

TDK-Thesis

Nguyen Duy Chien

Variable Projection Augmented Nonlinear Grey Bernoulli Networks

EÖTVÖS LORÁND UNIVERSITY

FACULTY OF INFORMATICS

DEPARTMENT OF NUMERICAL ANALYSIS



Author:

Nguyen Duy Chien

Computer Science BSc

III. Semester

Supervisor:

Dr. Péter Kovács

Habil. Associate Professor

Budapest, December 2024

Abstract

Time series forecasting and identification are crucial for real-world applications across diverse domains, including traffic management, financial analysis, healthcare, and energy systems. Although deep learning has gained prominence for its predictive power, traditional mathematical models continue to hold significance due to their interpretability and ability to work with small datasets.

In this thesis, we introduce a novel model-driven neural network architecture, VPNgbNET, which combines Variable Projection (VP) and the Nonlinear Grey Bernoulli Model (Ngb) to enhance forecasting accuracy and interpretability. By leveraging VP operators alongside the Ngb system, specifically tailored for time series forecasting, our approach enables the neural network to learn meaningful features, maintain interpretable parameters, and achieve a compact structure. This approach not only outperforms traditional mathematical models in terms of accuracy but also enhances the system's reliability and interpretability, making it more practical for real-world applications. This can be seen as a new paradigm for time series forecasting where we can replace Ngb by any other problem specific trajectory model. We discuss the underlying motivation, mathematical foundations, and implementation details of VPNgbNET, concluding with a case study on traffic pattern identification to illustrate its practical effectiveness and potential impact.

Contents

1	Introduction	3
2	VPNET: Variable Projection Networks	5
2.1	Variable projections	6
2.2	VPNet architecture	7
2.3	VP forward propagation	8
2.4	VP backpropagation	9
2.5	Network architecture	12
3	Nonlinear Grey Bernoulli Model for Daily Traffic Flow Pattern Identification	13
3.1	ODE-based nonlinear grey Bernoulli model	14
3.1.1	SLS and NLS Approaches for Parameters Estimation	15
3.2	IDE-based nonlinear grey Bernoulli model	17
3.2.1	Parameters and Initial Condition Estimation by SLS	18
3.2.2	Parameters and Initial Condition Estimation by NLS	20
3.3	Applications on daily traffic flow pattern identification	20
4	VPNgbNET: Variable Projection Augmented Nonlinear Grey Bernoulli Networks	22
4.1	VPNgbNET architecture	23
4.2	VP layer	24
4.3	Trajectory layer	25
4.3.1	ODE - based	25
4.3.2	IDE - based	26
4.4	Neural Networks and Residual Connection	27
4.5	VPNgbNET backpropagation	28
5	Experiments and Results	30
5.1	Simulation Data	30

5.2	Real Data: Daily traffic flow pattern identification	33
5.3	Real Data: Weekly traffic flow pattern identification	38
6	Summary and Discussion	43
6.1	Comparison	43
6.1.1	Comparison with Nonlinear Grey Bernoulli Model	43
6.1.2	Properties Comparison	44
6.2	Conclusion	44
6.3	Future Works	45
	Acknowledgements	47
A	Simulation Results	48
A.1	Experiment on Week identification	48
	Bibliography	50
	List of Figures	55
	List of Tables	57

Chapter 1

Introduction

Modern deep models have shown impressive performance in a variety of fields, but they have major drawbacks, including a strong dependence on big datasets and poor generalization skills. The key reasons for these constraints are the models' natural reliance on data-driven approaches and their restricted capacity to appropriately exploit domain-specific data [1]. Mathematical models have long been essential tools for describing and predicting the behavior of expert systems across various fields. Grounded in domain knowledge or physical laws, these models offer structured frameworks that help to understand and explain system dynamics. Through precise mathematical equations, they provide clarity and accuracy, enabling scientists and engineers to accurately capture and simulate real-world phenomena.

One widely-used approach for system identification and forecasting, especially in scenarios with limited data, is the nonlinear grey Bernoulli model (Ngb) [2]. This model, part of grey system theory, excels at handling incomplete or low-quality data by capturing nonlinear relationships and making forecasts based on limited information. It has been successfully applied in a variety of fields such as economic forecasting [3], environmental modeling [4], and energy demand prediction [5].

Despite advancements in simulating multiphysics problems, traditional mathematical models struggle with scalability, integrating noisy data, and recognizing complex patterns. They also face challenges in handling high-dimensional problems and complex mesh generation, limiting their overall effectiveness. To overcome these limitations, a knowledge-augmented deep learning paradigm has garnered increasing attention from researchers [1]. This approach combines domain knowledge with observational data in an effective way to provide data-efficient, generalizable, and interpretable deep learn-

ing algorithms. On the other hand, experienced knowledge is frequently expressed using less formal frameworks like probabilistic correlations, knowledge networks, and logical rules. Model-based deep learning techniques, sometimes referred to as model-driven AI, integrate mathematical frameworks and partial domain knowledge to enhance learning efficiency and performance, especially when working with small datasets [6].

For time series modeling to capture underlying patterns and relationships across many scales and dynamics, it is essential to integrate domain knowledge, physical constraints, and advanced data-driven methodologies to enhance both accuracy and scalability. Variable Projections Networks (VPNet) [7] offer a promising approach to addressing the challenges of high-dimensional data. By combining the power of projection-based methods with deep learning, VPNet efficiently reduces dimensionality and focuses the learning process on the most relevant features of the data. This makes VPNet highly effective for tasks where dimensionality reduction is crucial, such as in signal processing. Building on this capability, we integrate VPNet with the Ngb model, utilizing the VP layer for parameter estimation and the layer inspired by the Ngb model for forecasting. This integration, when applied to neural networks, enhances noise handling and complex pattern recognition, significantly improving the accuracy and robustness of predictions in time series applications.

The structure of this document is organized as follows. Chapter 2 provides an overview of the VPNet architecture, along with its mathematical foundations. Chapter 3 offers a detailed explanation of the nonlinear grey Bernoulli model and its application in identifying traffic patterns. Chapter 4 introduces our proposed model, highlighting its interpretability and accuracy. Chapter 5 presents an analysis of the experiments conducted, including a case study on traffic flow pattern identification and a discussion of the results obtained from applying our model. Chapter 6 concludes the work by comparing our model with existing methodologies and discussing potential directions for future research.

Chapter 2

VPNET: Variable Projection Networks

A traditional machine learning (ML) approach is to decompose the problem at hand into separate feature extraction and learning steps [8]. In this approach, the data is processed by selecting significant features based on domain knowledge, which are then passed to conventional ML models. This reduces the size and complexity of the data. Nevertheless, such manually chosen features are often ineffective for learning, as they may miss important patterns or introduce biases [8]. However, deep learning provides an alternative by automatically learning data representations, addressing some of the challenges of traditional methods. Features learned by neural networks can also be transferred as inputs for other procedures [9], for example discriminant correlation filters [10].

The increased number of hidden layers in Deep Neural Networks (DNNs) has enhanced their learning capabilities [11]. This layered architecture allows DNNs to work in stages: the initial layers extract features from the input data, while subsequent layers process these learned features. Among DNNs, Convolutional Neural Networks (CNNs) represent a specialized architecture that has emerged as the leading machine learning approach for processing 2D and 3D images and solving computer vision tasks [12, 13, 14]. The feature extraction layers here perform multiple convolutional filtering and pooling steps to reduce dimensions.

Despite their advantages, DNNs and CNNs present challenges, including increased computational complexity and training issues like overfitting and divergence. Due to numerous nonlinear interactions among parameters, these models are often considered "black-box" methods with limited interpretability, as their parameters lack clear physical significance. To address this limitation, the VPNet was designed to combine the expert

knowledge of traditional model-based approaches with the learning capabilities of neural networks. The proposed architecture is inspired by the so called model-driven NN concept, which is an emerging trend in signal processing [7].

2.1 Variable projections

VP [15] offers a methodological framework for handling nonlinear modeling problems that can be represented as

$$x \approx \sum_{k=0}^{n-1} c_k \Phi_k(\theta) = \Phi(\theta)c, \quad (2.1)$$

where $x \in \mathbb{R}^m$ and $\Phi_k \in \mathbb{R}^m$ indicate the input data to be approximated and the parametric function system, respectively. The notation $\Phi(\theta)$ denotes a matrix of size $\mathbb{R}^{m \times n}$ along with the function system itself. The function system Φ is composed of nonlinear parameters $\theta \in \mathbb{R}^p$ and linear parameters $c \in \mathbb{R}^n$. The nonlinear function is minimized by addressing the issue using the least-squares method, as described in the following equation

$$r(c, \theta) := \|x - \Phi(\theta)c\|_2^2. \quad (2.2)$$

The model's coefficients are linear when there are no nonlinear parameters (i.e., if θ is fixed). When r is minimized in relation to c , the well-known linear least-squares approximation appears. Orthogonal projection operators $P_{\Phi(\theta)}$ and Fourier coefficients can be used to express the ideal answer

$$c = \Phi^+(\theta)x \quad (2.3)$$

$$\hat{x} = P_{\Phi(\theta)}x = \Phi(\theta)\Phi^+(\theta)x \quad (2.4)$$

where $\Phi^+(\theta)$ denotes the Moore-Penrose pseudoinverse of matrix $\Phi(\theta)$. The principle is closely related to mathematical transformation techniques that can be read as orthogonal projections by a specified function system with a predefined θ , including Fourier and wavelet transforms. The coefficients c can be seen as features that VP extracted, whereas $\hat{x}$ is the outcome of dimension reduction and low-pass filtering. In the general case, minimizing r can be divided into two stages: first, minimizing with respect to the nonlinear parameters θ , and then calculating the linear parameters c using orthogonal projection.

Therefore, minimizing r is equivalent to minimizing the following VP functional, through the method of Golub and Pereyra [15]:

$$r_2(\theta) = \|x - \Phi(\theta)\Phi^+(\theta)x\|_2^2. \quad (2.5)$$

Mathematically, VP provides a framework for adaptive orthogonal transformations, enabling filtering and feature extraction through parametric function systems. Additionally, VP can serve as a solver for nonlinear optimization problems that are separable into linear and nonlinear parameters, expanding its potential applications [16, 17].

2.2 VPNet architecture

This architecture's main concept is to build a network that creates a composite model by combining the representation capabilities of VP and the prediction capabilities of NN [7]. The fundamental VPNet architecture is a feedforward neural network, where the initial layer(s) apply a VP operator, which is then passed through a fully connected, potentially DNN (see Figure 2.1). The structure is similar to that of CNNs in that the initial layer(s) of the network can be viewed as an integrated feature extraction method.

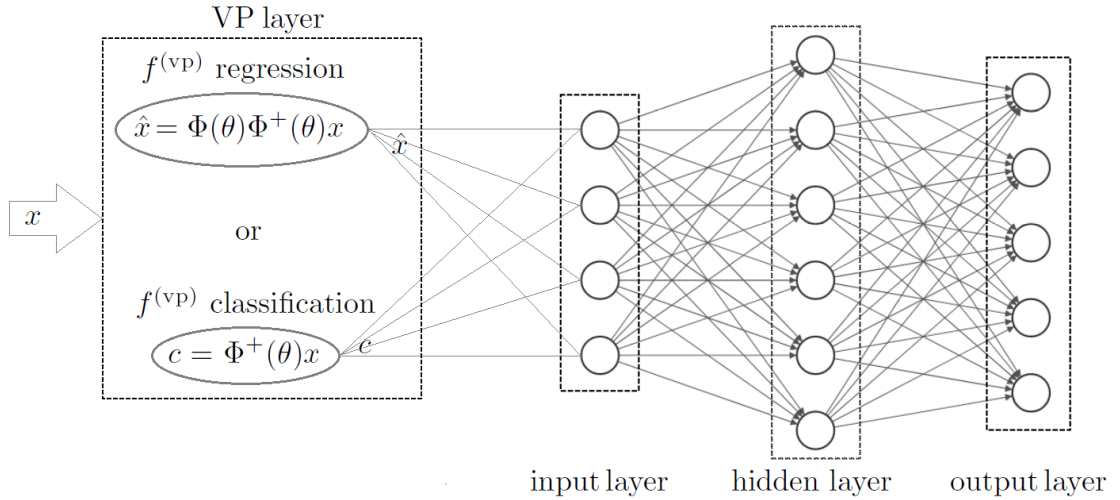


Figure 2.1: VPNET architecture [7]

The VP layer can behave in one of two ways, depending on its intended application. It performs either a filtering of the form [7]

$$f^{(vp)}(x) := \Phi(\theta)\Phi^+(\theta)x = \hat{x}, \quad (x \in \mathbb{R}^m), \quad (2.6)$$

or a feature extraction of the form

$$f^{(vp)}(x) := \Phi^+(\theta)x = c, \quad (x \in \mathbb{R}^m), \quad (2.7)$$

where $\theta \in \mathbb{R}^p$ denotes the nonlinear system parameters of the given function system Φ , as defined above.

These VP operators use the parametric system $\Phi(\theta)$ to refer to the orthogonal projection and the generic Fourier coefficients of the input x , as shown in Eq. (2.3, 2.4). Though feature extraction is appropriate for classification problems [7], our results in Chapter 4 demonstrate that feature extraction can also be used for parameters estimation for mathematical models, whereas the filtering method may be more appropriate for regression problems. The learnable parameters of the VP layer are included in the nonlinear system parameter vector θ .

The VPNet architecture introduces a model-driven approach for 1D signal-processing problems [7]. It incorporates domain knowledge through parameterized function systems, ensuring generality and interpretability, with VP layer parameters directly related to the shape of the input data. VPNet offers simplicity by reducing the model size compared to CNNs and DNNs, as the VP layer is governed by fewer parameters. Additionally, it supports compatibility by seamlessly integrating layers representing physical processes, enabling flexible model construction. Further details are provided in Chapter 4.

2.3 VP forward propagation

The low computational complexity comes from precomputing the nonlinearity and storing it in the matrix Φ . As a result, during evaluation, the VP layer only needs to perform a simple matrix multiplication, making the process much faster and more efficient. Furthermore, since the VP layer typically computes a relatively small number of features, subsequent layers can operate with reduced complexity [7]. Specifically, the weight matrix of a fully connected layer following the VP layer belongs to $\mathbb{R}^{n \times l}$, as opposed to $\mathbb{R}^{m \times l}$ without the VP layer. Here, n represents the number of coefficients, m denotes the length of the input signal, and l is the number of neurons in the fully connected layer. Since n is generally much smaller than m , the weight matrix size is significantly reduced for a fixed number of neurons l .

Solving the least-squares issue during training is the main source of computational

bottleneck in shallow neural networks with few hidden layers linked to the VP layer. This has an impact on VPNet’s numerical correctness and computing complexity [7]. Many function systems have been created to meet different application and analytical requirements. Notable examples include the Hermite System, which has been used specifically for medical signal processing [7], the Wavelet System [18], and the Malmquist–Takenaka System [19] for system identification problems.

2.4 VP backpropagation

Supervised learning involves training a model using labeled data, where the dataset consists of input-target pairs

$$(x_i, y_i) \quad \text{for } i = 1, 2, \dots, N.$$

Here, $x_i \in \mathbb{R}^m$ represents the input vector, and y_i is the corresponding target. For regression tasks, the target is a continuous vector, $y_i \in \mathbb{R}^s$, while for classification tasks, it can be represented as discrete labels, $y_i \in \mathbb{N}$, or as probabilities, $y_i \in [0, 1]^c$. This labeled data guides the model to learn the mapping from inputs to targets effectively. A general feed-forward neural network can be described as a composition of layer functions, where each layer applies a specific transformation to its input. This can be expressed mathematically as

$$NN_{\theta}(x) = (f_{\theta}^{(L)} \circ \dots \circ f_{\theta}^{(\ell)} \circ \dots \circ f_{\theta}^{(2)} \circ f_{\theta}^{(1)})(x),$$

where $x \in \mathbb{R}^m$ stands for the input samples, $f_{\theta}^{(\ell)}(x)$ denotes the function and $\theta^{(\ell)}$ denotes the parameters of the layer ℓ . Symbol θ denotes the set of parameters $\theta^{(\ell)}$. The layer functions $f^{(\ell)}$ can represent various operations, such as linear transformations, convolutional filters, nonlinear activation functions, pooling mechanisms, VP operators, and more [7]. Let

$$\hat{y}_i = NN_{\theta}(x_i) \quad (i = 1, 2, \dots, N)$$

indicate each input’s expected values. To train the network, an appropriate loss (or cost) function J is used to measure the error between predicted and target values, framing training as a minimization problem. Common loss functions include the Mean Squared Error (MSE), also called the least-squares cost function, which is typically used for regression problems where $y_i \in \mathbb{R}^s$, and the Binary Cross Entropy (BCE) loss, used for binary classification tasks where $y_k \in \{0, 1\}$. The following equations represent these two loss

functions

$$J_{MSE}(\theta) := \frac{1}{N} \sum_{i=1}^N \|y_i - \hat{y}_i\|_2^2, \quad (2.8)$$

$$J_{BCE}(\theta) := -\frac{1}{N} \sum_{i=1}^N (y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)). \quad (2.9)$$

Backpropagation, the widely-used technique for training feedforward networks, uses a stochastic gradient-descent optimization to minimize J (e.g. Adam [20], Adagrad [21], RMSprop [22]). Each layer parameter's gradient descent update formula is

$$\theta(\ell) := \theta(\ell) - \eta \frac{\partial J}{\partial \theta(\ell)}, \quad (2.10)$$

where $\eta > 0$ is called the learning rate. Backpropagation offers a recursive method of calculating the gradients using the chain rule, to put it simply

$$\begin{aligned} \frac{\partial J}{\partial f^{(\ell-1)}} &= \frac{\partial J}{\partial f^{(\ell)}} \cdot \frac{\partial f^{(\ell)}}{\partial f^{(\ell-1)}}, \\ \frac{\partial J}{\partial \theta(\ell)} &= \frac{\partial J}{\partial f^{(\ell)}} \cdot \frac{\partial f^{(\ell)}}{\partial \theta(\ell)}. \end{aligned}$$

Thus, the only partial derivatives of the layer function $f^{(\ell)}$ that need to be computed are those that relate to its parameters ($\frac{\partial f^{(\ell)}}{\partial \theta(\ell)}$) and its input ($\frac{\partial f^{(\ell)}}{\partial f^{(\ell-1)}}$). For the common layer types, these derivatives are typically well-known, and they can also be computed directly for the VP layers. The partial derivatives of the VP operators can be expressed in terms of their inputs and nonlinear parameters, providing a clear mathematical formulation for their behavior [15]. Specifically, for a filtering-type VP layer, as shown in Eq. (2.6), we have

$$f^{(vp)}(x) = \Phi(\theta)\Phi^+(\theta)x, \quad \frac{\partial f^{(vp)}}{\partial x} = [\Phi(\theta)\Phi^+(\theta)]^\top,$$

$$\frac{\partial f^{(vp)}}{\partial \theta_j} = \frac{\partial [\Phi(\theta)\Phi^+(\theta)]}{\partial \theta_j} x,$$

where

$$\partial [\Phi(\theta)\Phi^+(\theta)] = (I - \Phi\Phi^+)\partial\Phi\Phi^+ + [(I - \Phi\Phi^+)\partial\Phi\Phi^+]^T.$$

In the case of a feature-extraction-type VP layer in Eq. (2.7)

$$f^{(vp)}(x) = \Phi^+(\theta)x, \quad \frac{\partial f^{(vp)}}{\partial x} = [\Phi^+(\theta)]^T, \quad \frac{\partial f^{(vp)}}{\partial \theta_j} = \frac{\partial \Phi^+}{\partial \theta_j}x, \quad (2.11)$$

where

$$\partial \Phi^+ = -\Phi^+ \partial \Phi \Phi^+ + \Phi^+ [\Phi^+]^T \partial \Phi^T (I - \Phi \Phi^+) + (I - \Phi^+ \Phi) \partial \Phi^T [\Phi^+]^T \Phi^+.$$

Particularly in DNNs, a simple backpropagation implementation might result in numerical issues like divergence and overfitting. An ℓ_2 penalty on the weight parameters is usually inserted as a regularization term in the loss function to solve these problems. The resultant customized loss function is as follows

$$\begin{aligned} J_{\text{VPNet}}(\theta) &:= J(\theta) + \frac{\alpha}{N} \sum_{i=1}^N \frac{r_2(x_i; \theta^{(\text{VP})})}{\|x_i\|_2^2} \\ &= J(\theta) + \frac{\alpha}{N} \sum_{i=1}^N \frac{\|x_i - \Phi(\theta^{(\text{VP})})\Phi^+(\theta^{(\text{VP})})x_i\|_2^2}{\|x_i\|_2^2}, \end{aligned} \quad (2.12)$$

where $\alpha \geq 0$ controls the penalty effect.

The reasoning behind this regularization has two key aspects. First, it builds on prior findings that utilize VP as a feature extraction technique, indicating that precise VP approximations can yield meaningful features. Second, the optimal VPNet classifier is expected to capture the essential characteristics of the input signals, suggesting an effective approximation. Although this penalty term appears to challenge the backpropagation framework, the original method can be adjusted with a bypass step tailored specifically for the VP layer. The gradient with respect to the VP parameters is then updated as follows

$$\frac{\partial J_{\text{VPNet}}}{\partial \theta^{(\text{VP})}} = \frac{\partial J(\theta)}{\partial \theta^{(\text{VP})}} + \frac{\alpha}{N} \sum_{i=1}^N \frac{1}{\|x_i\|_2^2} \cdot \frac{\partial r_2}{\partial \theta^{(\text{VP})}}, \quad (2.13)$$

where

$$\partial r_2 = -2x_i^T (I - \Phi \Phi^+) \partial \Phi \Phi^+ x_i. \quad (2.14)$$

This approach enables the training of VPNet models in a manner similar to that of convolutional and fully connected neural networks (FCNNs) [7], thereby enhancing their scalability and applicability to a wide range of real-world problems. In Chapter 4, we

apply the same regularization technique to the output of a physics-based layer, ensuring that the model approximates the predictions of the mathematical model.

2.5 Network architecture

VPNet is structured with a VP layer as its foundational component, which is followed by fully connected neural networks (FCNNs). The outputs were understood as a probability distribution over expected output classes, while the inputs were $\mathbb{R}^m$ samples. The FC layers used nonlinear activation (SoftMax or ReLU) to carry out linear mappings. And offline backpropagation with Adam optimizer [20] was used for learning. The VP layer was of the feature-extraction type in Eq. (2.7).

In this thesis, we propose modifications to the basic architecture of VPNet to enhance its capability for dynamic system modeling. The key contributions are as follows:

- We define a new function system in the VP layer, which is derived from the nonlinear grey Bernoulli model (see Chapter 3).
- We introduced a physics-based component called the Trajectory layer, derived from the Ngb model (see Chapter 3).
- The VP and Trajectory layers are designed to capture the dynamics of daily traffic flow, while the connected deep neural network focuses on learning and modeling the noise.

Chapter 3

Nonlinear Grey Bernoulli Model for Daily Traffic Flow Pattern Identification

Time series data can display various patterns, with the inverted U-shape being a distinctive trend where values increase to a peak before subsequently declining [23]. This pattern is commonly observed across different domains, from economic cycles to population dynamics, representing a natural progression of growth followed by decay [24, 25].

The nonlinear grey Bernoulli model (Ngb) [2] has become an important tool for studying complex systems and has gained attention in areas like energy [26], traffic [27], and economics [28]. It improves on earlier models, like the grey Verhulst model [29], which was used to study patterns with an inverted U-shape over time. The Ngb model uses a simple and clear method to work with data, focusing on cases where the total sum of the data shows an S-shaped growth pattern. This makes it more flexible and accurate for different kinds of problems.

The nonlinear grey Bernoulli (Ngb) model works through three main steps: first, it accumulates the original time series to create the Cusum image; second, it fits the Bernoulli ordinary differential equation (ODE) to predict the Cusum values; and third, it restores these predictions to the original scale for final forecasts.

The fractional Cusum operator, new information priority Cusum operator, probabilistic Cusum operator, opposite direction Cusum operator, physics-preserving Cusum operator, and their combinations are some of the extensions of the Cusum operator that have been created to improve accuracy [30, 31, 32]. Additionally, by generalizing the Bernoulli ODE into more flexible forms, academics have broadened the whitening equation's use,

leading to a variety of extensions, including fractional-order, discrete-time, time-varying, and time-delayed variations [32, 5, 4, 3].

An integral equation (IE)-based direct modeling paradigm is provided by the Cusum operator, which is shown to be an Euler-based numerical approximation of an integral operator in linear grey system models [33]. Undoubtedly, the direct modeling paradigm offers a number of benefits, such as interpretability and uncertainty quantification. The first nonlinear grey Bernoulli model based on IDE was introduced by Lu and Xie [34]. However, in their research, the first element of the integral operator was specified as the observation at the initial time instant, which lacks theoretical support and introduces a logical flaw in the modelling process [30]. Furthermore, these models were unable to simultaneously estimate the initial conditions and structural parameters, leading to a more complex and inconsistent parameter estimation process. In this chapter, we will provide you with an outline of the concepts behind ordinary differential equation (ODE) [32] and integro-differential equation (IDE) [31], which were developed to address these limitations. Section 3.1 provides an overview and the mathematical foundation of the ODE-based model, while Section 3.2 introduces the IDE-based model as a solution to address these parameter estimation challenges.

3.1 ODE-based nonlinear grey Bernoulli model

The physics-preserving Cusum series [32] is defined as follows, assuming that the initial time series observations are $\{x(t_1), x(t_2), \dots, x(t_n)\}$

$$y(t_k) = \begin{cases} \eta_y, & k = 1 \\ \eta_y + \frac{1}{2} \sum_{i=2}^k h_i x(t_{i-1}) + \frac{1}{2} \sum_{i=2}^k h_i x(t_i), & k \geq 2, \end{cases} \quad (3.1)$$

where $\eta_y \in \mathbb{R}$ is an unknown parameter, and $h_k = t_k - t_{k-1}$ with $k \geq 2$ is the time interval.

Equation (3.1) provides a second-order accurate numerical approximation of the integral operator based on the trapezoid rule

$$y(t) = \eta_y + \int_{t_1}^t x(s) ds, \quad t \geq t_1, \quad (3.2)$$

where $\eta_y \in \mathbb{R}$ is an unknown parameter to guarantee the physics-preserving nature of this operator.

The whitening equation of the nonlinear grey Bernoulli model for the physics-preserving Cusum series $\{y(t_1), y(t_2), \dots, y(t_n)\}$ is defined as

$$\frac{dy(t)}{dt} = a_1 y(t) + a_2 [y(t)]^r, \quad y(t_1) = \eta_y, \quad t \geq t_1 \quad (3.3)$$

where a_1 , a_2 and $r \in \mathbb{R} \setminus \{0, 1\}$ are unknown structural parameters, η_y represents the unknown initial condition.

3.1.1 SLS and NLS Approaches for Parameters Estimation

Separable Nonlinear Least Squares (SLS) and Nonlinear Least Squares (NLS) are the two main approaches that can be used for parameter estimation in the model. While NLS is better suited for more complicated, nonlinear models and offers more flexibility in capturing complex interactions, SLS is frequently employed for linear models because of its efficiency and simplicity.

SLS Method

When Eq. (3.3) is subjected to the trapezoid rule-based scheme, the basic form of the nonlinear grey Bernoulli model is obtained as

$$\frac{y(t_k) - y(t_{k-1})}{h_k} = a_1 \left[\frac{1}{2} y(t_k) + \frac{1}{2} y(t_{k-1}) \right] + a_2 \left\{ \frac{1}{2} [y(t_k)]^r + \frac{1}{2} [y(t_{k-1})]^r \right\} + \varepsilon(k), \quad (3.4)$$

where $\varepsilon(k)$ indicates the error in numerical discretization. By treating a_1 and a_2 as functions of both the power coefficient r and the initial condition η_y , the pseudo regression in Eq. (3.4) produces the objective function

$$\mathcal{L}(r, \eta_y) = \sum_{k=2}^n \left\{ \frac{y(t_k) - y(t_{k-1})}{h_k} - a_1(r, \eta_y) \left[\frac{1}{2} y(t_k) + \frac{1}{2} y(t_{k-1}) \right] - a_2(r, \eta_y) \left\{ \frac{1}{2} [y(t_k)]^r + \frac{1}{2} [y(t_{k-1})]^r \right\} \right\}^2, \quad (3.5)$$

the minimization of which gives the estimates $(\hat{r}, \hat{\eta}_y)$ and subsequently, the estimates of a_1 and a_2 are obtained as

$$\begin{bmatrix} \hat{a}_1(\hat{r}, \hat{\eta}_y) \\ \hat{a}_2(\hat{r}, \hat{\eta}_y) \end{bmatrix} = (\Theta(\hat{r}, \hat{\eta}_y)^T \Theta(\hat{r}, \hat{\eta}_y))^{-1} \Theta(\hat{r}, \hat{\eta}_y)^T \mathbf{z}, \quad (3.6)$$

where

$$\mathbf{z} = \begin{pmatrix} \frac{y(t_2)-y(t_1)}{h_2} \\ \frac{y(t_3)-y(t_2)}{h_3} \\ \vdots \\ \frac{y(t_n)-y(t_{n-1})}{h_n} \end{pmatrix}, \quad (3.7)$$

$$\Theta(\hat{r}, \hat{\eta}_y) = \begin{pmatrix} \frac{1}{2}y(t_2) + \frac{1}{2}y(t_1) & \frac{1}{2}[y(t_2)]^{\hat{r}} + \frac{1}{2}[y(t_1)]^{\hat{r}} \\ \frac{1}{2}y(t_3) + \frac{1}{2}y(t_2) & \frac{1}{2}[y(t_3)]^{\hat{r}} + \frac{1}{2}[y(t_2)]^{\hat{r}} \\ \vdots & \vdots \\ \frac{1}{2}y(t_n) + \frac{1}{2}y(t_{n-1}) & \frac{1}{2}[y(t_n)]^{\hat{r}} + \frac{1}{2}[y(t_{n-1})]^{\hat{r}} \end{pmatrix}. \quad (3.8)$$

NLS Method

Since the trajectory of the ODE (see Eq. (3.3)) defines the regression function, the physics-preserving operator (see Eq. (3.1)) may be thought of as an implicit regression problem, meaning that the objective function is reached by the nonlinear regression

$$\mathcal{L}(a_1, a_2, r, \eta_y) = \sum_{k=1}^n [y(t_k) - y(t_k; a_1, a_2, r, \eta_y)]^2, \quad (3.9)$$

where $y(t; a_1, a_2, r, \eta_y)$ denotes the trajectory of the ODE. The estimates $\hat{a}_1, \hat{a}_2, \hat{r}, \hat{\eta}_y$ may be obtained after using Eq. (3.9) is minimized using a traditional nonlinear optimizer.

Lastly, the temporal response function corresponding to the Cusum series is stated as follows when the resulting estimations are plugged back into the trajectory

$$\hat{y}(t) = \left[\left(\hat{\eta}_y^{1-\hat{r}} + \frac{\hat{a}_2}{\hat{a}_1} \right) \exp(\hat{a}_1(1-\hat{r})(t-t_1)) - \frac{\hat{a}_2}{\hat{a}_1} \right]^{\frac{1}{1-\hat{r}}}, \quad t \geq t_1; \quad (3.10)$$

Applying Eq. (3.3) to the Cusum forecasts $\{\hat{y}(t_1), \hat{y}(t_2), \dots, \hat{y}(t_{n+p})\}$, we have the original forecasts $\{\hat{x}(t_1), \hat{x}(t_2), \dots, \hat{x}(t_{n+p})\}$, where $\{t_1, t_2, \dots, t_{n+p}\}$ is user-specified time instants, and $p \in \mathbb{N}$ denotes the forecasting horizon. Later, in Chapter 4, we'll introduce this as the Trajectory-ODE layer (see Section 4.3.1). When we apply the inverse integral operator, it gives us the time response function of the original series, which is expressed as

$$\begin{aligned} \hat{x}(t) = \frac{d}{dt}\hat{y}(t) &= (1 - \hat{r}) (\hat{a}_1 \hat{\eta}_y^{1-\hat{r}} + \hat{a}_2) \exp(\hat{a}_1(1 - \hat{r})(t - t_1)) \\ &\times \left[\left(\hat{\eta}_y^{1-\hat{r}} + \frac{\hat{a}_2}{\hat{a}_1} \right) \exp(\hat{a}_1(1 - \hat{r})(t - t_1)) - \frac{\hat{a}_2}{\hat{a}_1} \right]^{\frac{\hat{r}}{1-\hat{r}}}, \quad t \geq t_1. \end{aligned} \quad (3.11)$$

and this can be apply as Trajectory - IDE layer (see Section 4.3.2).

According to Wei and Xie [35], the nonlinear grey Bernoulli model employs an indirect modeling paradigm, which has drawbacks from a practical perspective. These include difficulties in understanding the propagation of measurement noise and parameter uncertainty. The following theorem helps adopt a direct modeling approach in order to handle this.

Theorem 1. *Under the condition (3.2), the ODE (3.3) is equivalent to the IDE*

$$\frac{d}{dt}x(t) = a_1x(t) + ra_2x(t) \left[\eta_y + \int_{t_1}^t x(s) ds \right]^{r-1}, \quad x(t_1) = \eta_x, \quad t \geq t_1, \quad (3.12)$$

where

$$\eta_x = a_1\eta_y + a_2\eta_y^r$$

is the unknown initial condition.

Theorem 1 shows that given an ODE (Eq. 3.3), we can always find an IDE (see Eq. 3.12) and their trajectories satisfy Eq. (3.2).

3.2 IDE-based nonlinear grey Bernoulli model

The following equation illustrates the relationship between noise and the resulting impact on the parameter estimation process

$$\tilde{x}(t_k) = x(t_k) + e(k), \quad (3.13)$$

where $x(t_k)$ and $e(k)$ denote the true value and measurement noise, respectively.

As a state equation, the IDE (Theorem 1) with an initial condition constraint is employed. The objective is to create trajectory and multi-step-ahead predictions after estimating the starting condition (η_y) and structural parameters (a_1, a_2, r) from noisy data.

3.2.1 Parameters and Initial Condition Estimation by SLS

By performing the integration of IDE (see Eq. 3.12) over the interval $[t_1, t]$, an integral equation is obtained

$$x(t) = a_1 \int_{t_1}^t x(s) ds + a_2 \left\{ \left[\eta_y + \int_{t_1}^t x(s) ds \right]^r - \eta_y^r \right\} + \eta_x, \quad (3.14)$$

The integral term is quantitatively estimated using the linear interpolation quadrature as

$$\int_{t_1}^{t_k} x(s) ds \approx \frac{1}{2} \sum_{i=2}^k h_i x(t_{i-1}) + \frac{1}{2} \sum_{i=2}^k h_i x(t_i). \quad (3.15)$$

Substituting Eq. (3.15) and the initial condition constraint $\eta_x = a_1 \eta_y + a_2 (\eta_y)^r$ into Eq. (3.14) yields

$$\begin{aligned} x(t_k) \approx & a_1 \left[\eta_y + \frac{1}{2} \sum_{i=2}^k h_i x(t_{i-1}) + \frac{1}{2} \sum_{i=2}^k h_i x(t_i) \right] \\ & + a_2 \left[\eta_y + \frac{1}{2} \sum_{i=2}^k h_i x(t_{i-1}) + \frac{1}{2} \sum_{i=2}^k h_i x(t_i) \right]^r \end{aligned} \quad (3.16)$$

By considering the noisy observations in Eq. (3.13) as approximations of the true values, leads to a pseudo regression

$$\begin{aligned} \tilde{x}(t_k) = & a_1 \left[\eta_y + \frac{1}{2} \sum_{i=2}^k h_i \tilde{x}(t_{i-1}) + \frac{1}{2} \sum_{i=2}^k h_i \tilde{x}(t_i) \right] \\ & + a_2 \left[\eta_y + \frac{1}{2} \sum_{i=2}^k h_i \tilde{x}(t_{i-1}) + \frac{1}{2} \sum_{i=2}^k h_i \tilde{x}(t_i) \right]^r \\ & + \varepsilon(k). \end{aligned} \quad (3.17)$$

where $\varepsilon(k)$ is the sum of the numerical discretization error and measurement noise.

Substituting $k = 2, \dots, n$ into Eq. (3.17) allows the objective function of the pseudo regression to be written in the following matrix form

$$J(a_1, a_2, r, \eta_y) = \left\| \tilde{\mathbf{x}} - \Xi(\tilde{\mathbf{x}}; r, \eta_y) \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \right\|_2^2 \quad (3.18)$$

where

$$\tilde{\mathbf{x}} = \begin{pmatrix} \tilde{x}(t_2) \\ \tilde{x}(t_3) \\ \vdots \\ \tilde{x}(t_n) \end{pmatrix},$$

and

$$\Xi(\tilde{x}; r, \eta_y) = \begin{pmatrix} \eta_y + \frac{1}{2} \sum_{i=2}^2 h_i \tilde{x}(t_{i-1}) + \frac{1}{2} \sum_{i=2}^2 h_i \tilde{x}(t_i) & [\eta_y + \frac{1}{2} \sum_{i=2}^2 h_i \tilde{x}(t_{i-1}) + \frac{1}{2} \sum_{i=2}^2 h_i \tilde{x}(t_i)]^r \\ \eta_y + \frac{1}{2} \sum_{i=2}^3 h_i \tilde{x}(t_{i-1}) + \frac{1}{2} \sum_{i=2}^3 h_i \tilde{x}(t_i) & [\eta_y + \frac{1}{2} \sum_{i=2}^3 h_i \tilde{x}(t_{i-1}) + \frac{1}{2} \sum_{i=2}^3 h_i \tilde{x}(t_i)]^r \\ \vdots & \vdots \\ \eta_y + \frac{1}{2} \sum_{i=2}^n h_i \tilde{x}(t_{i-1}) + \frac{1}{2} \sum_{i=2}^n h_i \tilde{x}(t_i) & [\eta_y + \frac{1}{2} \sum_{i=2}^n h_i \tilde{x}(t_{i-1}) + \frac{1}{2} \sum_{i=2}^n h_i \tilde{x}(t_i)]^r \end{pmatrix}. \quad (3.19)$$

The application of SLS [36] is made possible by the linear separability of the parameters a_1 and a_2 before minimizing the objective function (see Eq. 3.18). There are three phases in this process:

1. Parameters Estimation via SLS

From Eq. (3.18), it is easily derived that the least-squares estimates of the separable parameters are a_1, a_2 as functions of both η_y and r

$$\begin{bmatrix} \hat{a}_1(r, \eta_y) \\ \hat{a}_2(r, \eta_y) \end{bmatrix} = \left[\Xi(\tilde{x}; r, \eta_y)^T \Xi(\tilde{x}; r, \eta_y) \right]^{-1} \Xi(\tilde{x}; r, \eta_y)^T \tilde{x} \quad (3.20)$$

2. Parameters Estimation via NLS

$$J(r, \eta_y) = \left\| \tilde{x} - \Xi(\tilde{x}; r, \eta_y) \left[\Xi(\tilde{x}; r, \eta_y)^T \Xi(\tilde{x}; r, \eta_y) \right]^{-1} \Xi(\tilde{x}; r, \eta_y)^T \tilde{x} \right\|_2^2 \quad (3.21)$$

Gradient-based algorithms such as Gauss-Newton or Levenberg–Marquardt [37] may optimize r and η_y since $J(r, \eta_y)$ is differentiable.

3. Final Estimates of Individual Parameters

Their final estimates are obtained by substituting the final estimates (r and η_y) into the formula estimates of separable parameters in Eq. (3.20)

$$\begin{bmatrix} \hat{a}_1(r, \eta_y) \\ \hat{a}_2(r, \eta_y) \end{bmatrix} = \left[\Xi(\tilde{x}; r, \eta_y)^T \Xi(\tilde{x}; r, \eta_y) \right]^{-1} \Xi(\tilde{x}; r, \eta_y)^T \tilde{x}. \quad (3.22)$$

3.2.2 Parameters and Initial Condition Estimation by NLS

When using SLS, the noisy observation $\tilde{x}(t_k)$ is treated as a substitute for the true value $x(t_k)$. The pseudo regression in Eq. (3.17) is an errors-in-variables model, and if the measurement noise is ignored, it can result in inaccurate estimates [35, 38].

Eq. (3.13) can be viewed as a nonlinear regression

$$\tilde{x}(t_k) = x(t_k; a_1, a_2, r, \eta_y) + e(k), \quad (3.23)$$

where $x(t_k; a_1, a_2, r, \eta_y)$ denotes the trajectory of IDE (see Eq. 3.11). Since the initial condition is constrained, the closed-form trajectory is available. Then, the objective function for parameter estimation can be obtained as

$$J(a_1, a_2, r, \eta_y) = \sum_{k=1}^n [\tilde{x}(t_k) - x(t_k; a_1, a_2, r, \eta_y)]^2, \quad (3.24)$$

which is nonlinear in parameters and can be minimized by using a nonlinear optimization technique such as Levenberg-Marquardt algorithm [31].

3.3 Applications on daily traffic flow pattern identification

Traffic congestion, especially during rush hours, remains a pressing challenge for cities and urban areas worldwide [39]. Addressing and effectively managing this issue has become a critical priority in urban planning and transportation systems [40, 41].

The forecasts $\hat{x}(t_1), \hat{x}(t_2), \dots, \hat{x}(t_n)$ are obtained by matching the times in the data with the observation times, as estimating the state is an important step. The first statistic used is the Mean Absolute Percentage Error (MAPE), which is defined as

$$\text{MAPE} = \frac{1}{n} \sum_{k=1}^n \left| \frac{\hat{x}(t_k) - x(t_k)}{x(t_k)} \right| \times 100\%. \quad (3.25)$$

Table 3.1 shows the results of the nonlinear grey Bernoulli model (Ngb) applied to IDE and ODE using the NLS and SLS methods. The MAPE is calculated as the average performance across each day of the week.

In Chapter 5, we show that our proposed model outperforms all four approaches of the Ngb model on the test set, even when the Ngb model is trained and tested on a single example.

Day	ODE - SLS	ODE - NLS	IDE - SLS	IDE - NLS
	MAPE	MAPE	MAPE	MAPE
Monday	6.92%	6.9%	6.85%	6.83%
Tuesday	6.80%	6.97%	6.83%	6.77 %
Wednesday	7.77%	7.80%	7.65%	7.60%
Thursday	7.08%	7.25%	7.09%	7.08 %
Friday	7.76%	7.91%	7.77%	7.77 %
Saturday	8.12%	8.11%	8.06%	8.05 %
Sunday	8.18%	8.18%	8.17%	8.17 %

Table 3.1: Result of 4 method of Nonlinear Grey Bernoulli Model

The model has only been trained and tested on one example, so the results give a limited view of how well it works. While it is easy to interpret and offers insights into how it functions, relying on a single example makes it harder to scale the model or ensure it works reliably with different types of data. The key points are summarized below:

1. The nonlinear grey Bernoulli model can only be trained and assessed on a single sample because it is essentially a mathematical framework with a small number of parameters. Its scalability is hindered by this constraint, making it inappropriate for use with more complicated systems or bigger datasets.
2. In real-world applications, all variants of the Ngb model demonstrate inherent limitations. Both the ODE-based and IDE-based models are characterized by their inability to achieve high accuracy. While they generate smooth trajectories, they fail to effectively capture the fluctuations present in sample paths [30].

Chapter 4

VPNgbNET: Variable Projection Augmented Nonlinear Grey Bernoulli Networks

Inspired by Variable Projection Networks (VPNet) [7], the nonlinear grey Bernoulli model (Ngb) based IDE [31], and ODE [32], the Standard Least Squares (SLS) approach, sometimes referred to as Variable Projection, may be applied to fit r and η_y and estimate the parameters a_1 and a_2 . This makes it simple to scale up the model by integrating it into the VPNet framework. Variable Projection Augmented Nonlinear Grey Bernoulli Networks (VPNgbNET) represent an innovative and advanced architecture that integrates several powerful methodologies. It combines the strength of Variable Projection (VP) techniques, the flexibility of nonlinear grey Bernoulli models (Ngb) when working with limited data, and the efficiency of Fully Connected Neural Networks (FCNNs) and Residual Networks (ResNet), resulting in a robust model designed to handle complex, nonlinear relationships in data. This hybrid approach leverages the mathematical rigor of VP for dimensionality reduction and optimization, the adaptability of Ngb for dealing with uncertainty and incomplete information, and the deep learning capabilities of FCNNs and ResNet to capture intricate patterns and make accurate predictions.

Time series forecasting involves predicting future values of target variables based on patterns learned from past data. To make the problem manageable, learning methods typically use data from a fixed-length window of previous time points as explicit input. Many useful temporal inferencing problems have been considered including filtering, smoothing, predictions of unobserved past events or alternative histories. Efficient and accurate

predictions of the future have significant applications in many different domains: finance, traffic, engineering, healthcare, weather, and more. Better predictions can lead to better investments, better management of traffic anomalies, improved management of resources.

4.1 VPNgbNET architecture

The components of the VPNgbNET architecture are structured as follows:

VP Layer : Based on the Ngb System, designed to estimate precise parameters for the Trajectory layer. This approach allows for accurate parameterization, making it easier to scale the basic mathematical model and improve its ability to capture complex system dynamics.

Trajectory Layer : This approach can be based on either ODE or IDE. By predefining the forecasting time frame, the Trajectory layer is capable of generating time-based regression outputs. Additionally, this framework is flexible and can be adapted to incorporate other time-based models or domain-specific physics-based models, depending on the problem requirements.

Fully Connected Neural Networks : For managing noise and complex pattern recognition, a basic approach can involve one or two fully connected (FC) layers using activation functions such as ReLU or Linear. For more complex problems, the model can be expanded by incorporating the Recurrent Neural Networks (RNNs) family or other architectures designed for sequence learning. Additionally, larger and more sophisticated architectures can be developed to address more challenging tasks, offering greater flexibility in capturing intricate patterns within the data.

Residual Connection : The output of the trajectory layer is passed forward and added to the neural network's output. While this resembles the residual connections in ResNet [42], which help with training deep networks, here it enhances interpretability by explicitly incorporating domain-driven patterns into the model's predictions.

Figure 4.1 shows the VPNgbNET model architecture, explaining how it learns, makes forecasts, and handles noise and complex patterns in the data.

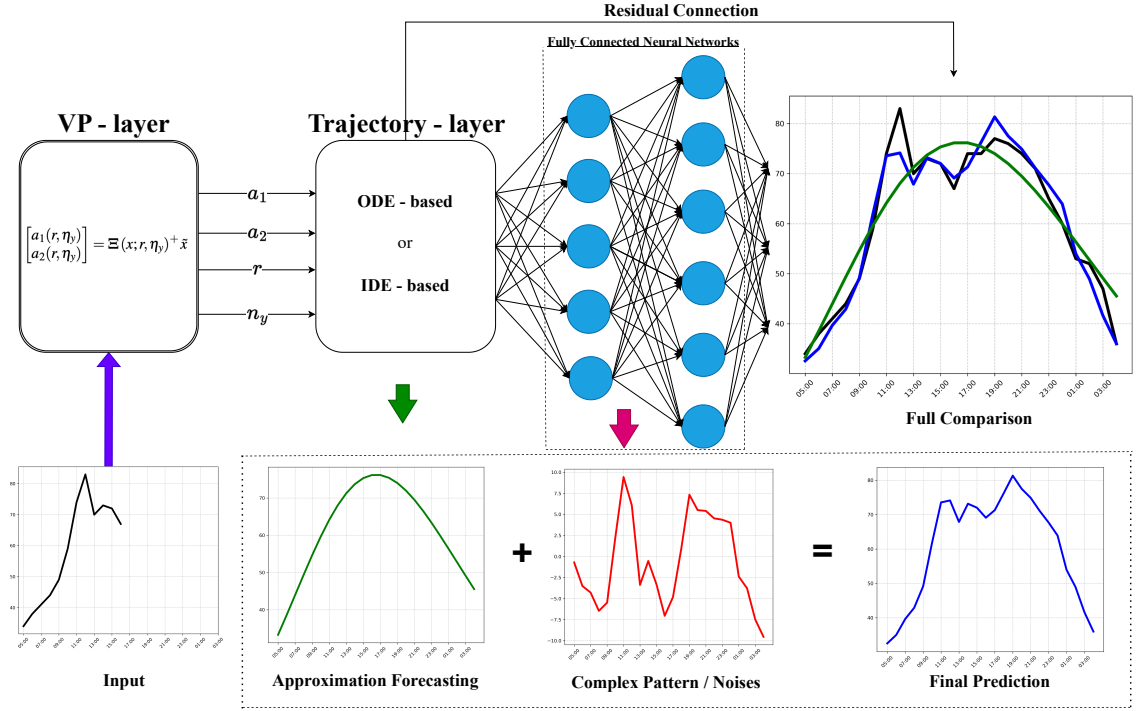


Figure 4.1: VPngbNET architecture

4.2 VP layer

The input x is time series data representing daily traffic flow. Each time frame can be represented in units, such as minutes or hours

$$\mathbf{x} = \begin{pmatrix} x(t_1) \\ x(t_2) \\ \vdots \\ x(t_n) \end{pmatrix},$$

The system matrix used in the VP layer, derived from the nonlinear grey Bernoulli system, is defined as follows

$$\Xi(x; r, \eta_y) = \begin{pmatrix} \eta_y + \frac{1}{2} \sum_{i=2}^2 h_i x(t_{i-1}) + \frac{1}{2} \sum_{i=2}^2 h_i x(t_i) & [\eta_y + \frac{1}{2} \sum_{i=2}^2 h_i x(t_{i-1}) + \frac{1}{2} \sum_{i=2}^2 h_i x(t_i)]^r \\ \eta_y + \frac{1}{2} \sum_{i=2}^3 h_i x(t_{i-1}) + \frac{1}{2} \sum_{i=2}^3 h_i x(t_i) & [\eta_y + \frac{1}{2} \sum_{i=2}^3 h_i x(t_{i-1}) + \frac{1}{2} \sum_{i=2}^3 h_i x(t_i)]^r \\ \vdots & \vdots \\ \eta_y + \frac{1}{2} \sum_{i=2}^n h_i x(t_{i-1}) + \frac{1}{2} \sum_{i=2}^n h_i x(t_i) & [\eta_y + \frac{1}{2} \sum_{i=2}^n h_i x(t_{i-1}) + \frac{1}{2} \sum_{i=2}^n h_i x(t_i)]^r \end{pmatrix}. \quad (4.1)$$

We define

$$\tilde{x} = \begin{pmatrix} x(t_2) \\ x(t_3) \\ \vdots \\ x(t_n) \end{pmatrix},$$

And the estimate parameters is define as

$$\begin{bmatrix} \hat{a}_1(r, \eta_y) \\ \hat{a}_2(r, \eta_y) \end{bmatrix} = \Xi(x; r, \eta_y)^+ \tilde{x}, \quad (4.2)$$

where $\Xi(x; r, \eta_y)^+$ denotes the Moore-Penrose pseudoinverse of the system matrix $\Xi(x; r, \eta_y)$.

The parameters r and η_y are learnable to optimize the VP layer. Features a_1 , a_2 , r , and η_y are extracted and passed forward to the next layer. At this stage, a_1 , a_2 , r , and η_y serve as refined parameter estimates for the Trajectory layer.

This approach can be seen as a new paradigm for parameter estimation in traditional mathematical models. In classical models, such as the nonlinear grey Bernoulli model, training and testing are typically limited to a single example of data. This limitation hinders their ability to scale effectively, as they struggle to generalize well across varying inputs. By incorporating Variable Projection (VP), the model's scalability is significantly improved. With VP, the model can handle different inputs and generate corresponding outputs before proceeding to the trajectory layer. This enhances the model's adaptability, enabling it to scale efficiently. Furthermore, VP facilitates the integration of advanced techniques like Fully Connected Neural Networks (FCNNs).

4.3 Trajectory layer

Two types of trajectory working specifically for daily particular traffic flow are introduced in [31] (see Eq. (3.11, 3.10)). We provide two different kinds of layers that may be used with our approach.

4.3.1 ODE - based

The trajectory of the Cusum series, as demonstrated in Chapter 3, is defined as

$$\hat{y}(t) = \left[\left(\hat{\eta}_y^{1-\hat{r}} + \frac{\hat{a}_2}{\hat{a}_1} \right) \exp(\hat{a}_1(1-\hat{r})(t-t_1)) - \frac{\hat{a}_2}{\hat{a}_1} \right]^{\frac{1}{1-\hat{r}}}, \quad t \geq t_1, \quad (4.3)$$

then using inverse equation to transform the Cusum series into the original series, we obtain

$$\hat{x}(t) = \frac{dy(t)}{dt} = a_1 y(t) + a_2 [y(t)]^r, \quad y(t_1) = \eta_y, \quad t \geq t_1. \quad (4.4)$$

The variable t represents any time step, which can correspond to daily, weekly, or other intervals, with steps measured in hours, minutes, or seconds.

The ODE-based model uses two main functions: the first is the Cusum series trajectory, which tracks cumulative changes over time, and the second is the inverse function, which transforms the Cusum series back to the original time series for forecasting. However, in our experiment in Chapter 5, we see that the ODE-based approach, due to its use of simpler functions, results in a time complexity similar to that of the IDE approach of VPNGbNET (see Section 4.3.2).

4.3.2 IDE - based

In section 3.2, the trajectory of the original series is defined as

$$\begin{aligned} \hat{x}(t) = \frac{d}{dt} \hat{y}(t) &= (1 - \hat{r}) (\hat{a}_1 \hat{\eta}_y^{1-\hat{r}} + \hat{a}_2) \exp(\hat{a}_1 (1 - \hat{r})(t - t_1)) \\ &\times \left[\left(\hat{\eta}_y^{1-\hat{r}} + \frac{\hat{a}_2}{\hat{a}_1} \right) \exp(\hat{a}_1 (1 - \hat{r})(t - t_1)) - \frac{\hat{a}_2}{\hat{a}_1} \right]^{\frac{\hat{r}}{1-\hat{r}}}, \quad t \geq t_1. \end{aligned} \quad (4.5)$$

The approach of the Trajectory layer is take estimate parameters form VP layer and try to fit with Trajectory. It can be seen as an NLS approach for parameters estimation. By substituting the estimates of parameters into the continuous-time trajectory

$$\hat{x}(t; \hat{a}_1, \hat{a}_2, \hat{r}, \hat{\eta}_y)$$

with the initial time $t \geq t_1$. Subsequently, given the discrete time instants $\{t_{k_1}, t_{k_2}, \dots, t_{k_s}\}$, we have the corresponding forecasts

$$\{\hat{x}(t_{k_1}), \hat{x}(t_{k_2}), \dots, \hat{x}(t_{k_s})\},$$

where $n \leq k_s \in \mathbb{N}^+$.

This method is implemented as a layer, after which the resulting output is passed forward to the neural network, incorporating residual connections for enhanced learning and improved information flow.

4.4 Neural Networks and Residual Connection

Noise frequently exhibits complex and recurrent patterns in real-world time series data that are strongly related to certain temporal contexts, such as recurrent association between last Friday and this Friday or pattern of weekend and weekday. Despite their apparent random nature, these patterns usually include important information that can greatly affect forecasting precision and system comprehension. These subtle noise patterns are frequently overlooked by traditional mathematical models, which are mainly intended to capture prevailing trends and larger system dynamics. Noise frequently exhibits complex and recurrent patterns in real-world time series data that are strongly related to certain temporal contexts, such a recurrent association between last Friday and this Friday or pattern of weekend and dayweek. Despite their apparent random nature, these patterns usually include important information that can greatly affect forecasting precision and system comprehension. These subtle noise patterns are frequently overlooked by traditional mathematical models, which are mainly intended to capture prevailing trends and larger system dynamics. The limitation comes from their inability to capture the complex and frequently nonlinear relationships present in noise, which leads to worse prediction accuracy in situations when these patterns are significant. On the other hand, neural networks offer a reliable framework for identifying the underlying complexity of noise patterns as well as the main trends in time series data. They are ideal for managing the complex structure of noise that conventional methods miss because of their ability to model non-linear connections and learn directly from data. Using neural networks makes it feasible to include noise as a crucial part of the forecasting process, improving the robustness and performance of the model.

The proposed model formulates the output as $y = \hat{y} + \hat{y}_{noise}$, where y represents the final prediction, $\hat{y}$ is the output of the Trajectory layer, and $\hat{y}_{noise}$ accounts for the learned noise component. This structure aligns conceptually with the ResNet architecture [42], where residual learning is employed to enhance model performance by explicitly modeling deviations or finer details. The rationale behind employing this Residual Architecture lies in its ability to effectively separate the main trend from the noise components in the data. Specifically, the Trajectory layer is designed to capture and output the primary trend associated with each specific day. This design leverages the inherent regularity in daily patterns, where each day tends to exhibit a consistent overall shape.

By isolating the main trend within the Trajectory layer, the residual component, repre-

senting the noise, is left for further modeling by subsequent layers. If the entire task were directly assigned to Fully Connected Neural Networks (FCNNs) without the inclusion of residual connections, it would result in an intractable problem. In our experiment, the FCNNs would struggle to distinguish between noise and underlying patterns, leading to overfitting. As a result, noise or overly complex patterns from the training set could be inadvertently transferred to the test set, undermining the model's ability to generalize effectively. This not only reduces their performance but also compromises the interpretability of the model. By employing a residual approach, the architecture ensures a more explainable and efficient solution, enabling the FCNNs to focus solely on fitting the residual noise patterns while maintaining a clear separation of the trend and noise components.

A wide range of architectures could be explored to learn the sequence noise effectively, including sophisticated models from the Recurrent Neural Network (RNN) family, such as Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRU), as well as other advanced variants of ResNet. These architectures are particularly suited to capturing temporal dependencies and intricate relationships in sequential data. However, for the purposes of this study, we limit our exploration to a straightforward approach using basic FCNNs. This choice provides a baseline to assess the feasibility of the proposed framework without introducing the complexity of more advanced architectures.

4.5 VPNgbNET backpropagation

Common loss functions for time series problem are the Mean Squared Error (MSE), that is, the least-squares cost function.

Here, we provide a general regularization that, in the case of a forecasting task, may be applied to a single feature-extraction VP layer and Trajectory Layer. Our suggested adjusted loss function is

$$J_{\text{VPNgbNET}}(\hat{y}_{\text{final}}, y) := J_{\text{MSE}}(\hat{y}_{\text{final}}, y) + \frac{\alpha}{N} J_{\text{MSE}}(\hat{y}, y) \quad (4.6)$$

where $\alpha \geq 0$ controls the penalty effect and $\hat{y}$ is the output of Trajectory layer.

By introducing the error back into the prediction $\hat{y}$, we ensure that the Trajectory layer's output closely approximates the forecasted time series. This approach enhances the interpretability of the model, as the Trajectory layer focuses on providing a clear representation of the forecast. Subsequently, the FCNNs refine the output by managing noise and other complexities, thereby improving the overall accuracy of the model.

The backpropagation technique is used to update the parameters, as explained in Section 2.4 for VPNet. This iterative optimization procedure uses gradient descent to modify the model's parameters in order to reduce the difference between expected and actual results. The architecture makes use of backpropagation to make sure that every layer of the model including those connected to the Variable Projections Networks is methodically improved for overall accuracy and resilience. This method facilitates effective learning and adaptability, especially when working with high dimensional inputs or complicated data patterns.

Chapter 5

Experiments and Results

Time series data often have two main problems: not enough data and older data becoming less useful for predicting future values as time passes [28]. Most deep learning models need a lot of data to work well, but older data doesn't help much when predicting the next day or near future [43]. Our model is designed to handle small datasets and solve both of these problems effectively.

To evaluate the model's performance, we will conduct two separate experiments: simulation part and real traffic data. Simulation part is for testing the model ability to identify the physical trajectory. Additional experiments show that the proposed model is explainable and more robust on real-world data.

5.1 Simulation Data

Numerical experiment are designed and conducted to evaluate both the VPNgbNET-ODE and the VPNgbNET-IDE model. The noisy observations are generated by following Eq. (3.13): $\{x(t_k)\}_{k=1}^n$ from the trajectory ODE (see Eq. (3.10)) and the trajectory IDE (see Eq. (3.11)) and $\{e(k)\}_{k=1}^n$ from a Gaussian white noise process with mean 0 and standard deviation σ . Using noise-variance ratio (*nvr*) to measure noise level, the standard deviation is

$$\sigma = nvr \sqrt{\text{Var}(\{x(t_k)\}_{k=1}^n)}. \quad (5.1)$$

Here, the noise-free trajectory provides accurate information about the true system that can serve as a reference, allowing us to measure the performance of diverse paradigms on noisy observations.

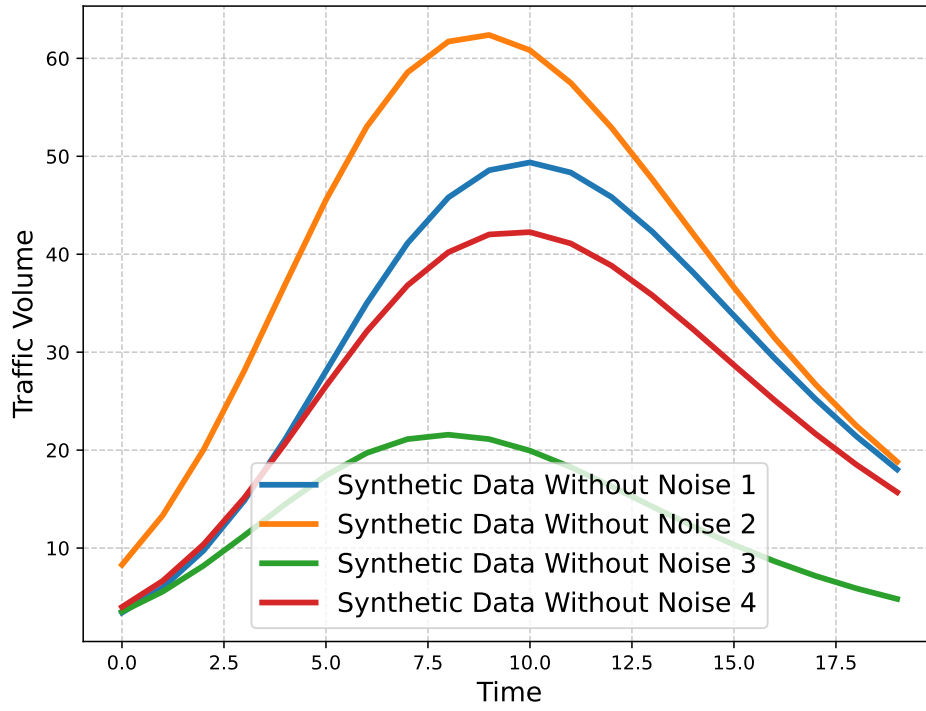


Figure 5.1: Original Simulation based on Trajectory function

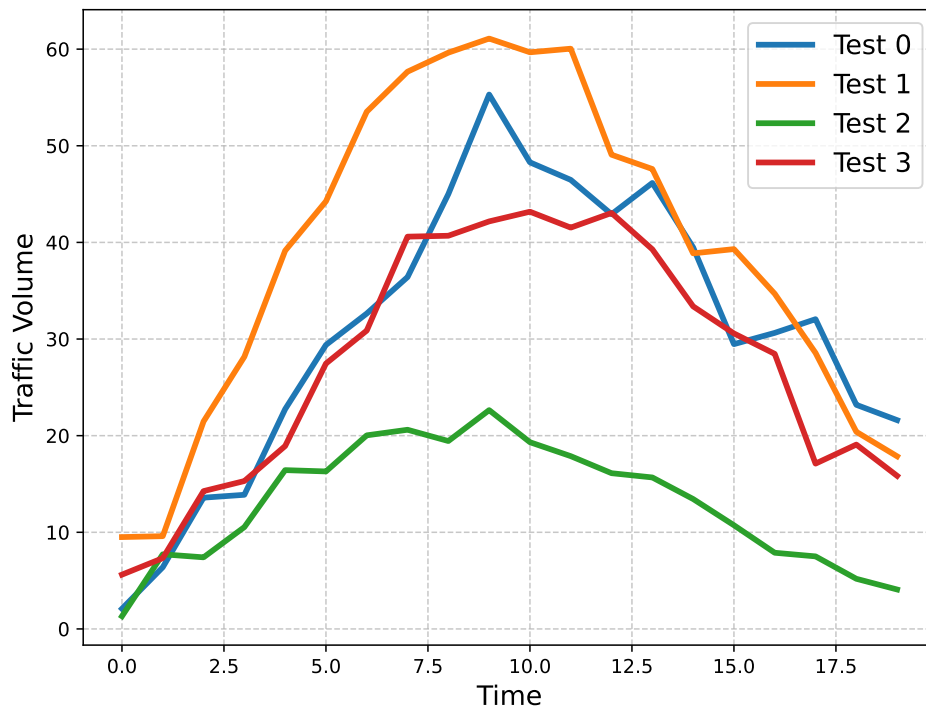


Figure 5.2: Simulation data after adding noise

Without loss of generality, we set the true parameter to a_1, a_2, r, η_y radomly and the time interval to $h = 0.7$. The noise level nvr is set randomly between 10% and 20%.

Figure 5.1 show the original trajectory from the test set data. These datasets were chosen for testing because of their variety of forms and attributes. We aim to assess the model's ability to recognize trends. Figure 5.2 illustrates the test data after noise has been added. Figure 5.3 shows the complete data, separated into training and test sets.

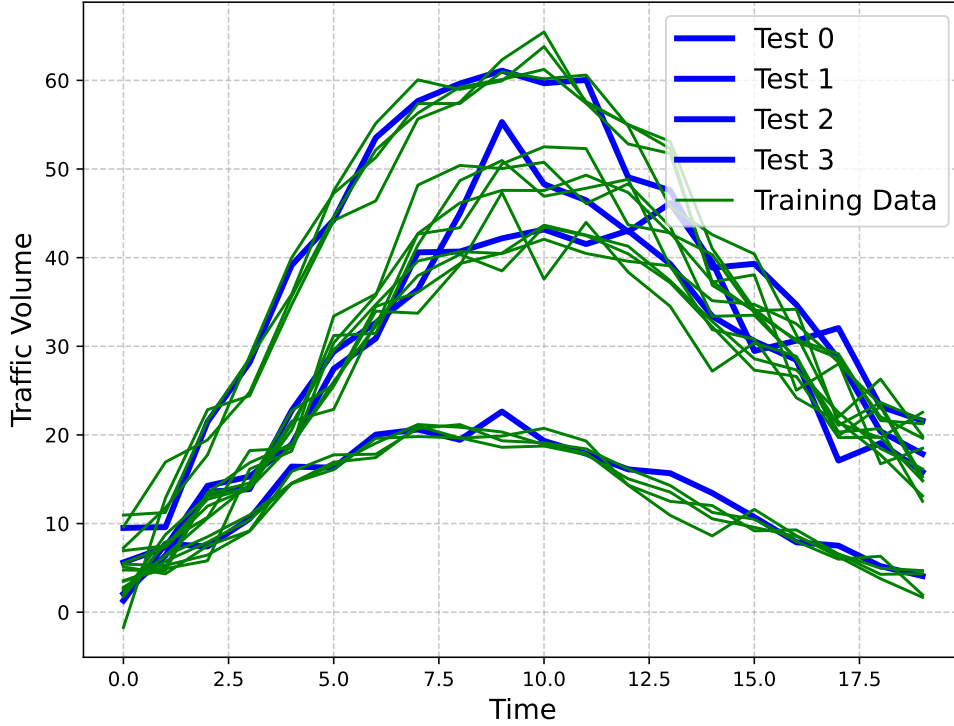


Figure 5.3: Full Data, split for training and testing

We trained a single model, VPNgbNET, using the complete training dataset and evaluated its performance on the test set. Table 5.1 reports a performance of MAPE equal to 9.5% on the test set. However, the accuracy results are lower when tested on real-world data. One reason for this discrepancy is that the noise in the simulation is introduced randomly. In contrast, as discussed later in Section 5.2, real-world data reveals that each shape corresponds to non-random, periodic noise patterns.

Predicted	VPNgbNET - ODE	VPNgbNET - IDE
	MAPE	MAPE
Average Maape	9.47%	9.47%
Test 1	10.56%	10.71%
Test 2	8.32%	8.28%
Test 3	9.73%	9.71%
Test 4	8.30%	8.19%

Table 5.1: Model performance results on the test set

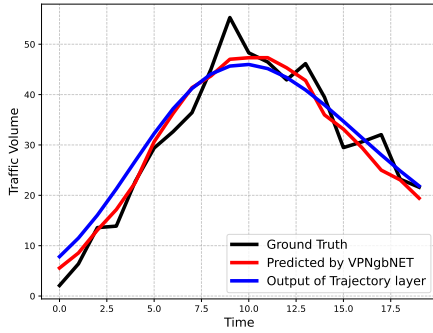


Figure 5.4: Test 1 Prediction by VPNgbNET

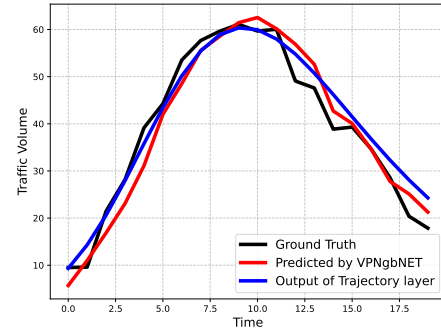


Figure 5.5: Test 2 Prediction by VPNgbNET

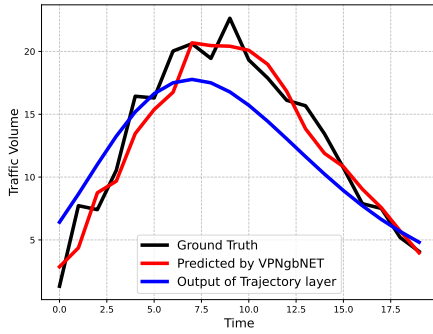


Figure 5.6: Test 3 Prediction by VPNgbNET

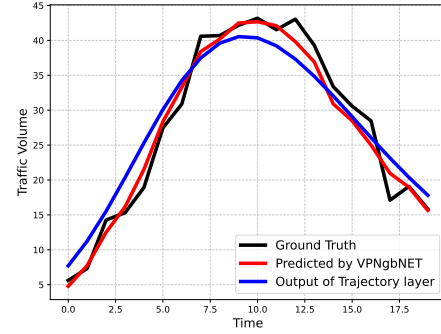


Figure 5.7: Test 4 Prediction by VPNgbNET

5.2 Real Data: Daily traffic flow pattern identification

The traffic dataset is available at kaggle.com/traffic-prediction-dataset and comes from the Kaggle community. Here, five weeks' worth of data from October 16, 2016,

to November 19, 2016 are chosen without losing generality. Figure 5.8 shows the real data collected from one junction, with similar patterns seen at other junctions. The data has a clear inverted U-shape pattern over a specific time frame, repeating daily.

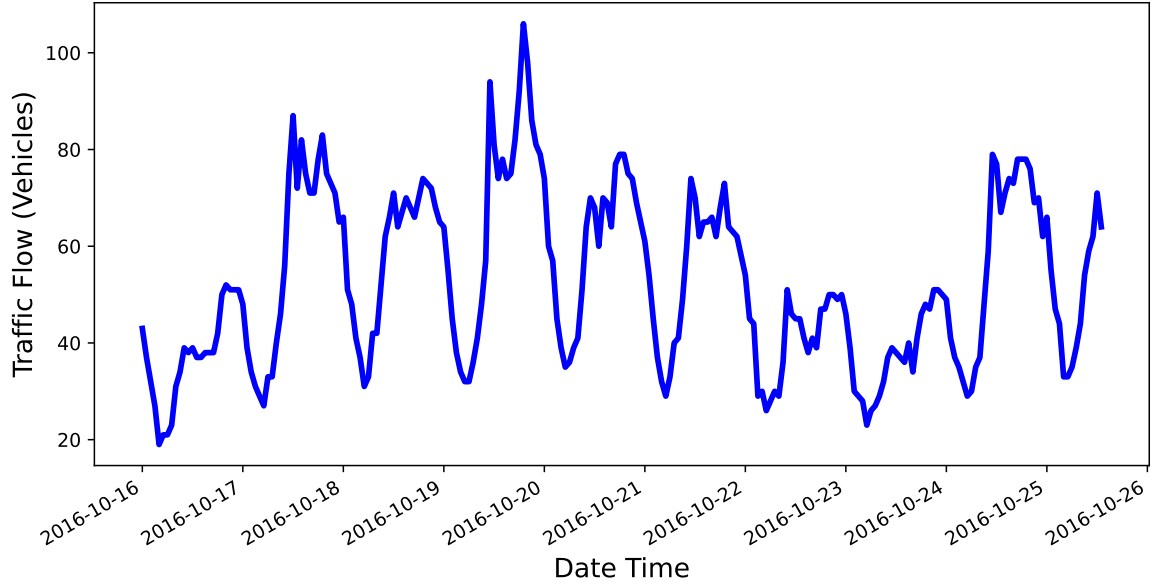


Figure 5.8: Real traffic flow dataset

The performance evaluation was conducted using a five-fold cross-validation framework. Namely, the model is developed separately for each day of the week and evaluated across five iterations. In each evaluation, data is divided into four weeks for training and one week for testing. The outcomes are presented below.

Table 5.2 presents the results evaluated using MAPE (Eq. 3.25). The average MAPE across all five weeks is reported in the table.

The figures below illustrate the performance of VPNgbNET-ODE and VPNgbNET-IDE for each day of the week. The left side displays the ODE-based results of VPNgbNET, while the right side shows the IDE-based results. Each figure clearly presents the ground truth, the approximate forecast from the Trajectory layer, and the final prediction from VPNgbNET.

The trajectory output closely matches the actual traffic volume patterns, making the model easier to interpret. Furthermore, the use of neural networks to manage noise greatly enhances the accuracy of the predictions.

Predicted Day	VPNgbNET - ODE	VPNgbNET - IDE
	MAPE	MAPE
Monday	4.96%	4.99%
Tuesday	4.79%	4.82%
Wednesday	5.65%	5.48%
Thursday	4.93%	4.96%
Friday	6.4%	6.51%
Saturday	5.31%	5.12%
Sunday	4.37%	4.33%

Table 5.2: Results of VPNgbNET specific for each day

Monday

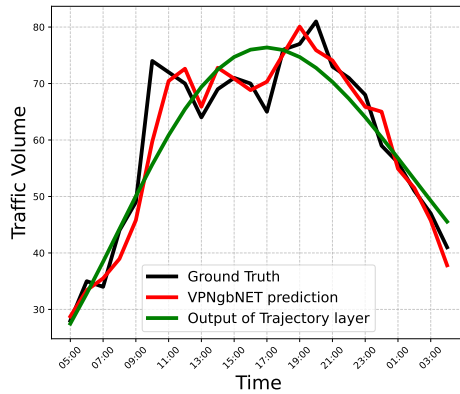


Figure 5.9: Prediction by VPNgbNET-ODE

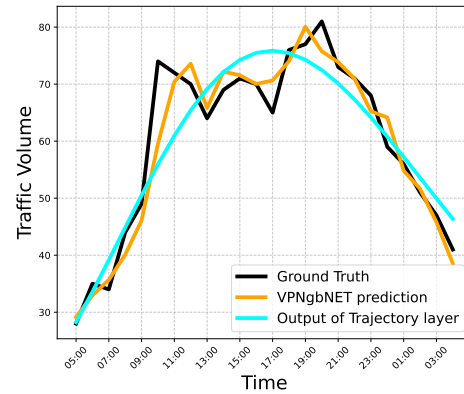


Figure 5.10: Prediction by VPNgbNET-IDE

Tuesday

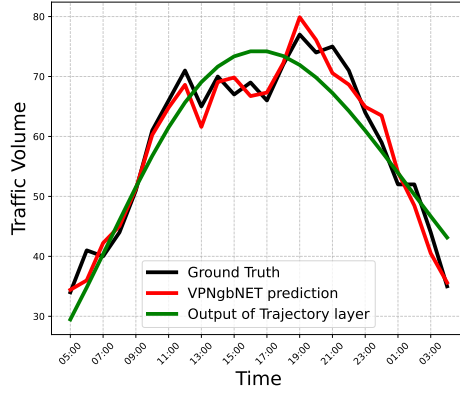


Figure 5.11: Tuesday prediction by VPNgbNET-ODE

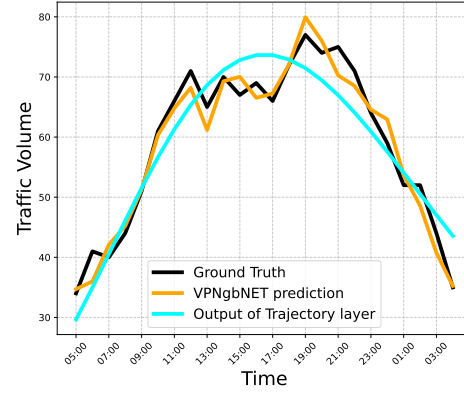


Figure 5.12: Tuesday prediction by VPNgbNET-IDE

Wednesday

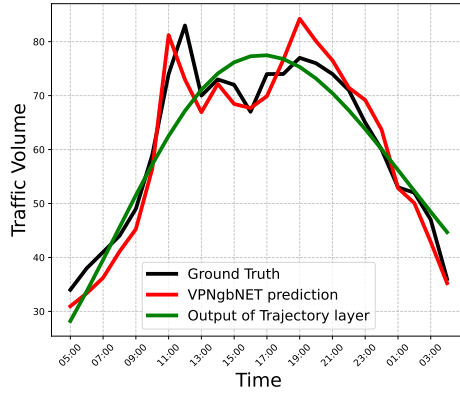


Figure 5.13: Wednesday prediction by VPNgbNET-ODE

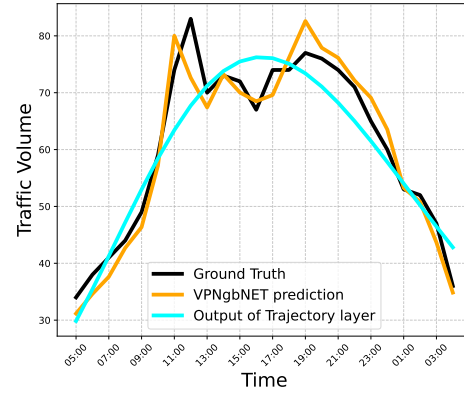


Figure 5.14: Wednesday prediction by VPNgbNET-IDE

Thursday

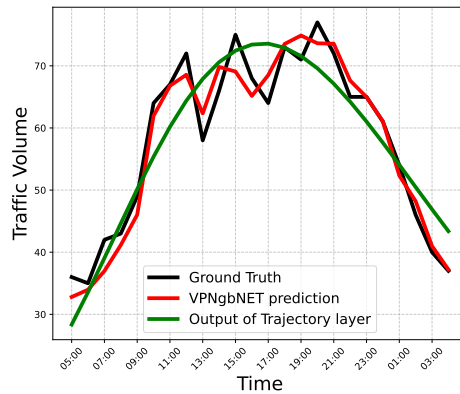


Figure 5.15: Thursday prediction by VPNgbNET-ODE

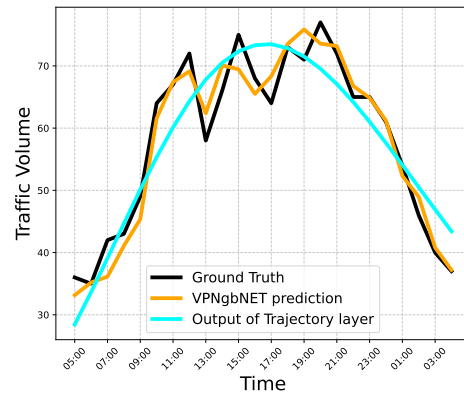


Figure 5.16: Thursday prediction by VPNgbNET-IDE

Friday

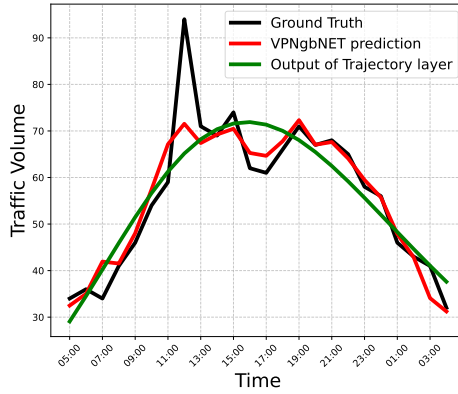


Figure 5.17: Friday prediction by VPNgbNET-ODE

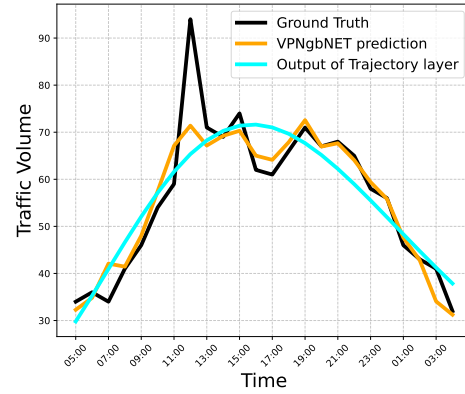


Figure 5.18: Friday prediction by VPNgbNET-IDE

Saturday

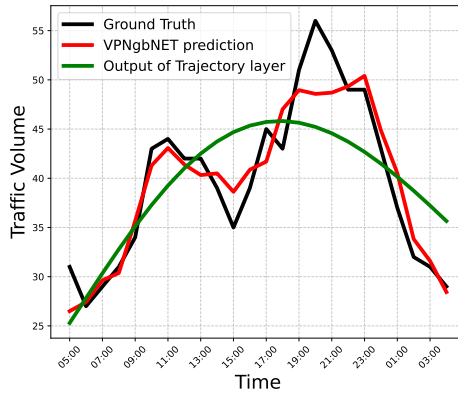


Figure 5.19: Saturday prediction by VPNgbNET-ODE

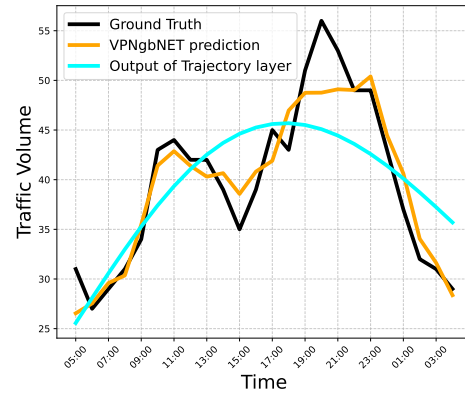


Figure 5.20: Saturday prediction by VPNgbNET-IDE

Sunday

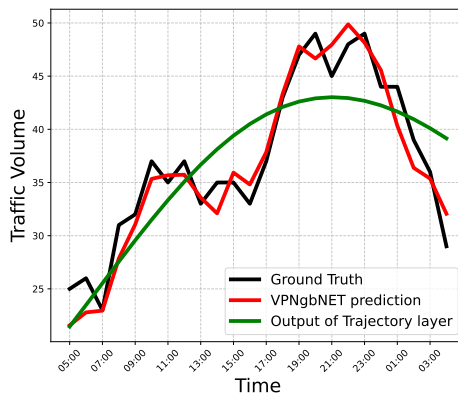


Figure 5.21: Sunday prediction by VPNgbNET-ODE

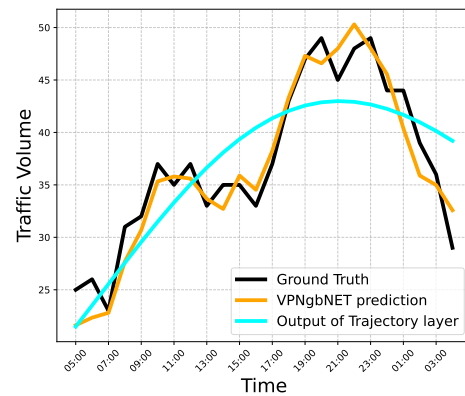


Figure 5.22: Sunday prediction by VPNgbNET-IDE

5.3 Real Data: Weekly traffic flow pattern identification

In this configuration, we expanded both the training and testing datasets to cover a broader range of time. The model was trained on data spanning an entire week, with four weeks of data allocated for training and one week reserved for testing

Figure 5.23 illustrates the prediction performance on the test set, which consists of data from a full week. The blue line represents the ground truth, while the red line shows the predictions made by the VPNgbNET-ODE model. The green line represents the output of the trajectory layer, which provides physics-based predictions derived from the Ngb model. Similarly, in Figure 5.24, the blue line represents the ground truth, the yellow line indicates the output of the trajectory layer, and the brown line shows the prediction from VPNgbNET-IDE.

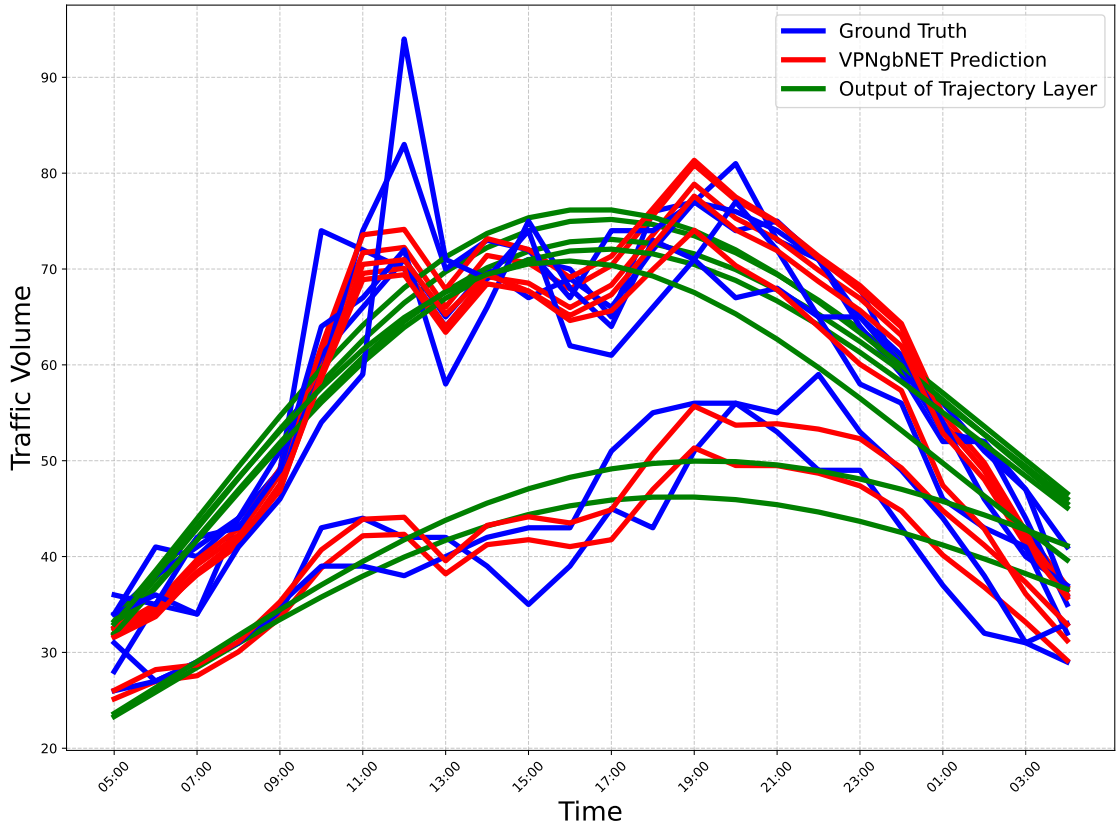


Figure 5.23: Full Testing Prediction of VPNgbNET - ODE

Table 5.3 presents the performance of VPNgbNET on the test set. Following the rules of five-fold cross-validation, the experiment was repeated five times, with each repetition involving selecting one specific week for testing and using the remaining weeks for training. The Average MAPE was calculated by taking the mean of the results across all five repetitions. For each day's MAPE, we used data from a five-week period, with the

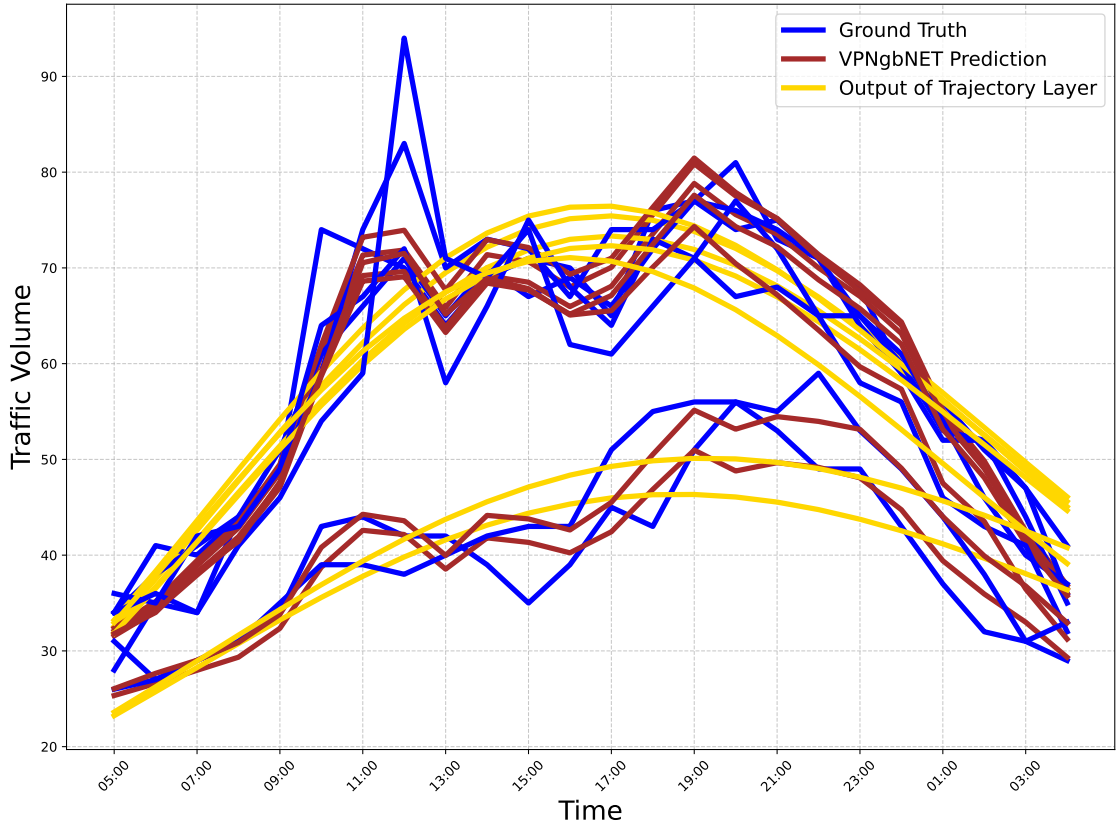


Figure 5.24: Full Testing Prediction of VPNgbNET - IDE

fifth week (the farthest in the timeline) selected as the test set. This approach ensures that the model's performance is evaluated on the most distant week, providing a robust and reliable measure of its ability to forecast future data effectively.

The results indicate that the MAPE remains consistent with the initial configuration, where the model was trained on data from specific days. This consistency underscores the model's robustness and scalability across various training scenarios.

The figures below show how the model performed on the test set. They include the actual data, the model's predictions, and the approximate forecasts from the trajectory layer. These results highlight how accurate and effective the model is at making predictions.

Predicted Day	VPNgbNET - ODE	VPNgbNET - IDE
	MAPE	MAPE
Average Maape	4.81%	4.83%
Monday	4.74%	4.79%
Tuesday	4.15%	4.17%
Wednesday	3.40%	3.55%
Thursday	4.44%	4.53%
Friday	6.27%	6.13%
Saturday	6.32%	6.16%
Sunday	4.45%	4.41%

Table 5.3: Result of the model produced in the Test set

Monday

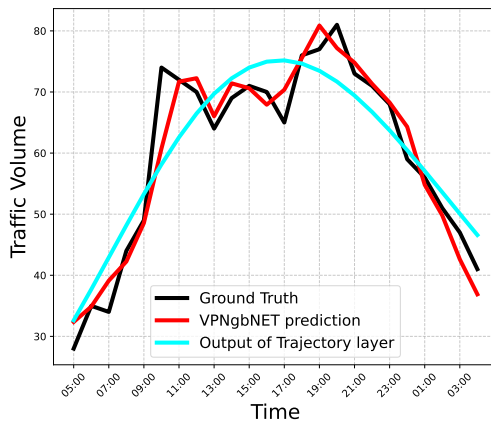


Figure 5.25: VPNgbNET - ODE

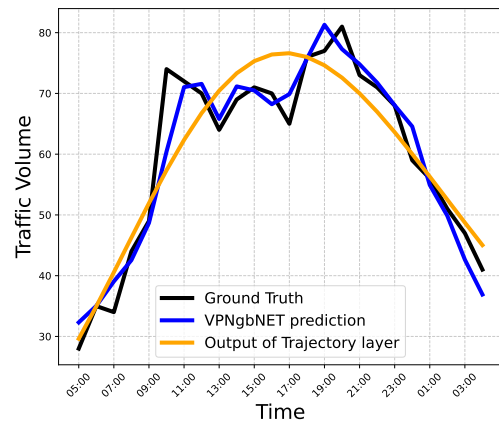


Figure 5.26: VPNgbNET - IDE

Tuesday

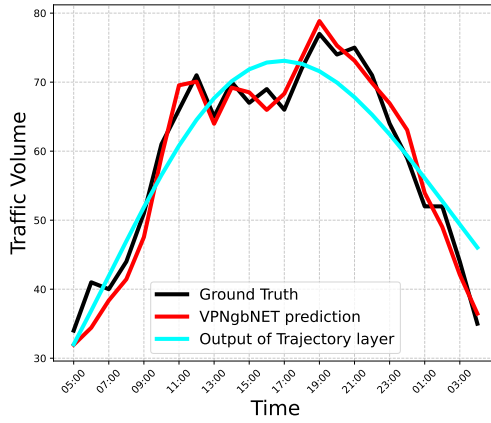


Figure 5.27: VPNgbNET - ODE

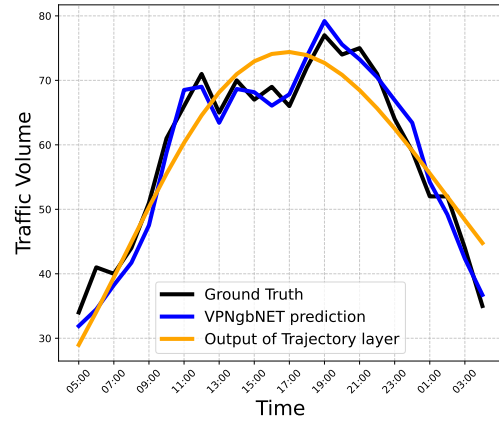


Figure 5.28: VPNgbNET - IDE

Wednesday

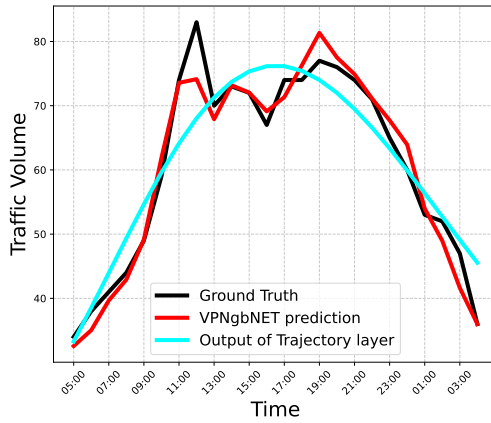


Figure 5.29: VPNgbNET - ODE

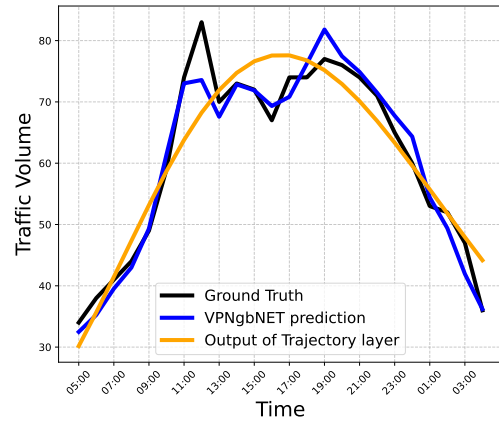


Figure 5.30: VPNgbNET - IDE

Thursday

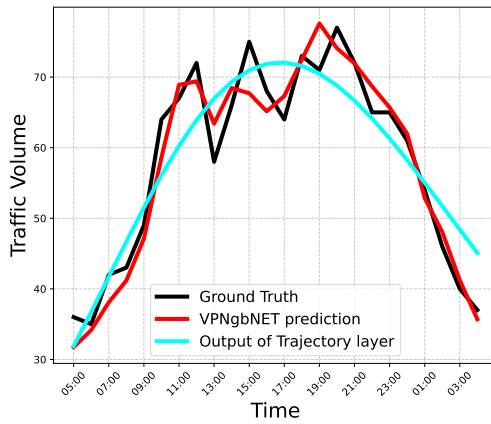


Figure 5.31: VPNgbNET - ODE

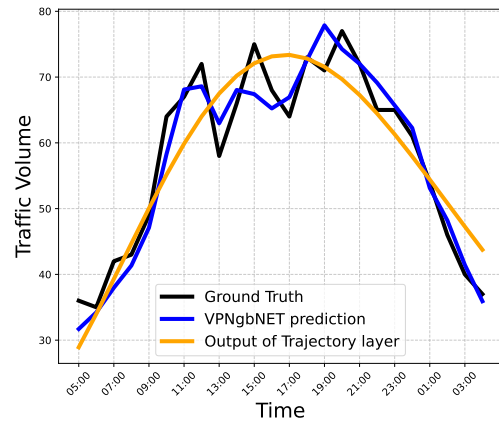


Figure 5.32: VPNgbNET - IDE

Friday

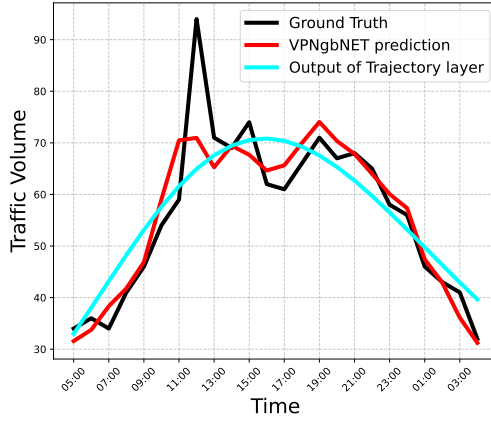


Figure 5.33: VPNgbNET - ODE

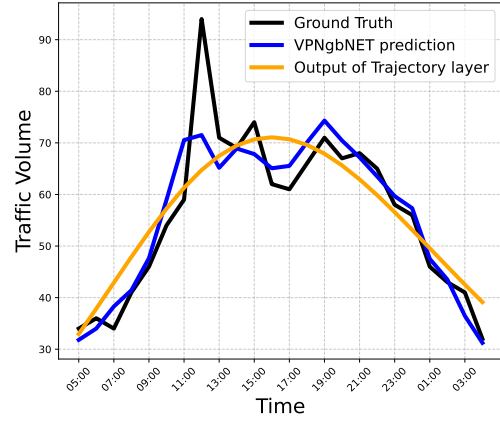


Figure 5.34: VPNgbNET - IDE

Saturday

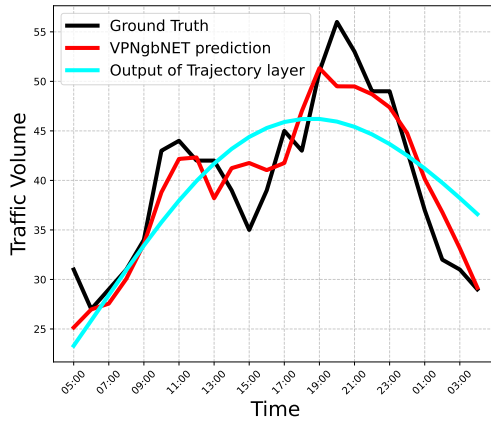


Figure 5.35: VPNgbNET - ODE

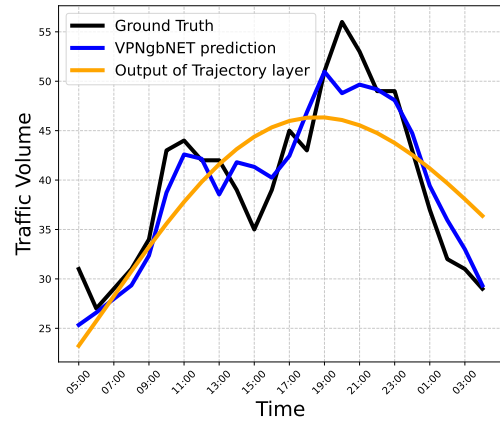


Figure 5.36: VPNgbNET - IDE

Sunday

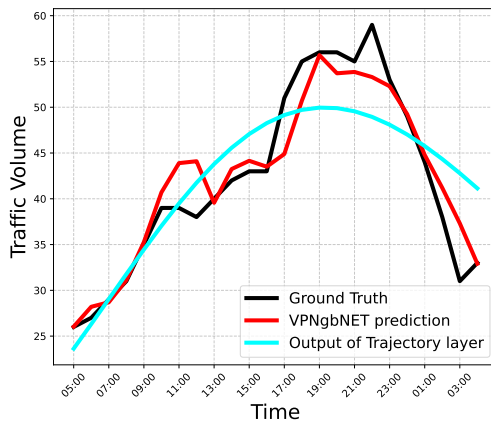


Figure 5.37: VPNgbNET - ODE

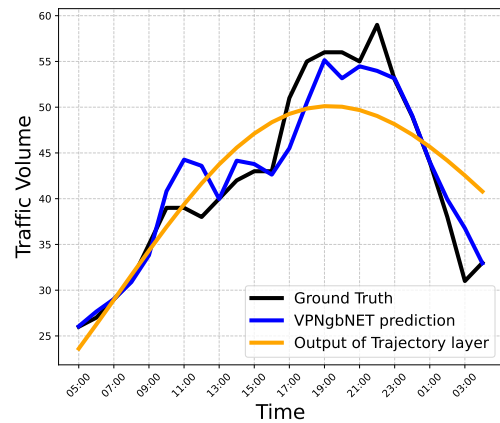


Figure 5.38: VPNgbNET - IDE

Chapter 6

Summary and Discussion

6.1 Comparison

6.1.1 Comparison with Nonlinear Grey Bernoulli Model

Note that in this comparison, VPNgbNET is specified for each day of the week, trained on 80% of the data, and tested on the remaining 20%. The nonlinear grey Bernoulli model, whether IDE-based or ODE-based, is trained and tested on the same data for each example. The results in Table 6.1 demonstrate the accuracy and scalability of our model.

Predicted Day	VPNgbNET - ODE	VPNgbNET - IDE	ODE - SLS	ODE - NLS	IDE - SLS	IDE - NLS
	MAPE	MAPE	MAPE	MAPE	MAPE	MAPE
Monday	4.96%	4.99%	6.92%	6.90%	6.85%	6.83%
Tuesday	4.79%	4.82%	6.80%	6.97%	6.83%	6.77 %
Wednesday	5.65%	5.48%	7.77%	7.80%	7.65%	7.60%
Thursday	4.93%	4.96%	7.08%	7.25%	7.09%	7.08 %
Friday	6.40%	6.51%	7.76%	7.91%	7.77%	7.77 %
Saturday	5.31%	5.12%	8.12%	8.11%	8.06%	8.05 %
Sunday	4.37%	4.33%	8.18%	8.18%	8.17%	8.17 %

Table 6.1: Full comparison of VPNgbNET with nonlinear grey Bernoulli model

6.1.2 Properties Comparison

Our findings show that our approach outperforms classical mathematical models in several important areas, including consistency and adaptability accross different datasets and training scenarios, in addition to achieving greater accuracy and resilience. Furthermore, our approach is more scalable, meaning that it can manage bigger, more complex datasets without significant performance drop. Together with its decent accuracy, its scalability makes it especially well-suited for use in real-world scenarios where a flexible and reliable approach is needed due to changing conditions and a variety of data inputs.

Model	Accuracy	Explainability	Scalability
<i>VPNgbNET</i>	✓	✓	✓
<i>IDE and ODE model</i>	×	✓	×
<i>Deep neural networks</i>	✓	×	✓

Table 6.2: Comparison of model accuracy, explainability, and scalability

In the context of time series analysis, challenges such as limited data availability, unreliability of unseen data, and poor data quality are prevalent. While mathematical models are still commonly employed, they often fall short in terms of accuracy and scalability. Our model integrates the strengths of both deep learning and traditional mathematical approaches, effectively addressing these challenges and providing a more reliable and scalable solution.

6.2 Conclusion

In summary, this study has presented the background and motivation underpinning the development of Variable Projection Augmented Nonlinear Grey Bernoulli Networks (VPNgbNET). The findings demonstrate that VPNgbNET effectively captures the daily trends in traffic flow, achieving high forecasting accuracy. Furthermore, the proposed model outperforms traditional approaches, including the nonlinear grey Bernoulli model (Ngb), as well as ordinary differential equation (ODE) and integral differential equation (IDE)-based methods, highlighting its superior performance in this context.

Our primary findings are organized as follows:

- We have introduced an innovative method for parameter estimation by incorporating a nonlinear grey Bernoulli system into the VP layer inspired by VPNet architecture. This approach enhances the efficiency and accuracy of parameter inference, particularly in the context of dynamic systems, enabling more reliable modeling and prediction.
- We propose a novel Trajectory layer, derived from a time-step-defined trajectory function, which serves as a physics-based layer representing the evolution of time series for forecasting. This approach effectively captures temporal dynamics in complex systems, enhancing the model's robustness and interpretability.
- To address issues associated with noisy or fluctuating data, we have designed neural networks with integrated residual connections, which are specifically tailored to effectively manage these challenges. Our network architecture is designed to maintain robustness and performance even when faced with major changes in the data over time.
- Our model achieves high performance while remaining transparent, offering valuable insights into underlying processes. Additionally, it can be trained on relatively small datasets, overcoming the data scarcity challenges typical of conventional deep learning models. This makes it particularly suitable for real-world applications where data may be limited or costly to obtain.
- The performance of the proposed VPNgbNET was evaluated in a case-study of daily traffic flow estimation using both simulated and real-world data. The proposed method outperformed four state-of-the-art methods which are based on ODE/IDE and NLS/SLS formulations of the nonlinear grey Bernoulli model.

6.3 Future Works

The VPNgbNET framework is highly flexible and may be used successfully in many simple applications in several sectors. For instance, the structural features of VPNgbNET are well suited for tasks like anticipating peak values, such as daily energy consumption, web server traffic, financial market trends, or demand forecasting in supply chains, in the context of time-series data. For these kinds of applications, the framework is especially well-suited as it can model temporal relationships and spot patterns in data that fluctuates.

More precise forecasts and well-informed decision-making in various domains are made possible by VPNgBNET capturing both short-term fluctuations and long-term trends.

Additionally, the system's flexibility allows for the substitution of the Trajectory layer with other models, such as the nonlinear grey Verhulst model (Ngv) [44]. This adaptability extends its applicability to a wide range of dynamic systems. It can also be utilized in conjunction with established approaches like Volterra series models [45] and nonlinear autoregressive models, ARIMA and SARIMA, opening new avenues for research and innovation.

The proposed framework could also benefit from the noise handling abilities of advanced architectures, such as Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTMs), and Residual Networks (ResNet) [42]. These architectures have proven to be effective in managing complex data patterns and mitigating the effects of noise in various domains. Interestingly, our experimental results highlight the efficacy of a comparatively simple neural networks layer with ReLU activation, which shows good performance even with its simple design.

Beyond standard neural network architectures, the integration of the Variable Projection (VP) layer with Physics-Informed Neural Networks (PINNs) [46] introduces a highly robust and versatile framework for modeling systems governed by physical principles. Rather than replacing the physical trajectory, we can build a PINN following the VP layer. This approach utilizes the intrinsic structure of physical laws to constrain the solution space, thereby enhancing the model's ability to handle noisy or sparse datasets. Moreover, it simultaneously ensures higher accuracy and interpretability, making the model more robust and reliable in real-world applications. This combination bridges the gap between data-driven and theory-driven modeling, enabling the system to address a broader range of scientific and engineering challenges effectively. This integration highlights the effectiveness of combining advanced deep learning methods with domain-specific expertise, opening new pathways for research into dynamic systems and complex phenomena characterized by uncertainty or incomplete data. The resulting framework is powerful not only for theoretical exploration, but also for practical applications where precision and insight are critical.

Acknowledgments

I would like to sincerely thank Dr. Péter Kovács for his exceptional guidance and mentorship throughout this research in the Modelling Lab. His groundbreaking work on Variable Projection Networks, which earned him the Hojjat Adeli Award, has inspired this study. The guidance, support, and insightful feedback from Dr. Kovács have been incredibly valuable to my development as a researcher. Learning from a mentor who is so dedicated and innovative has been an honor.

Appendix A

Simulation Results

A.1 Experiment on Week identification

The implementation codes are available at <https://github.com/Felixnguyen33/VPNgbNET>.

The detailed results of experiments in Section 5.3 on real data.

Set Up	VPNgbNET - ODE		
	Train Set	Test Set	MAPE
Experiment 1	Week 1,2,3,4	Week 5	5.00%
Experiment 2	Week 1,2,3,5	Week 4	5.80%
Experiment 3	Week 1,2,4,5	Week 3	5.91%
Experiment 4	Week 1,3,4,5	Week 2	6.75%
Experiment 5	Week 2,3,4,5	Week 1	5.53%

Table A.1: Result of experiment of VPNbgNET - ODE

Set Up	VPNgbNET - IDE		
	Train Set	Test Set	MAPE
Experiment 1	Week 1,2,3,4	Week 5	5.02%
Experiment 2	Week 1,2,3,5	Week 4	5.93%
Experiment 3	Week 1,2,4,5	Week 3	5.90%
Experiment 4	Week 1,3,4,5	Week 2	6.63%
Experiment 5	Week 2,3,4,5	Week 1	5.50%

Table A.2: Result of experiment of VPNbgNET - IDE

Bibliography

- [1] Zijun Cui; Tian Gao; Kartik Talamadupula and Qiang Ji. “Knowledge-augmented Deep Learning and Its Applications: A Survey”. In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1 (2022).
- [2] Chun-I Chen; Hong Long Chen and Shuo-Pei Chen. “Forecasting of foreign exchange rates of Taiwan’s major trading partners by novel nonlinear Grey Bernoulli model NGBM(1, 1)”. In: *Communications in Nonlinear Science and Numerical Simulation* (2008).
- [3] Chun-I Chen. “Application of the novel nonlinear grey Bernoulli model for forecasting unemployment rate”. In: *Chaos Solitons and Fractals* (2008).
- [4] Qingping He; Xin Ma; Lanxi Zhang; Wanpeng; Tianzi Li. “The nonlinear multi-variable grey Bernoulli model and its applications”. In: *Applied Mathematical Modelling* (2024).
- [5] Yong Wang; Rui Yang; Lang Sun. “A novel structure adaptive discrete grey Bernoulli model and its application in renewable energy power generation prediction”. In: *Expert Systems with Applications* (2024).
- [6] Nir Shlezinger; Jay Whang; Yonina C. Eldar and Alexandros G. Dimakis. “Model-Based Deep Learning”. In: *arxiv* (2022).
- [7] Péter Kovács; Gergő Bognár; Christian Huber and Mario Huemer. “VPNET: Variable Projection Networks”. In: *International Journal of Neural Systems* 32.1 (Aug. 2022).
- [8] E. Alpaydin. “Introduction To Machine Learning”. In: *The MIT Press* (2020).
- [9] E. Alpaydin. “Deep Learning in neural networks”. In: *An overview, Neural Netw.* (2015).

- [10] T. Yang; C. Cappelle; Y. Ruichek and M. El Bagdouri. “Multi-object tracking with discriminant correlation filter based deep learning tracker”. In: *Integr. Comput Aided Eng* (2019).
- [11] G. E. Hinton and R. R. Salakhutdinov. “Reducing the dimensionality of data with neural networks”. In: *Science* (2006).
- [12] A. Krizhevsky; I. Sutskever and G. E. Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in Neural Information Processing Systems* (2019).
- [13] M. Leming; J. M. Gorriz and J. Suckling. “Ensemble deep learning on large, mixed-site fmri datasets in autism and other tasks”. In: *International Journal of Neural System* (2020).
- [14] S. Kiranyaz; T. Ince and M. Gabbouj. “Real-time patient-specific ECG classification by 1-d convolutional neural networks”. In: *IEEE Transactions Biomedical Engineering* (2020).
- [15] G. H. Golub and V. Pereyra. “The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate”. In: *SIAM J. Num. Anal. (SINUM)* (1973).
- [16] G. H. Golub and V. Pereyra. “The variable projection method and its applications”. In: *Inverse Problem* (2003).
- [17] G.-Y. Chen; M. Gan; S. Wang and C. L. P. Chen. “Insights into algorithms for separable nonlinear least squares problems”. In: *IEEE Transactions and Images Processing* (2020).
- [18] Attila Miklós Ámon; Kristian Fenech; Péter Kovács; Tamás Dózsa. “Continuous wavelet transform based variable projection networks”. In: *32nd European Signal Processing Conference (EUSIPCO)* (2024).
- [19] Tamás Dózsa; Mátyás Szabari; Alexandros Soumelidis and Péter Kovács. “Pole Identification using discrete Laguerre expansion and variable projection”. In: *IFAC* (2023).
- [20] D. P. Kingma and Adam J. Ba. “A method for stochastic optimization”. In: *arxiv* (2014).

- [21] J. Duchi; E. Hazan and Y. Singer. “Adaptive subgradient methods for online learning and stochastic of variable projection”. In: *IEEE Transaction Biomedical Engineering* (2021).
- [22] T. Tieleman and G. Hinton. “RmsProp: Divide the gradient by a running average of its recent magnitude”. In: *COURSERA: Neural networks for machine learning* (2012).
- [23] Yuxia Zhang; Minghui Zhou; Audris Mockus and Zhi Jin. “Thinking about U: Theorizing and Testing U- and inverted U-shaped Relationships in Strategy Research”. In: *Strategic Management Journal* (2016).
- [24] Yuxia Zhang; Minghui Zhou; Audris Mockus and Zhi Jin. “Companies’ Participation in OSS Development—An Empirical Study of OpenStack”. In: *IEEE Transactions on Software Engineering* (2021).
- [25] Saadettin Aydin. “Time Series Analysis and Some Applications in Medical Research”. In: *Journal of Mathematics and Statistics Studies* (2022).
- [26] Chong Liu; Tongfei Lao; Wen-Ze Wu; Wanli Xie; Hegui Zhu. “An optimized non-linear grey Bernoulli prediction model and its application in natural gas production”. In: *Expert Systems with Applications* (2022).
- [27] Xie Ming; Wu Lifeng; Li Bin; Li Zechen. “A novel hybrid multivariate nonlinear grey model for forecasting the traffic-related emissions”. In: *Applied Mathematical Modelling* (2020).
- [28] Wang ZX. “A weighted non-linear grey Bernoulli model for forecasting non-linear economic time series with small data sets”. In: *Economic Cybernetics Studies and Research* (2017).
- [29] Chun-I Chen. “Introduction to Grey System Theory”. In: *Journal of Grey System* (2008).
- [30] Dongfang Ma; Xiang Song and Pu Li. “Daily Traffic Flow Forecasting Through a Contextual Convolutional Recurrent Neural Network Modeling Inter- and Intra-Day Traffic Patterns”. In: *IEEE Transactions and Images Processing* (2020).
- [31] Lu Yang; Bentao Su; Naiming Xie and Baolei Wei. “An IDE-based nonlinear grey Bernoulli model and applications to daily traffic flow pattern identification”. In: *Expert System With Applications* (2024).

- [32] Wei Baolei and Lu Yang. “Nonlinear grey Bernoulli model with physics- preserving cusum operator”. In: *Expert Systems with Applications* (2023).
- [33] Baolei Wei; Naiming Xie; Lu Yang. “Understanding cumulative sum operator in grey prediction model with integral matching”. In: *Communications in Nonlinear Science and Numerical Simulation* (2020).
- [34] Lu Yang; Naiming Xie. “Integral matching-based nonlinear grey Bernoulli model for forecasting the coal consumption in China”. In: *Soft Computing* (2021).
- [35] Baolei Wei; Naiming Xie. “Parameter estimation for grey system models: A nonlinear least squares perspective”. In: *Communications in Nonlinear Science and Numerical Simulation* (2021).
- [36] Wei Baolei. “Parameter estimation strategies for separable grey system models with comparisons and applications”. In: *Applied Mathematical Modelling* (2023).
- [37] Min Gan; C. L. Philip Chen; Guang-Yong Chen and Long Chen. “On Some Separated Algorithms for Separable Nonlinear Least Squares Problems”. In: *IEEE Transactions on Cybernetics* (2018).
- [38] Dattner I. “Differential equations in data analysis”. In: *Computational Statistics* (2020).
- [39] Mahmood S. N. “Traffic Pattern Classification in Smart Cities Using Deep Recurrent Neural Networks”. In: *Sustainability* (2023).
- [40] Li H. “Identifying Traffic Congestion Patterns Using Clustering Methods. Sustainability”. In: *arxiv* (2023).
- [41] Fu; Yang. “The Role of Traffic Pattern Identification in Urban Mobility”. In: *ResearchGate* (2023).
- [42] Kaiming He; Xiangyu Zhang; Shaoqing Ren and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *Computer Vision and Pattern Recognition* (2015).
- [43] Rakshitha Godahewaa; Christoph Bergmeira; Geoffrey I. Webba; Rob J. Hyndmanb; Pablo Montero-Mansoc. “Monash Time Series Forecasting Archive”. In: *arxiv* (2021).

- [44] Bo Zeng; Mingyu Tong and Xin Ma. “A new-structure grey Verhulst model: Development and performance comparison”. In: *Applied Mathematical Modelling* (2020).
- [45] Heng Zhang. “Volterra Series: Introduction and Application”. In: *RF Communication Circuits and Systems* (2015).
- [46] M. Raissi; P. Perdikaris and G.E. Karniadakis. “Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations”. In: *Nature* (2022).

List of Figures

2.1	VPNET architecture [7]	7
4.1	VPNgbNET architecture	24
5.1	Original Simulation based on Trajectory function	31
5.2	Simulation data after adding noise	31
5.3	Full Data, split for training and testing	32
5.4	Test 1 Prediction by VPNgbNET	33
5.5	Test 2 Prediction by VPNgbNET	33
5.6	Test 3 Prediction by VPNgbNET	33
5.7	Test 4 Prediction by VPNgbNET	33
5.8	Real traffic flow dataset	34
5.9	Prediction by VPNgbNET-ODE	35
5.10	Prediction by VPNgbNET-IDE	35
5.11	Tuesday prediction by VPNgbNET-ODE	36
5.12	Tuesday prediction by VPNgbNET-IDE	36
5.13	Wednesday prediction by VPNgbNET-ODE	36
5.14	Wednesday prediction by VPNgbNET-IDE	36
5.15	Thursday prediction by VPNgbNET-ODE	36
5.16	Thursday prediction by VPNgbNET-IDE	36
5.17	Friday prediction by VPNgbNET-ODE	37
5.18	Friday prediction by VPNgbNET-IDE	37
5.19	Saturday prediction by VPNgbNET-ODE	37
5.20	Saturday prediction by VPNgbNET-IDE	37
5.21	Sunday prediction by VPNgbNET-ODE	37
5.22	Sunday prediction by VPNgbNET-IDE	37
5.23	Full Testing Prediction of VPNgbNET - ODE	38
5.24	Full Testing Prediction of VPNgbNET - IDE	39

5.25	VPNgbNET - ODE	40
5.26	VPNgbNET - IDE	40
5.27	VPNgbNET - ODE	41
5.28	VPNgbNET - IDE	41
5.29	VPNgbNET - ODE	41
5.30	VPNgbNET - IDE	41
5.31	VPNgbNET - ODE	41
5.32	VPNgbNET - IDE	41
5.33	VPNgbNET - ODE	42
5.34	VPNgbNET - IDE	42
5.35	VPNgbNET - ODE	42
5.36	VPNgbNET - IDE	42
5.37	VPNgbNET - ODE	42
5.38	VPNgbNET - IDE	42

List of Tables

3.1	Result of 4 method of Nonlinear Grey Bernoulli Model	21
5.1	Model performance results on the test set	33
5.2	Results of VPNgbNET specific for each day	35
5.3	Result of the model produced in the Test set	40
6.1	Full comparison of VPNgbNET with nonlinear grey Bernoulli model . . .	43
6.2	Comparison of model accuracy, explainability, and scalability	44