predict trajectories and sequences. They benefit from parallel processing, which often makes them faster and more computationally efficient than their RNN counterparts. However, CNNs might not always capture long-term temporal dependencies as effectively as GRUs or LSTMs.

In contrast to these models, Reservoir Computing offers a unique approach. Our reservoir model maintains a large, dynamically rich, yet fixed recurrent layer, training only the output weights. This setup enables efficient temporal data processing with significantly reduced training complexity. Since the reservoir only trains the output layer, it also significantly reduces the training time and complexity.

Comparative analyses, as depicted in Fig. 5(a) and (b), illustrate this point. Here, we evaluate the training and validation losses of our reservoir model against GRUs, LSTMs, and CNNs. The results indicate that our reservoir computing model achieves substantially lower losses than the other models, demonstrating its efficacy in learning from temporal data.

Furthermore, as shown in Fig. 6, the training time of our reservoir model is markedly less, clocking in at only 2 seconds. This is in stark contrast to the training times of LSTM (83s), GRU (104s), and CNN (1700s). Such a significant reduction in training duration without compromising on performance underscores the potential of reservoir computing as a highly efficient alternative for processing temporal sequences. The comparison with the state-of-the-art models from Table I shows that our work has a significantly low loss and an extremely high accuracy of 99%.

C. Determining Accuracy of Model

The categorization of combined driving scores into distinct states such as 'dangerous', 'okay', 'safe', and 'very good' is a crucial step in our analysis. This process involves normalizing the scores and then categorizing them based on predefined ranges. This categorization transforms the continuous score data into discrete classes that represent various levels of driving proficiency. The categorized variables thus obtained serve as the basis for making predictions about driver safety.

We employ a Support Vector Machine (SVM) classifier for this predictive task. SVM is renowned for its robustness, especially in scenarios where the distinction between classes is not immediately clear-cut. Its ability to find the optimal hyperplane that separates different classes makes it particularly effective for our purpose. In our analysis, the SVM classifier demonstrates high accuracy, achieving a remarkable 99.3% in identifying the categories of drivers. This level of accuracy underscores the SVM's capability to handle complex classification tasks with a high degree of precision.

The features fed into the classifier are derived from the reservoir computing model. Reservoir computing, known for its efficiency in processing temporal data, extracts meaningful patterns from the input features. These patterns, which encapsulate crucial information about driving behavior over time, are then used as inputs to the SVM classifier. The classifier, in turn, utilizes these inputs to differentiate between good and bad drivers.

By leveraging the strengths of both reservoir computing and SVM, our approach provides a nuanced understanding of driver behavior. The reservoir helps in capturing the temporal dynamics of driving data, while the SVM effectively categorizes these dynamics into distinct classes of driving quality. This synergistic use of reservoir computing for feature extraction and SVM for classification forms the backbone of our system, enabling us to reliably identify various categories of drivers based on their driving scores.

VI. Conclusion

In this work, we presented a novel approach for trajectory prediction by employing a combination of autoencoder and reservoir computing techniques. The autoencoder, serving as the initial stage of our model, effectively compresses and reconstructs the input features, achieving a minimal Mean Absolute Error (MAE) loss of 0.02. This performance surpasses that of more conventional models such as LSTM, GRU, and

CNN, highlighting the efficacy of our method in capturing the essential characteristics of trajectory data. Furthermore, the integration of reservoir computing substantially enhances the model's capability to process temporal dynamics, a critical aspect of trajectory prediction. This combination not only improves prediction accuracy but also significantly reduces computational overhead, as evidenced by the remarkably low training time of just 2 seconds. Such efficiency is particularly advantageous in real-time or resource-constrained environments. The application of the SVM classifier in our model further reinforces its robustness, achieving an impressive accuracy of 99.3%. This high accuracy rate indicates the model's strong discriminative power in classifying different trajectory patterns, making it highly reliable for practical applications in trajectory prediction. Overall, our approach demonstrates a significant advancement in trajectory prediction, offering a balance of high accuracy, low computational cost, and rapid processing. This makes it an excellent choice for various applications, ranging from autonomous vehicle navigation to traffic management systems, where accurate and efficient trajectory prediction is paramount.

REFERENCES

[1] L. Chen, Y. Li, C. Huang, Y. Xing, D. Tian, L. Li, Z. Hu, S. Teng, C. Lv, J. Wang, D. Cao, N. Zheng, and F.-Y. Wang, "Milestones in autonomous driving and intelligent vehicles—part i: Control, computing system design, communication, hd map, testing, and human behaviors," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 9, pp. 5831–5847, 2023.

[2] F. Xu, F. Xu, J. Xie, C.-M. Pun, H. Lu, and H. Gao, "Action recognition framework in traffic scene for autonomous driving system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 22 301–22 311, 2021.

[3] Y. Li, H. Wang, L. M. Dang, T. N. Nguyen, D. Han, A. Lee, I. Jang, and H. Moon, "A deep learning-based hybrid framework for object detection and recognition in autonomous driving," *IEEE Access*, vol. 8, pp. 194 228–194 239, 2020.

[4] Y. Ma, Z. Wang, H. Yang, and L. Yang, "Artificial intelligence applications in the development of autonomous vehicles: A survey," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 2, pp. 315–329, 2020.

[5] W. Wang, L. Wang, C. Zhang, C. Liu, L. Sun *et al.*, "Social interactions for autonomous driving: A review and perspectives," *Foundations and Trends® in Robotics*, vol. 10, no. 3-4, pp. 198–376, 2022.

[6] M. R. Bachute and J. M. Subhedar, "Autonomous driving architectures: insights of machine learning and deep learning algorithms," *Machine Learning with Applications*, vol. 6, p. 100164, 2021.

[7] E. Marti, M. A. De Miguel, F. Garcia, and J. Perez, "A review of sensor technologies for perception in automated driving," *IEEE Intelligent Transportation Systems Magazine*, vol. 11, no. 4, pp. 94–108, 2019.

[8] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, 2016.

[9] S. S. Banerjee, S. Jha, J. Cyriac, Z. T. Kalbarczyk, and R. K. Iyer, "Hands off the wheel in autonomous vehicles?: A systems perspective on over a million miles of field data," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018, pp. 586–597.

[10] D. Cao, X. Wang, L. Li, C. Lv, X. Na, Y. Xing, X. Li, Y. Li, Y. Chen, and F.-Y. Wang, "Future directions of intelligent vehicles: Potentials, possibilities, and perspectives," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 1, pp. 7–10, 2022.

[11] Y. Huang, J. Du, Z. Yang, Z. Zhou, L. Zhang, and H. Chen, "A survey on trajectory-prediction methods for autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 652–674, 2022.

[12] F. Leon and M. Gavrilescu, "A review of tracking and trajectory prediction methods for autonomous driving," *Mathematics*, vol. 9, no. 6, p. 660, 2021.

[13] J. Liu, X. Mao, Y. Fang, D. Zhu, and M. Q.-H. Meng, "A survey on deep-learning approaches for vehicle trajectory prediction in autonomous driving," in *2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2021, pp. 978–985.

[14] V. Lefkopoulos, M. Menner, A. Domahidi, and M. N. Zeilinger, "Interaction-aware motion prediction for autonomous driving: A multiple model kalman filtering scheme," *IEEE Robotics and Automation Letters*, vol. 6, no. 1, pp. 80–87, 2020.

[15] R. Zhang, L. Cao, S. Bao, and J. Tan, "A method for connected vehicle trajectory prediction and collision warning algorithm based on v2v communication," *International Journal of Crashworthiness*, vol. 22, no. 1, pp. 15–25, 2017.

[16] K. Okamoto, K. Berntorp, and S. Di Cairano, "Driver intention-based vehicle threat assessment using random forests and particle filtering," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 13 860–13 865, 2017, 20th IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896317329063

[17] Y. Guo, V. V. Kalidindi, M. Arief, W. Wang, J. Zhu, H. Peng, and D. Zhao, "Modeling multi-vehicle interaction scenarios using gaussian random field," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 3974–3980.

[18] P. Kumar, M. Perrollaz, S. Lefevre, and C. Laugier, "Learning-based approach for online lane change intention prediction," in *2013 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2013, pp. 797–802.

[19] H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, J. Schneider, D. Bradley, and N. Djuric, "Deep kinematic models for kinematically feasible vehicle trajectory predictions," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 563–10 569.

[20] F.-C. Chou, T.-H. Lin, H. Cui, V. Radosavljevic, T. Nguyen, T.-K. Huang, M. Niedoba, J. Schneider, and N. Djuric, "Predicting motion of vulnerable road users using high-definition maps and efficient convnets," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1655–1662.

[21] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[22] A. Zyner, S. Worrall, and E. Nebot, "A recurrent neural network solution for predicting driver intention at unsignalized intersections," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1759–1764, 2018.

[23] F. Altché and A. de La Fortelle, "An lstm network for highway trajectory prediction," in *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)*. IEEE, 2017, pp. 353–359.

[24] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *arXiv preprint arXiv:1605.07678*, 2016.

[25] Y. Huang and Y. Deng, "A hybrid model utilizing principal component analysis and artificial neural networks for driving drowsiness detection," *Applied Sciences*, vol. 12, no. 12, p. 6007, 2022.

[26] S. K. Chada, D. Görges, A. Ebert, R. Teutsch, and C. G. Min, "Learning-based driver behavior modeling and delay compensation to improve the efficiency of an eco-driving assistance system," in *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2022, pp. 415–422.

[27] A. Parizad and C. J. Hatziadoniu, "Cyber-attack detection using principal component analysis and noisy clustering algorithms: A collaborative machine learning-based framework," *IEEE Transactions on Smart Grid*, vol. 13, no. 6, pp. 4848–4861, 2022.

[28] J. Liu, L. N. Boyle, and A. G. Banerjee, "An inverse reinforcement learning approach for customizing automated lane change systems," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 9, pp. 9261–9271, 2022.

[29] F. De Vita, G. Nocera, D. Bruneo, and S. K. Das, "A novel echo state network autoencoder for anomaly detection in industrial iot systems," *IEEE Transactions on Industrial Informatics*, 2022.

[30] H. Zhang and D. V. Vargas, "A survey on reservoir computing and its interdisciplinary applications beyond traditional machine learning," *IEEE Access*, 2023.

[31] J. Kato, G. Tanaka, R. Nakane, and A. Hirose, "Proposal of reconstructive reservoir computing to detect anomaly in time-series signals," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–6.

[32] L. Wang, S. Chen, F. Chen, Q. He, and J. Liu, "B-detection: Runtime reliability anomaly detection for mec services with boosting lstm autoencoder," *IEEE Transactions on Mobile Computing*, 2023.
[33] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," *arXiv preprint arXiv:1312.6026*, 2013.
[34] K. Nakajima and I. Fischer, *Reservoir computing*. Springer, 2021.
[35] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, "Advances in optimizing recurrent networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 8624–8628.
[36] C. R. Vogel, *Computational methods for inverse problems*. SIAM, 2002.
[37] N. Soures and D. Kudithipudi, "Deep liquid state machines with neural plasticity for video activity recognition," *Frontiers in neuroscience*, vol. 13, p. 686, 2019.
[38] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer science review*, vol. 3, no. 3, pp. 127–149, 2009.
[39] C. Gallicchio, A. Micheli, and L. Pedrelli, "Deep reservoir computing: A critical experimental analysis," *Neurocomputing*, vol. 268, pp. 87–99, 2017.
[40] N. D. Irimia, M. Luchian, F. I. Lazar, and A. Ipatiov, "Performant fault tolerant control by using space vector modulation (svm) technique of a five phases bldc motor for autonomous driving applications," in *2022 10th International Conference on Systems and Control (ICSC)*, 2022, pp. 317–322.
[41] G. Hwang, D. Jung, Y. Goh, and J.-M. Chung, "Personal driving style-based adas customization in diverse traffic environments using svm for public driving safety," in *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*, 2022, pp. 1938–1940.
[42] E. Ali, "Global positioning system (gps): Definition, principles, errors, applications & dgps," *no. April*, 2020.
[43] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-drive: driving directions based on taxi trajectories," in *Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems*, 2010, pp. 99–108.
[44] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 316–324.
[45] S. Liu, L. Liu, and Y. Yi, "Quantized reservoir computing for spectrum sensing with knowledge distillation," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 15, no. 1, pp. 88–99, 2023.
[46] L. Li, L. Liu, Z. Zhou, and Y. Yi, "Reservoir computing meets extreme learning machine in real-time mimo-ofdm receive processing," *IEEE Transactions on Communications*, vol. 70, no. 5, pp. 3126–3140, 2022.
[47] L. Li, W. Zhao, C. Xu, C. Wang, Q. Chen, and S. Dai, "Lane-change intention inference based on rnn for autonomous driving on highways," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 6, pp. 5499–5510, 2021.
[48] M. Schreiber, V. Belagiannis, C. Gläser, and K. Dietmayer, "Dynamic occupancy grid mapping with recurrent neural networks," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 6717–6724.
[49] G.-Z. Tiron and M.-S. Poboroniuc, "Neural network based traffic sign recognition for autonomous driving," in *2019 International Conference on Electromechanical and Energy Systems (SIELMEN)*, 2019, pp. 1–5.
[50] P.-Y. Hsu, M.-L. Huang, W.-Y. Wang, and H.-H. Chiang, "Traffic agent trajectory prediction using a time sequence deep learning model with trajectory mapping for autonomous driving," in *2021 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, 2021, pp. 1–2.
[51] Z. Zhong, Y. Luo, and W. Liang, "Stgm: Vehicle trajectory prediction based on generative model for spatial-temporal features," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 18 785–18 793, 2022.
[52] L. Ye, Z. Wang, X. Chen, J. Wang, K. Wu, and K. Lu, "Gsan: Graph self-attention network for learning spatial–temporal interaction representation in autonomous driving," *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9190–9204, 2021.