Deep Latent Regularity Network for Modeling Stochastic Partial Differential Equations

Shiqi Gong[1*†], Peiyan Hu[1], Qi Meng[2†], Yue Wang[2], Rongchan Zhu[3], Bingguang Chen[1], Zhiming Ma[1], Hao Ni[4, 5†], Tie-Yan Liu[2]

[1] Academy of Mathematics and Systems Science, Chinese Academy of Sciences
[2] Microsoft Research AI4Science
[3] Bielefeld University
[4] Department of Mathematics, University College London
[5] The Alan Turing Institute

Abstract

Stochastic partial differential equations (SPDEs) are crucial for modelling dynamics with randomness in many areas including economics, physics, and atmospheric sciences. Recently, using deep learning approaches to learn the PDE solution for accelerating PDE simulation becomes increasingly popular. However, SPDEs have two unique properties that require new design on the models. First, the model to approximate the solution of SPDE should be generalizable over both initial conditions and the random sampled forcing term. Second, the random forcing terms usually have poor regularity whose statistics may diverge (e.g., the space-time white noise). To deal with the problems, in this work, we design a deep neural network called *Deep Latent Regularity Net* (DLR-Net). DLR-Net includes a regularity feature block as the main component, which maps the initial condition and the random forcing term to a set of regularity features. The processing of regularity features is inspired by regularity structure theory and the features provably compose a set of basis to represent the SPDE solution. The regularity features are then fed into a small backbone neural operator to get the output. We conduct experiments on various SPDEs including the dynamic Φ^4_1 model and the stochastic 2D Navier-Stokes equation to predict their solutions, and the results demonstrate that the proposed DLR-Net can achieve SOTA accuracy compared with the baselines. Moreover, the inference time is over 20 times faster than the traditional numerical solver and is comparable with the baseline deep learning models.

Introduction

Stochastic partial differential equations (SPDEs) are crucial for modeling dynamics in many areas including economics (Barone-Adesi and Whaley 1987), physics (Uhlenbeck and Ornstein 1930), atmospheric sciences (Hasselmann 1976), biology (Wilkinson 2018), etc. SPDEs, generalized from PDEs, take random forcing terms into consideration, and they are widely used to study the statistical mechanics of dynamical systems. Plenty of scientific models are formulated in the form of SPDEs, such as Φ^4 model arising in the stochastic quantisation of quantum field theory (Hairer 2014a), and stochastic Navier-Stokes equations modeling the statistics of turbulent flows (Buckmaster and Vicol 2019) in atmospheric science. Since SPDEs are so important in scientific areas, studying their dynamics from both theoretical and numerical aspects is of vital importance in both mathematics and physics.

Recently, integrating AI techniques with differential equations to accelerate the dynamics simulation is very popular and there have been many deep learning models arisen (Kovachki et al. 2021; Lu, Jin, and Karniadakis 2019). A promising setup is using deep neural networks to model the solution of parametric PDEs, which maps from input functions (e.g., initial functions) to their corresponding solutions. For example, Neural Operators (Kovachki et al. 2021) and DeepONet (Lu, Jin, and Karniadakis 2019) are two network architectures that can model the PDE operator with theoretical guarantees on universal approximation. These models are trained by supervised learning, relying on pre-given training samples, which are usually generated by the traditional numerical solver. However, SPDEs usually have poor regularity w.r.t the time variable for function-valued noise and singularity w.r.t space for space-time white noise that these models do not take into consideration. Directly applying these kinds of models to learn SPDEs' solutions will meet two problems. The first is the universality of these models to approximate the map from the random forcing to SPDEs' solutions is not guaranteed (e.g., the space-time white noise does not lie in the function spaces in Assumption 1 in Neural Operator (Li et al. 2020c).). The second is that the sample complexity will be large if the ground truth mapping is rough, i.e., it requires more training samples, but the computational cost of acquiring samples relies on numerical solvers which may be expensive.

The model feature vectors (Chevyrev, Gerasimovics, and Weber 2021) inspired by the regularity structure theory (Hairer 2014b) compose a set of bases of SPDEs' mild solution. They have improved regularity compared to the random forcing and have previously been treated as a fixed

*This work was done when he was visiting Microsoft Research.
†Corresponds to Shiqi Gong (gongshiqi15@mails.ucas.ac.cn), Qi Meng (meq@microsoft.com), Hao Ni (h.ni@ucl.ac.uk).

feature transformation, on top of which a model may be built. (Chevyrev, Gerasimovics, and Weber 2021) was the first work to propose the model feature vector for learning the SPDEs. However, (Chevyrev, Gerasimovics, and Weber 2021) used the regularity structure as a deterministic feature extraction and only considered the linear model, which significantly limits the effectiveness of this feature to predict the SPDE solution. Prior knowledge is needed to determine the number of feature vectors, and its computational cost grows exponentially as the order of the features becomes large.

In this paper, we propose a novel approach that combines the advantages of the model feature vectors with modern deep learning frameworks. We propose *Deep Latent Regularity Net* (abbrev. DLR-Net) to model dynamics given by SPDEs. In contrast to (Chevyrev, Gerasimovics, and Weber 2021), the proposed Latent Regularity Net is a trainable layer, which can be stackable to multi-layers to extract the high order feature information and tackle the dimension issue of model vector features, hence leading to significant performance boost. The stacked regularity feature block as the main component, consecutively encodes the random forcing to regularity features. We prove that the output features can approximate SPDEs' mild solutions and that the terms of the features can be selected in a data-dependent way. Moreover, the regularity feature block has fewer parameters compared to the deep-learning-based baselines, which is helpful to reduce the sample complexity in learning.

Our Contributions. We introduce the DLR-Net to model the dynamics of parametric SPDEs. This deep-learning-based method has three advantages as follows:

- The DLR-Net utilizes more information from the equations themselves as we select kernels in regularity feature blocks as semigroup operators from the linear part of the SPDEs, which is physics-informed, resolution-invariant, and then leads to lower loss.
- The DLR-Net is efficient for inferring the solution of SPDE. With accelerating the regularity feature calculation, DLR-Net is over 20 times faster than traditional numerical solver and is comparable with the baseline deep learning models.
- We test the DLR-Net on the dynamic Φ^4_1 model, reaction-diffusion equation with linear multiplicative noise, and the stochastic 2D Navier-Stokes equation. Using the DLR-Net, both the testing accuracy and computational time are enhanced. Specifically, the error is one order of magnitude lower than other baselines.

Related Work

There have been several popular deep-learning-based methods for modeling the dynamics driven by differential equations (Lu, Jin, and Karniadakis 2019; Patel et al. 2021; Kovachki et al. 2021; Li et al. 2020b; Bhattacharya et al. 2020; Nelsen and Stuart 2021; Li et al. 2020a).

Neural ODE and Its Variants. The neural ordinary differential equations (Neural ODEs) (Chen et al. 2018) seek to learn a function f_θ that models an ordinary differential equation $dx_t = f_\theta(x_t, t)dt$. Afterwards, several variants such as Neural CDE(Kidger et al. 2020), Neural RDE(Morrill et al. 2021), and Neural jump SDE(Jia and Benson 2019) were proposed for modelling differential equations with different structures and regularity. For example, Neural RDE is proposed for solving rough differential equations, which has poor regularity w.r.t. time. These models are designed for modelling the temporal dynamics, while they do not consider the effectiveness to extract the spatial information, which limits their performance to process signals varying both in space and time, e.g., PDE operators.

Neural Operator. The Neural Operators (Kovachki et al. 2021) are generalizations of neural networks capable of modelling the maps between spaces of functions, e.g. Fourier Neural Operator (FNO) (Li et al. 2020b). DeepONet is a representative architecture designed with universal approximation to approximate an operator. One of the applications of neural operators is to map a family of functions to the corresponding solutions of parametric PDEs. By taking the randomness into consideration, neural SPDE (Salvi and Lemercier 2021) - a neural operator architecture that takes the randomness into consideration is proposed for modelling the SPDE operator.

Path Signature and Model Feature Vectors. Path signature (Chevyrev and Kormilitzin 2016) is a powerful tool in the analysis of time-ordered data. In the rough path theory, path signature can well characterize the path up to a natural equivalence relation. Model feature vectors (Chevyrev, Gerasimovics, and Weber 2021) are the multi-dimensional generalization of path signature, i.e., from the temporal dimension to the spatial-temporal dimension. Both path signature and model feature vectors are used as features in applications for modelling spatial-temporal data, such as a solution of CDE (Morrill et al. 2021) and SPDE (Chevyrev, Gerasimovics, and Weber 2021; Hu et al. 2022) respectively. Because the computational complexity will dramatically increase when calculating high-order signatures, prior knowledge is needed on determining the degree of the features.

In this paper, we propose a deep neural network structure to generate regularity features. It is inspired by feature engineering for SPDEs proposed by (Chevyrev, Gerasimovics, and Weber 2021), but our design contains trainable weights, which goes beyond feature engineering. Since the *regularity features* consist more prior (including the kernel, the initial condition, and the forcing) of the SPDE, it is expected to have a better generalization and lower sample complexity.

Preliminary

Mild Solution of SPDE

We consider an SPDE on $[0, T] \times D$ with the following form

$$\begin{aligned} \partial_t u - \mathcal{L}u &= \mu(u, \partial_1 u, \cdots, \partial_d u) + \sigma(u, \partial_1 u, \cdots, \partial_d u)\xi, \\ u(0, x) &= u_0(x), \end{aligned} \tag{1}$$

where $x \in D \subset \mathbb{R}^d$, $t \in [0, T]$, $\partial_i := \partial/\partial x_i, i = 1, \cdots, d$, $\mathcal{L}$ is a linear differential operator, $\xi : [0, T] \times D \to \mathbb{R}$ is a stochastic forcing, $u_0 : D \to \mathbb{R}$ is the initial condition,

$\mu, \sigma : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}$ are two functions. Under local Lipschitz conditions on μ, σ with respect to suitable norm, this SPDE has a unique mild solution (Hairer 2014b; Salvi and Lemercier 2021):

$$\begin{aligned} u_t &= e^{t\mathcal{L}}u_0 + \int_0^t e^{(t-s)\mathcal{L}}\mu(u_s, \partial_1 u_s, \cdots, \partial_d u_s)\,\mathrm{d}s \\ &+ \int_0^t e^{(t-s)\mathcal{L}}\sigma(u_s, \partial_1 u_s, \cdots, \partial_d u_s)\xi\,\mathrm{d}s\,\mathrm{d}y, \end{aligned} \quad (2)$$

where $u_t(\cdot) := u(t, \cdot), t \in [0, T]$. The space-time white noise ξ is in space of distributions with negative regularity, i.e., $C^{-\frac{(d+2)}{2}-\epsilon}, \epsilon > 0$ with regularity $\frac{-(d+2)}{2}-$. Taking $\mathcal{L} := \Delta = \sum_{i=1}^d \partial_i^2$ as an example, the following theorem tells the regularity after operating $e^{t\Delta}$.[1]

Theorem 1 (Lemma A7 in (Gubinelli, Imkeller, and Perkowski 2015))**.** *Let $C^\alpha(D)$ be the Hölder space with $\alpha \in \mathbb{R}$ and $u_t \in C^\alpha(D)$. For every $\delta \geq 0$, there exists a constant B independent of u_t such that:*

$$\|e^{t\Delta}u_t\|_{C^{\alpha+\delta}} \leq B \cdot t^{-\frac{\delta}{2}}\|u_t\|_{C^\alpha}.$$

Model Feature Vectors

The concept *model* in the regularity structures theory (Hairer 2014b) is a collection of *model feature vectors*, which are multi-dimensional signals designed to approximate the mild solutions of SPDEs even with low regularity regimes. The motivation comes from the Picard theorem and Taylor expansion. According to the representation of the mild solution in Eqn.(2), we define two linear operators $I[f]_t = \int_0^t e^{(t-s)\mathcal{L}}f_s\,\mathrm{d}s$ and $I_c[g]_t = e^{t\mathcal{L}}g$ for any function $f : [0, T] \times D \to \mathbb{R}$ and $g : D \to \mathbb{R}$. Picard theorem shows that the following recursive sequence approximates the solution u of equation (1) as $n \to \infty$

$$\begin{aligned} u_t^0 &= I_c[u_0]_t, \\ u_t^{n+1} &= I_c[u_0]_t + I[\mu(u^n) + \sigma(u^n\xi)]_t. \end{aligned} \quad (3)$$

Using Taylor expansion on μ and σ, we then have the recursive sequence that can approximate u_t as $m, l, n \to \infty$[2]

$$\begin{aligned} u_t^{0,m,l} =& I_c[u_0]_t, \\ u_t^{n+1,m,l} =& I_c[u_0]_t + \sum_{k=0}^m \frac{\mu^{(k)}(0)}{k!} I[(u^{n,m,l})^k]_t \\ &+ \sum_{k=0}^l \frac{\sigma^{(k)}(0)}{k!} I[(u^{n,m,l})^k\xi]_t. \end{aligned} \quad (4)$$

Then, the solution of SPDE can be approximated by the weighted sum of the features $I[(u^{n,m,l})^{k_1}]$, $I[(u^{n,m,l})^{k_2}\xi], k_1 = 0, \cdots, m; k_2 = 0, \cdots, l$, where we call n as the height, and m, l as the width in the approximation. Motivated by this, (Chevyrev, Gerasimovics, and Weber 2021) develops a tool for feature engineering of SPDEs. By the regularity structure theory, the model feature vectors are obtained by integrals of functionals of u_0 and ξ (as I and I_c are convolution operations). The regularity is improved according to Theorem 1, i.e.,

$$\|I[u]_t\|_{C^{\alpha+2}} \leq B \cdot \sup_{t\in[0,T]} \|u_t\|_{C^\alpha}. \quad (5)$$

Generation of the Model Feature Vectors

We briefly review the generation process of the model feature vectors (Chevyrev, Gerasimovics, and Weber 2021). Consider the tuple of non-negative integers $\alpha = (m, l) \in \mathbb{N}^2$ and $n \in \mathbb{N}$. Given an initial feature $s : [0, T] \times D \to \mathbb{R}$, and forcing ξ, the feature set is recursively generated as

$$\begin{aligned} \mathcal{S}_\alpha^0(s, \xi) =& \{s\}; \\ \mathcal{S}_\alpha^n(s, \xi) =& \{I[\xi^j \prod_{i=1}^k \partial^{a_i} f_i] : f_i \in \mathcal{S}_\alpha^{n-1}, a_i, j \in \{0, 1\}, \\ & k \in \mathbb{N}, 1 \leq k + j \leq mI_{j=0} + \ell I_{j>0}\} \cup \mathcal{S}_\alpha^{n-1}, \end{aligned} \quad (6)$$

where n denotes the height of the features, (m, l) denotes the width of the features, which corresponds to the n, m, l in Eqn.(4). Taking $s = I_c[u_0]$, $\mathcal{S}_\alpha^n(I_c[u_0], \xi)$ gives the feature set of Eqn.(1).

Next, we discuss the universality and complexity of the model feature vectors.

Proposition 1. *For the given SPDE in Eqn.(1) and $f_i \in \mathcal{S}_\alpha^n$, as $n, m, l \to \infty$, there exists coefficients c_i to make $\sum_{f_i \in \mathcal{S}_\alpha^n} c_i f_i$ tend to the mild solution of the SPDE.*

Proof: We only need to prove every term in the summation on the right side of Eqn.(4) is contained in $\mathcal{S}_\alpha^n$, which can be proved by hypothesis testing.

Complexity. As the number of model feature vectors grows exponentially as n increases, the computational complexity also grows exponentially. Therefore, it requires prior knowledge to determine the order n to achieve less information loss and computational cost. It motivates us to design DLR-Net to better tradeoff the approximation accuracy and efficiency.

Regularity Feature Transformation as a Layer in Neural Network

In this section, we introduce the structure of the regularity feature block (RF block). We combine model feature vectors and neural networks to obtain better expressiveness in learning SPDEs. The regularity feature block also benefits from the transformation $I[\cdot]$ in the generation of model features, since the regularity of the input function can be improved according to Eqn.(5).

For computing $I[\cdot]$ and $I_c[\cdot]$ emerging in above model features, equivalently to $I[f]_t = \int_0^t e^{(t-s)\mathcal{L}}f_s\,\mathrm{d}s$ and $I_c[g]_t = e^{t\mathcal{L}}g$, the operations $I[f]$ and $I_c[g]$ satisfy the following

[1] For general linear operator $\mathcal{L}$, similar results can be established with different scaling of t, i.e., $t^{-\beta\delta}$. (Gubinelli, Imkeller, and Perkowski 2015)

[2] Here, we assume $\mu(u)$ and $\sigma(u)$ does not depend on ∂u for simplifying the description. The following constructed features are applicable to μ, σ that depend on ∂u.

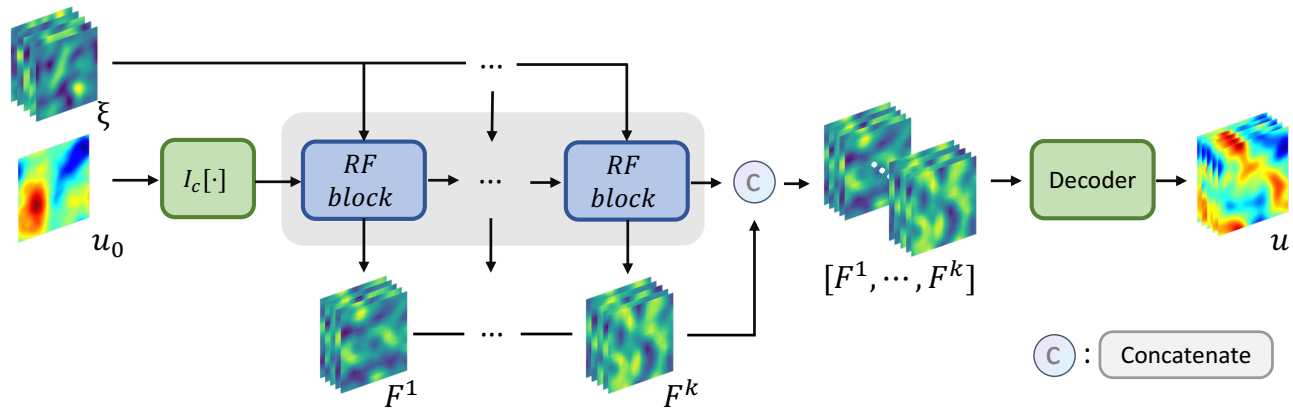

Figure 1: Architecture of the Deep Latent Regularity Network. DLR-Net takes an initial condition u_0 and a stochastic forcing ξ as inputs. The input undergoes a series of transformations through a stack of Regularity Feature (RF) blocks, and the resulting output features $F^1, \cdots, F^k$ are concatenated. Subsequently, DLR-Net decodes the feature vectors to output the prediction of the mild solution u.

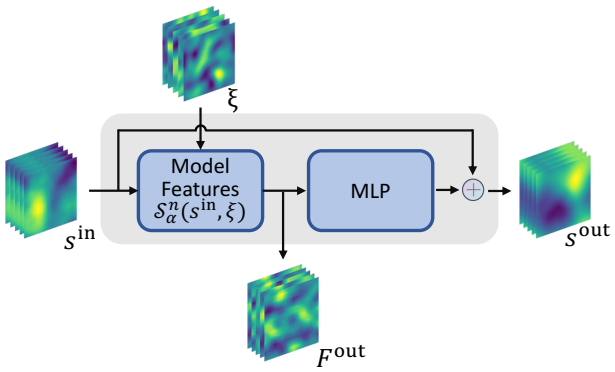

Figure 2: Architecture of the regularity feature block. RF block comprises an initial feature s^{in}, which can either be $I_c[u_0]$ or the output $\hat{s}^{\text{out}}$ of another RF block, and a stochastic forcing term ξ as inputs. The block then generates the model features F^{out} and the latent variable s^{out} using a MLP.

equations:

$$(\partial_t - \mathcal{L})I[f] = f \quad \text{and} \quad I[f]_0 = 0, \tag{7}$$
$$(\partial_t - \mathcal{L})I_c[g] = 0 \quad \text{and} \quad I_c[g]_0 = g. \tag{8}$$

By discretizing above equations, the values of $I[\cdot]$ and $I_c[\cdot]$ can be obtained in an iterative manner.

Discretization. To numerically calculate the value of $I[f]$, we discretize the space-time domain $[0, T] \times D$ with $D \subset \mathbb{R}^d$ onto the grid $\mathcal{G} = O_T \times O_{X_1} \times \cdots \times O_{X_d}$, with $0 = t_0 < t_1 < \cdots < t_k < \cdots < t_K = T$ and $t_{k+1} - t_k = \delta_t$. Now we consider to evaluate the functions and operators on this grid. The operator $\mathcal{L}$ can be approximated by a finite-dimensional tensor using finite-difference. (We put detailed examples in Appendix.) Given the grid, the $I[f]$ is approximated by numerical method. Here, we use the implicit Euler scheme, i.e.,

$$(I[f]_{t_{k+1}} - I[f]_{t_k}) - \delta_t \cdot \mathcal{L}I[f]_{t_{k+1}} = u_{t_{k+1}} \cdot \delta_t. \tag{9}$$

By rearranging the formula, we have

$$I[f]_{t_{k+1}} = (I[f]_{t_k} + u_{t_{k+1}} \cdot \delta_t) \cdot M, \tag{10}$$

where the tensor M is to approximate $(Id - \mathcal{L} \cdot \delta_t)^{-1}$, where Id denote the identity function, i.e., $Id[f] = f, \forall f$. In the following context, when we mention $I[\cdot]$, we refer its discretized version as above.

Using the discretized spatial grids $\mathcal{G}$, we discretize the operator $\mathcal{L}$ as a finite-dimensional tensor with the same size as the spatial grids. For example, if $\mathcal{L}$ equals the 1d Laplace operator, i.e., $\mathcal{L} = \Delta$, the discretized operator using finite difference is

$$\Delta^{\text{dis}} = \frac{1}{\epsilon^2} \begin{pmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & 1 & -2 & 1 \\ 0 & \cdots & 0 & 1 & -2 \end{pmatrix}, \tag{11}$$

where ϵ is the discretized step. Then we calculate kernel $M = (Id - \Delta^{\text{dis}} \cdot \delta_t)^{-1}$. Given the temporal grids, the time integral in Eqn.(7) and Eqn.(8) can be equivalently represented as

$$I[f]_{t_{k+1}} = (f_{t_1} \cdot M^k + f_{t_2} \cdot M^{k-1} + \cdots + f_{t_k} \cdot M) \cdot \delta_t \tag{12}$$

$$I_c[g]_{t_{k+1}} = g \cdot M^{k+1} \tag{13}$$

Remark: *To make the pre-processed matrix M applicable to different resolution, we upsample the training data by linear interpolation to fit the size of M.*

With the pre-calculated M, we introduce the computing process of the model feature vectors in Eqn.(6).

Algorithm 1: Generation of *Model Feature Vectors*

Input: Initial feature s, forcing ξ
Parameter: Height n, $\alpha = (m, l)$, discretized operator $\mathcal{L}^{\text{dis}}$
Output: Feature set $\mathcal{S}_\alpha^n(s, \xi)$

Calculate $M = (Id - \mathcal{L}^{\text{dis}} \cdot \delta t)^{-1}$
Initial function set $\mathcal{S}_\alpha^0 = \{s\}$
for $p = 1, \cdots, n$ **do**
  Generate set $\mathcal{Z}_\alpha^p = \{\xi^j \prod_{i=1}^k \partial^{a_i} f_i : f_i \in \mathcal{S}_\alpha^{n-1}, a_i, j \in \{0, 1\}, k \in \mathbb{N}, 1 \le k + j \le m I_{j=0} + \ell I_{j>0}\}$
  for $k = 0, 1, \cdots, K-1$ **do**
    For $f \in \mathcal{Z}_\alpha^p$, $I[f]_{t_{k+1}} = (I[f]_{t_k} + f_{t_k} \cdot \delta_t) \cdot M$
  end for
  $\mathcal{S}_\alpha^p = \{I[f], f \in \mathcal{Z}_\alpha^p\} \cup \mathcal{S}_\alpha^{p-1}$
end for

Computing of Model Feature Vectors. Here we present the algorithm to compute model feature vectors iteratively in Algorithm 1.

Next, we introduce the regularity feature block as shown in Figure 2.

Structure of the Regularity Feature Block. Given a feature set $\mathcal{S}_\alpha^n$, RF block maps the initial feature s^{in} and the stochastic forcing ξ to model features F^{out} and a latent variable s^{out}. Note that all of them are defined on the grid $\mathcal{G}$. Leveraging the property of $\mathcal{S}_\alpha^n$, the regularity feature block contains two component layers: the first layer is the regularity feature transformation on (s^{in}, ξ); the second layer is a multi-layer perceptron (MLP), which includes linear transformation plus non-linear activation functions, i.e.,

$$F^{\text{out}} = \mathcal{S}_\alpha^n(s^{\text{in}}, \xi), \tag{14}$$

$$s^{\text{out}} = V s^{\text{in}} + \text{MLP}_W\left(F^{\text{out}}\right), \tag{15}$$

where $s^{\text{in}}, s^{\text{out}} \in \mathbb{R}^{\mathcal{G}}$, $F^{\text{out}} \in \mathbb{R}^{N_F \times \mathcal{G}}$, $N_F = |\mathcal{S}_\alpha^n(\cdot, \cdot)|$ is fixed given n, α, and $V \in \mathbb{R}^{\mathcal{G} \times \mathcal{G}}$ is the learnable weights. Here, $\text{MLP}_W(F^{\text{out}})$ is a multi-layer perceptron network with learnable weights W and non-linear activation functions.

Deep Latent Regularity Network

We move on to introducing the architecture of the Deep Latent Regularity Network as shown in Figure 1. For given SPDEs which have the form in Eqn.(1), our goal is to learn the mapping from (u_0, ξ) to u where u_0 is assumed to be generated by a parametric distribution. We assume the SPDE is partially observed, i.e., the linear part $\partial_t - \mathcal{L}$ is known.

Encoder $I_c[\cdot]$. Let X_i, K be the number of grids on the i-th spatial dimension and temporal dimension, respectively. For each sample $u_0 \in \mathbb{R}^{X_1 \times \cdots \times X_d}$ with dimension equal to the spatial grids, the input u_0 is encoded by the $I_c[\cdot] : \mathbb{R}^{X_1 \times \cdots \times X_d} \to \mathbb{R}^{X_1 \times \cdots \times X_d \times K}$ operator, i.e.,

$$I_c[u_0] = u_0 \cdot \mathcal{M}_c, \tag{16}$$

where $\mathcal{M}_c = (Id, M, M^2, \cdots, M^K)$.

Connected Regularity Feature Blocks. A sequence of connected RF blocks is used to extract the model features of the target SPDEs. Given k blocks in total, we use $s^i, F^i, i = 1, \cdots, k$ to denote the outputs of the ith RF block. $(I_c[u_0], \xi)$ is fed into the first RF block, while (s^i, ξ) is used as the input of $(i+1)$th block. Next, we concatenate the model features as $[F^1, F^2, \cdots, F^k]$ to be the input of the following decoder.

Decoder. The features output by the regularity feature blocks can be combined with several types of the decoder that can deal with the temporal series. Considering the efficiency, we select the Fourier layer in neural Fourier operator (Li et al. 2020b) as the decoder. Each feature is regarded as a channel and the neural Fourier operator projects the features to the solution u. For details about the Fourier layer, readers can refer to Appendix and (Li et al. 2020b).

Loss Function. The DLR-Net is trained in a supervised way. Given m samples $\{(u_0^i, \xi^i; u^i), i = 1, \cdots, m\}$, we optimize the loss function

$$L(\theta) = \sum_{i=1}^m \|u^i - F_\theta^{\text{DLR}}(u_0^i, \xi^i)\|_2^2, \tag{17}$$

where F_θ^{DLR} denotes the DLR-Net with learnable weights θ. Optimizers such as Adam can be used to minimize the loss.

Experiments

In this section, we evaluate the performance of DLR-Net on three SPDEs including the dynamic Φ_1^4 model, reaction-diffusion equation with linear multiplicative forcing, and 2D Navier-Stokes equation with additive noise. We report the accuracy of two supervised operator learning settings: $\xi \to u$ with observed forcing and fixed initial condition; $(u_0, \xi) \to u$ with observed forcing ξ and different u_0 sampled from a distribution. In addition, the number of parameters and the inference time per sample are measured as well.

Dynamic Φ_1^4 Model

We first consider the dynamic Φ_1^4 model with the periodic boundary condition as in (Chevyrev, Gerasimovics, and Weber 2021). It takes the form

$$\begin{aligned} \partial_t u - \Delta u &= 3u - u^3 + \xi, \quad (t, x) \in [0, T] \times D \\ u(t, 0) &= u(t, 1), \quad \text{(Periodic BC)} \\ u_0(x) &= u(0, x) = x(1 - x) + \kappa \eta(x), \end{aligned} \tag{18}$$

where ξ is the space-time white noise. Following the settings in (Salvi and Lemercier 2021; Chevyrev, Gerasimovics, and Weber 2021), we choose $T = 0.05$, $D = [0, 1]$, $\eta(x) = \sum_{k=-10}^{k=10} \frac{a_k}{1+|k|^2} sin(\lambda^{-1} k \pi (x - 0.5))$, with $a_k \sim \mathcal{N}(0, 1)$, $\lambda = 2$, and $\kappa = 0$ or 0.1 corresponding to the initial condition is fixed or not. The time and space domain is evenly divided into 50 and 128 points to make the grid $O_T \times O_X$. Space-time white noise ξ and reference solution u on this grid are given by numerical simulator in (Chevyrev, Gerasimovics, and Weber 2021) on this grid.

Model	#Para	Inference time (ms)	$N = 1000$		$N = 10000$	
			$\xi \mapsto u$	$(u_0, \xi) \mapsto u$	$\xi \mapsto u$	$(u_0, \xi) \mapsto u$
Numerical Solver	$\times$	4.088	$\times$	$\times$	$\times$	$\times$
NCDE	272672	0.851	0.112	0.127	0.056	0.072
NRDE	2164356	0.127	0.129	0.150	0.070	0.083
NCDE-FNO	48769	2.079	0.071	0.066	0.066	0.069
DeepONet	159600	0.062	0.126	$\times$	0.061	$\times$
FNO	6301761	0.235	0.032	0.030	0.027	0.024
NSPDE	265089	0.277	0.009	0.012	0.006	0.006
DLR-Net	133178	0.182	**0.0009**$_{\pm 0.0001}$	**0.0027**$_{\pm 0.0004}$	**0.0004**$_{\pm 0.0001}$	**0.0008**$_{\pm 0.0001}$

Table 1: Dynamic Φ_1^4 model. We show the $l2$ errors of the baselines and our model in two settings with training data size $N = 1000$ or 10000. Baseline $l2$ errors are based on (Salvi and Lemercier 2021). Symbol $\times$ means not applicable. Results for DLR-Net are averaged on 3 runs.

For this equation, we use two RF blocks with height $n = 2$ and $\alpha = (3, 1)$ in the feature sets. In the decoder layer, we use 4-layer 2d-FNO with $s_1 = 16, s_2 = 16$ and $width = 8$.

The result is shown in Table 1. We consider two settings, in both of which our architecture outperforms other benchmarks a lot with a small network and shorter inference time.

Reaction-Diffusion Equation with Linear Multiplicative Forcing

As the dynamic Φ_1^4 model is a parabolic equation with additive forcing, we then consider a parabolic equation with multiplicative forcing as in (Chevyrev, Gerasimovics, and Weber 2021), which is given by

$$\begin{aligned} \partial_t u - \Delta u &= 3u - u^3 + \sigma u \xi, \quad (t, x) \in [0, T] \times D \\ u(t, 0) &= u(t, 1), \quad \text{(Periodic BC)} \\ u_0(x) &= u(0, x) = x(1 - x) + \kappa \eta(x), \end{aligned} \tag{19}$$

where ξ is the space-time white noise scaled by $\sigma = 0.1$. T, D, $\eta(x)$, grid and numerical simulator are chosen to be the same as in the Φ_1^4 model.

For this equation, we use two RF blocks with height $n = 2$ and $\alpha = (3, 2)$ in their feature sets. As for the decoder, we use 4-layer 2d-FNO with $s_1 = 16, s_2 = 16$, and $width = 8$.

As the parameter numbers and inference time is almost the same with the dynamic Φ_1^4 model, we list the $l2$ errors in Table 2. The results show that our model achieves the lowest error in both of the two settings. The experiments on the two equations clearly show the effectiveness of DLR-Net.

Stochastic 2D Navier-Stokes Equation

In this section, we aim to evaluate our model on the 2D Navier-Stokes equation for a viscous, incompressible fluid in vorticity form

$$\begin{aligned} \partial_t w - \nu \Delta w &= -u \cdot \nabla w + f + \sigma \xi, \ (t, x) \in [0, T] \times D \\ \omega(0, x) &= \omega_0(x) \end{aligned} \tag{20}$$

where u is the velocity field, $\omega = \nabla \times u$ is the vorticity, ω_0 is the initial vorticity, $f = 0.1(\sin(2\pi(x_1 + x_2)) + \cos(2\pi(x_1 + x_2)))$ is the deterministic forcing as in (Li et al. 2020b), ξ is the random forcing rescaled by $\sigma = 0.05$ as in (Salvi and Lemercier 2021), $T = 1$, $D = [0, 1]^2$, and the viscosity coefficient $\nu = 10^{-4}$. We generate the reference solution as (Salvi and Lemercier 2021) do. Specifically, the time domain is evenly divided into 1000 points, and the 2D space domain is evenly divided into a 64×64 grid. Numerical simulation is then performed on this grid to generate 1000 reference solution samples, which are further downsampled on 100 evenly divided time points and 16×16 evenly spaced spatial grid to be our training dataset. More details can be found in Appendix.

Our target is to model the vorticity ω, which is harder to learn compared with the velocity u. For this equation, we use two RF blocks with height $n = 2$ and $\alpha = (2, 1)$ in their feature sets. In the decoder layer, we use 4-layer 3d-FNO with $s_1 = 8, s_2 = 8, s_3 = 8$ and $width = 8$.

We first train and evaluate our model on 16×16 spatial grid. As shown in Table 3, the error of our model is the lowest in both of the settings. Besides, we transfer our model trained on 16×16 grid to evaluate on 64×64 grid. Figure 3 shows the numerical simulation references and the predictions of our model. Although our model is trained on a coarse grid, it works well on a higher-resolution grid.

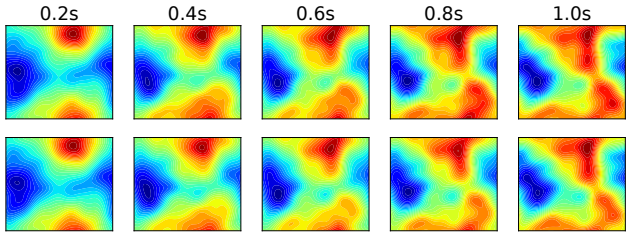

Figure 3: 2D stochastic Navier-Stokes equation. We train the model on 16×16 grid and evaluate on 64×64 grid. (Top): Numerical simulation references. (Bottom): Predictions of DLR-Net.

Ablation Studies

In this section, we conduct ablation studies on feature vector complexity by controlling the height n and α in Eqn. (6) of the feature vector generation process. We report the model inference time and $l2$ error on $\xi \mapsto w$ in Table 4 and Ta-

Model	$N = 1000$		$N = 10000$	
	$\xi \mapsto u$	$(u_0, \xi) \mapsto u$	$\xi \mapsto u$	$(u_0, \xi) \mapsto u$
NCDE	0.016	0.087	0.010	0.059
NRDE	0.023	0.584	0.023	0.641
NCDE-FNO	0.015	0.034	0.017	0.019
DeepONet	0.023	$\times$	0.023	$\times$
FNO	0.0036	0.0063	0.0035	0.0037
NSPDE	0.0016	0.0062	0.0012	0.0026
DLR-Net	**0.0006**$_{\pm 0.0001}$	**0.0028**$_{\pm 0.0005}$	**0.0002**$_{\pm 0.0001}$	**0.0006**$_{\pm 0.0002}$

Table 2: Reaction-Diffusion equation. We compare the $l2$ errors of the baselines and our model with training data size $N = 1000$ or 10000. Symbol $\times$ means not applicable. Results for DLR-Net are averaged on 3 runs.

Model	Inference time (ms)	$\xi \mapsto \omega$	$(\omega_0, \xi) \mapsto \omega$
Numerical Solver	18.18	$\times$	$\times$
NCDE	0.683	0.513	0.723
NRDE	-	-	-
NCDE-FNO	0.269	0.571	0.615
DeepONet	0.170	0.492	$\times$
FNO	0.657	0.084	0.073
NSPDE	0.449	0.052	0.067
DLR-Net	0.896	**0.024**$_{\pm 0.0001}$	**0.020**$_{\pm 0.001}$

Table 3: Stochastic 2D Navier-Stokes equation. We compare the $l2$ errors of the baselines and our model in two settings with 1000 training samples. We train and test these models on downsampled 16×16 grid. Symbol $\times$ means not applicable, and - means memory limit exceeded. Results for DLR-Net are averaged on 3 runs.

$\boldsymbol{n}$	Inference time (ms)	$\xi \mapsto w$
1	0.125	0.0015
2	0.182	0.0009
3	0.394	0.0011

Table 4: Ablation study over the height n of the feature set $\mathcal{S}^n_\alpha$ on Dynamic Φ^4_1 model with training data size $N = 1000$. $\alpha = (m, l)$ is fixed to be $(3, 1)$.

$\boldsymbol{m}$	$\boldsymbol{l}$	Inference time (ms)	$\xi \mapsto w$
1	1	0.141	0.0014
2	1	0.161	0.0012
3	1	0.182	0.0009
3	2	0.192	0.0014
3	3	0.213	0.0009

Table 5: Ablation study over $\alpha = (m, l)$ of the feature set $\mathcal{S}^n_\alpha$ on dynamic Φ^4_1 model with training data size $N = 1000$. n is fixed to be 2.

ble 5. The results demonstrate the robustness of DLR-Net to different values of n and α.

Discussion

DLR-Net provides an efficient framework for incorporating regularity features into deep neural operators. While the motivation for DLR-Net comes from the roughness of SPDEs, the feature calculation can also be applied to PDEs by setting ξ to 1 for all t and x in $\mathcal{S}^n_\alpha$. The benefits of using DLR-Net for PDEs will be investigated in future work. Another extension is to incorporate different discretization schemes, such as finite volume methods, as sub-modules of the DLR-Net framework. This will allow for more flexibility in modeling and simulating a wide range of real-world applications. Additionally, future work will explore the combination of regularity feature transformation with autoregressive models, such as LSTM, to efficiently simulate SPDEs over long time periods. A potential limitation of the DLR-Net is its reliance on a pre-calculated discretized operator M, which is dependent on the resolution of the data. To address this limitation, a possible solution under varying settings is to approximate M using a model, such as a neural network, and pre-train the model for different operators to ensure generalizability. We may explore it in future research.

Conclusion

In this work, we introduce DLR-Net as a strong SPDE-solving tool. By incorporating the Regularity Feature block, DLR-Net combines the strengths of Neural Operators and regularity structures, thus addressing their limitations. Experiments demonstrate that DLR-Net can not only learn the solution operators $(u_0, \xi) \mapsto u$ of SPDEs but also achieve significantly lower error than other deep learning models. In future work, DLR-Net will be extended to include different discretization schemes and will be applied to solve a wider range of PDEs or SPDEs relevant to real-world applications.

Acknowledgements

Hao Ni is supported by the EPSRC under the program grant EP/S026347/1 and the Alan Turing Institute under the EPSRC grant EP/N510129/1.

References

Barone-Adesi, G.; and Whaley, R. E. 1987. Efficient Analytic Approximation of American Option Values. *Journal of Finance*, 42(2): 301–320.

Bhattacharya, K.; Hosseini, B.; Kovachki, N. B.; and Stuart, A. M. 2020. Model reduction and neural networks for parametric pdes. *arXiv preprint arXiv:2005.03180*.

Buckmaster, T.; and Vicol, V. 2019. Convex integration and phenomenologies in turbulence.

Chen, T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. 2018. Neural Ordinary Differential Equations. *CoRR*, abs/1806.07366.

Chevyrev, I.; Gerasimovics, A.; and Weber, H. 2021. Feature engineering with regularity structures. *arXiv preprint arXiv:2108.05879*.

Chevyrev, I.; and Kormilitzin, A. 2016. A primer on the signature method in machine learning. *arXiv preprint arXiv:1603.03788*.

Gubinelli, M.; Imkeller, P.; and Perkowski, N. 2015. Paracontrolled distributions and singular PDEs. In *Forum of Mathematics, Pi*, volume 3. Cambridge University Press.

Hairer, M. 2014a. Regularity structures and the dynamical Φ^4_3 model. *Current Developments in Mathematics*, 2014(1): 1–49.

Hairer, M. 2014b. A theory of regularity structures. *Invent. Math*, 198(2): 269–504.

Hasselmann, K. 1976. Stochastic climate models Part I. Theory. *Tellus*, 28(6): 473–485.

Hu, P.; Meng, Q.; Chen, B.; Gong, S.; Wang, Y.; Chen, W.; Zhu, R.; Ma, Z.-M.; and Liu, T.-Y. 2022. Neural Operator with Regularity Structure for Modeling Dynamics Driven by SPDEs. *arXiv preprint arXiv:2204.06255*.

Jia, J.; and Benson, A. R. 2019. Neural jump stochastic differential equations. *Advances in Neural Information Processing Systems*, 32.

Kidger, P.; Morrill, J.; Foster, J.; and Lyons, T. J. 2020. Neural Controlled Differential Equations for Irregular Time Series. *CoRR*, abs/2005.08926.

Kovachki, N. B.; Li, Z.; Liu, B.; Azizzadenesheli, K.; Bhattacharya, K.; Stuart, A. M.; and Anandkumar, A. 2021. Neural Operator: Learning Maps Between Function Spaces. *CoRR*, abs/2108.08481.

Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Stuart, A.; Bhattacharya, K.; and Anandkumar, A. 2020a. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33: 6755–6766.

Li, Z.; Kovachki, N. B.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A. M.; and Anandkumar, A. 2020b. Fourier Neural Operator for Parametric Partial Differential Equations. *CoRR*, abs/2010.08895.

Li, Z.; Kovachki, N. B.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A. M.; and Anandkumar, A. 2020c. Neural Operator: Graph Kernel Network for Partial Differential Equations. *CoRR*, abs/2003.03485.

Lu, L.; Jin, P.; and Karniadakis, G. E. 2019. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*.

Morrill, J.; Salvi, C.; Kidger, P.; and Foster, J. 2021. Neural rough differential equations for long time series. In *International Conference on Machine Learning*, 7829–7838. PMLR.

Nelsen, N. H.; and Stuart, A. M. 2021. The random feature model for input-output maps between banach spaces. *SIAM Journal on Scientific Computing*, 43(5): A3212–A3243.

Patel, R. G.; Trask, N. A.; Wood, M. A.; and Cyr, E. C. 2021. A physics-informed operator regression framework for extracting data-driven continuum models. *Computer Methods in Applied Mechanics and Engineering*, 373: 113500.

Salvi, C.; and Lemercier, M. 2021. Neural Stochastic Partial Differential Equations. *CoRR*, abs/2110.10249.

Uhlenbeck, G. E.; and Ornstein, L. S. 1930. On the Theory of the Brownian Motion. *Phys. Rev.*, 36: 823–841.

Wilkinson, D. 2018. *Stochastic Modelling for Systems Biology, Third Edition*, volume 1. Taylor & Francis Ltd.