Improved Network Traffic Prediction in VANETs using Stacking Ensemble Learning

Tabbah Nicola
ING3 IA2
CY Tech
Cergy, France
tabbahnico@cy-tech.fr

Sagdullin Damir
ING3 IA2
CY Tech
Cergy, France
sagdullind@cy-tech.fr

Laiolo Léo
ING3 IA2
CY Tech
Cergy, France
laiololeo@cy-tech.fr

***Abstract*—The amount of data generated in intelligent transportation systems (ITS) continues to increase. This is an opportunity to highlight the need for predictive traffic in vehicular networks. Accurate estimation of network connectivity in vehicular ad hoc networks (VANETs) is important for traffic monitoring and control.**

We conducted a comparative study on the performance of machine learning (ML) and deep learning (DL) algorithms in predicting network traffic in VANETs. Focusing on regression and binary classification functions, we examine current literature to identify state-of-the-art methods for network traffic prediction.

In this study, we specifically focus on Stacking Ensemble Learning (SEL) and propose a novel SEL model that integrates diverse ML and DL models for network traffic prediction in VANETs. Our findings aim to provide insights into the effectiveness of SEL for network traffic prediction and offer guidance for researchers and engineers in designing and optimizing intelligent transportation systems.

***Index Terms*—Machine Learning, Deep Learning, VANET, Network Traffic Prediction, Comparative Analysis, Stacking Ensemble Learning, Regression**

I. Introduction

The intelligent transportation (ITS) field is undergoing a revolutionary change, entering in a new era of efficiency and innovation in urban transportation. ITS integrates advanced technologies and aims to improve all transportation by optimizing traffic, ensuring safety and reducing the impact on environment. In this context, vehicle ad hoc networks (VANETs) appears to be important component for effective dynamic communication of vehicles. However, despite their promise, VANETs also have unique challenges, such as the intermittent nature of communication, varying vehicle densities, other numerous external factors and the need to make decisions on the fly. Solving these problems requires a deep understanding of traffic patterns in such networks.

The incorporation of artificial intelligence (AI) in understanding and predicting network traffic brings forth some advantages over classic statistical methods that significantly elevate the efficiency and responsiveness of ITS. AI's approach benefit lies in ability to discern complex patterns and relationships within vast datasets, enabling a nuanced understanding of dynamic traffic behaviors. The capacity of AI to process information swiftly and make real-time decisions aligns seamlessly with the demands of dynamic transportation environments.

In the realm of AI predictions, machine learning (ML) and deep learning (DL) technologies play key role, offering diverse methodologies specific to prediction tasks. For supervised learning, particularly in binary classification scenarios where the objective is to determine the presence or absence of network traffic, ML algorithms are often used. These models, trained on labeled datasets, leverage patterns and features to make predictions. On the other hand, in the context of unsupervised learning, regression methods come to the forefront, aiming to predict the quantity or intensity of network traffic. Unsupervised models delve into the intrinsic structures of data without labeled examples, making them able to predict continuous variables like traffic volume. For both approaches, the data crucial for predicting network traffic in VANETs typically includes Vehicle-to-Vehicle (V2V) and Vehicle-to-Roadside (V2R) communications. V2V communication involves interactions and information exchange among vehicles on the road, providing real-time updates about their positions, speeds, and road conditions. This direct communication network forms a dynamic and responsive system, enabling vehicles to share critical data to enhance safety and traffic efficiency. On the other hand, V2R communication involves the exchange of information between vehicles and roadside infrastructure, such as traffic lights or monitoring systems. This data often includes traffic signal timings, roadwork alerts, and other relevant information that contributes to understanding of the traffic environment. Beyond V2V and V2R, additional data sources, like the Internet of Things (IoT), can be harnessed to enrich the predictive models.

While various ML and DL algorithms have been explored for network traffic prediction in VANETs, Stacking Ensemble Learning (SEL) offers a promising approach to further improve prediction accuracy by combining the strengths of diverse models. This study delves into the potential of SEL for network traffic prediction by proposing a novel SEL model that integrates a carefully selected set of ML and DL base and meta models. We evaluate this model against individual baseline algorithms by analyzing the residual distributions of the models to gain a deeper understanding of their error profiles and identify potential areas for improvement. Our

findings aim to contribute to the advancement of intelligent transportation systems by providing researchers and engineers with guidance on selecting and optimizing network traffic prediction models.

II. Background and related work

In the subsequent sections, we will present a chronological overview of selected articles addressing the challenge of network traffic prediction. Our approach will unfold in a systematic order, allowing readers to navigate through the evolving landscape of methodologies proposed over time. Towards the conclusion of our exploration, we will consolidate our findings into a comparative table. This table will succinctly highlight the similarities and differences among the various methods studied, highliting their respective strengths and weaknesses. Our goal is to provide a concise reference for readers, emphasizing potential areas of improvement in the pursuit of more effective and resilient network traffic prediction models.

A. Network Traffic Prediction based on Diffusion Convolutional Recurrent Neural Networks [1]

The introduction of Diffusion Convolutional Recurrent Neural Networks (DCRNN) has significantly advanced traffic prediction techniques, eclipsing traditional approaches like ARIMA and standard machine learning methods that struggled with the intricate spatio-temporal dynamics of traffic data. By ingeniously integrating graph convolutions to understand spatial relationships within traffic networks with recurrent neural networks to capture temporal fluctuations, DCRNN offers unprecedented accuracy in forecasting traffic conditions. Despite the challenges of requiring substantial computational resources and extensive datasets for optimal training, the emergence of DCRNN marks a pivotal leap forward, promising a new era in traffic management and urban planning by harnessing the power of advanced machine learning to address complex urban challenges.

B. VANET Traffic Prediction Using LSTM with Deep Neural Network Learning [2]

The paper presents Long Short Term Memory (LSTM) AI model architecture for network traffic prediction in VANETs. The main advantage of this architecture is the ability of LSTMs to preserve long-term relationships. Which is beneficial in the context of VANET where lots of parameters change over time. To evaluate the performance of this approach, a simulation of VANET environment was used. In which different packet rates were tested and it was shown that model performed better with lower package rates, outperforming classic statistical methods. However no explanations were given concerning lower performance with higher package rates, nor possible structural improvements of LSTM architecture. As for the next steps, only training model on larger dataset was pointed out.

C. Network Traffic Prediction Model Considering Road Traffic Parameters Using Artificial Intelligence Methods in VANET [3]

This proposed model introduces the RF-GRU-NTP algorithm, a fusion of Random Forest (RF) and Gated Recurrent Unit (GRU), aimed at predicting network traffic (NT) based on both road traffic (RT) and V2V communication. The goal is to predict the presence of network traffic : it's a binary classification problem. The model uses predictions of road traffic from network traffic data, creating a combined approach to the interconnected dynamics of vehicular environments. Simulation results reveal superior performance in execution time and prediction accuracy compared to alternative algorithms.

Highlighting the inherent connection between road traffic and network traffic, this paper distinguishes itself from others that often consider these factors independently. The goal is to predict network traffic by simultaneously examining road traffic, offering a more comprehensive understanding of the interdependence between these variables.

Utilizing GPS datasets from CRAWDAD GATECH/VEHICULAR [4], the article distinguishes between V2V data for road traffic prediction (RTP) and V2R data for network traffic prediction (NTP). The work involves applying ML algorithms (RF, KNN, SVM, NB) to the V2R dataset for NTP, and DL algorithms (LSTM, Bi-LSTM, GRU) to the V2V dataset for RTP. The combined model, RF-GRU-NTP, extracts essential features of combined V2V and V2R dataset using RF and predicts NT with GRU based on both road and network features.

The RF-GRU-NTP model outperforms standalone algorithms, showcasing 70% less execution time compared to LSTM or Bi-LSTM. However, the study can be critiqued for testing a limited set of models and we can proposee exploring more modern models, such as transformers or large language models (LLM), to enhance predictive capabilities. Moreover, authors propose as well implementing their model on big data ie bigger volume and flow of data.

D. Fed-NTP: A Federated Learning Algorithm for Network Traffic Prediction in VANET [5]

Because VANET data come from drivers and pedestrians, some researches tried to keep the privacy of the network users. In this paper they implemented Federated Learning (FL) algorithm to predict the sender speed from traffic information. The proposed model was trained on decentralized datasets generated with a dataset [4] containing 39,998 records of VANET communication. The FL was based on LSTM though the dataset isn't large enough to train DL model. Moreover the proposed FL model was compared to only one other FL model based on Gated Recurrent Unit (GRU) and three DL model trained on centralized data : LSTM, RNN and GRU. Despite the lack of comparison and data, this paper showed that it is possible to predict traffic network while keeping the privacy of the users.

TABLE I
RELATED PAPERS COMPARISON

Paper	Proposed method	Contribution	Studied data	Prediction type
Network Traffic Prediction based on Diffusion Convolutional Recurrent Neural Networks	DCRNN	Predicting traffic loads in telecom networks due to its efficiency to capture topological properties of the network.	Network data	regression
VANET Traffic Prediction Using LSTM with Deep Neural Network Learning	LSTM	Introduce LSTM in network traffic prediction in VANET context. Test it in a simulated environment with different packet rates.	Dataset from VANET simulation	regression
Network Traffic Prediction Model Considering Road Traffic Parameters Using Artificial Intelligence Methods in VANET	RF-GRU-NTP more accurate than pure algorithms	Take into account road traffic parameters for network traffic prediction	V2V, V2R	binary classification
Fed-NTP: A Federated Learning Algorithm for Network Traffic Prediction in VANET	Federated learning algorithm (Fed-NTP)	Network sender speed prediction while conserving data privacy	V2V, V2R	regression
An Ensemble-Based Machine Learning Model for Forecasting Network Traffic in VANET	Stacking Ensemble Learning with Booster Model (STK–EBM)	Comparison of ML and EL to predict existence of traffic	V2V, V2R	binary classification
Digital Twin for Transportation Big Data: A Reinforcement Learning-Based Network Traffic Prediction Approach	GAN, VANET, Deep Q Reinforcement Learning	The integration of Deep Q Learning and GAN to enhance traffic prediction in VANETs. High performance even with difficult contions(COVID).	V2V, V2I, V2R	regression

E. An Ensemble-Based Machine Learning Model for Forecasting Network Traffic in VANET [6]

Ensemble based machine learning model for forecasting network traffic in VANET was proposed in this paper. ML models have much more efficient prediction time than DL. The combination of ML algorithms permitted to achieve higher performance in all the classification metrics compared to baseline ML models. The used dataset was the same than in the previous paper, which contains few data for ML. Moreover, the combined ML models were optimized for baseline prediction instead of optimizing the hole combination. This paper showed that Ensemble Learning (EL) can increase ML efficiency. It would be interesting to compare it with DL baseline models then with EL based on DL.

F. Digital Twin for Transportation Big Data: A Reinforcement Learning-Based Network Traffic Prediction Approach [7]

The realm of predicting network traffic has seen a flurry of innovation recently, with a shift towards employing deep learning and machine learning techniques that promise not just better accuracy but also a smarter approach to handling data. Reinforcement learning stands out in this landscape, offering a flexible way to adapt to the unpredictable nature of network traffic. It's like the method learns from its environment, getting better over time. On another front, Generative Adversarial Networks (GANs) have been making waves for their ability to churn out data that mimics real-world traffic patterns, a boon for creating more reliable models. Merging these two powerful tools has set a new standard in the quest for smarter network management. The synergy between reinforcement learning and GANs is not just a technical milestone; it represents a leap towards networks that can predict and adapt, almost as if they're thinking ahead. This isn't just about machines crunching numbers; it's about crafting solutions that evolve and learn, much like we do.

G. Summary of related work network traffic prediction

This subsection summarizes all the related works on traffic network prediction by Artificial Intelligence (AI) described in this paper.
The Table. 1 represents a summary of the paper described in the previous subsections.
Based on those researches we'll implement their models on a uniformed setup to allow a fair comparison.

III. METHODOLOGY

From the approaches described in the previous section, we decided to extend the work on Stacking Ensemble Learning (SEL) by changing prediction type to regression as well as improving model architecture and using richer vehicle-to-anything (V2X) dataset.

This section delves into the methodology employed in this study, specifically focusing on the implementation of a SEL model based on ML and DL for predicting network traffic in VANETs.

A. Ensemble Learning

Ensemble learning (EL) represents a powerful machine learning paradigm that strategically combines multiple individual models to achieve superior predictive performance. The underlying principle of EL lies in the concept of "wisdom of the crowd," where the collective knowledge of diverse models often surpasses the capabilities of any single model in isolation. By aggregating predictions from multiple models, EL effectively mitigates the risks associated with over-fitting

and model bias, leading to more robust and generalizable predictions.

Network traffic prediction is a complex task. It is so complex that ML models usually aren't able to handle it perfectly. To compensate their weaknesses, we combine ML models. This is what we call Ensemble Learning (EL).

In this study, we specifically leverage the Stacking Ensemble Learning approach, which combines the strengths of both bagging and boosting to achieve optimal predictive performance.

B. Stacking Ensemble Learning

Stacking, also known as stacked generalization, is a sophisticated EL technique that employs a layered architecture to combine predictions from multiple base models. The key components of a stacking model include:

1) Base Models:
   These are the individual ML models that form the foundation of the ensemble. In our study, we utilize a diverse set of base models, including Logistic Regression, Support Vector Machines (SVM), and Random Forests, to capture different aspects of the data.
2) Meta-Model:
   The meta-model sits atop the base models and learns to combine their predictions to generate the final prediction. Typically, a simple model like Logistic Regression is used as the meta-model.

The training process of a stacking model involves two stages:

1) Training Base Models:
   Each base model is trained independently on the training data.
2) Training Meta-Model:
   The predictions of the base models on the training data are used as input features to train the meta-model.

By learning from the combined predictions of the base models, the meta-model can effectively identify patterns and relationships that individual models might miss, leading to improved overall prediction accuracy.

C. Machine Learning Regressors

Machine learning provides a wide array of regression algorithms, each tailored to different types of data and modeling objectives. In this section, we explore various regressors, including Random Forest, Gradient Boosting, Ridge Regression, SVR, Extra Trees, XGBoost, LightGBM, Elastic Net, K-Nearest Neighbors (KNN), Decision Tree, AdaBoost, Bagging, Multi-layer Perceptron (MLP), Lasso Regression, Bayesian Ridge, Passive Aggressive, Huber Regression, Lasso Lars, Orthogonal Matching Pursuit (OMP), and Automatic Relevance Determination (ARD).

1) Passive Aggressive: Passive Aggressive is an online learning algorithm for regression that updates model parameters incrementally, making it suitable for streaming data and dynamic environments.

2) Orthogonal Matching Pursuit (OMP): OMP is a greedy algorithm for sparse signal reconstruction that sequentially selects features to minimize the residual error at each step.

3) Linear Regression: Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables by fitting a linear equation to the observed data. The model assumes a linear relationship between the predictors and the target variable, with the goal of minimizing the sum of squared differences between the observed and predicted values.

4) Huber Regression: Huber Regression combines aspects of linear regression and robust statistics by minimizing a combination of squared errors and absolute errors, providing robustness to outliers.

5) Lasso Regression: Lasso Regression adds L1 regularization to linear regression, promoting sparsity by driving some coefficients to zero. It is effective for feature selection and model interpretability.

6) Lasso Lars: Lasso Lars (Least Angle Regression) is a variant of Lasso Regression that efficiently computes the entire solution path, making it suitable for high-dimensional datasets.

7) Ridge Regression: Ridge Regression adds L2 regularization to linear regression, preventing overfitting by penalizing large coefficients. It is useful for handling multicollinearity in datasets.

8) Elastic Net: Elastic Net combines L1 and L2 regularization to overcome the limitations of Ridge and Lasso Regression. It encourages sparse solutions while handling multicollinearity.

9) Bayesian Ridge: Bayesian Ridge is a Bayesian variant of Ridge Regression that estimates coefficients using a probabilistic framework, providing uncertainty estimates along with point estimates.

10) SVR (Support Vector Regression): SVR adapts the principles of Support Vector Machines to regression tasks. It finds the hyperplane that best fits the data within a specified margin, utilizing a subset of training examples as support vectors.

11) KNN (K-Nearest Neighbors): KNN predicts the target value by averaging the values of its k-nearest neighbors. It is a simple and intuitive non-parametric algorithm but can be computationally expensive for large datasets.

12) Multi-layer Perceptron (MLP): MLP is a type of artificial neural network with multiple layers of nodes (neurons) that learn complex patterns in the data through forward and backward propagation of errors.

13) ARD (Automatic Relevance Determination): ARD is a Bayesian regression method that automatically determines the relevance of features by estimating their posterior probabilities, facilitating automatic feature selection.

14) Decision Tree: Decision Tree recursively partitions the feature space into subsets, making binary decisions at each node based on feature values. It is interpretable and can capture complex interactions in the data.

15) Regression Tree: A regression tree is a decision tree-based machine learning algorithm used for predicting continuous target variables. It recursively partitions the feature space into regions based on feature values, aiming to minimize the variance of the target variable within each partition. Each leaf node of the tree represents a predicted value for the target variable, typically the mean value of the target variable within that region.

16) Random Forest: Random Forest is an ensemble learning method that constructs multiple decision trees during training and averages their predictions for regression tasks. It mitigates overfitting and improves robustness by randomly selecting subsets of features at each split.

17) Extra Trees: Extra Trees is similar to Random Forest but introduces additional randomness by selecting random feature splits, leading to faster training times and potentially improved performance.

18) Gradient Boosting: Gradient Boosting builds predictive models sequentially by combining weak learners, in a stepwise fashion. It minimizes a loss function by adding trees that correct errors made by previous models, resulting in strong predictive performance. In this paper, Regression Trees were used as weak learners.

19) XGBoost: XGBoost is an optimized implementation of Gradient Boosting that incorporates regularization, tree pruning, and parallel computation to enhance model performance and efficiency.

20) LightGBM: LightGBM is a gradient boosting framework that focuses on speed and efficiency. It uses a novel tree construction algorithm and histogram-based splitting to achieve faster training times.

21) AdaBoost: AdaBoost combines multiple weak learners to create a strong classifier. It assigns higher weights to misclassified examples, focusing subsequent models on difficult-to-classify instances. In this paper, Decision Trees were used as weak learners.

22) Bagging: Bagging (Bootstrap Aggregating) builds multiple models using bootstrapped samples of the training data and aggregates their predictions, reducing variance and improving generalization. In this paper, Decision Trees were used as weak learners.

These regressors offer a diverse toolkit for modeling various types of data and addressing different regression challenges. Understanding their principles and characteristics is essential for selecting the most appropriate algorithm for a given modeling task and optimizing its performance effectively.

D. Dataset description [8]

This study utilizes the Berlin V2X dataset [8], a collection of vehicular network data gathered in Berlin, Germany. The dataset presentet in parquet format contains various measurements related to cellular network performance, vehicle kinematics, and environmental conditions.

For our analysis, we specifically focus on downlink data rate (datarate) measurements as the target variable for network traffic prediction.

Also note that dataset provides data provided by 2 cell operators. Measurements are not equivalent for both of them. So we decided to work only with operator 1.

Then, we chose 39 measurements across three main categories for the prediction task :

1) Cellular Network Parameters (22 features):
   These features provide information about the cellular network's performance and signal quality experienced by the vehicles. They are further divided into two subcategories:
   a) Primary Serving Cell (PCell) Features (11 features):
      - PCell_RSRP_max:
        Maximum Reference Signal Received Power (dBm). Indicates the strongest signal strength received from the primary serving cell.
      - PCell_RSRQ_max:
        Maximum Reference Signal Received Quality (dB). Represents the quality of the received signal from the primary serving cell.
      - PCell_RSSI_max:
        Maximum Received Signal Strength Indicator (dBm). Measures the overall power received from the primary serving cell, including signal and noise.
      - PCell_SNR_1 & PCell_SNR_2:
        Signal-to-Noise Ratio (dB) on two different antennas. Indicates the ratio of signal power to noise power received from the primary serving cell.
      - PCell_Downlink_Num_RBs:
        Number of Resource Blocks allocated for downlink transmission. Represents the amount of bandwidth dedicated to data transmission from the primary serving cell to the vehicle.
      - PCell_Downlink_TB_Size:
        Transport Block size (bits) for downlink transmission. Indicates the amount of data transmitted in each transmission block from the primary serving cell.
      - PCell_Downlink_Average_MCS:
        Average Modulation and Coding Scheme used for downlink transmission. Represents the efficiency of data transmission based on the modulation and coding scheme employed.
      - PCell_Downlink_Average_MCS:
        Bandwidth available for downlink transmission from the primary serving cell (PCell) to the user equipment (e.g., a vehicle) in megahertz
      - PCell_Cell_Identity:
        Unique identifier of the primary serving cell.
      - PCell_freq_MHz:
        Frequency (MHz) of the primary serving cell.
   b) Secondary Serving Cell (SCell) Features (10 features):

These features mirror the PCell features but pertain to the secondary serving cell, which provides additional network coverage and capacity.

2) Vehicle Kinematics (5 features):
These features describe the movement and state of the vehicles within the network:
   - Latitude & Longitude:
     Geographic coordinates (degrees) of the vehicle.
   - Altitude:
     Height (meters) of the vehicle above sea level.
   - speed_kmh:
     Vehicle speed (kilometers per hour).
   - COG:
     Course Over Ground (degrees). Represents the vehicle's direction of movement relative to true north.
3) Environmental Conditions (12 features):
These features capture the surrounding environment's influence on network performance and traffic flow:
   - precipIntensity:
     Precipitation intensity (millimeters per hour).
   - precipProbability:
     Probability of precipitation (%).
   - temperature: Air temperature (degrees Celsius).
   - apparentTemperature:
     Apparent temperature (degrees Celsius), considering factors like humidity and wind speed.
   - dewPoint:
     Dew point temperature (degrees Celsius).
   - humidity:
     Relative humidity (%).
   - pressure:
     Atmospheric pressure (millibars).
   - windSpeed:
     Wind speed (meters per second).
   - cloudCover:
     Cloud coverage (%).
   - uvIndex:
     Ultraviolet index.
   - visibility:
     Visibility (kilometers).
   - Traffic Jam Factor:
     An indicator of traffic congestion levels (%).

E. Data preprocessing

Before trying to predict network traffic, we need to preprocess our data. In this paper, we applied the following process.

1) Drop duplicated rows: so our model won't be biased by the most common cases and should be able to understand extreme cases.
2) Remove incomplete data: so our EL won't be biased by the inference function that could have been used. After first two steps we got a dataset containing 11546 rows.
3) Split data so we have one dataset to train our model and another one to evaluate it. Since our dataset is large enough, we decided to use only 10% for testing. So we end up with 10391 rows for training and 1155 rows for testing our SEL model.
4) Standard scaling: so EL doesn't have to deal with too big values. The formula below is applied, where X is the non treated dataset, X_p corresponds to the treated data which follows a Gaussian distribution.

$$X_p = \frac{X - \mu}{\sigma}$$

$$\text{with } X_p \sim \mathcal{N}(\mu,\, \sigma^2)$$

We tested different combinations of standard and min-max scaling applied to features columns and/or datarate target column. But applying standard scaling to only features columns showed the best results. Essentially, scaling had little effect on tree based methods but improved by much performance of neural network based models like MLP.

F. Protocol to find the optimal SEL model

This section outlines the protocol for constructing an optimal Stacking Ensemble Learning (SEL) model for datarate regression in the context of VANETs. The objective is to develop a model that surpasses the performance of a chosen baseline model.

1) Choice and test of a baseline model
The first step involves selecting and evaluating a baseline model. This model serves as a reference point for comparing the performance of the SEL model. Typically, a commonly used and well-understood model, such as linear regression or a basic decision tree, is chosen as the baseline. Evaluating its performance on the data rate regression task establishes a benchmark for subsequent comparisons.
2) Test base models alone and choose relevant ones
Next, we individually test various base models described in section above that will potentially contribute to the SEL model. These models can include different types of algorithms, such as linear regression, support vector machines, random forests, and even deep learning models or other SEL models.
It's crucial to analyze the residual distribution of each base model. For the SEL model to reach its full potential, it should incorporate models that excel in different input scenarios or, in other words, make different types of errors. Simply choosing the best-performing base models may not necessarily improve the SEL model's performance. Instead, we should seek models that exhibit diverse residual distributions while still producing relevant results. However, models that generate nonsensical predictions will not contribute positively to the ensemble, even if their errors are unique.
3) Test chosen base models in a SEL model
After selecting base models with diverse and relevant error profiles, we integrate them into the SEL model. This involves training the base models on the training data and using their predictions as input features for

the meta-model. We then compare the performance of the SEL model to the individual baseline models to assess the potential improvement in datarate regression accuracy.

4) Test meta models
   The meta-model plays a critical role in the SEL model by learning how to optimally combine the predictions from the base models. Choosing the right meta-model is essential for maximizing the ensemble's performance. Typically, simpler models like linear regression or logistic regression are preferred as meta-models due to their ability to generalize well and avoid overfitting the combined predictions from the base models. We test different meta-model options and select the one that yields the best overall performance on the data rate regression task.
5) Optimize hyper-parameters
   Hyper-parameters are settings that control the learning process of both base and meta models. Optimizing these hyper-parameters can significantly improve the performance of the SEL model. Techniques like grid search or randomized search can be employed to systematically explore different hyper-parameter combinations and identify the settings that lead to the best results.

This protocole helps us build a strong SEL model for predicting network traffic in VANETs. By carefully picking different types of models that make different kinds of mistakes, choosing the right model to combine their predictions, and fine-tuning the settings of all the models, we can get the most accurate predictions possible. This way, we take advantage of the best parts of each model and put them together to make a better prediction than any single model could do on its own.

IV. Experimental results and performance analysis

In this section, we will describe the experimental results obtained with our proposed model and compare it with other ML algorithms.

A. Model Implementation

Our implementation environment is a Python Jupyter Notebook, located on the cloud. This Notebook is hosted by Google Collaboratory. It is a tool often used for ML research. We set Python to version 3 and ran our code on CPU. This hardware setting decreases the computation time of our SEL. The Google Collaboratory Notebook is located on a Google Drive, next to a copy of the dataset. Hosting both code and dataset on the same Google Drive allows to run the code completely on the cloud.

Moreover, to implement models discussed earlier we used sklearn library that is very convenient and provides tools to easily build and evaluate SEL models.

B. Evaluation Metrics

Evaluation metrics are crucial tools for assessing the performance of predictive models. In this study, we employ several commonly used metrics: R^2 (Coefficient of Determination), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE).

- *R^2 (Coefficient of Determination):*
  The coefficient of determination, denoted by R^2, measures the proportion of the variance in the dependent variable that is predictable from the independent variables. It is calculated as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

  where y_i represents the observed values, $\hat{y}_i$ represents the predicted values, $\bar{y}$ represents the mean of the observed values and n is the number of observations.
- *Mean Squared Error (MSE):*
  The Mean Squared Error (MSE) quantifies the average squared difference between the predicted values and the actual values. It is calculated as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

  where y_i represents the observed values, $\hat{y}_i$ represents the predicted values, and n is the number of observations.
- *Root Mean Squared Error (RMSE):*
  Root Mean Squared Error (RMSE) is the square root of the MSE, providing a measure of the average magnitude of the error. It is calculated as:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- *Mean Absolute Percentage Error (MAPE):*
  Mean Absolute Percentage Error (MAPE) measures the accuracy of a forecasting method by calculating the average percentage difference between the predicted and actual values. It is calculated as:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100$$

  where y_i represents the observed values, $\hat{y}_i$ represents the predicted values, and n is the number of observations.
- *Residuals and Density Analysis*
  In addition to traditional metrics, we conducted a novel analysis involving residuals and density. By plotting the density of residuals and fitting a Gaussian distribution, we aimed to assess the goodness of fit of the predictive model. Deviations from Gaussianity in the distribution of residuals may indicate areas for model improvement or data preprocessing. The formula to calculate residuals is:

$$e_i = y_i - \hat{y}_i$$

  where y_i represents the observed values and $\hat{y}_i$ represents the predicted values.

These evaluation metrics provide valuable insights into the performance of predictive models and help in comparing different algorithms or parameter settings. Plus, understanding residuals and their density distributions provides valuable insights into the ensemble model's performance and aids in diagnosing potential issues such as heteroscedasticity or bias in the predictions.

C. Proposed model

We tried different EL before finding the best combination of ML. Each of our EL was compared to the baseline model Random Forest. Indeed, Random Forest was the baseline ML algorithm that obtained the most interesting scores on evaluation metrics compared to: Passive Agressive, Orthogonal Matching Pursuit, Huber Regression, Lasso Regression, Lasso Lars, Ridge Regression, Elastic Net, Bayesian Ridge, SVR, KNN, MLP, ARD, Decision Tree, Extra Trees, Gradient Boosting, XGBoost, LightGBM, AdaBoost and Bagging.

The final architecture of our proposed SEL is represented by the figure 1.

We listed the hyper-parameters of ML composing the proposed SEL in this table II.

D. Results analysis

In this section, we explain all the ML evaluation scores, contained in this table III.

We can see that the majority of tested ML have a R^2 near 1. This means that they understand data variability, contrary to models with a low score. Negative R^2 score shows that the model couldn't understand the data, which is an elimination criterion for our SEL.

RMSE score seems to follow R^2 score. Here, this metric doesn't give more information to explain our ML performance.

MAPE score seems more interesting and is easier to explain. It is an error average in percent that shows that most of the tested ML predict values that are less than 15 % close to the true values. Which means that we can predict data rate with a marge of 15 %.

We observe that MAPE and R^2 score aren't fully corelated. Indeed, the SEL composed of all the tested ML (SEL_all_ML) has a better R^2 score than our proposed SEL (optimal_SEL) while it is the opposite with MAPE score.

Residual density 2 illustrates the residual distribution which shows visually how far our model can be from the perfect predictions. This metric is much more interesting with EL. It shows if some base learners are able to predict correctly alone and how their predictions influence the overall EL. For the proposed SEL 4, we keep ML that are efficient alone but the SEL containing all tested ML 3 has to deal with models that have residual densities far from the optimal normal Gaussian distribution. It means that the meta learner might be polluted by untruthful predictions, impacting its own performance. Nonetheless it is important to have various base model distributions in SEL model.

We choose to base our final comparison on MAPE score because this metric shows best the prediction error impact of our models. Therefore, we consider that the optimized SEL is more efficient because its error marge is less than 12 %. This error percentage seems low but we have to keep in mind that the predict values are really big numbers, which means that the marge error is big as well. Models can be better optimized.

V. Conclusion and Future Work

This study explored the potential of Stacking Ensemble Learning (SEL) for network traffic prediction in VANETs. While our optimal SEL model achieved slightly better performance metrics compared to the baseline Random Forest model, the improvement was not as significant as anticipated. This suggests that further exploration and refinement of the SEL architecture are necessary to achieve substantial performance gains.

Interestingly, the SEL model that incorporated all tested base models performed almost as well as the optimal SEL model. This highlights the ability of SEL to effectively leverage the strengths of diverse models while mitigating the impact of outlier predictions. However, it also indicates that our selection process for the optimal SEL model did not yield significant advantages in terms of prediction accuracy. To address the performance limitations of our optimal SEL model, several ways for future work can be considered:

1) Feature selection:
   Employing feature selection techniques on the all dataset columns (more than 100) to identify the most relevant variables could enhance prediction accuracy by providing more relevant data.
2) Missing data imputation:
   Completing missing data instead of dropping rows could retain valuable information and potentially improve model performance. The dataset size would increase from 11546 to around 19000 rows.
3) Deep learning base models:
   Incorporating deep learning models like CNNs and LSTMs as base models could capture complex temporal and spatial patterns in the data, leading to better predictions.
4) Training base models on different data subsets:
   Training different base models on data from different operators or data subsets could increase prediction diversity and potentially improve the SEL model's overall performance.
5) Meta-model exploration and optimization:
   Investigating different meta-models and optimizing their hyper-parameters could further enhance the ensemble's performance.
6) GPU acceleration:
   Utilizing libraries like PyTorch or TensorFlow with GPU acceleration could enable faster training and allow for more extensive experimentation with different model architectures and hyper-parameters using for example grid search or randomized search.

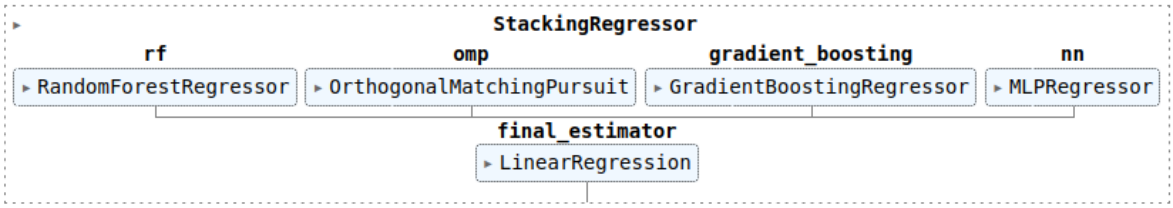

Fig. 1. Proposed Stacking Ensemble Learning model

TABLE II
SEL'S PARAMETERS

Learner	**ML**	**Hyper-parameters** (Non-default values in bold)
Base	MLP	hidden_layer_sizes=(100,), activation='relu', **solver='lbfgs'**, alpha=0.0001, batch_size='auto', learning_rate='constant', learning_rate_init=0.001, power_t=0.5, max_iter=200, shuffle=True, random_state=None, tol=0.0001, verbose=False, warm_start=False, momentum=0.9, nesterovs_momentum=True, early_stopping=False, validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10, max_fun=15000
Base	Random Forest	n_estimators=100, criterion='squared_error', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=1.0, max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, **n_jobs=-1**, **random_state=1337**, verbose=0, warm_start=False, ccp_alpha=0.0, max_samples=None, monotonic_cst=None
Base	Orthogonal Matching Pursuit	n_nonzero_coefs=None, tol=None, fit_intercept=True, precompute='auto'
Meta	Linear Regression	fit_intercept=True, copy_X=True, n_jobs=None, positive=False

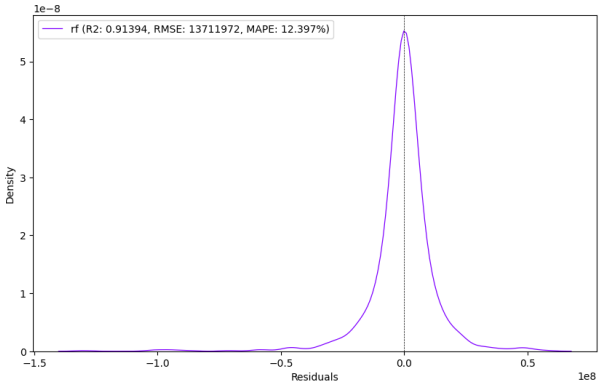

Fig. 2. Random Forest residual density

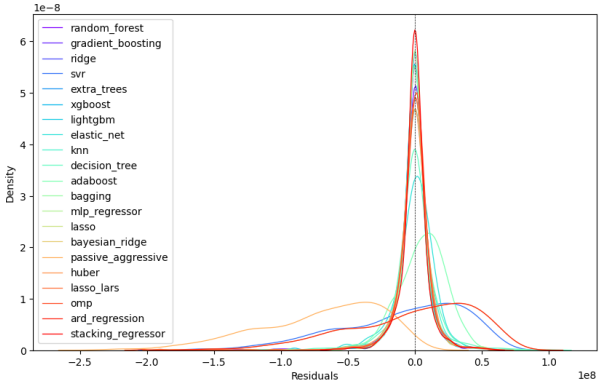

Fig. 3. Residuals densities comparison

TABLE III
MODELS' PERFORMANCE METRICS

Model	R^2	**RMSE**	**MAPE %**
random_forest	0.91394	13711972	12.397
gradient_boosting	0.90703	14251880	13.754
ridge	0.88150	16090160	13.354
svr	−0.03890	47641290	74.696
extra_trees	0.92815	12529195	12.374
xgboost	0.93378	12028337	12.505
lightgbm	0.92572	12739167	12.720
elastic_net	0.86334	17279224	19.496
knn	0.91521	13610499	15.922
decision_tree	0.84690	18288623	17.694
adaboost	0.82801	19383954	38.786
bagging	0.91453	13664538	12.418
mlp_regressor	0.94234	11223278	12.750
lasso	0.88151	16089440	13.357
bayesian_ridge	−0.00060	46754795	83.836
passive_aggressive	−2.19606	83561071	81.087
huber	0.87882	16271158	14.173
lasso_lars	0.88072	16143095	13.314
omp	0.87146	16757817	13.018
ard_regression	−0.00060	46754795	83.836
SEL_all_ML	0.94380	11080227	12.132
optimal_SEL	0.94340	11119842	11.910

REFERENCES

[1] D. Andreoletti, S. Troia, F. Musumeci, S. Giordano, G. Maier, and M. Tornatore, "Network traffic prediction based on diffusion convolutional recurrent neural networks," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2019, pp. 246–251.

[2] A. Refaee and A. Koucheryavy, "Vanet traffic prediction using lstm with deep neural network learning," pp. 281–294, 12 2020.

[3] S. S. Sepasgozar and S. Pierre, "Network traffic prediction model con-

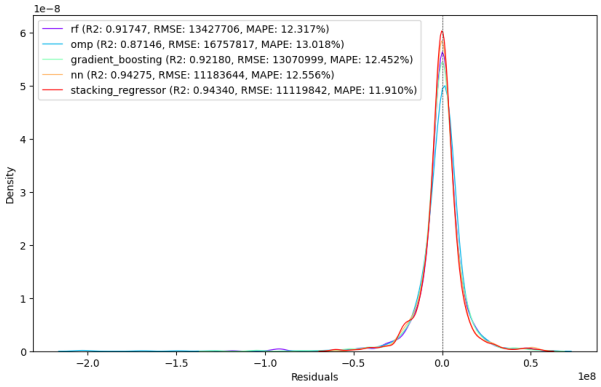

Fig. 4. SEL residuals densities

sidering road traffic parameters using artificial intelligence methods in vanet," *IEEE Access*, vol. 10, pp. 8227–8242, 2022.

[4] R. M. Fujimoto, R. Guensler, M. P. Hunter, H. Wu, M. Palekar, J. Lee, and J. Ko, "Crawdad gatech/vehicular (v. 2006-03-15)," 2022. [Online]. Available: https://dx.doi.org/10.15783/C74S3Z

[5] S. S. Sepasgozar and S. Pierre, "Fed-ntp: A federated learning algorithm for network traffic prediction in vanet," *IEEE Access*, vol. 10, pp. 119 607–119 616, 2022.

[6] P. A. D. Amiri and S. Pierre, "An ensemble-based machine learning model for forecasting network traffic in vanet," *IEEE Access*, vol. 11, pp. 22 855–22 870, 2023.

[7] L. Nie, X. Wang, Q. Zhao, Z. Shang, L. Feng, and G. Li, "Digital twin for transportation big data: A reinforcement learning-based network traffic prediction approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 1, pp. 896–906, 2024.

[8] R. Hernangómez, P. Geuer, A. Palaios, D. Schäufele, C. Watermann, K. Taleb-Bouhemadi, M. Parvini, A. Krause, S. Partani, C. Vielhaus, M. Kasparick, D. F. Külzer, F. Burmeister, F. H. P. Fitzek, H. D. Schotten, G. Fettweis, and S. Stańczak, "Berlin V2X: A Machine Learning Dataset from Multiple Vehicles and Radio Access Technologies," in *2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*, Florence, Italy, Jun. 2023, pp. 1–5.