

TaylorNet: Learning PDEs from Non-Grid Data

Andrzej Dulny

Paul Heinisch

Andreas Hotho

Anna Krause

CAIDAS, University of Würzburg

ANDRZEJ.DULNY@UNI-WUERZBURG.DE

PAUL.HEINISCH@STUD-MAIL.UNI-WUERZBURG.DE

HOTHO@INFORMATIK.UNI-WUERZBURG.DE

ANNA.KRAUSE@INFORMATIK.UNI-WUERZBURG.DE

Editors: Cecília Coelho, Bernd Zimmering, M. Fernanda P. Costa, Luís L. Ferrás, Oliver Niggemann

Abstract

Modeling data obtained from dynamical systems has gained attention in recent years as a challenging task for machine learning models. Previous approaches assume the measurements to be distributed on a grid. However, for real-world applications like weather prediction, the observations are taken from arbitrary locations within the spatial domain. In this paper, we propose TaylorNet – a novel machine learning method that is designed to overcome this challenge. Our algorithm uses the multidimensional Taylor expansion of a dynamical system at each observation point to estimate the spatial derivatives to perform predictions. TaylorNet is able to accomplish two objectives simultaneously: accurately forecast the evolution of a complex dynamical system and explicitly reconstruct the underlying differential equation describing the system. We evaluate our model on a variety of advection-diffusion equations with different parameters and show that it performs similarly to equivalent approaches on grid-structured data while being able to process unstructured data as well.

Keywords: Partial Differential Equations - Dynamic System - NeuralPDE - Taylor Expansion - Deep Learning - Physics - Symbolic Neural Network

1. Introduction

Dynamical systems like weather ([Rasp et al. \(2020\)](#); [Bauer et al. \(2015\)](#)), chemical reactions ([Zhang and Ma \(2020\)](#)) or wave propagation ([Karlbauer et al. \(2019\)](#)) are an essential part of our environment. Analyzing these systems may lead to knowledge of how they evolve and a better understanding of the system itself ([Raol et al. \(2004\)](#); [Close et al. \(2001\)](#)). Dynamic systems are described by ODEs (ordinary differential equations) with only time derivatives or PDEs (partial differential equations) containing time and spatial derivatives ([Kuehn \(2019\)](#)). These equations play a vital role in many disciplines and describe the physical laws governing the system.

Modeling dynamic systems has gained attention in recent contributions as an interesting and challenging topic ([Karlbauer et al. \(2019\)](#); [Praditia et al. \(2021\)](#); [Li et al. \(2020a\)](#); [So et al. \(2021\)](#)). While traditional approaches rely on analytical or numerical solutions to the governing equations, there has been growing interest in leveraging data driven methods (particularly deep learning) to augment or even replace traditional modeling frameworks ([Azizzadenesheli et al. \(2024\)](#)). Extracting the underlying differential equation from observed data has emerged as a distinct and complementary task to forecasting the system’s future states, aiming not just to predict but to uncover interpretable and generalizable representations of the system’s dynamics ([Brunton et al. \(2016\)](#); [Raissi et al. \(2017b\)](#)).

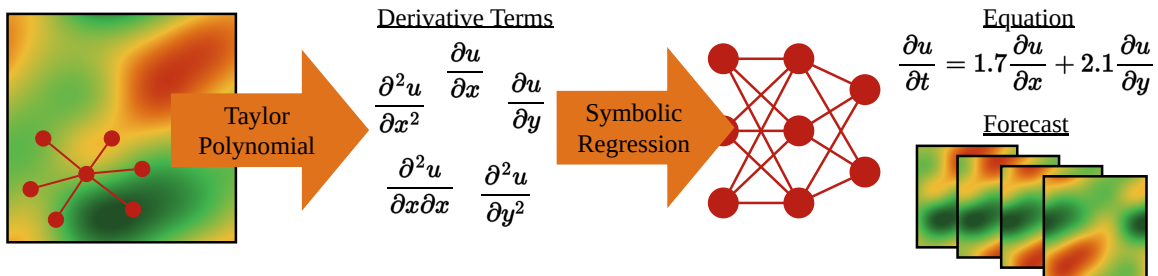


Figure 1: In TaylorNet measurements of a system at arbitrary points in space are used with a Taylor polynomial to estimate spatial derivatives for each point. These are used with a symbolic regression network to estimate the parameters of a partial differential equation and perform predictions of future states.

Available machine learning models require the observations to be structured on a grid. In this case, convolutional neural networks (Long et al. (2019); Goodfellow et al. (2016)) or neural operators (Kovachki et al. (2023)), dominant for this type of data, can be used to forecast the state of the system. However, in real-world applications, the state of a dynamical system at a specific time is measured at arbitrary points in space (point *cloud*). Data, like weather dynamics (Rasp et al. (2020); Bauer et al. (2015)), chemical reactions (Zhang and Ma (2020)), and wave propagation (Karlbauer et al. (2019)), can rarely be measured on a grid.

Our objective is to develop a machine learning method able to learn from non-grid data as we often measure it in reality. To this end, we propose the TaylorNet, which is based on the approximation of spatial derivatives with the Taylor polynomial (Gölsu and Sezer (2006)). TaylorNet does not require gridded data and instead computes the spatial derivatives $\frac{\partial^{(q+r)}u}{\partial x^q \partial y^r}$ using arbitrary neighboring points. Our approach, as summarized in Figure 1, consists of two steps: Firstly, the spatial derivatives are calculated using the Taylor polynomial evaluated at several neighboring points. Secondly, the differential equation is constructed using a linear symbolic regressor which assigns coefficients to different terms of the equation. By learning the coefficients of each spatial derivative in the equation, our model is able to both accurately forecast future states of the system (Dulny et al. (2022); Cai et al. (2021)) and reconstruct the underlying differential equation.

We evaluate our model on several 2D advection-diffusion equations (Dehghan (2004)):

$$\frac{\partial u}{\partial t} = \alpha_{10} \frac{\partial u}{\partial x} + \alpha_{01} \frac{\partial u}{\partial y} + \alpha_{20} \frac{\partial^2 u}{\partial x^2} + \alpha_{02} \frac{\partial^2 u}{\partial y^2} \quad (1)$$

with different coefficients $\alpha_{ij} \in \mathbb{R}$.

Our contributions can be summarized as follows:

1. we propose a novel approach to learning dynamical systems from data based on Taylor polynomials
2. we show that our model TaylorNet is able to perform well on both grid and non-grid data alike

- we demonstrate that TaylorNet is able to both predict future states of the dynamical system and extract its governing equation

We make the code containing our model and all experiments publicly available. ¹

2. Related Work

Previous work on learning dynamical systems from observations has mainly focused on grid-structured data. [Dulny et al. \(2022\)](#) propose the NeuralPDE model which combines CNNs with a differentiable method of lines solver to parametrize the underlying PDEs. Similarly, [Ayed et al. \(2019\)](#) propose a hidden-state neural-based model to forecast dynamical systems using a ResNet to parametrize the equation. The Finite Volume Neural Network (FINN) by [Praditia et al. \(2021\)](#) is based on the Finite Volume Method and predicts the evolution of diffusion-type systems by explicitly modeling the flow between grid points. [Li et al. \(2021\)](#) propose the Fourier Neural Operator which learns the simulations of physical processing using convolutions in Fourier space.

Another line of research focuses on extracting the differential equation which describes the evolution of the dynamical system. [Long et al. \(2018\)](#) propose the PDE-Net model which uses learnable convolutional filters to estimate single derivative terms together with a linear regression layer to reconstruct the coefficients of the equation. An extension of this approach is the PDE-Net 2.0 ([Long et al. \(2019\)](#)) which additionally features a symbolic regression network capable of including non-linear terms in the equations. [Raissi et al. \(2017a\)](#) propose a physics-informed deep learning approach that is able to fit the coefficients of a known equation type using automatic differentiation.

The task of predicting the evolution of a physical system from non-grid structured data has only recently started gaining attention. Recently [Dulny et al. \(2023\)](#) proposed a benchmark for learning dynamical systems from low-resolution non-grid observations on which several graph neural network and point cloud ([Zhao et al. \(2021\)](#)) based models were evaluated. [Iakovlev et al. \(2020\)](#) use a graph message passing approach to learn predictions for an advection-diffusion problem, as well as the heat equation and Burger’s equation from data. The multipole graph neural operator proposed by [Li et al. \(2020b\)](#) can also be used to learn dynamical systems from unstructured data, however, the authors only evaluate it on grid observations.

To the best of our knowledge, our proposed approach TaylorNet is the first model capable of simultaneously forecasting the evolution of a dynamical system while also extracting the differential equations describing its behavior from non-grid observations.

3. TaylorNet

Given measurements of an evolving physical system $u: \Omega \times T \rightarrow \mathbb{R}^d$ at specific locations $\mathbf{p}_1, \dots, \mathbf{p}_n$ within the spatial domain $\Omega \subset \mathbb{R}^2$. The measurements are available at time points $T = \{t_1, t_2, t_3, \dots, t_\tau\}$ for $T \subset \mathbb{R}$. In principle, our approach works with arbitrary spaced time steps, but for simplicity, we assume $t_{i+1} - t_i = \text{const}$. We denote the known state of the system at a given time t as $u(t) := [u(\mathbf{p}_1, t), \dots, u(\mathbf{p}_n, t)] \in \mathbb{R}^{n \times d}$. The system is

1. <https://github.com/badulion/taylornet>

assumed to evolve according to an underlying PDE of the form in equation 1. We consider the task of predicting the evolution of the system (*forecasting*) as well as reconstructing the underlying differential equation (*reconstruction*).

Our proposed model, TaylorNet, is based on the Taylor polynomial (Sezer et al. (2005)), which offers an approximation for functions at a single point in space. We calculate the polynomial for the function $u(\cdot, t_i)$ at each point \mathbf{p}_i and its neighboring points $\mathcal{N}(\mathbf{p}_i)$ and solve a set of linear equations with the derivatives of the underlying PDE model as unknowns (Gülsu and Sezer (2006)). We combine this knowledge with learnable parameters to conduct an Euler step (Biswas et al. (2013)) and compute the future state of the model.

3.1. Taylor Polynomial Approximation

For a given point $\mathbf{p}_0 \in \mathbb{R}^2$ with coordinates x_0 and y_0 in our measurements $u(t_i)$ we compute the derivatives at \mathbf{p}_0 using the Taylor approximation.

Theorem 1 (Taylor Approximation Duistermaat and Kolk (2004)) *A function of two variables f whose partial derivatives all exist up to the Q^{th} order within a neighborhood U of the point (x_0, y_0) can be approximated by the Q^{th} -degree Taylor polynomial of f at the point (x, y) in the neighbourhood of (x_0, y_0) as follows:*

$$f(x, y) = f(x_0, y_0) + P_Q(x, y) + o(h^{Q+1}) \quad (2)$$

where

$$P_Q(x, y) = \sum_{q=0}^Q \sum_{r=0}^{Q-q} \frac{(x-x_0)^q (y-y_0)^r}{q!r!} \cdot \frac{\partial^{(q+r)} f}{\partial x^q \partial y^r}(x_0, y_0) \quad (3)$$

and $h = \sqrt{(x-x_0)^2 + (y-y_0)^2}$.

We can leverage Theorem 1 to calculate the partial derivatives of a function f at a given point \mathbf{p}_0 using available data at K neighboring points $\mathbf{p}_1, \dots, \mathbf{p}_K, \mathbf{p}_i = (x_i, y_i)$. By expanding the function at the neighboring points \mathbf{p}_i using the Taylor approximation and treating the partial derivatives $u_{x^q y^r} = \frac{\partial^{(q+r)} f}{\partial x^q \partial y^r}$ as unknowns we arrive at K different linear equations of the form:

$$f(x_i, y_i) - f(x_0, y_0) = \sum_{q+r \leq Q; q, r \geq 0} \alpha^{(i)}(q, r) u_{x^q y^r} \quad (4)$$

with $\alpha^{(i)}(q, r) = \frac{(x_i-x_0)^q (y_i-y_0)^r}{q!r!}$.

Note that the left-hand side of the equation and the coefficients $\alpha(q, r)$ can be computed with the available data. For the the physical system $u(t)$ as the function f at a given point in time t and several neighboring points \mathbf{p}_i , we construct and solve the following linear equation system:

$$\Delta \mathbf{u} = \mathbf{A} \mathbf{D} \quad (5)$$

where $\mathbf{A} \in \mathbb{R}^{K \times K}$ contains all the computed factors $\alpha^{(i)}(q, r)$, $\mathbf{D} \in \mathbb{R}^K$ containing the derivatives $u_{x^q y^r}$ we want to solve for, and $\Delta \mathbf{u} = [u(t)(x_1, y_1) - u(t)(x_0, y_0), \dots, u(t)(x_K, y_K) - u(t)(x_0, y_0)] \in \mathbb{R}^K$. Note that the method of computation for the derivatives \mathbf{D} does not require the data points to be structured grid-like.

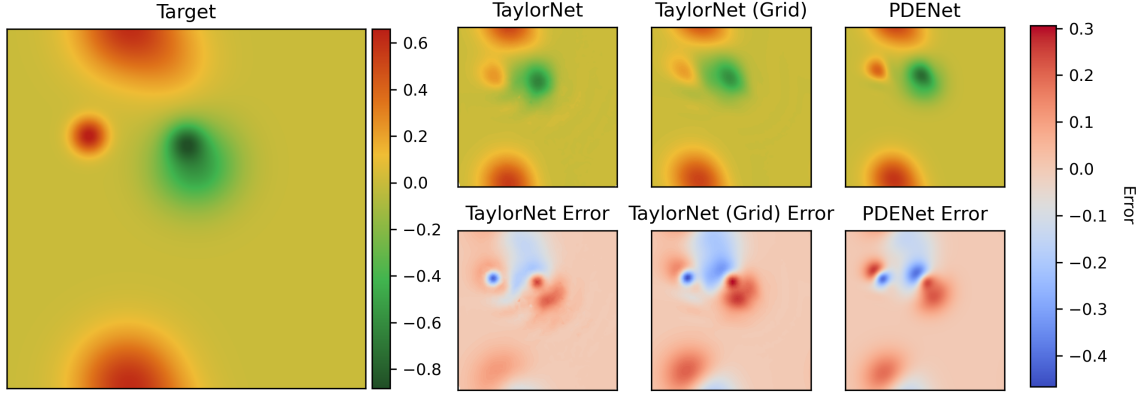


Figure 2: Comparison of the prediction error after after $L = 32$ prediction steps of the TaylorNet, TaylorNet (grid) and the PDENet models.

To calculate all partial derivatives $u_{x^q y^r}$ up to order Q , the linear system Equation (5) needs to contain as many equations as there are partial derivatives ($\frac{(Q+1)(Q+2)}{2}$). However, in practice, we find that over-specifying the equation system leads to numerically more stable results (cf. Section 4.3). For this reason, we use an ordinary least squares solver to estimate the derivatives $\hat{\mathbf{D}}$ for a given point \mathbf{p}_0 :

$$\hat{\mathbf{D}} = \min_{\mathbf{D} \in \mathbb{R}^K} \|\mathbf{A}\mathbf{D} - \Delta\mathbf{u}\|_2 \quad (6)$$

where $\|\cdot\|_2$ is the l_2 norm.

3.2. Architecture TaylorNet

We predict the state of the system one step Δt into the future by performing one forward Euler step:

Let $u(t)$ be the state of the system at time t . Then we predict the state at time $t + \Delta t$ as:

$$\hat{u}(t + \Delta t) = w_{0,0} \cdot u(t) + \Delta t \cdot \sum_{q=0}^Q \sum_{r=0}^{Q-q} w_{q,r} \cdot \frac{\partial^{(q+r)}}{\partial x^q \partial y^r} u(t) \quad (q+r \neq 0) \quad (7)$$

where the derivatives are estimated with the Taylor approximation at each point. The learnable weights are denoted as $w_{q,r}$ with q associated with the order of the derivative in x , and r the derivative in y . The weights thus represent the parameters of the underlying PDE model and can be easily extracted.

To enable the algorithm to make long-term predictions we perform several prediction steps sequentially using the output of the previous step as input for the current. For all prediction steps the weights are shared thus keeping the coefficients constant during prediction. Given the input data $u(t)$ we update the parameters $w_{q,r}$ by minimizing the error of the L -prediction steps $\frac{1}{L} \sum_{i=1}^L (u(t + i \cdot \Delta t) - \hat{u}(t + i \cdot \Delta t))^2$.

Table 1: Prediction MSE after 32 prediction steps compared with the PDE-Net evaluated on equations (1)-(5) with a previously unseen initial conditions.

Resolution	Model	(1)	(2)	(3)	(4)	(5)
16	TaylorNet	1.4e-02	2.1e-04	5.5e+01	3.0e-03	8.4e-04
	TaylorNet (grid)	1.4e-02	1.9e-04	4.3e-03	3.2e-03	8.9e-04
	PDENet	7.9e-03	4.5e-05	2.1e-03	1.6e-03	2.9e-04
	CNN	4.3e-03	3.4e-05	2.4e-03	1.1e-03	2.0e-04
32	TaylorNet	7.4e-02	6.7e-05	2.7e-03	2.0e-03	3.9e-04
	TaylorNet (grid)	1.2e-02	7.2e-05	2.9e-03	2.1e-03	4.3e-04
	PDENet	5.8e-03	3.6e-05	1.7e-03	1.3e-03	2.3e-04
	CNN	4.1e-03	8.3e-05	1.6e-03	1.2e-03	2.1e-04
64	TaylorNet	8.1e-03	3.8e-05	2.8e-03	1.5e-03	2.5e-04
	TaylorNet (grid)	8.4e-03	3.5e-05	2.0e-03	1.5e-03	2.5e-04
	PDENet	4.8e-03	3.9e-05	1.6e-03	1.3e-03	2.1e-04
	CNN	4.4e-03	5.0e-05	1.7e-03	1.3e-03	3.3e-04

4. Experiments

In this section, we test the TaylorNet’s performance on grid and non-grid data, and study the influence of the number of neighbors used in the computation of the derivatives. We further compare the TaylorNet to the existing PDE-Net (Long et al. (2019, 2018)), which is also capable of both reconstructing the equation and forecasting the system, albeit only on grid data, as well as a simple CNN baseline which can only forecast, but not reconstruct the equation.

4.1. Data

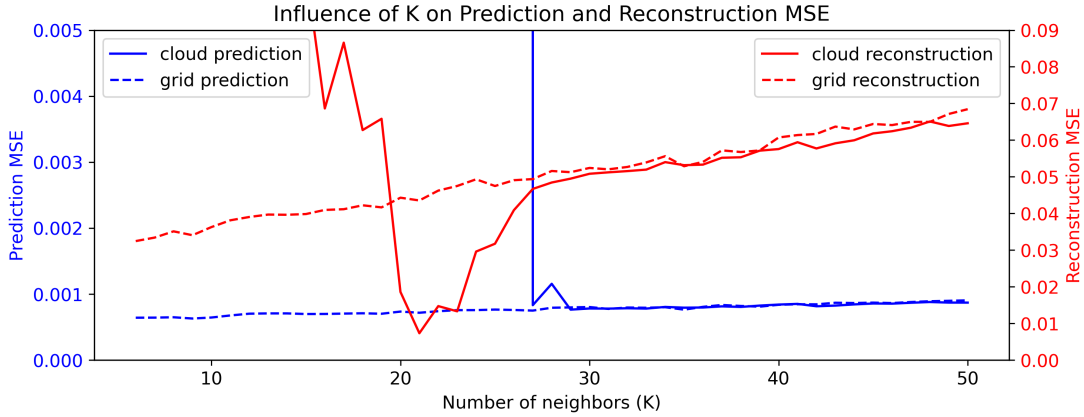
For our experiments, we generate data by numerically solving an advection-diffusion equation using the DynaBench suite (Dulny et al. (2023)). The data is simulated on a grid with periodic boundary conditions of the grid size 128×128 , which is later downsampled to 16^2 , 32^2 and 64^2 points, structured either on a grid (for TaylorNet (grid), PDE-Net and CNN) or arbitrarily (for TaylorNet) The solution is saved with a time discretization of $\Delta t = 0.1$. We run our experiments on 10 different equations with different parameters. More details can be found in Appendix A.

4.2. Training and Testing

During training, we use $L = 8$ prediction steps and set the maximum order of the derivatives to two. We use a kernel size of 3×3 for the PDE-Net and $K = 30$ neighbors for the TaylorNet. We evaluate the models on their performance on the forecasting by computing the MSE (Mean Squared Error) between the computed and the correct trajectory. For the reconstruction task we compute the average squared error over all derivative coefficients up to the order of two.

Table 2: Learned PDE parameters for the first four advection-diffusion equations on a resolution of 64×64

model	Equation	Δ (MSE)
Equation (1)	$u_t = -1.000u_x + 1.000u_y + 0.000u_{xx} + 0.000u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = -1.409u_x + 1.408u_y + 0.104u_{xx} + 0.105u_{yy} - 0.170u_{xy}$	0.077
TaylorNet (grid)	$u_t = -1.420u_x + 1.419u_y + 0.109u_{xx} + 0.108u_{yy} - 0.175u_{xy}$	0.081
PDENet	$u_t = -1.247u_x + 1.247u_y + 0.028u_{xx} + 0.028u_{yy} - 0.041u_{xy}$	0.025
Equation (2)	$u_t = +0.000u_x + 0.000u_y + 1.000u_{xx} + 1.000u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = -0.000u_x + 0.000u_y + 1.260u_{xx} + 1.260u_{yy} - 0.001u_{xy}$	0.027
TaylorNet (grid)	$u_t = -0.001u_x - 0.000u_y + 1.251u_{xx} + 1.250u_{yy} + 0.003u_{xy}$	0.025
PDENet	$u_t = +0.000u_x + 0.000u_y + 0.865u_{xx} + 0.871u_{yy} + 0.006u_{xy}$	0.007
Equation (3)	$u_t = +2.900u_x - 0.300u_y + 0.500u_{xx} + 0.200u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = +3.645u_x - 0.379u_y + 0.780u_{xx} + 0.265u_{yy} - 0.038u_{xy}$	0.129
TaylorNet (grid)	$u_t = +3.646u_x - 0.379u_y + 0.783u_{xx} + 0.258u_{yy} - 0.033u_{xy}$	0.130
PDENet	$u_t = +3.512u_x - 0.364u_y + 0.750u_{xx} + 0.245u_{yy} - 0.029u_{xy}$	0.089
Equation (4)	$u_t = +0.200u_x + 2.900u_y + 1.600u_{xx} + 0.300u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = +0.250u_x + 3.640u_y + 1.985u_{xx} + 0.534u_{yy} + 0.019u_{xy}$	0.151
TaylorNet (grid)	$u_t = +0.249u_x + 3.648u_y + 1.989u_{xx} + 0.537u_{yy} + 0.016u_{xy}$	0.154
PDENet	$u_t = +0.242u_x + 3.513u_y + 1.266u_{xx} + 0.467u_{yy} + 0.014u_{xy}$	0.103
Equation (5)	$u_t = -1.200u_x + 0.000u_y + 0.500u_{xx} + 0.900u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = -1.495u_x - 0.000u_y + 0.657u_{xx} + 1.127u_{yy} + 0.001u_{xy}$	0.033
TaylorNet (grid)	$u_t = -1.496u_x - 0.001u_y + 0.658u_{xx} + 1.132u_{yy} + 0.002u_{xy}$	0.033
PDENet	$u_t = -1.454u_x - 0.000u_y + 0.626u_{xx} + 1.077u_{yy} - 0.001u_{xy}$	0.022

Figure 3: Reconstruction MSE (red) and forecast MSE (blue) in relation to the number of neighbors (K) on grid and non-grid (cloud) data.

4.3. Hyperparameter study

We further investigate the influence of the number of neighbors ($K \in \{5, 50\}$) as described in Section 3.1. This overspecifies the equation system and increases the available data to compute the derivatives. We use equation (3) from Table 2 with 4096 points and $L = 8$ prediction steps during training and evaluate the same way as in Section 4.2.

4.4. Results

Our experiments demonstrate that TaylorNet is effective for both forecasting and equation reconstruction tasks, particularly when handling non-grid data. As shown in Table 1, TaylorNet performs competitively with models specifically designed for grid-structured data, such as CNN-based models and PDENet. While those models tend to achieve slightly lower errors in most scenarios, they are limited to structured data, whereas TaylorNet provides the added flexibility of operating on arbitrarily sampled spatial observations.

In the differential equation reconstruction task, shown in Table 2, TaylorNet consistently identifies the correct structure of the governing PDE. Although it tends to overestimate the magnitudes of the coefficients, the model successfully distinguishes between relevant and irrelevant terms in the equation. This behavior is also observed in PDENet, though typically with less deviation. Importantly, the reconstruction performance remains consistent across both grid and non-grid data, indicating TaylorNet’s robustness to sampling irregularities.

An ablation study (Figure 3) further highlights the impact of the number of neighboring points K used for Taylor approximation. On grid data, increasing K beyond a certain point degrades performance due to oversmoothing, as redundant spatial information blurs local features. For scattered (cloud) data, the behavior differs: at very low values of K , the model struggles to learn meaningful dynamics, likely due to insufficient spatial information. As K increases, the quality of reconstruction improves, though forecasting remains a challenge, likely due to numerical instabilities during rollout. Finally, for $K \geq 30$, the performance of TaylorNet on cloud data approaches that of its performance on grid data for both tasks.

Overall, our findings suggest that fewer neighbors are needed for accurate modeling on grid data due to the regular spatial distribution. In contrast, non-uniform sampling in cloud data requires larger neighborhoods to mitigate numerical instability introduced by irregular point spacing. We hypothesize that the marginal performance advantage of CNNs and PDENet on grid data stems from their use of explicitly structured spatial relationships—either through fixed finite difference stencils or learned convolutional kernels, and the relatively large number of neighbors required for TaylorNet to achieve rollout stability.

Additional results can be seen in Appendices B and C.

5. Conclusion and Future Work

In this paper, we have proposed a novel architecture for learning dynamical systems from non-grid data capable of both forecasting the evolution of the system as well as reconstructing the governing equation. We have shown in several experiments on linear advection-diffusion equations, that TaylorNet is capable of accurately solving both tasks on both grid and non-grid observations. For gridded data, our model performs on par with the other models capable of forecasting and equation reconstruction. In future work, we intend to extend the approach to other types of equations and study the influence of noise on the stability and accuracy of the TaylorNet.

Acknowledgments

The project underlying this publication was funded by the German Federal Ministry of Education and Research under the grant number 16DKWN0099B (MAGNET4Cardiac7T). The responsibility for the content of this publication lies with the authors.

References

- Ibrahim Ayed, Emmanuel de Bézenac, Arthur Pajot, Julien Brajard, and Patrick Gallinari. Learning dynamical systems from partial observations. *CoRR*, abs/1902.11136, 2019.
- Kamyar Azizzadenesheli, Nikola Kovachki, Zongyi Li, Miguel Liu-Schiaffini, Jean Kossaifi, and Anima Anandkumar. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, 6(5):320–328, 2024. ISSN 2522-5820.
- Peter Bauer, Alan Thorpe, and Gilbert Brunet. The quiet revolution of numerical weather prediction. *Nature*, 525(7567):47–55, 2015.
- BN Biswas, Somnath Chatterjee, SP Mukherjee, and Subhradeep Pal. A discussion on euler method: A review. *Electronic Journal of Mathematical Analysis and Applications*, 1(2):294–317, 2013.
- Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.
- Keaton J Burns, Geoffrey M Vasil, Jeffrey S Oishi, Daniel Lecoanet, and Benjamin P Brown. Dedalus: A flexible framework for numerical simulations with spectral methods. *Physical Review Research*, 2(2):023068, 2020.
- Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.
- Charles M Close, Dean K Frederick, and Jonathan C Newell. *Modeling and analysis of dynamic systems*. John Wiley & Sons, 2001.
- Mehdi Dehghan. Numerical solution of the three-dimensional advection–diffusion equation. *Applied Mathematics and Computation*, 150(1):5–19, 2004.
- Johannes Jisse Duistermaat and Johan AC Kolk. *Multidimensional real analysis I: differentiation*, volume 86. Cambridge University Press, 2004.
- Andrzej Dulny, Andreas Hotho, and Anna Krause. Neuralpde: Modelling dynamical systems from data. In *KI 2022: Advances in Artificial Intelligence: 45th German Conference on AI, Trier, Germany, September 19–23, 2022, Proceedings*, pages 75–89. Springer, 2022.
- Andrzej Dulny, Andreas Hotho, and Anna Krause. Dynabench: A benchmark dataset for learning dynamical systems from low-resolution data, 2023.

- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- Mustafa Gülsu and Mehmet Sezer. A taylor polynomial approach for solving differential-difference equations. *Journal of Computational and Applied Mathematics*, 186(2):349–364, 2006.
- Valerii Iakovlev, Markus Heinonen, and Harri Lähdesmäki. Learning continuous-time pdes from sparse data with graph neural networks. *arXiv preprint arXiv:2006.08956*, 2020.
- Matthias Karlbauer, Sebastian Otte, Hendrik Lensch, Thomas Scholten, Volker Wulfmeyer, and Martin V Butz. A distributed neural network architecture for robust non-linear spatio-temporal prediction. *arXiv preprint arXiv:1912.11141*, 2019.
- Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.
- Christian Kuehn. *PDE dynamics: an introduction*, volume 23. SIAM, 2019.
- Jun Li, Gan Sun, Guoshuai Zhao, and H Lehman Li-wei. Robust low-rank discovery of data-driven partial differential equations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 767–774, 2020a.
- Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole Graph Neural Operator for Parametric Partial Differential Equations. In *Advances in Neural Information Processing Systems*, volume 33, pages 6755–6766. Curran Associates, Inc., 2020b. URL <https://proceedings.neurips.cc/paper/2020/hash/4b21cf96d4cf612f239a6c322b10c8fe-Abstract.html>.
- Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=c8P9NQVtmn0>.
- Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. Pde-net: Learning pdes from data. In *International conference on machine learning*, pages 3208–3216. PMLR, 2018.
- Zichao Long, Yiping Lu, and Bin Dong. Pde-net 2.0: Learning pdes from data with a numeric-symbolic hybrid deep network. *Journal of Computational Physics*, 399:108925, 2019.
- Timothy Praditia, Matthias Karlbauer, Sebastian Otte, Sergey Oladyshkin, Martin V Butz, and Wolfgang Nowak. Finite volume neural network: modeling subsurface contaminant transport. *arXiv preprint arXiv:2104.06010*, 2021.

- Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part II): Data-driven discovery of nonlinear partial differential equations, November 2017a. URL <http://arxiv.org/abs/1711.10566>. type: article.
- Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations, 2017b. URL <https://arxiv.org/abs/1711.10566>.
- Jitendra R Raol, Gopalrathnam Girija, and Jatinder Singh. *Modelling and parameter estimation of dynamic systems*, volume 65. Iet, 2004.
- Stephan Rasp, Peter D Dueben, Sebastian Scher, Jonathan A Weyn, Soukayna Mouatadid, and Nils Thuerey. Weatherbench: a benchmark data set for data-driven weather forecasting. *Journal of Advances in Modeling Earth Systems*, 12(11):e2020MS002203, 2020.
- Mehmet Sezer, Aysen Karamete, and Mustafa Gülsu. Taylor polynomial solutions of systems of linear differential equations with variable coefficients. *International Journal of Computer Mathematics*, 82(6):755–764, 2005.
- Chi Chiu So, Tsz On Li, Chufang Wu, and Siu Pang Yung. Differential spectral normalization (dsn) for pde discovery. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9675–9684, 2021.
- Jun Zhang and Wenjun Ma. Data-driven discovery of governing equations for fluid dynamics based on molecular simulation. *Journal of Fluid Mechanics*, 892:A5, 2020.
- Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16259–16268, 2021.

Appendix A. Equation parameters

To evaluate the performance of TaylorNet, we generate synthetic datasets based on a class of two-dimensional advection-diffusion equations using the DynaBench benchmark suite. The general form of the partial differential equation (PDE) used to describe the dynamics is:

$$u_t = \alpha_1 u_x + \alpha_2 u_y + \alpha_3 u_{xx} + \alpha_4 u_{yy}$$

where $u(x, y, t)$ denotes the system state at spatial coordinates (x, y) and time t , and the coefficients α_i control the relative influence of advection and diffusion in each spatial direction. Each equation is initialized with a random smooth initial condition, which is constructed by summing several two-dimensional Gaussian functions with randomly sampled centers and variances. This setup allows for a rich variety of spatial patterns and behaviors over time.

The simulations span a temporal window from $t = 0$ to $t = 10$, with a fixed time step of $\Delta t = 0.1$, resulting in 101 time frames per trajectory. The evolution of the PDEs is numerically solved using the pseudo-spectral solver Dedalus (Burns et al. (2020)), which provides high-precision integration suitable for analyzing fine-grained spatiotemporal dynamics. Each equation is generated using 1000 different initial conditions. The trajectories are split into 800 train, 100 validation and 100 test trajectories.

Table 3 lists the specific equations used in our experiments.

Table 3: Full description of the equations used to generate the data. The first equation (1) describes a purely advective system, whereas equation (2) describes pure diffusion. The remaining equations (3)-(10) describe the mixed advection-diffusion process with different parameters

(1)	$u_t =$	$-1.000u_x$	$+1.000u_y$	$+0.000u_{xx}$	$+0.000u_{yy}$	$+0.000u_{xy}$
(2)	$u_t =$	$+0.000u_x$	$+0.000u_y$	$+1.000u_{xx}$	$+1.000u_{yy}$	$+0.000u_{xy}$
(3)	$u_t =$	$+0.100u_x$	$-1.700u_y$	$+0.100u_{xx}$	$+0.700u_{yy}$	$+0.000u_{xy}$
(4)	$u_t =$	$+0.200u_x$	$+2.900u_y$	$+1.600u_{xx}$	$+0.300u_{yy}$	$+0.000u_{xy}$
(5)	$u_t =$	$+1.600u_x$	$-0.400u_y$	$+0.800u_{xx}$	$+0.000u_{yy}$	$+0.000u_{xy}$
(6)	$u_t =$	$+2.900u_x$	$-0.300u_y$	$+0.500u_{xx}$	$+0.200u_{yy}$	$+0.000u_{xy}$
(7)	$u_t =$	$-1.900u_x$	$+0.700u_y$	$+0.000u_{xx}$	$+1.700u_{yy}$	$+0.000u_{xy}$
(8)	$u_t =$	$+1.000u_x$	$+1.000u_y$	$+1.000u_{xx}$	$+1.000u_{yy}$	$+0.000u_{xy}$
(9)	$u_t =$	$-1.200u_x$	$+0.000u_y$	$+0.500u_{xx}$	$+0.900u_{yy}$	$+0.000u_{xy}$
(10)	$u_t =$	$+0.000u_x$	$+1.600u_y$	$+0.300u_{xx}$	$+1.300u_{yy}$	$+0.000u_{xy}$

Appendix B. Additional equation reconstruction results

In this supplementary material section we include additional equation reconstruction results for resolutions 16×16 , 32×32 and 64×64 (in full).

Table 4: Learned PDE parameters for different advection-diffusion equations on a resolution of 16×16

model	Equation	Δ (MSE)
Equation (1)	$u_t = -1.000u_x + 1.000u_y + 0.000u_{xx} + 0.000u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = -2.103u_x + 2.100u_y + 0.635u_{xx} + 0.620u_{yy} - 0.701u_{xy}$	0.741
TaylorNet (grid)	$u_t = -2.237u_x + 2.233u_y + 0.605u_{xx} + 0.606u_{yy} - 0.789u_{xy}$	0.881
PDENet	$u_t = -1.472u_x + 1.469u_y + 0.094u_{xx} + 0.094u_{yy} - 0.070u_{xy}$	0.093
Equation (2)	$u_t = +0.000u_x + 0.000u_y + 1.000u_{xx} + 1.000u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = +0.003u_x - 0.003u_y + 1.761u_{xx} + 1.751u_{yy} + 0.002u_{xy}$	0.229
TaylorNet (grid)	$u_t = -0.005u_x + 0.001u_y + 1.812u_{xx} + 1.795u_{yy} - 0.018u_{xy}$	0.258
PDENet	$u_t = -0.000u_x + 0.000u_y + 1.294u_{xx} + 1.292u_{yy} + 0.001u_{xy}$	0.034
Equation (3)	$u_t = +2.900u_x - 0.300u_y + 0.500u_{xx} + 0.200u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = +5.076u_x - 0.533u_y + 2.251u_{xx} + 0.435u_{yy} - 0.292u_{xy}$	1.600
TaylorNet (grid)	$u_t = +5.233u_x - 0.551u_y + 2.343u_{xx} + 0.365u_{yy} - 0.301u_{xy}$	1.804
PDENet	$u_t = +3.740u_x - 0.403u_y + 0.860u_{xx} + 0.268u_{yy} - 0.039u_{xy}$	0.170
Equation (4)	$u_t = +0.200u_x + 2.900u_y + 1.600u_{xx} + 0.300u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = +0.353u_x + 4.921u_y + 3.273u_{xx} + 1.666u_{yy} + 0.142u_{xy}$	1.759
TaylorNet (grid)	$u_t = +0.350u_x + 5.053u_y + 3.448u_{xx} + 1.699u_{yy} + 0.158u_{xy}$	2.011
PDENet	$u_t = +0.257u_x + 3.789u_y + 2.027u_{xx} + 0.617u_{yy} + 0.024u_{xy}$	0.215
Equation (5)	$u_t = -1.200u_x + 0.000u_y + 0.500u_{xx} + 0.900u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = -1.986u_x + 0.000u_y + 1.130u_{xx} + 1.655u_{yy} - 0.001u_{xy}$	0.317
TaylorNet (grid)	$u_t = -2.048u_x + 0.001u_y + 1.141u_{xx} + 1.731u_{yy} + 0.000u_{xy}$	0.364
PDENet	$u_t = -1.553u_x - 0.000u_y + 0.690u_{xx} + 1.159u_{yy} - 0.002u_{xy}$	0.046
Equation (6)	$u_t = +1.600u_x - 0.400u_y + 0.800u_{xx} + 0.000u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = +2.676u_x - 0.750u_y + 1.784u_{xx} - 0.029u_{yy} - 0.247u_{xy}$	0.462
TaylorNet (grid)	$u_t = +2.856u_x - 0.788u_y + 1.977u_{xx} - 0.101u_{yy} - 0.243u_{xy}$	0.637
PDENet	$u_t = +2.030u_x - 0.580u_y + 1.086u_{xx} + 0.014u_{yy} - 0.025u_{xy}$	0.060
Equation (7)	$u_t = -1.900u_x + 0.700u_y + 0.000u_{xx} + 1.700u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = -3.666u_x + 1.275u_y + 0.642u_{xx} + 4.856u_{yy} - 0.547u_{xy}$	2.824
TaylorNet (grid)	$u_t = -3.791u_x + 1.317u_y + 0.550u_{xx} + 5.347u_{yy} - 0.465u_{xy}$	3.555
PDENet	$u_t = -2.699u_x + 0.878u_y + 0.225u_{xx} + 2.060u_{yy} - 0.057u_{xy}$	0.171
Equation (8)	$u_t = +0.000u_x + 1.600u_y + 0.300u_{xx} + 1.300u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = -0.000u_x + 2.622u_y + 0.502u_{xx} + 2.601u_{yy} + 0.020u_{xy}$	0.555
TaylorNet (grid)	$u_t = -0.003u_x + 2.709u_y + 0.459u_{xx} + 2.751u_{yy} + 0.012u_{xy}$	0.672
PDENet	$u_t = -0.001u_x + 2.020u_y + 0.396u_{xx} + 1.715u_{yy} + 0.002u_{xy}$	0.071
Equation (9)	$u_t = +0.100u_x - 1.700u_y + 0.100u_{xx} + 0.700u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = +0.178u_x - 2.869u_y + 0.164u_{xx} + 1.742u_{yy} - 0.059u_{xy}$	0.494
TaylorNet (grid)	$u_t = +0.189u_x - 2.989u_y + 0.120u_{xx} + 1.862u_{yy} - 0.074u_{xy}$	0.605
PDENet	$u_t = +0.136u_x - 2.180u_y + 0.138u_{xx} + 0.980u_{yy} - 0.006u_{xy}$	0.062
Equation (10)	$u_t = +1.000u_x + 1.000u_y + 1.000u_{xx} + 1.000u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = +1.621u_x + 1.622u_y + 1.921u_{xx} + 1.904u_{yy} + 0.294u_{xy}$	0.505
TaylorNet (grid)	$u_t = +1.669u_x + 1.658u_y + 1.951u_{xx} + 1.933u_{yy} + 0.297u_{xy}$	0.549
PDENet	$u_t = +1.273u_x + 1.275u_y + 1.311u_{xx} + 1.310u_{yy} + 0.038u_{xy}$	0.069

Table 5: Learned PDE parameters for different advection-diffusion equations on a resolution of 32×32

model	Equation	Δ (MSE)
Equation (1)	$u_t = -1.000u_x + 1.000u_y + 0.000u_{xx} + 0.000u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = -1.670u_x + 1.671u_y + 0.234u_{xx} + 0.232u_{yy} - 0.355u_{xy}$	0.227
TaylorNet (grid)	$u_t = -1.709u_x + 1.710u_y + 0.240u_{xx} + 0.242u_{yy} - 0.382u_{xy}$	0.254
PDENet	$u_t = -1.315u_x + 1.314u_y + 0.057u_{xx} + 0.057u_{yy} - 0.056u_{xy}$	0.042
Equation (2)	$u_t = +0.000u_x + 0.000u_y + 1.000u_{xx} + 1.000u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = -0.001u_x + 0.000u_y + 1.371u_{xx} + 1.368u_{yy} - 0.004u_{xy}$	0.055
TaylorNet (grid)	$u_t = -0.001u_x - 0.000u_y + 1.389u_{xx} + 1.385u_{yy} - 0.003u_{xy}$	0.060
PDENet	$u_t = +0.000u_x + 0.000u_y + 1.247u_{xx} + 1.247u_{yy} - 0.002u_{xy}$	0.024
Equation (3)	$u_t = +2.900u_x - 0.300u_y + 0.500u_{xx} + 0.200u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = +4.011u_x - 0.418u_y + 1.101u_{xx} + 0.304u_{yy} - 0.088u_{xy}$	0.325
TaylorNet (grid)	$u_t = +4.033u_x - 0.426u_y + 1.132u_{xx} + 0.300u_{yy} - 0.093u_{xy}$	0.344
PDENet	$u_t = +3.559u_x - 0.371u_y + 0.765u_{xx} + 0.251u_{yy} - 0.030u_{xy}$	0.103
Equation (4)	$u_t = +0.200u_x + 2.900u_y + 1.600u_{xx} + 0.300u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = +0.269u_x + 3.970u_y + 2.170u_{xx} + 0.799u_{yy} + 0.050u_{xy}$	0.345
TaylorNet (grid)	$u_t = +0.271u_x + 4.004u_y + 2.201u_{xx} + 0.824u_{yy} + 0.042u_{xy}$	0.372
PDENet	$u_t = +0.246u_x + 3.576u_y + 1.991u_{xx} + 0.516u_{yy} + 0.022u_{xy}$	0.132
Equation (5)	$u_t = -1.200u_x + 0.000u_y + 0.500u_{xx} + 0.900u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = -1.619u_x + 0.002u_y + 0.770u_{xx} + 1.241u_{yy} - 0.004u_{xy}$	0.073
TaylorNet (grid)	$u_t = -1.641u_x - 0.002u_y + 0.783u_{xx} + 1.265u_{yy} + 0.002u_{xy}$	0.081
PDENet	$u_t = -1.474u_x - 0.000u_y + 0.646u_{xx} + 1.116u_{yy} - 0.001u_{xy}$	0.029
Equation (6)	$u_t = +1.600u_x - 0.400u_y + 0.800u_{xx} + 0.000u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = +2.197u_x - 0.600u_y + 1.236u_{xx} - 0.009u_{yy} - 0.094u_{xy}$	0.119
TaylorNet (grid)	$u_t = +2.227u_x - 0.614u_y + 1.282u_{xx} - 0.028u_{yy} - 0.090u_{xy}$	0.136
PDENet	$u_t = +1.958u_x - 0.518u_y + 1.025u_{xx} + 0.010u_{yy} - 0.024u_{xy}$	0.039
Equation (7)	$u_t = -1.900u_x + 0.700u_y + 0.000u_{xx} + 1.700u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = -2.879u_x + 0.989u_y + 0.306u_{xx} + 2.574u_{yy} - 0.150u_{xy}$	0.385
TaylorNet (grid)	$u_t = -2.915u_x + 0.996u_y + 0.300u_{xx} + 2.700u_{yy} - 0.140u_{xy}$	0.445
PDENet	$u_t = -2.431u_x + 0.857u_y + 0.110u_{xx} + 2.062u_{yy} - 0.051u_{xy}$	0.090
Equation (8)	$u_t = +0.000u_x + 1.600u_y + 0.300u_{xx} + 1.300u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = +0.001u_x + 2.142u_y + 0.406u_{xx} + 1.867u_{yy} + 0.004u_{xy}$	0.125
TaylorNet (grid)	$u_t = -0.002u_x + 2.155u_y + 0.400u_{xx} + 1.910u_{yy} - 0.003u_{xy}$	0.138
PDENet	$u_t = -0.000u_x + 1.952u_y + 0.371u_{xx} + 1.656u_{yy} + 0.001u_{xy}$	0.051
Equation (9)	$u_t = +0.100u_x - 1.700u_y + 0.100u_{xx} + 0.700u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = +0.143u_x - 2.319u_y + 0.139u_{xx} + 1.121u_{yy} - 0.021u_{xy}$	0.113
TaylorNet (grid)	$u_t = +0.144u_x - 2.343u_y + 0.133u_{xx} + 1.157u_{yy} - 0.027u_{xy}$	0.125
PDENet	$u_t = +0.125u_x - 2.082u_y + 0.126u_{xx} + 0.916u_{yy} - 0.007u_{xy}$	0.039
Equation (10)	$u_t = +1.000u_x + 1.000u_y + 1.000u_{xx} + 1.000u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = +1.328u_x + 1.327u_y + 1.401u_{xx} + 1.399u_{yy} + 0.074u_{xy}$	0.108
TaylorNet (grid)	$u_t = +1.334u_x + 1.333u_y + 1.409u_{xx} + 1.409u_{yy} + 0.081u_{xy}$	0.113
PDENet	$u_t = +1.224u_x + 1.225u_y + 1.264u_{xx} + 1.263u_{yy} + 0.033u_{xy}$	0.048

Table 6: Learned PDE parameters for different advection-diffusion equations on a resolution of 64×64

model	Equation	Δ (MSE)
Equation (1)	$u_t = -1.000u_x + 1.000u_y + 0.000u_{xx} + 0.000u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = -1.409u_x + 1.408u_y + 0.104u_{xx} + 0.105u_{yy} - 0.170u_{xy}$	0.077
TaylorNet (grid)	$u_t = -1.420u_x + 1.419u_y + 0.109u_{xx} + 0.108u_{yy} - 0.175u_{xy}$	0.081
PDENet	$u_t = -1.247u_x + 1.247u_y + 0.028u_{xx} + 0.028u_{yy} - 0.041u_{xy}$	0.025
Equation (2)	$u_t = +0.000u_x + 0.000u_y + 1.000u_{xx} + 1.000u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = -0.000u_x + 0.000u_y + 1.260u_{xx} + 1.260u_{yy} - 0.001u_{xy}$	0.027
TaylorNet (grid)	$u_t = -0.001u_x - 0.000u_y + 1.251u_{xx} + 1.250u_{yy} + 0.003u_{xy}$	0.025
PDENet	$u_t = +0.000u_x + 0.000u_y + 0.865u_{xx} + 0.871u_{yy} + 0.006u_{xy}$	0.007
Equation (3)	$u_t = +2.900u_x - 0.300u_y + 0.500u_{xx} + 0.200u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = +3.645u_x - 0.379u_y + 0.780u_{xx} + 0.265u_{yy} - 0.038u_{xy}$	0.129
TaylorNet (grid)	$u_t = +3.646u_x - 0.379u_y + 0.783u_{xx} + 0.258u_{yy} - 0.033u_{xy}$	0.130
PDENet	$u_t = +3.512u_x - 0.364u_y + 0.750u_{xx} + 0.245u_{yy} - 0.029u_{xy}$	0.089
Equation (4)	$u_t = +0.200u_x + 2.900u_y + 1.600u_{xx} + 0.300u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = +0.250u_x + 3.640u_y + 1.985u_{xx} + 0.534u_{yy} + 0.019u_{xy}$	0.151
TaylorNet (grid)	$u_t = +0.249u_x + 3.648u_y + 1.989u_{xx} + 0.537u_{yy} + 0.016u_{xy}$	0.154
PDENet	$u_t = +0.242u_x + 3.513u_y + 1.266u_{xx} + 0.467u_{yy} + 0.014u_{xy}$	0.103
Equation (5)	$u_t = -1.200u_x + 0.000u_y + 0.500u_{xx} + 0.900u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = -1.495u_x - 0.000u_y + 0.657u_{xx} + 1.127u_{yy} + 0.001u_{xy}$	0.033
TaylorNet (grid)	$u_t = -1.496u_x - 0.001u_y + 0.658u_{xx} + 1.132u_{yy} + 0.002u_{xy}$	0.033
PDENet	$u_t = -1.454u_x - 0.000u_y + 0.626u_{xx} + 1.077u_{yy} - 0.001u_{xy}$	0.022
Equation (6)	$u_t = +1.600u_x - 0.400u_y + 0.800u_{xx} + 0.000u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = +2.015u_x - 0.527u_y + 1.060u_{xx} - 0.003u_{yy} - 0.043u_{xy}$	0.052
TaylorNet (grid)	$u_t = +2.010u_x - 0.529u_y + 1.060u_{xx} - 0.008u_{yy} - 0.038u_{xy}$	0.051
PDENet	$u_t = +1.937u_x - 0.496u_y + 1.021u_{xx} + 0.005u_{yy} - 0.021u_{xy}$	0.034
Equation (7)	$u_t = -1.900u_x + 0.700u_y + 0.000u_{xx} + 1.700u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = -2.498u_x + 0.878u_y + 0.138u_{xx} + 2.105u_{yy} - 0.042u_{xy}$	0.115
TaylorNet (grid)	$u_t = -2.507u_x + 0.887u_y + 0.136u_{xx} + 2.120u_{yy} - 0.043u_{xy}$	0.120
PDENet	$u_t = -2.334u_x + 0.845u_y + 0.080u_{xx} + 1.612u_{yy} - 0.043u_{xy}$	0.045
Equation (8)	$u_t = +0.000u_x + 1.600u_y + 0.300u_{xx} + 1.300u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = +0.001u_x + 1.991u_y + 0.378u_{xx} + 1.667u_{yy} + 0.003u_{xy}$	0.059
TaylorNet (grid)	$u_t = -0.000u_x + 1.989u_y + 0.371u_{xx} + 1.670u_{yy} - 0.002u_{xy}$	0.059
PDENet	$u_t = +0.000u_x + 1.934u_y + 0.347u_{xx} + 1.374u_{yy} + 0.000u_{xy}$	0.024
Equation (9)	$u_t = +0.100u_x - 1.700u_y + 0.100u_{xx} + 0.700u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = +0.114u_x - 0.752u_y + 0.271u_{xx} + 0.375u_{yy} - 0.056u_{xy}$	0.207
TaylorNet (grid)	$u_t = +0.127u_x - 2.130u_y + 0.125u_{xx} + 0.943u_{yy} - 0.009u_{xy}$	0.049
PDENet	$u_t = +0.121u_x - 2.058u_y + 0.122u_{xx} + 0.906u_{yy} - 0.006u_{xy}$	0.034
Equation (10)	$u_t = +1.000u_x + 1.000u_y + 1.000u_{xx} + 1.000u_{yy} + 0.000u_{xy}$	
TaylorNet	$u_t = +1.239u_x + 1.240u_y + 1.270u_{xx} + 1.267u_{yy} + 0.031u_{xy}$	0.052
TaylorNet (grid)	$u_t = +1.242u_x + 1.242u_y + 1.268u_{xx} + 1.268u_{yy} + 0.028u_{xy}$	0.052
PDENet	$u_t = +1.209u_x + 1.209u_y + 0.861u_{xx} + 0.868u_{yy} + 0.032u_{xy}$	0.025

Appendix C. Additional forecasting results

Table 7: Prediction MSE after 32 prediction steps compared with the PDE-Net evaluated on all equations (1)–(10) with previously unseen initial conditions. The resolution (R) 128×128 contains the original simulation data structured on a grid.

R	Model	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
16	TaylorNet	1.4e-02	2.1e-04	5.5e+01	3.0e-03	8.4e-04	2.6e-03	5.8e-03	1.2e-03	1.9e-03	5.8e-04
	TaylorNet (grid)	1.4e-02	1.9e-04	4.3e-03	3.2e-03	8.9e-04	2.6e-03	5.8e-03	6.5e-04	1.9e-03	5.8e-04
	PDENet	7.9e-03	4.5e-05	2.1e-03	1.6e-03	2.9e-04	9.7e-04	3.3e-03	2.2e-04	6.8e-04	1.8e-04
	CNN	4.3e-03	3.4e-05	2.4e-03	1.1e-03	2.0e-04	6.6e-04	1.8e-03	1.6e-04	4.9e-04	1.3e-04
32	TaylorNet	7.4e-02	6.7e-05	2.7e-03	2.0e-03	3.9e-04	1.5e-01	4.2e-03	3.7e-04	8.8e-04	2.4e-04
	TaylorNet (grid)	1.2e-02	7.2e-05	2.9e-03	2.1e-03	4.3e-04	1.3e-03	4.5e-03	3.3e-04	9.4e-04	2.5e-04
	PDENet	5.8e-03	3.6e-05	1.7e-03	1.3e-03	2.3e-04	7.0e-04	2.4e-03	1.9e-04	5.3e-04	1.5e-04
	CNN	4.1e-03	8.3e-05	1.6e-03	1.2e-03	2.1e-04	9.5e-04	2.0e-03	1.7e-04	5.0e-04	2.6e-04
64	TaylorNet	8.1e-03	3.8e-05	2.8e-03	1.5e-03	2.5e-04	7.9e-04	2.9e-03	2.1e-04	6.6e-03	1.6e-04
	TaylorNet (grid)	8.4e-03	3.5e-05	2.0e-03	1.5e-03	2.5e-04	7.8e-04	2.9e-03	2.1e-04	6.0e-04	1.6e-04
	PDENet	4.8e-03	3.9e-05	1.6e-03	1.3e-03	2.1e-04	6.3e-04	2.1e-03	1.6e-04	5.0e-04	1.7e-04
	CNN	4.4e-03	5.0e-05	1.7e-03	1.3e-03	3.3e-04	6.4e-04	7.8e-03	2.4e-04	6.0e-04	2.0e-03
128	TaylorNet (grid)	5.8e-03	3.0e-05	1.7e-03	1.3e-03	2.3e-04	6.5e-04	2.4e-03	1.8e-04	5.1e-04	1.4e-04
	PDENet	4.4e-03	2.4e-03	2.3e-03	3.9e-03	1.7e-03	1.1e-03	3.8e-03	1.9e-03	9.2e-04	2.7e-03
	CNN	4.1e-03	6.4e-03	8.3e-03	2.8e-03	1.4e-03	3.8e-03	9.5e+02	1.4e-02	5.9e-04	6.0e+00

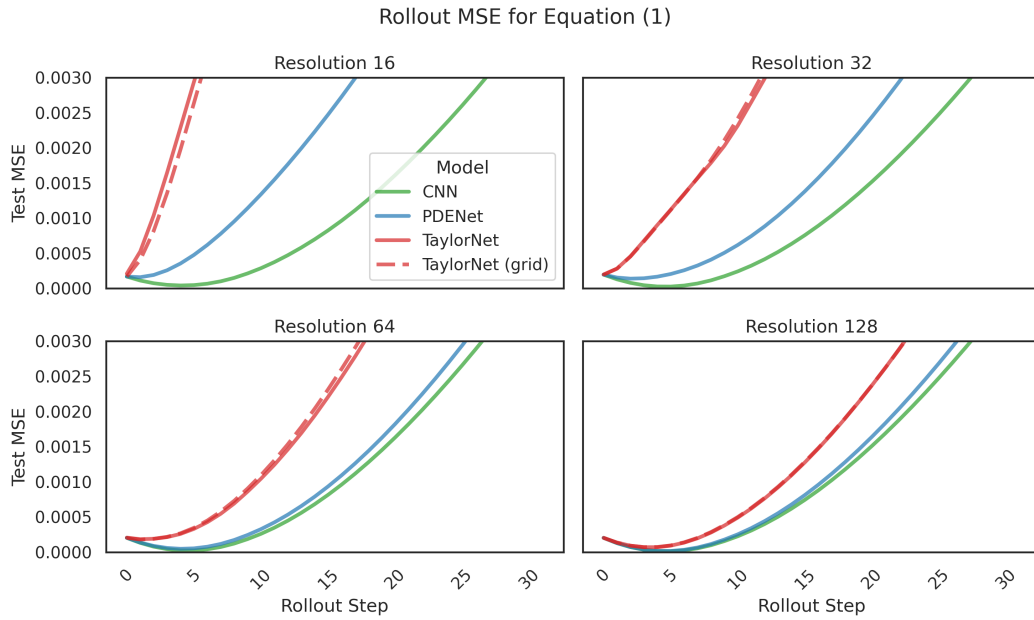


Figure 4: MSE for different resolutions along the rollout trajectory (Rollout Step) for the Equation (1).

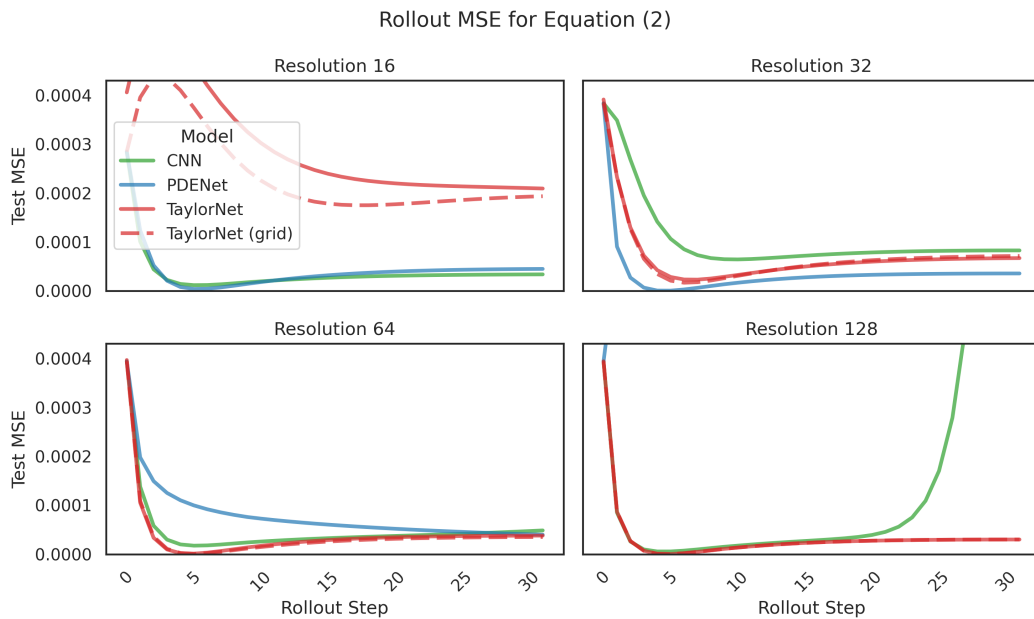


Figure 5: MSE for different resolutions along the rollout trajectory (Rollout Step) for the Equation (2).

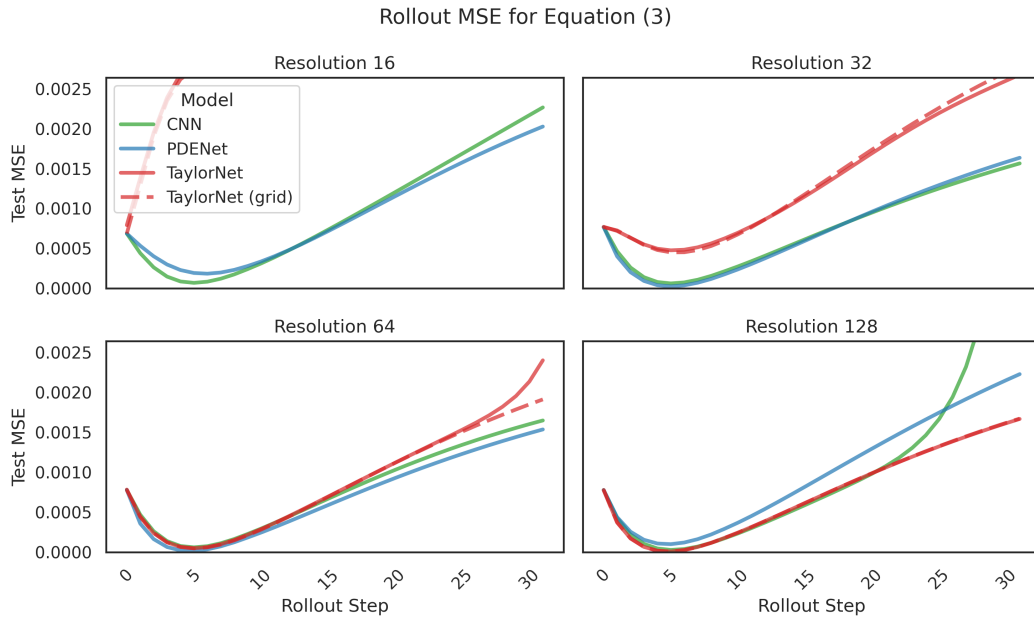


Figure 6: MSE for different resolutions along the rollout trajectory (Rollout Step) for the Equation (3).

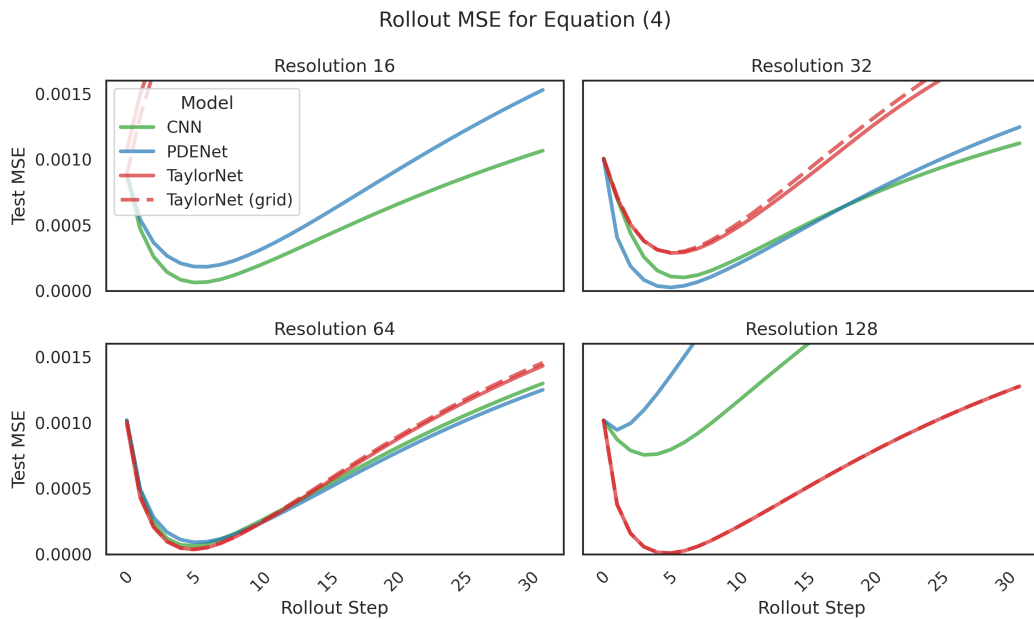


Figure 7: MSE for different resolutions along the rollout trajectory (Rollout Step) for the Equation (4).

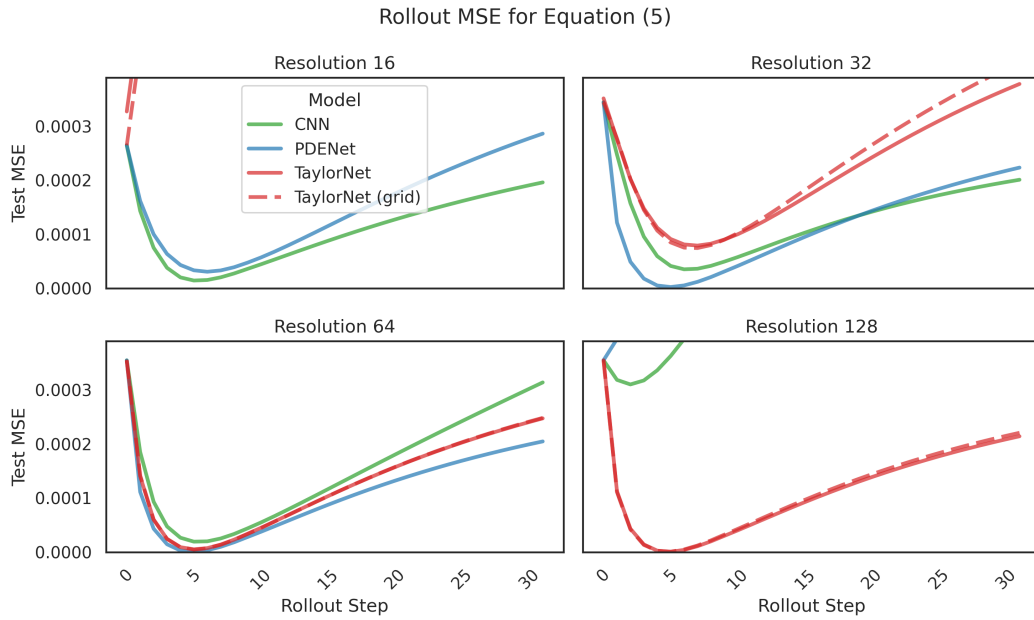


Figure 8: MSE for different resolutions along the rollout trajectory (Rollout Step) for the Equation (5).

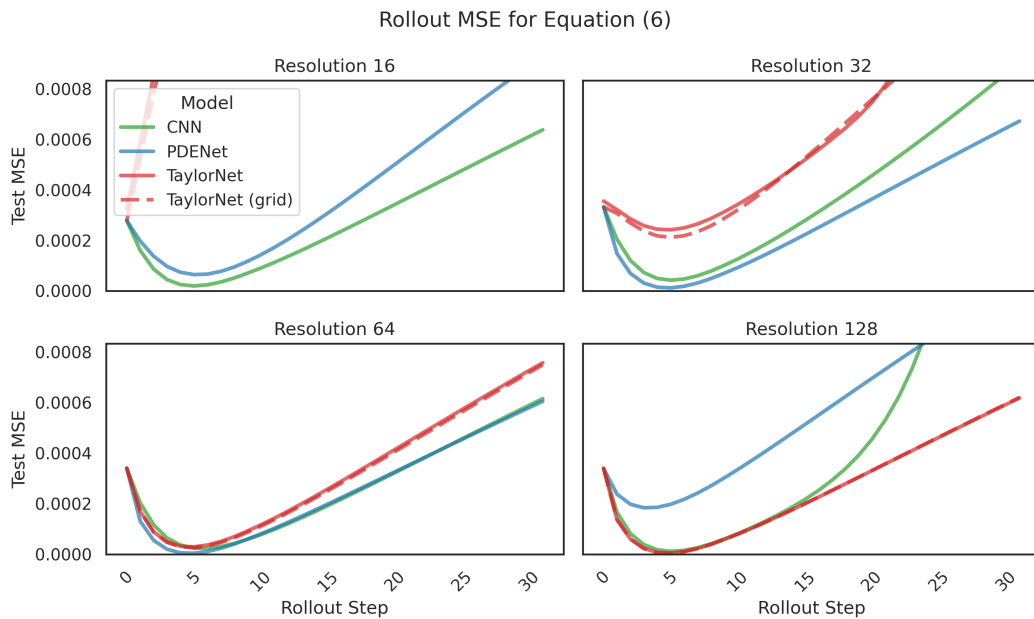


Figure 9: MSE for different resolutions along the rollout trajectory (Rollout Step) for the Equation (6).

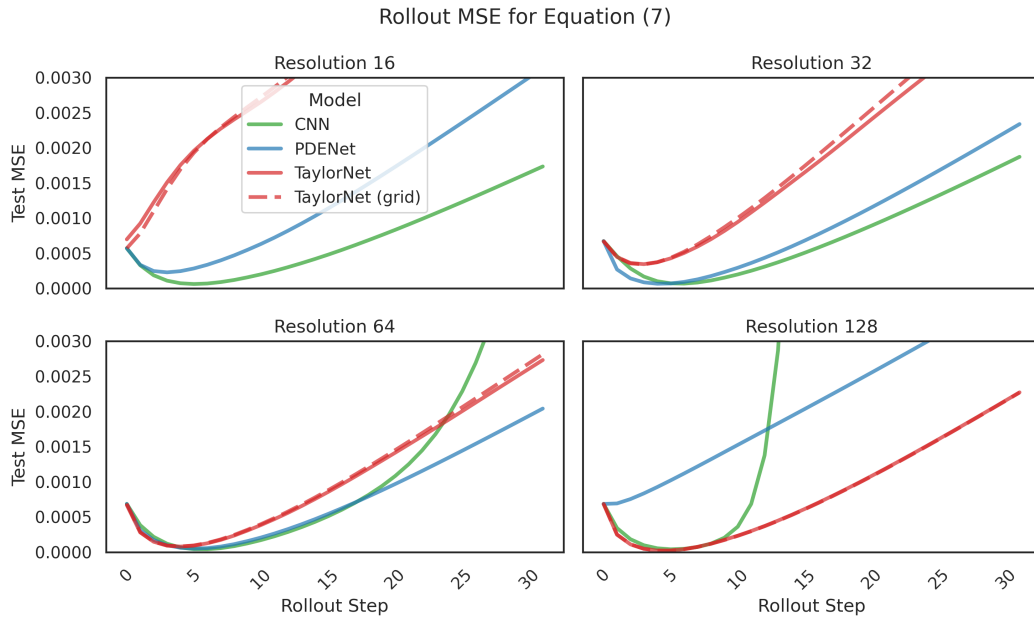


Figure 10: MSE for different resolutions along the rollout trajectory (Rollout Step) for the Equation (7).

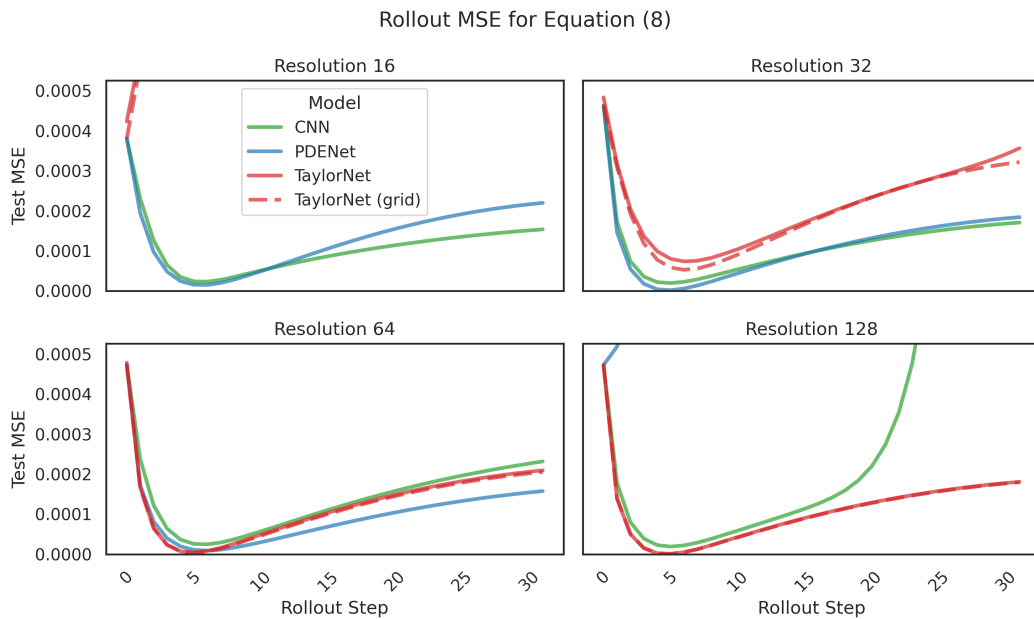


Figure 11: MSE for different resolutions along the rollout trajectory (Rollout Step) for the Equation (8).

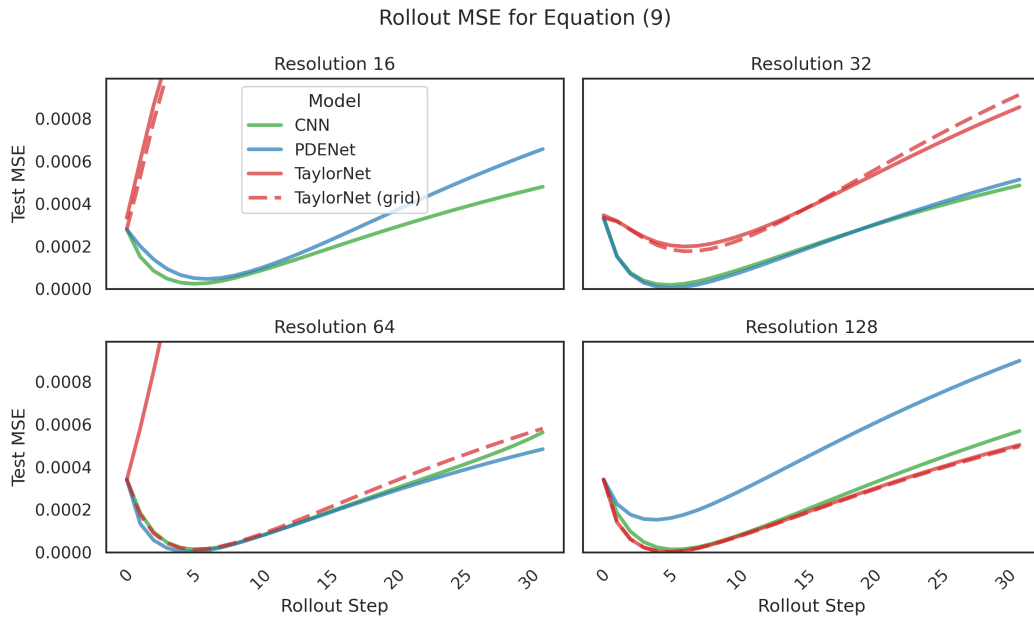


Figure 12: MSE for different resolutions along the rollout trajectory (Rollout Step) for the Equation (9).

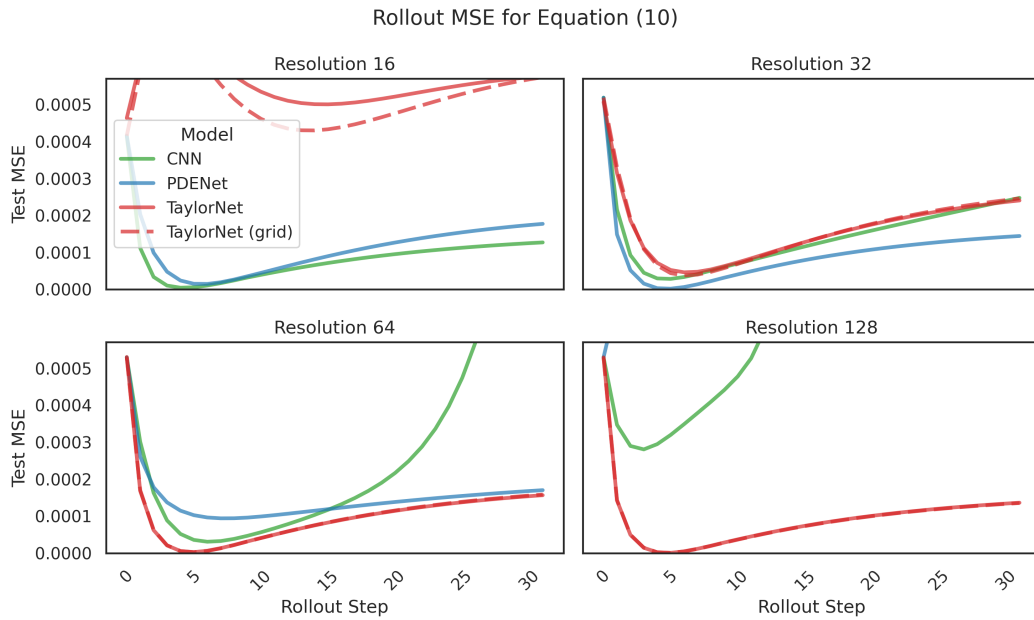


Figure 13: MSE for different resolutions along the rollout trajectory (Rollout Step) for the Equation (10).