A New Class of Polynomial Activation Functions of Deep Learning for Precipitation Forecasting

Jiachuan Wang
Hong Kong University of Science and Technology
Hong Kong, China
jwangey@cse.ust.hk

Lei Chen
Hong Kong University of Science and Technology
Hong Kong, China
leichen@cse.ust.hk

Charles Wang Wai Ng
Hong Kong University of Science and Technology
Hong Kong, China
cecwwng@ust.hk

ABSTRACT

Precipitation forecasting, modeled as an important chaotic system in earth system science, is not explicitly solved with theory-driven models. In recent years, deep learning models have achieved great success in various applications including rainfall prediction. However, these models work in an image processing manner regardless of the nature of a physical system. We found that the non-linearity relationships learned by deep learning models, which mostly rely on the activation functions, are commonly weighted piecewise continuous functions with bounded first-order derivatives. In contrast, the polynomial is one of the most widely used classes of functions for theory-driven models, applied to numerical approximation, dynamic system modeling, *etc.*. Researchers started to use the polynomial activation functions (*Pac*s in short) for neural networks from the 1990s. In recent years, with bloomed researches that apply deep learning to scientific problems, it is weird that such a powerful class of basis functions is rarely used. In this paper, we investigate it and argue that, even though polynomials are good at information extraction, it is too fragile to train stably. We finally solve its serious data flow explosion problem with Chebyshev polynomials and prepended normalization, which enables networks to go deep with *Pac*s. To enhance the robustness of training, a normalization called Range Norm is further proposed. Performance on synthetic dataset and summer precipitation prediction task validates the necessity of such a class of activation functions to simulate complex physical mechanisms. The new tool for deep learning enlightens a new way of automatic theoretical physics analysis.

CCS CONCEPTS

• **Applied computing** → **Physical sciences and engineering**;
• **Computing methodologies** → **Artificial intelligence**.

KEYWORDS

Neural networks; polynomials; precipitation; heterogeneous data.

ACM Reference Format:
Jiachuan Wang, Lei Chen, and Charles Wang Wai Ng. 2022. A New Class of Polynomial Activation Functions of Deep Learning for Precipitation Forecasting. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3488560.3498448

WSDM '22, February 21–25, 2022, Tempe, AZ, USA

ACM ISBN 978-1-4503-9132-0/22/02…$15.00
https://doi.org/10.1145/3488560.3498448

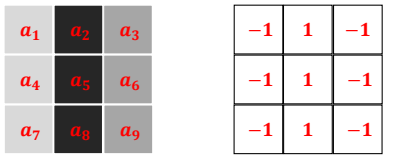
1 INTRODUCTION

Deep learning, a large scale machine learning architecture, shows powerful ability in real-world applications, such as image processing [14, 33, 51], natural language processing [29, 49, 52], and recommendation systems [11, 55, 58].

Precipitation, especially the heavy summer rainfall, has great importance for the prevention of natural disasters and the production of agriculture and industry [50]. As a famous chaotic system, the theories of rainfall are still not explicitly resolved such as convection [57]. In the widely used WRF (Weather Research and Forecasting) model[47], rainfall is divided into different cases according to season and location and solved with empirical model and parameter setting [16, 19, 20]. In recent years, deep learning models are applied to precipitation prediction problems based on radar maps, which is treated as an alternative indicator [45, 46, 53, 54]. However, these models process radar images similar to video prediction task in a computer vision (CV) mode, which do not engage other weather data and omit the underlying physical relationships.

To simulate complex systems such as precipitation, deep learning models combine linear cells such as fully-connected layer and convolutional layer with non-linear operators, mainly provided by the class of module called activation functions. Researchers have paid great efforts to analyze and design different activation functions. We can divide the widely used activation functions into two major clusters, (1) the gate-like functions. These functions map the input into limited ranges (e.g. $[0, 1]$ and $[-1, 1]$), such as sigmoid function, tanh function, and softsign function. (2) Nonsaturating activation functions, which do not approximate to a constant when the input goes larger, such as Relu, leaky relu, elu, and selu functions [9, 31, 39]. Most of them are asymptotically linear with bounded first-order derivatives. For many fields such as image processing and language processing, using only asymptotically linear functions is reasonable, as vectors for pixels and words are of dimensional homogeneity. It makes sense to model the effects of input features on the output as a weighted sum with functions like relu and leaky relu. However, the inputs of physical modeling are heterogeneous. Their relationships can be much more complex, such as multiply and exponent. Based on these facts, we propose an example to illustrate our motivation.

Example 1.1. Convolutional Neural Networks (CNN) is one of the most widely used layers of deep learning models, which outputs



(a) Pixels extracted for a 3 × 3 kernel **(b) A kernel to detect vertical edge**

Figure 1: An Illustrative Example

the weighted sum of feature dimensions in each kernel. Figure.1 illustrates an application of CNN in an image processing task. We display a CNN kernel that learns to extract vertical edge as a local feature, which could be further used in detecting high-level features such as hair and outline. The sample with pixels in Figure.1a for a CNN kernel with shape= 3×3 is an input with a vertical edge. The kernel weight in Figure.1b will detect this property by giving a large output. The weighted sum of input pixels a_1 to a_9 effectively detects the vertical edge; however, it hardly provides an intuitive clue for edge detection by adding a feature of multiplication (e.g. $a_1 \times a_3$) and exponent (e.g. $a_2^{a_6}$) between inputs.

In contrast, for physical problems including precipitation forecasting, there are various widely accepted nonlinear physical relationships. For example, to calculate the pressure P given the amount of substance n and absolute temperature t according to ideal gas law, which is used in WRF[47] model for rainfall:

$$P = \frac{nRt}{V},$$

we need multiplication for inputs n and t other than linear or sublinear functions. A quadratic polynomial module can perfectly do the task by learning parameters $W = \begin{bmatrix} 0 & 0 & 0 & \frac{R}{V} & 0 & 0 \end{bmatrix}$, so that

$$P = W \begin{bmatrix} 1 & n & t & nt & n^2 & t^2 \end{bmatrix}^{\mathrm{T}} = \frac{nRt}{V}$$

This fact raises an interesting problem: **are we misled by the activation functions in other fields while we model the complex non-linear functionality in the physical system of precipitation, which could be a detour or even a dead-end towards the target?**

As polynomials are served as one of the most important basis functions in science and engineering for dozens of years, such as approximation [43], simulation of dynamic system [28], *etc.*, we focus on a class of activation functions, named as **P**olynomial-**ac**tivation functions (*Pac*s in short), aiming to powerfully capture the complex non-linearity of physical models. In contrast with the popularity of deep learning with multiple layers, we found that *Pac*s are studied from 1990s but rarely used nowadays and mainly applied to networks with single hidden layer [34, 37, 38]. We conducted experiments with *Pac*s and argue that, deep network is hard to train stably with them. Compared with saturating functions, a polynomial approximator is nonsaturating and without bounded first- nor even second-order derivatives, which grows more aggressively with larger inputs. This property makes *Pac*s greatly vulnerable to data flow explosion problems. Tons of experiments are conducted directly using polynomial activation functions or append normalization after *Pac*s, which results in exploded loss during training. In this paper, the issue is handled in two phases. We first design *Pac*s based on the Chebyshev polynomials of the first kind, which are the "extremal" polynomials for various properties in approximation theory [44]. With input in the domain $[-1, 1]$, Chebyshev polynomials output features in the same range. On the other hand, to feed such a restricted input to *Pac*s, we prepend other than append normalization to *Pac*s so that the input range is narrowed down for stable data propagation. A new normalization called Range Norm is proposed to further enhance the robustness.

To validate the effectiveness of our *Pac*s and Range Norm to approximate nonlinear functions such as polynomials, experiments on a synthetic dataset are conducted. The result shows that *Pac*s outperform various activation functions while its approximation ability is greatly improved with Range Norm. To deal with the spatiotemporal data, previous papers designed a class of modules called ConvRNN, which has shown a powerful effect on handling these tasks [45, 46, 53, 54]. We apply *Pac*s to these ConvRNNs, which serve as the activation functions after their inner convolutions of state and input. The *Pac*s-based ConvRNNs are tested in the summer precipitation prediction task based on a benchmark HKO-7 [46] with spatiotemporal radar echo data. The dataset is further enhanced with heterogeneous WRF data to gain the amount of rainfall in the next few hours [47]. In fact, previous precipitation forecasting models refined by *Pac*s and Range Norm all achieve great improvement on performance and validate the effectiveness of our modules.

Here we summarize our main contributions:

- We apply and investigate the **P**olynomial-**ac**tivation functions (*Pac*s) to boost the simulation of physical processes using deep learning model. We find that the data flow explosion problem is serious when the network goes deep.
- We tackle the training instability problem of *Pac*s-based deep learning model with 1) Chebyshev polynomials as basis functions, and 2) placing the normalization before polynomial operators for stable data propagation.
- A new normalization Range Norm is designed for robust training with *Pac*s in Section.3.
- We generate synthetic datasets and validate the effectiveness of *Pac*s and Range Norm compared with commonly used activation functions and layer normalization.
- We conduct extensive experiments on the rainfall prediction task by applying *Pac*s and Range Norm to various models and show the effectiveness of *Pac*s in Section.4.

We review related works in Section.2 and conclude this paper in Section.5.

2 BACKGROUND AND RELATED WORKS

2.1 Precipitation Forecasting with Deep Learning

Precipitation, which is one of the most important research fields in earth science [40], affects civil production and living all over the world [50, 57]. Traditionally, the domain experts from physics and engineering construct numerical models such as WRF (Weather Research and Forecasting) model [47] based on their domain knowledge as theory-driven solutions. However, lots of underlying mechanisms are still unclear [57]. Model designs and parameter settings still rely on empirical conjectures [16, 19, 20]. On the other hand,

coupling multiple weather processes to the weather system results in a larger and larger structure, which results in high computation cost and huge accumulation error [47].

Deep learning, serving as a data-driven black box, shows a powerful effect to handle these problems in recent years. As the radar image is a strong indicator for precipitation, Shi *et al.* first propose the encoding-forecasting framework and Convolutional LSTM, which combines CNN and RNN to handle spatiotemporal image series, to predict accurate radar images in the next 2 hours as precipitation nowcasting [45]. Their framework and modules are further improved. A module called Trajectory Gated Recurrent Unit (TrajGRU) is proposed with a learned convolution kernel as the hidden state could be aggregated along certain trajectories such as the Typhoon Eye [46]. Wang *et al.* integrate spatiotemporal information with additional memory state in the recurrent neural network as PredRNN, which is updated by both low-level stacked layer and state in previous time step [53]. A module called Memory In Memory (MIM) is also applied to radar image prediction, which adds non-stationary and stationary memory cells in PredRNN [54]. It models the target system as a non-stationary process of polynomial, which is approximated with state differencing as degree reduction. In contrast, our activation functions *Pac*s increase the degree of polynomials to model more complex physical interrelationships in the precipitation system.

While deep learning is applied to various earth science problems [7, 36, 56], integrating theory-driven and data-driven methods is another important direction to handle these earth science problems [5, 18, 27]. Bezenac *et al.* design a hybrid model, which uses the framework of a physical model for water surface temperature with its motion field simulation part replaced by an encoder-decoder deep learning model [5]. Gentine *et al.* use a neural network to derive intermediary rainfall-related variables including longwave and shortwave heating tendency, which is further fed into a numerical model for convection rain [18].

However, those models are framework-wise combinations of physical knowledge and deep learning. To be more specific, deep learning models are used to enrich or replace some or all blocks in the theory-driven numerical models, while the structures of applied deep learning models are simply duplicated from those used for NLP and CV tasks, such as FNN and CNN with variants of relu (e.g. elu, selu, leaky relu, etc.) and gated-like functions (e.g. sigmoid, tanh, softmax, etc.) as activation functions [9, 31, 39]. These modules may not match with the functionality of the physical system for precipitation, which limits their ability of approximation.

In contrast, *Pac* is a class of meta operators that has been underestimated while polynomial functions are used everywhere in theory-driven models [28, 43]. Lee and Jeng [34] feed inputs into all the product terms of Chebyshev polynomials with degrees no larger than $p(o)$ as augmentation and pass them through one fully-connected layer. Ma and Khorasani [37] propose a single-layer network with Hermite polynomials as activation functions. Their model is trained incrementally in order to use higher degree polynomials to decrease residual error. López-Rubio *et al.* [38] propose piecewise *Pac*s, which adapts the separation nodes for each degree of polynomial. It is limited to a single hidden layer network for the differentiability issue. These applications of *Pac*s are limited to single hidden layer. With experiments, we found that ***deep*** network is not converged with *Pac*s even after appending normalization. Our paper solves this problem using a new class of *Pac*s with specially selected basis functions and prepended normalization layer to pave a highway for precipitation and even physical modeling.

2.2 Convolutional Recurrent Neural Network

Recurrent Neural Network (RNN) and its variants are proposed for series data [8, 22]. The simplest RNN can be written as:

$$\begin{aligned} s_t &= \Phi(W_x \cdot x_t + W_s \cdot s_{t-1}) \\ y_t &= \sigma(W_y \cdot s_t) \end{aligned} \tag{1}$$

where the hidden state s_t is updated every step based on input x_t to generate output y_t. W_x, W_s, and W_y are learnable parameters operated with tensors by matrix multiplication '$\cdot$'. $\Phi(\cdot)$ and $\sigma(\cdot)$ are activation functions. Variants of RNNs involve more hidden states with complex interactions.

Convolutional Recurrent Neural Network (ConvRNN) is a class of modules to handle spatiotemporal data such as radar echo datasets, including ConvLSTM, ConvGRU, ST-LSTM, MIM, etc. [45, 46, 53, 54]. The temporal relationships are captured by an RNN. In each step, the input is a 2-dimensional image instead of a vector or scalar. In ConvRNN, the input is operated by convolution. The module is widely used for image processing[33, 51]. For example, we can modified the original RNN in Equation.1 to:

$$\begin{aligned} S_t &= \Phi(W_x * X_t + W_s * S_{t-1}) \\ Y_t &= \sigma(W_y * S_t) \end{aligned} \tag{2}$$

where '$*$' refers to the convolution operator. State S_t, input X_t, and output Y_t are matrices. In this way, researchers adapt various RNN models to ConvRNN, such as ConvLSTM and ConvGRU [45, 46]. Spatiotemporal states and complex interactions are added to further improve them [53, 54]. In our paper, *Pac*s are applied to the state and input convolutions in these ConvRNNs.

2.3 Approximation Ability of Deep Learning

Early in the 1980s, many researchers investigated the approximation ability of a multilayer feedforward perceptron (MLP) model through the theory of density [17, 21, 24, 26]. Cybenko first showed the universal approximation ability of a single hidden layer perceptron with sigmoidal functions as the activation function [12]. After that, many works extended the ranges of activation functions with universality [2, 23, 41, 48]. Leshno *et al.* prove that a single hidden layer MLP model can approximate any function if and only if its activation function is nonpolynomial [35]. Note that this theory does not ruin the power of a polynomial operator in deep learning. The universality of nonpolynomial functions is proven by spanning to different degrees of polynomials and applying Stone-Weierstrass Theorem for approximation [10]. To represent a linear space of algebraic polynomials of higher degree, a single hidden layer perceptron needs more hidden units [2]. A deep learning model with polynomial activation functions can be universal with enough number of layers [13], which is another direction of design network compared with using more hidden units [32].

Another important point is that, the existence of a universal approximator does not imply the hardness of training such a model. Many earliest proved activation functions with universality are not

commonly used nowadays, such as exponential and trigonometric functions [17, 48]. On the contrary, lots of research fields use polynomials to approximate complex systems [28, 43]. In addition, combining polynomial modules with the widely used piecewise linear activation functions results in piecewise polynomial approximation, which also shows great power in various applications [6, 15]. These successful implementations motivate us to refine the application of *Pac*s in deep learning models.

3 PROBLEM DEFINITION AND SOLUTION

3.1 Precipitation Forecasting Task

In this paper, we focus on the precipitation forecasting task with heterogeneous spatiotemporal weather data. At each time step $t \in T$, we have weather data from an area cut into $H \times W$ grids. There are two types of data sources. One is the radar image for rainfall, which can be represented as a tensor $X_t \in \mathbb{R}^{H \times W \times 1}$. Another is the simulation result from the WRF model, which covers D types of weather data with shape $A_t \in \mathbb{R}^{H \times W \times D}$. Given t_{in} time clips of radar image and WRF data, $X_1, X_2, \cdots, X_{t_{in}}$ and $A_1, A_2, \cdots, A_{t_{in}}$, precipitation forecasting task requires us to find a model to predict the next t_{out} clips of radar images $X_{t_{in}+1}, X_{t_{in}+2}, \cdots, X_{t_{in}+t_{out}}$.

3.2 *Pac*s

As mentioned in Section.2, polynomial operators are rarely applied to neural networks. In the following subsections, we first formally define our *Pac*s based on Chebyshev polynomials with prepended normalization. Then we clarify our motivation to use Chebyshev polynomials based on the drawbacks and experimental results of other *Pac*s. Finally, we discuss the advantages of prepending Range Norm layer for *Pac*s.

Given the output X from a layer of a linear neural network, such as FNN and CNN layer, an activation function σ is added after it as a mapping:

$$\sigma : X \to Y(\mathbb{R}^n \to \mathbb{R}^n), \tag{3}$$

where σ is applied pointwise so that input and output are of the same shape. We denote our *Pac*s of degree n as p_n. The output of p_n can be written as:

$$Y = p_n(X) = T_n(Norm(X)), \tag{4}$$

where T_n is the Chebyshev polynomials of the first kind with degree n and $Norm$ is the normalization. As *Pac*s are more powerful but furious to extract features, we further design a new normalization called Range Norm RN for stable data propagation. Formally,

$$RN(X) = \begin{cases} X & \text{if } Var(X) = 0; \\ \dfrac{2X - max(X) - min(X)}{max(X) - min(X)} & \text{otherwise,} \end{cases} \tag{5}$$

where $min(X)$ and $max(X)$ are the smallest and the largest value of the input X. Var refers to the variance, where $Var(X) = 0$ is a special case that all the input features are of the same value. The idea of designing the Range Norm is to make the output strictly fall into the range $[-1, 1]$. It is inspired by the idea of standardization which is used for data processing.

3.3 Why Chebyshev Polynomials

Compared with the deterministic approaches for parametric approximation, deep learning does not only require a model of high approximation ability, but also a model that could be trained stably and effectively. We have conducted experiments for multiple types of polynomial functions and different ways to combine them with normalizations and regularizations. Most of them are failed to be trained stably and result in overflow quickly. For more detailed descriptions, please refer to subsection 4.2. Finally, we find Chebyshev polynomials as an excellent class of functions for a stable training process combined with a prepended normalization module. Chebyshev polynomials of degree n, denoted as T_n, can be defined in a recurrence relation [44]:

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x) \end{aligned} \tag{6}$$

For input $x \in [-1, 1]$, Chebyshev polynomials can be expressed in trigonometric form [44], $T_n(cos\alpha) = cos(n\alpha)$ with $cos\alpha = x$. For example:

$$\begin{aligned} cos(0 \cdot \alpha) &= 1 && \Rightarrow T_0(x) = 1, \\ cos(1 \cdot \alpha) &= cos\alpha && \Rightarrow T_1(x) = x, \\ cos(2 \cdot \alpha) &= 2cos^2\alpha - 1 && \Rightarrow T_2(x) = 2x^2 - 1 \end{aligned}$$

Based on the above equations, one can conclude the following two properties of $T_n(x)$ when $x \in [-1, 1]$.

- $T_n(x)$ has n roots in the range $[-1, 1]$, $x = cos\frac{(2k-1)\pi}{2n}, k = 1, 2, \cdots, n$. This can be varified through the trigonometric form as $T_n(x) = cos(n\alpha) = cos(n\frac{(2k-1)\pi}{2n}) = 0$. As a polynomial function of degree n has exactly n roots, all the roots of T_n are located in $[-1, 1]$.
- The maximal absolute value of $T_n(x)$ in the range $[-1, 1]$ is 1. Similarly, from the trigonometric form, we can conclude that $|T_n(x)| = |T_n(cos\alpha)| = |cos(n\alpha)| \leq 1$.

With these two properties, Chebyshev polynomial works as a very stable mapping in the range of $[-1, 1]$ for data propagation. With the help of normalization, the output of our *Pac*s takes the advantage of a polynomial operator and conquers the vulnerability to data explosion.

Another important feature of the Chebyshev polynomials is that, in the domain $[-1, 1]$, among all the polynomials of degree n with absolute value bounded by 1, T_n has the largest possible leading coefficient [44]. This fact leads to a more powerful mapping with larger first-order derivatives, where $T_n'(1) = n^2$. With the furious growth of Chebyshev polynomials out of domain $[-1, 1]$, one can see the importance of a prepended Range Norm, which strictly narrows down the scale to $[-1, 1]$ and guarantees stability.

Note that previous paper [37] emphasizes that Hermite polynomials are orthogonal over the interval $(-\infty, \infty)$ and thus better than Chebyshev polynomials, which is orthogonal over the interval $[-1, 1]$. They argue that a confined range of input could hurt the ability of approximation. However, concrete proof and comparison are still missing. We show that *Pac*s work well at the approximation for polynomials after normalization, where stricter confined input range even greatly improved the results in subsection 4.1.

3.4 Why Prepended Range Norm

Normalization layers, which are proposed to handle the problem called internal covariate shift, make data propagate robustly and speed up the training process [25]. Compared with batch normalization which is usually used for CNN, layer normalization was proposed for RNN to make the features of each group with normalized mean and standard deviation [3]. This process sacrifices the distribution information (mean and std) to construct a smooth solution space for training. As mentioned above, polynomials are powerful in approximation but fragile during training. For Chebyshev polynomials which serve as a stable mapping function from domain $[-1, 1]$ to $[-1, 1]$, a prepended normalization can press the range of input close to 0, such as layer normalization. However, values out of $[-1, 1]$ still exist in the output of layer normalization. In contrast, range normalization output values in $[-1, 1]$ strictly. This process could lose more information but greatly improves the stability. Our experiments also validate this idea.

Most of the implementations place normalizations after activation functions to refine the final output. In contrast, our normalization is added before the Chebyshev polynomial operators. As a prepended normalization limits the scale of input, Chebyshev polynomial operators can output a stable data flow automatically. We also conduct experiments with appended normalization, which results in more and more skew distribution of output and fast data explosion during training.

4 EXPERIMENTAL STUDY

We make our code publicly available with finetuned parameters on Github [1]. We also set a fixed random key and train the models deterministically. All the results can be repeated exactly with the same server and python version.

4.1 Synthetic Dataset

As mentioned in Section.2, we argue that traditional activation functions are not powerful to learn complex non-linear systems such as physical problems. As polynomial functions are common in the governing equations of all kinds of research fields, we rely on *Pac*s to capture these functionalities. Here, we construct a synthetic dataset to validate that, with the help of Range Norm, *Pac*s can outperform various activation functions in learning the polynomials.

Dataset. Each input sample has d dims, randomly generated from a uniform distribution in the range $[-1, 1]$. Its output is a multivariable polynomial, of which terms are with weights uniformly selected from $[-r_w, r_w]$ and with degrees up to 3 and 4.

Model and Implementation Details. All the deep learning models are of h_u hidden units. The input is fed into n_l fully-connected layers for embedding, where the first layer is of size $d \times h_u$ and the following layers are of sizes $h_u \times h_u$. Each layer is appended by an activation function f with normalization. Resnet is used for each layer except for the first layer. Finally, a fully-connected layer of size $h_u \times 1$ maps the embedding to the output without any activation fuction.

The parameters we tested are listed in Table.1. *Pac*s of degree 2 and 3 are tested, denoted as *Pac*2 and *Pac*3.

We use MSE ($\frac{\sum_{i=1}^{N}(Y_i - \tilde{Y}_i)^2}{N}$) as the metric. As each combination of f and normalization has 16 subtasks, we rank the 10 combinations by MSE from 1 to 10 (Rank=1 for the best) for each task and derive the average rank for evaluation. Section A in the appendix shows more details about training.

Table 1: Compared Settings For Synthetic Dataset.

Parameters	Description	Settings
d	number of input variables	8
h_u	number of hidden units	[8, 16, 32, 64]
n_l	number of embedding layers	[2, 3, 4, 5]
r_w	range of the weight of objective function	5
f	activation function used	Relu, Sigmoid, SoftPlus, Pac2, Pac3
norm	type of normalization used	Layer Norm, Range Norm

Experimental Results. We show the performance of combinations of activation functions and normalizations for the approximation of polynomials of degree 3 in Table.2 and degree 4 in Table.3.

The performances of our *Pac*s (*Pac*2 and *Pac*3) with Range Norm rank the top 2 in both the approximations of polynomials of degree 3 and 4, which are marked with **bold font** and underline. In addition, *Pac*2 with Layer Norm gets the third-best rank for the target polynomial of degree 3 (3.62 compared with other scores $\geq$ 4.00) and 4 (2.88 compared with other scores $\geq$ 4.44). *Pac*3 works poorly at both tasks with Layer Norm, which results in extremely large losses ($\geq$ 100) sometimes. This is due to the instability as *Pac*3 is of a higher degree. In contrast, the results of *Pac*s are significantly improved with Range Norm. *Pac*3 even ranks the top-1 in both tasks. This indicates that activation functions of the higher degree of polynomials are more powerful for feature extraction as long as the stability is guaranteed.

4.2 Precipitation Forecasting

Dataset. Deep learning for precipitation forecasting based on large size radar echo data is first studied using the dataset from HKO (Hong Kong Observatory) [45], which is further published as a benchmark HKO-7 [46]. It covers the radar images from 2009 to 2015 captured every 6 minutes, where the original rainfall intensity is mapped to the scale of $[0, 255]$. Each image has 480×480 pixels for an area of size $512km \times 512km$ centered in Hong Kong. We continue to use the mask for denoising ground clutter and sun spikes. For more details, please refer to [46]. Summer heavy rainfall is critical to many natural disasters such as landslide. In addition, predicting hours of precipitation ahead is important for an early warning system [4]. Thus, we focus on hourly precipitation forecasting. In addition, we use weather data simulated by WRF model, which covers a smaller region including Hong Kong island with 162×213 pixels. These data are calculated with timestamp one hour, in May to August from 2011 to 2015. There are 4 types of data, water vapor mixing ratio $QVAPOR$, temperature T, zonal wind speed U, and meridional wind speed V. After data analysis, we discard temperature as $QVAPOR$ and temperature are highly correlated and $QVAPOR$ is a more direct indicator for the formation of condensation nucleus. Neither zonal nor meridional wind speed has a high correlation with precipitation. We generate the absolute wind speed $W = \sqrt{U^2 + V^2}$ which has a stable positive correlation with precipitation in each year (2011 to 2015). To combine the two data sources for one task, we extract the corresponding 162×213 pixels from each original radar image matching with the WRF data. Input and output radar images are of one-hour interval combined with $QVAPOR$ and generated W WRF data in the same hours. (e.g., a training sample of length 2 could be radar images at 12:06 and 13:06 with WRF data at 12:00

Table 2: The results for the synthetic dataset of degree 3. The best and the second-best average ranks and scores for each task are marked with bold font and underline. "↓" denotes the lower the better.

Model	MSE ↓																Average. Rank ↓
	$h_d = 8$				$h_d = 16$				$h_d = 32$				$h_d = 64$				
	$n_l = 2$	$n_l = 3$	$n_l = 4$	$n_l = 5$	$n_l = 2$	$n_l = 3$	$n_l = 4$	$n_l = 5$	$n_l = 2$	$n_l = 3$	$n_l = 4$	$n_l = 5$	$n_l = 2$	$n_l = 3$	$n_l = 4$	$n_l = 5$	
Relu+LN	18.71	15.00	20.19	20.72	10.53	13.74	10.86	12.43	10.99	9.895	9.429	8.801	8.050	8.414	8.133	6.586	6.50
Sigmoid+LN	15.52	17.88	16.06	16.66	9.635	14.30	9.453	10.86	8.663	8.705	7.412	7.973	6.345	5.048	4.602	6.096	4.94
SoftPlus+LN	14.92	13.71	19.61	17.22	10.82	9.762	9.467	10.69	8.850	6.555	8.658	7.218	7.553	9.016	8.828	8.600	5.44
Pac2+LN	13.03	12.93	15.16	23.98	9.580	8.925	9.116	10.94	5.158	5.925	8.512	5.175	1.438	2.818	3.924	6.603	3.62
Pac3+LN	21.69	14.23	25.88	27.15	12.01	13.13	12.65	15.50	9.669	11.12	11.128	18.50	8.844	10.72	13.31	531.3	7.81
Relu+RN	19.73	15.08	14.23	17.21	10.82	11.74	9.648	9.797	7.024	6.379	6.629	6.230	3.634	2.783	3.218	3.153	4.00
Sigmoid+RN	33.78	31.59	35.38	37.41	33.46	41.71	33.72	26.87	31.07	29.66	25.87	31.95	25.90	33.03	26.89	28.03	9.94
SoftPlus+RN	20.90	20.86	27.14	34.00	20.13	25.74	23.97	17.22	13.81	19.45	22.05	19.26	21.86	22.40	16.44	16.62	8.88
Pac2+RN	**12.00**	**12.14**	**12.22**	**12.43**	8.409	**7.242**	**6.262**	**7.764**	**2.807**	4.621	5.119	7.269	6.749	6.055	5.126	5.172	2.31
Pac3+RN	12.57	13.479	12.93	16.28	**7.928**	8.856	8.195	8.795	2.979	**3.386**	**2.401**	**3.972**	**0.972**	**1.285**	**1.301**	**1.577**	**1.56**

Table 3: The results for the synthetic dataset of degree 4. The best and the second-best average ranks and scores for each task are marked with bold font and underline. "↓" denotes the lower the better.

Model	MSE ↓																Average. Rank ↓
	$h_d = 8$				$h_d = 16$				$h_d = 32$				$h_d = 64$				
	$n_l = 2$	$n_l = 3$	$n_l = 4$	$n_l = 5$	$n_l = 2$	$n_l = 3$	$n_l = 4$	$n_l = 5$	$n_l = 2$	$n_l = 3$	$n_l = 4$	$n_l = 5$	$n_l = 2$	$n_l = 3$	$n_l = 4$	$n_l = 5$	
Relu+LN	42.79	27.86	40.63	32.23	26.85	25.58	25.49	21.75	21.44	23.26	20.27	22.97	17.94	18.54	18.01	19.77	6.94
Sigmoid+LN	36.78	29.73	38.07	31.10	24.50	23.84	22.78	23.21	17.30	15.94	17.06	17.31	12.87	12.05	11.75	13.54	5.12
SoftPlus+LN	36.54	26.78	27.91	30.30	24.19	24.89	22.84	19.57	18.30	18.96	15.71	19.23	17.95	15.38	17.83	20.35	5.06
Pac2+LN	33.73	28.07	27.18	29.32	21.74	20.98	20.46	18.14	11.68	14.97	11.07	14.98	9.453	11.31	6.338	8.528	2.88
Pac3+LN	54.68	32.48	34.19	30.98	26.67	28.08	91.18	24.22	24.69	21.51	25.56	246.4	21.52	21.85	155.7	26.37	8.00
Relu+RN	39.27	31.15	31.31	29.13	26.14	22.90	24.21	20.17	15.07	17.73	16.93	14.86	10.25	12.39	8.652	9.547	4.44
Sigmoid+RN	69.12	45.19	52.22	49.72	55.75	55.17	41.74	38.90	46.97	44.97	47.10	56.48	50.14	45.28	50.07	51.20	9.81
SoftPlus+RN	42.34	35.59	34.64	35.22	41.11	31.28	32.90	29.50	34.42	40.37	34.43	45.50	32.51	34.57	36.23	33.91	8.50
Pac2+RN	**29.62**	**24.27**	**25.09**	**22.35**	**18.68**	**19.43**	19.95	16.74	**8.687**	**8.682**	**6.880**	7.512	12.34	14.31	16.58	15.35	**2.12**
Pac3+RN	31.39	25.05	29.03	37.12	18.77	19.82	**19.45**	**14.33**	9.496	9.047	7.988	**6.476**	**5.055**	**5.015**	**4.285**	**3.797**	**2.12**

and 13:00.) Our task uses 3-hour's radar images and WRF data to predict the next 6-hour's precipitation radar images. To the best of our knowledge, this is the first work to predict large-scale hourly precipitation based on heterogeneous weather data.

Compared Models. To verify the generalization ability of our *Pac*s, we compare models based on three frameworks applied to precipitation radar image prediction [46, 53, 54]. Examples of these frameworks are given in Figure.2.

- **ConvGRU** [46]. The authors propose various models to fit convolution layers into RNNs in a framework called encoding-forecasting structure shown in Figure.2a. ConvGRU is a light and effective one of them with GRU (Gated recurrent unit) as RNN block.
- **PredRNN** [53]. Based on Convolutional RNN, they propose a layer called ST-LSTM, which adds the memory state for the traditional LSTM block. Their framework allows the information transported temporally forward and spatially upward (C, H states in red and M state in black in Figure.2b).
- **MIM** [54]. It shares a similar framework with PredRNN shown in Figure.2c. A new layer call MIM is proposed with stationary cell S and non-stationary memory cell N. These cells use state differencing between H in blue and red in Figure.2c to model the orders of a dynamic system.

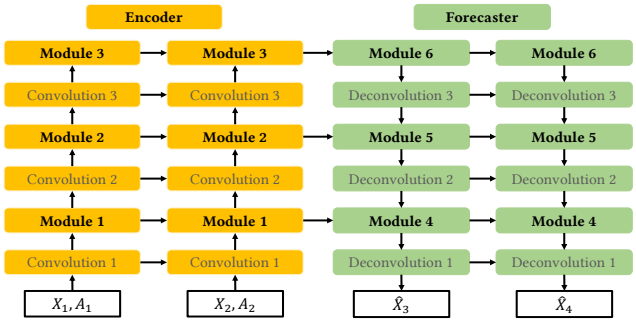
Recall that our *Pac*s are applied into ConvGRU, ST-LSTM, and MIM for the input and state convolutions. The adjusted ConvRNN are denoted as modules 1 to 10 (in black bold format) in the three frameworks in Figure.2, where more details about the parameter setting are listed in Section.Bin the appendix.

Based on the replacement, we propose the following groups of models for comparison.

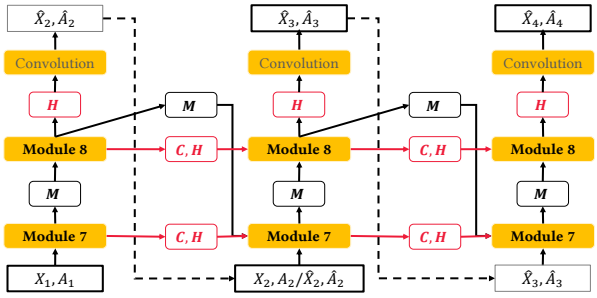
- **Baselines**. The original three models, ConvGRU, PredRNN, and MIM, without any modification.
- ***Pac*s-based Models with Layer Norm**. Using the 2^{nd} and 3^{rd} degree *Pac*s with Layer Norm for input and state convolution of the modules 1 to 10 in Figure.2 as stated above. We denote them as ConvGRU/PredRNN/MIM-*Pac*2/*Pac*3-LN.
- ***Pac*s-based Models with Range Norm**. Adjusted models based on *Pac*s with Range Norm , which are named as ConvGRU/PredRNN/MIM-*Pac*2/*Pac*3-RN.
- **Other Failed *Pac*s-based Models**. We repeat the setting of the above groups but remove or append normalization for the various polynomial operators, including power series x^n, Chebyshev polynomials, and Hermite polynomials [37]. We also test adding l1 and l2 regularizations of different weights to them. In addition, we use a special loss to punish too large output after activation. As all of their testing losses are either too large or even divergent, we do not display them in the evaluation part. Note that the second- and third-order Hermite polynomials used in [37] are not convergent even after prepending Range Norm.

For the more detailed comparison between *Pac*s, Relu, and Sigmoid as activation function with Range Norm and Layer Norm, please refer to Section.C in our appendix.

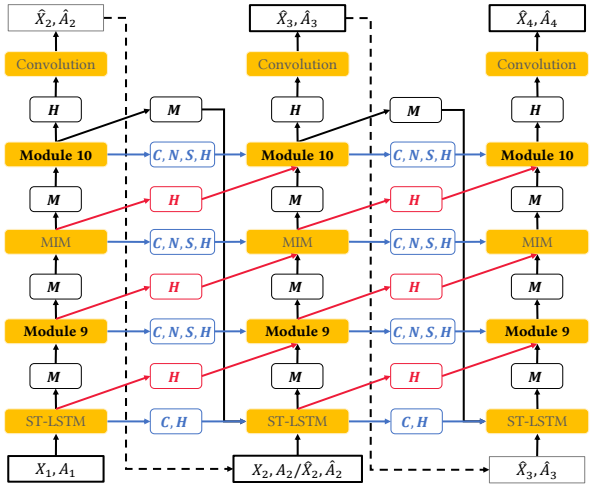
Implementation Details. We follow the previous work to evaluate the results based on multiple levels of precipitation intensity with threshold $r_1 = 0.5, r_2 = 2, r_3 = 5, r_4 = 10$, and $r_5 = 30$ [46]. Based on whether the values are in or out of an interval, we can calculate the TP (prediction=in, truth=in), TN (prediction=out, truth=out), FP (prediction=in, truth=out), and FN (prediction=out, truth=in). Two metrics CSI = $\frac{\text{TP}}{\text{TP+FN+FP}}$ and HSS = $\frac{\text{TP}\times\text{TN}-\text{FN}\times\text{FP}}{(\text{TP+FN})(\text{FN+TN})+(\text{TP+FP})(\text{FP+TN})}$ are used to evaluate the performance of the models for rainfall with different levels of intensities. In addition, we use the two metrics *B-MSE* and *B-MAE* that pay more effort to the heavy rainfall, which has a great impact on civil lives [46]. To be more specific, with regard to the rainfall intensity r in



(a) Example for the framework of ConvGRU.



(b) Example for the framework of PredRNN.



(c) Example for the framework of MIM.

Figure 2: The 3 tested frameworks, ConvGRU, PredRNN, and MIM. By applying *Pac*s of degrees 2 and 3 to the input and state convolutions in their original ConvRNN modules 1 to 10, we get 6 new models ConvGRU-Pac2, ConvGRU-Pac3, PredRNN-Pac2, PredRNN-Pac3, MIM-Pac2, and MIM-Pac3. All the examples use 2 timesteps of input (X_1, X_2, A_1, A_2) to predict the precipitation in the next 2 timesteps(X_3, X_4). (a) An example for the ConvGRU model with 3 levels in their encoder-forecaster framework. Modules 1 to 6 are ConvGRU layers in their original paper. (b) The example for the framework of PredRNN with 2 layers. Module 7 and 8 are applied with *Pac*s compared with the original ST-LSTM layer in their paper. (c) A framework of MIM with 1 ST-LSTM layer and 3 MIM-related layers. In their paper, modules 9 and 10 are two MIM layers.

each pixel, we have a weight function: $w(r) = \begin{cases} 1, & r < 2 \\ 2, & 2 \le r < 5 \\ 5, & 5 \le r < 10 \\ 10, & 10 \le r < 30 \\ 30, & r \ge 30 \end{cases}$. Then given the pixel set $P = [1, 162] \times [1, 213]$ and mask M, where pixel $p \in M$ refers to $p \in P$ is masked and vice versa, the *B-MSE* and *B-MAE* are defined as B-MSE= $\frac{1}{T_{out}} \sum_{t=1}^{T_{out}} \sum_{m \in P \setminus M} w(r_{t,p})(r_{t,p} - \tilde{r}_{t,p})^2$ and B-MAE= $\frac{1}{T_{out}} \sum_{t=1}^{T_{out}} \sum_{m \in P \setminus M} w(r_{t,p}) |r_{t,p} - \tilde{r}_{t,p}|$, where T_{out} is the number of images to predict and $r_{t,p}$ and $\tilde{r}_{t,p}$ are the ground truth and prediction result of the intensity in pixel p and t^{th} image. We also add average rank among all the metrics to compare all the models. Details of training configurations are in the appendix (Section A)

Experimental Results. The evaluation results are shown in Table.4. Each score is the average result of 3 repetitions, where **bold font** is used for the best scores and underlined scores are the second-best ones. For each framework, we compare eight variants. #-pure-long refers to the original models with only radar image (pure) as input, while #-wrf-long uses the concatenation of radar image and WRF data (wrf) as input data. #-pure and #-wrf have the same setting but run for the iterations a quarter of the original ones. With fewer iterations, all the frameworks have better performance as the models are overfitting under the original configuration. Thus, we keep this setting for the rest of the models, which are *Pac*s-based with both radar image and WRF data as input data. #-*Pac2*-LN and #-*Pac3*-LN embed *Pac*s of degrees 2 and 3 with Layer Norm into the original models according to Figure.2. #-*Pac2*-RN and #-*Pac3*-RN are the *Pac*s-based models with Range Norm.

B-MSE and B-MAE evaluate the prediction error of various intensities of precipitation as a whole. For the average scores of three frameworks, *Pac2* with Layer Norm outperforms all the 4 original models by reducing B-MSE by 5.2% to 17.5% and reducing B-MAE by 1.7% to 7.3%, while *Pac*3 with Layer Norm reduces B-MSE by 8.9% to 20.8% and reduces B-MAE by 0.3% to 6.0%. In addition, with Range Norm, *Pac*2 achieves 11.3% to 22.9% reduction on B-MSE and 1.7% to 7.4% reduction on B-MAE, while *Pac*3 outperforms the original models by decreasing B-MSE by 13.6% to 24.8% and decreasing B-MAE by 3.4% to 9.0%. The improvement validates the effectiveness of polynomial operators and shows the importance of Range Norm for *Pac*s.

The CSI and the HSI are defined on multiple levels of precipitation intensity r. For the light to middle rain (*i.e.*, $0.5 \le r \le 2$, $2 \le r \le 5$, and $5 \le r \le 10$), *Pac*s-based models overwhelmingly beat the original models. By ranking the 4 *Pac*-based models among 8 models in each framework, all the 4 models achieve top-5 ranks for each score, while for 38.9% scores the 4 models get all the top-4 ranks. With Range Norm, *Pac*s perform better and get the 17 best scores out of 18. For heavier rains (*i.e.*, $10 \le r \le 30$, $30 \le r$), the improvement of *Pac*s are not so stable. However, the best scores still belong to *Pac*-based models, which improve the best scores from those of the original models by 4.8% ($10 \le r \le 30$) and 58.8% ($30 \le r$) for the two CSI scores and 4.9% and 58.6% for HSI scores.

Overall, for all the metrics in all the groups, the two *Pac*s-based models with Range Norm get 61.1% of the top 1 scores while the

Table 4: The evaluation results for precipitation forecasting. In each group, bold font are the best B-MSE and B-MAE, where ↓ denotes the lower the better and ↑ for the higher the better. Underlined results are the second-best scores.

Framework	Group	B-MSE ↓	B-MAE ↓	CSI ↑					HSS ↑					Average Rank↓
				$r \geq 0.5$	$r \geq 2$	$r \geq 5$	$r \geq 10$	$r \geq 30$	$r \geq 0.5$	$r \geq 2$	$r \geq 5$	$r \geq 10$	$r \geq 30$	
ConvGRU	ConvGRU-pure-long	4282 ± 69	7300 ± 55	0.2024	0.1593	0.1070	0.0589	0.0147	0.2814	0.2338	0.1661	0.0965	0.0263	18.67
	ConvGRU-wrf-long	4298 ± 87	7312 ± 96	0.2043	0.1581	0.1048	0.0571	0.0145	0.2845	0.2322	0.1627	0.0936	0.0262	19.67
	ConvGRU-pure	3886 ± 240	7039 ± 191	0.2274	0.1759	0.1177	0.0727	0.0222	0.3085	0.2527	0.1785	0.1166	0.0400	11.83
	ConvGRU-wrf	3925 ± 238	7097 ± 199	0.2224	0.1717	0.1162	0.0702	0.0178	0.3004	0.2462	0.1763	0.1126	0.0320	13.33
	ConvGRU-*Pac*2-LN	3468 ± 123	6723 ± 142	0.2479	0.1924	0.1294	**0.0802**	**0.0289**	0.3351	0.2771	0.1975	**0.1294**	**0.0514**	4.08
	ConvGRU-*Pac*3-LN	3478 ± 319	7182 ± 94	0.2415	0.1745	0.1042	0.0521	0.0094	0.3235	0.2535	0.1621	0.0861	0.0167	17.58
	ConvGRU-*Pac*2-RN	3300 ± 139	6897 ± 56	0.2578	0.1955	0.1248	0.0621	0.0057	0.3468	0.2841	0.1938	0.1023	0.0103	9.92
	ConvGRU-*Pac*3-RN	**3163 ± 67**	**6678 ± 77**	**0.2643**	**0.2123**	**0.1401**	0.0791	0.0136	**0.3534**	**0.3071**	**0.2164**	0.1290	0.0243	**2.83**
PredRNN	PredRNN-pure-long	4202 ± 71	7182 ± 94	0.2105	0.1646	0.1112	0.0624	**0.0169**	0.2933	0.2415	0.1713	0.1011	**0.0299**	15.33
	PredRNN-wrf-long	4269 ± 76	7277 ± 60	0.1992	0.1567	0.1099	0.0604	0.0130	0.2764	0.2299	0.1697	0.0980	0.0227	19.42
	PredRNN-pure	3762 ± 84	6901 ± 38	0.2363	0.1790	0.1185	0.0682	0.0149	0.3223	0.2589	0.1806	0.1096	0.0265	12.92
	PredRNN-wrf	3655 ± 123	6775 ± 91	0.2423	0.1886	0.1291	**0.0765**	0.0143	0.3286	0.2726	0.1973	**0.1234**	0.0257	9.25
	PredRNN-*Pac*2-LN	3620 ± 209	6880 ± 79	0.2376	0.1838	0.1258	0.0759	0.0152	0.3211	0.2643	0.1912	0.1217	0.0272	9.92
	PredRNN-*Pac*3-LN	3488 ± 91	6752 ± 44	0.2461	0.1906	0.1273	0.0681	0.0108	0.3333	0.2754	0.1946	0.1102	0.0195	10.58
	PredRNN-*Pac*2-RN	3258 ± 34	6732 ± 60	**0.2524**	**0.1935**	**0.1309**	0.0591	0.0073	**0.3372**	**0.2791**	**0.2039**	0.0980	0.0134	9.00
	PredRNN-*Pac*3-RN	**3167 ± 102**	**6692 ± 23**	0.2510	0.1928	0.1224	0.0639	0.0072	0.3355	0.2788	0.1891	0.1055	0.0132	10.25
MIM	MIM-pure-long	4245 ± 88	7246 ± 18	0.2068	0.1583	0.1077	0.0614	0.0174	0.2871	0.2313	0.1654	0.0992	0.0307	16.67
	MIM-wrf-long	4336 ± 104	7300 ± 108	0.1913	0.1521	0.1053	0.0561	0.0136	0.2683	0.2243	0.1631	0.0913	0.0238	21.00
	MIM-pure	3805 ± 106	6884 ± 110	0.2375	0.1806	0.1186	0.0698	0.0181	0.3244	0.2608	0.1802	0.1116	0.0323	11.08
	MIM-wrf	3648 ± 71	6760 ± 42	0.2383	0.1887	0.1295	0.0756	**0.0182**	0.3244	0.2729	0.1976	0.1214	**0.0324**	7.42
	MIM-*Pac*2-LN	3556 ± 170	6683 ± 110	0.2449	0.1914	0.1310	**0.0793**	0.0123	0.3309	0.2750	0.1991	**0.1270**	0.0224	7.58
	MIM-*Pac*3-LN	**3259 ± 16**	6643 ± 28	**0.2546**	0.1932	0.1290	0.0699	0.0075	0.3383	0.2760	0.1965	0.1127	0.0136	7.83
	MIM-*Pac*2-RN	3396 ± 144	6645 ± 64	0.2529	0.1926	0.1269	0.0744	0.0150	0.3393	0.2765	0.1927	0.1188	0.0268	6.33
	MIM-*Pac*3-RN	3375 ± 104	**6556 ± 136**	0.2542	**0.1962**	**0.1319**	0.0781	0.0148	**0.3403**	**0.2814**	**0.2009**	0.1257	0.0264	3.50

two *Pac*s-based models with Layer Norm get 22.2% and the other 4 original models get 16.7%. Accounting for both the best and the second-best scores, models using *Pac*s cover 56.9% scores with Range Norm and 26.4% with Layer Norm. The average rank shows a comparison for all the 24 models in three frameworks. The best two models are ConvGRU and MIM based on our *Pac*s with Norm Range (rank 2.83 and 3.50 on average with 12 scores). In each framework, the average ranks of the two *Pac*-based models with Range Norm are better than the models using Layer Norm. This result validates our conjecture that by eliminating the instability using Range Norm, *Pac*s realize their full potential of the great power for feature extraction. With Layer Norm, *Pac*s still beat the two groups of original models. The great improvement of the precipitation prediction on all three frameworks indicates the reliability of our modules in various deep learning structures.

Discussion and Result Summary. In many designations of deep learning modules, researchers make the trade-off between stable training and expressive features. For example, layer normalization and batch normalization sacrifice the mean and std information of input data for smooth solution space.

In this work, we apply polynomial activation functions to extract meaningful non-linear features. However, the stability of training is weakened. We choose Chebyshev polynomials as the basis functions so that the output is robust with confined input, which can be gathered with the help of a prepended layer normalization.

However, to tame the furious *Pac*s, Layer Norm is still not harsh enough. In addition, the Range Norm further enhances the stability to feed *Pac*s with input in $[-1, 1]$. Such a transformation could lose more distribution information. On the synthetic dataset, other activation functions except for the nonsaturating Relu function work worse with Range Norm. However, *Pac*s are greatly improved even with this information loss, which should be the contributions of valuable extracted features of polynomial operators. It is promising to apply such a powerful operator to various tasks as we design the Range Norm to maintain its robustness.

Here, We conclude the experimental results. On the synthetic dataset, our *Pac*s outperform all the traditional activation functions to learn the class of polynomial objective functions with the help of Range Norm. For the precipitation forecasting task, all the frameworks are greatly improved with our *Pac*s, while Range Norm is more powerful than Layer Norm. The results show the effectiveness of our module to simulate the real precipitation system.

5 CONCLUSION

In this paper, we investigate the polynomial activation functions (*Pac*s) and design a novel class of *Pac*s for deep neural network applied to precipitation forecasting with heterogeneous data. In general, deep learning relies on activation functions to simulate nonlinearity. Compared with other activation functions with bounded first-order derivatives, *Pac*s serves as a polynomial approximator to simulate complex physical relationships in the rainfall system. Previous related studies suffer from the training instability problem and thus failed to apply *Pac*s to deep networks. Our structure solves this problem with Chebyshev polynomials and prepended normalization. We further propose a normalization called Range Norm to enhance stability. We validate the effectiveness of *Pac*s and Range Norm on a synthetic dataset and an enhanced benchmark HKO-7. On the synthetic dataset, *Pac*s outperform all the other activation functions to learn the polynomial objective functions with the help of Range Norm. On three frameworks for precipitation forecasting, models improved by our *Pac*s all achieve better performances and show the effectiveness of *Pac*s and Range Norm for real-world physical problems. Our work unearths an underestimated but powerful tool in deep learning to handle the complex physical relationship with heterogeneous data.

ACKNOWLEDGMENTS

This work is partially supported by Hong Kong RGC AOE Project AoE/E-603/18, Theme-based project TRS T41-603/20R, National Key Research and Development Program of China Grant No. 2018AAA0101100, the Hong Kong RGC GRF Project 16202218, CRF Project C6030-18G, C1031-18G, C5026-18G, China NSFC No. 61729201, Guangdong Basic and Applied Basic Research Foundation 2019B151530001 , Hong Kong ITC ITF grants ITS/044/18FX and ITS/470/18FX, Microsoft Research Asia Collaborative Research Grant, HKUST-NAVER/LINE AI Lab, Didi-HKUST joint research lab, HKUST-Webank joint research lab grants.

REFERENCES

[1] 2021. [Online] Code. https://github.com/dominatorX/Pacs/.
[2] P. Allan. 1999. Approximation theory of the MLP model in neural networks. *Acta Numerica* 8 (1999), 143–195.
[3] J. L. Ba, J. R. Kiros, and G. E. Hinton. 2016. Layer Normalization. (2016).
[4] Rex L Baum and Jonathan W Godt. 2010. Early warning of rainfall-induced shallow landslides and debris flows in the USA. *Landslides* 7, 3 (2010), 259–272.
[5] E De Bezenac, A. Pajot, and P. Gallinari. 2017. Deep Learning for Physical Processes: Incorporating Prior Scientific Knowledge. *Journal of Statistical Mechanics Theory and Experiment* 2019, 12 (2017).
[6] MŠ Birman and M. Z. Solomjak. 1967. Piecewise-Polynomial Approximations of Functions of the Classes. *Mathematics of the USSR-Sbornik* 2, 3 (1967), 295–317.
[7] N. D. Brenowitz and C. S. Bretherton. 2018. Prognostic Validation of a Neural Network Unified Physics Parameterization. *Geophysical Research Letters* 45 (2018).
[8] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*. ACL, 1724–1734.
[9] Djork-Arné Clevert, T. Unterthiner, and S. Hochreiter. 2015. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *Computer Science* (2015).
[10] N. E. Cotter. 1990. The Stone-Weierstrass theorem and its application to neural networks. *IEEE Transactions on Neural Networks* 1, 4 (1990), 290–5.
[11] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *RecSys*. ACM, 191–198.
[12] G. Cybenko. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems* 2, 4 (1989), 303–314.
[13] Bhaskar DasGupta and Georg Schnitger. 1992. The Power of Approximation: A Comparison of Activation Functions. In *NIPS*. Morgan Kaufmann, 615–622.
[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *CoRR* abs/2010.11929 (2020).
[15] S. Durrleman and R. Simon. 1989. Flexible regression models with cubic splines. *Statistics in Medicine* 8, 5 (1989), 551–561.
[16] J. Fritsch and C. Chappell. 1980. Numerical Prediction of Convectively Driven Mesoscale Pressure Systems. Part I: Convective Parameterization. *J.atmos* 37, 3 (1980), págs. 198–202.
[17] A. R. Gallant and H. White. 1988. There Exists A Neural Network That Does Not Make Avoidable Mistakes. In *Neural Networks, 1988., IEEE International Conference on*.
[18] P. Gentine, M. Pritchard, S. Rasp, G. Reinaudi, and G. Yacalis. 2018. Could Machine Learning Break the Convection Parameterization Deadlock? *Geophysical Research Letters* 45, 11 (2018).
[19] G. A. Grell and Dezs Dévényi. 2002. A generalized approach to parameterizing convection combining ensemble and data assimilation techniques. *Geophys.res.lett* 29, 14 (2002).
[20] G. A. Grell and S. R. Freitas. 2014. A scale and aerosol aware stochastic convective parameterization for weather and air quality modeling. *Atmospheric Chemistry and Physics* 13, 9 (2014), 5233–5250.
[21] R. Hecht-Nielsen. 1987. Kolmogorov's mapping neural network existence theorem. *IEEE. Press* (1987).
[22] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (1997), 1735–1780.
[23] K. Hornik. 1991. Approximation capabilities of multilayer feedforward networks. *Neural Networks* 4, 2 (1991), 251–257.
[24] K. Hornik, M. Stinchcombe, and H. White. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* 2, 5 (1989), 359–366.
[25] S. Ioffe and C. Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *JMLR.org* (2015).
[26] Irie and Miyake. 1988. Capabilities of three-layered perceptrons. In *Proc IEEE International Conference on Neural Nerworks*. 641–648 vol.1.
[27] V. Jochem, S. Neus, R. Juan, M. M. Jordi, V. Jorge, C. V. Gustau, and M José. 2016. Emulation of Leaf, Canopy and Atmosphere Radiative Transfer Models for Fast Global Sensitivity Analysis. *Remote Sensing* 8, 8 (2016), 673.
[28] H. K. Khalil. 2002. Nonlinear Systems. 38, 6 (2002), 1091–1093.
[29] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP*. ACL, 1746–1751.
[30] D. Kingma and J. Ba. 2014. Adam: A Method for Stochastic Optimization. *Computer Science* (2014).
[31] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-Normalizing Neural Networks. In *NIPS*. 971–980.
[32] M. Kozel. 2015. Deep Learning for Image Recognition. (2015).
[33] A. Krizhevsky, I. Sutskever, and G. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*.
[34] T. T. Lee and J. T. Jeng. 1998. The Chebyshev-polynomials-based unified model neural networks for function approximation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* (1998).
[35] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. [n.d.]. Multilayer Feedforward Networks with a Non-Polynomial Activation Function Can Approximate Any Function. *Social Science Electronic Publishing* ([n. d.]).
[36] Y. Liu, E. Racah, Prabhat, J. Correa, and W. Collins. 2016. Application of Deep Convolutional Neural Networks for Detecting Extreme Weather in Climate Datasets. (2016).
[37] Liying, Khorasani, and K. 2005. Constructive Feedforward Neural Networks Using Hermite Polynomial Activation Functions. *IEEE Transactions on Neural Networks* 16, 4 (2005), 821–833.
[38] Ezequiel Lopez-Rubio, F. Ortega-Zamorano, Enrique Dominguez, and Jose Munoz-Perez. 2019. Piecewise Polynomial Activation Functions for Feedforward Neural Networks. *Neural Processing Letters* 50, 1 (2019), 121–147.
[39] A. L. Maas, A. Y. Hannun, and A. Y. Ng. 2013. Rectifier Nonlinearities Improve Neural Network Acoustic Models. (2013).
[40] Markus, Reichstein, Gustau, Camps-Valls, Bjorn, Stevens, Martin, Jung, Joachim, and Denzler. 2019. Deep learning and process understanding for data-driven Earth system science. *Nature* (2019).
[41] H. N. Mhaskar and C. A. Micchelli. 1992. Approximation by superposition of sigmoidal and radial basis functions. *Advances in Applied Mathematics* 13, 3 (1992), 350–373.
[42] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*. 8024–8035.
[43] Powell and M. Jd. 2015. Approximation theory and methods. *Cambridge University Press* (2015).
[44] TJ Rivlin. 1974. The Chebyshev Polynomials, Pure and Applied Mathematics.
[45] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *NIPS*. 802–810.
[46] Xingjian Shi, Zhihan Gao, Leonard Lausen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2017. Deep Learning for Precipitation Nowcasting: A Benchmark and A New Model. In *NIPS*. 5617–5627.
[47] W. C. Skamarock, J. B. Klemp, J. Dudhia, D. O. Gill, and J. G. Powers. 2005. A Description of the Advanced Research WRF Version 2. *Ncar Technical* (2005).
[48] M. Stinchcombe and H. White. 1990. Approximating and learning unknown mappings using multilayer feedforward networks with bounded weights. In *Ijcnn International Joint Conference on Neural Networks*.
[49] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *NIPS*. 3104–3112.
[50] D. L. Swain, B. Langenbrunner, J. D. Neelin, and A. Hall. 2018. Increasing precipitation volatility in twenty-first-century California. *Nature Climate Change* 8, 5 (2018).
[51] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *CVPR*. IEEE Computer Society, 1–9.
[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*. 5998–6008.
[53] Yunbo Wang, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and Philip S. Yu. 2017. PredRNN: Recurrent Neural Networks for Predictive Learning using Spatiotemporal LSTMs. In *NIPS*. 879–888.
[54] Yunbo Wang, Jianjin Zhang, Hongyu Zhu, Mingsheng Long, Jianmin Wang, and Philip S. Yu. 2019. Memory in Memory: A Predictive Neural Network for Learning Higher-Order Non-Stationarity From Spatiotemporal Dynamics. In *CVPR*. Computer Vision Foundation / IEEE, 9154–9162.
[55] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed H. Chi. 2019. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *RecSys*. ACM, 269–277.
[56] M. A. Zaytar and C. E. Amrani. 2016. Sequence to Sequence Weather Forecasting with Long Short-Term Memory Recurrent Neural Networks. *International Journal of Computer Applications* 143, 11 (2016), 7–11.
[57] X. Zhang, F. W. Zwiers, G. Li, H. Wan, and A. J. Cannon. 2017. Complexity in estimating past and future extreme short-duration rainfall. *Nature Geoscience* 10, 4 (2017), 255–259.
[58] Guorui Zhou, Xiaoqiang Zhu, Chengru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep Interest Network for Click-Through Rate Prediction. In *KDD*. ACM, 1059–1068.

A TRAINING CONFIGURATION

For the synthetic dataset, the Adam optimizer [30] with a learning rate of 0.1 are applied for all the models. The training is conducted on one machine with 4 NVIDIA V100 GPUs. All the codes are implemented in Pytorch [42]. The batch size is 2048 and the maximum number of iterations is 2^{17}. We apply Early Stopping to all the models, which checks every 512 iterations and stops after 5 checkpoints without improvement. Each model has 16 subtasks. The results are the average scores of 5 repetitions for each task.

For the precipitation forecasting task, the models are trained on two machines, one with 4 NVIDIA V100 GPUs and another with 4 RTX 3090 GPUs. All the codes are implemented in Pytorch [42]. We use Adam optimizer [30] with learning rate 10^{-4} for ConvGRU models and 10^{-3} for PredRNN and MIM models. All the models are learning to minimize the sum of B-MSE and B-MAE (i.e., $0.00005(B\text{-}MSE + B\text{-}MAE)$). The batch size is fixed to 4 for all the models. We reduce the maximum running iterations (80,000 for ConvGRU and 160,000 for PredRNN and MIM) to one quarter (20,000 for ConvGRU and 40,000 for PredRNN and MIM), which has better results. Each model is repeated three times. All the other parameter settings are the same as the original papers.

B THE DETAILS OF MODEL CONFIGURATIONS

Here are the setting of frameworks for precipitation forecasting. We show the information for ConvGRU in Table.5, PredRNN in Table.6, and MIM in Table.7.

C VARYING ACTIVATION FUNCTION FOR RAINFALL PREDICTION

We display our result in Table.8. On the three frameworks (ConvGRU, PredRNN, and MIM), our *Pac*s are compared with Relu and Sigmoid functions, which are normalized by Layer Norm and Range Norm. *Pac*s-based models get the highest scores for each framework, where *Pac*3 with Range Norm is particularly powerful, which have 5 out of 6 best scores and 1 second-best score. *Pac*s-based models also get almost all the second-best scores except the B-MAE of Relu-based model on PredRNN with Layer Norm (B-MAE= 6710). The result further validates the effectiveness of our *Pac*s compared with commonly used activation functions. On the other hand, most of the models with Relu and Sigmoid activation functions do not have better results with Range Norm compared with Layer Norm. The adeptness of Range Norm and other activation functions still needs to be investigated in future work.

Table 5: The parameter setting of the models based on ConvGRU. All the structures are the same as Table 14 in their paper [46] with adjustments according to our input and output size. The ConvGRU layer has inner convolutions for input-to-state and state-to-state transition. We use *Module A* to denote these adjustable ConvGRU layers in the table, which are the original ConvGRU layers in baseline and *Pac*s-based ConvGRUs in the improved models. *Conv* and *Deconv* are convolution and deconvolution layers. *Ker(in), Str(in), Pad(in)* are the kernel size, stride and padding for the input-to-state convolution. *Ker(st)* and *Dil(st)* are the kernel and dilation size for the state-to-state convolution. *In/Out M* and *In/Out F* are the number of input/output sizes for object and feature dims. Value in parentheses (i.e., 3(1)) refers to the model with only radar echo data as input (only 1 feature dim). *Input* and *State* are the input and state for each layer.

Name	Ker(in)	Str(in)	Pad(in)	Ker(st)	Dil(st)	In/Out M	In/Out F	Layer	Input	State
econv1	7×7	5×5	1×1	-	-	$162\times213/32\times42$	3(1)/8	Conv	input	-
emodule1	3×3	1×1	1×1	5×5	1×1	$32\times42/32\times42$	8/64	Module A	econv1	-
econv2	5×5	3×3	1×1	-	-	$32\times42/10\times14$	64/64	Conv	emodule1	-
emodule2	3×3	1×1	1×1	5×5	1×1	$10\times14/10\times14$	64/192	Module A	econv1	-
econv3	3×3	2×2	1×1	-	-	$10\times14/5\times7$	192/192	Conv	emodule2	-
emodule3	3×3	1×1	1×1	3×3	1×1	$5\times7/5\times7$	192/192	Module A	econv3	-
fmodule1	3×3	1×1	1×1	3×3	1×1	$5\times7/5\times7$	192/192	Module A	-	emodule3
fdeconv1	2×2	2×2	0×0	-	-	$5\times7/10\times14$	192/192	Deconv	fmodule1	-
fmodule2	3×3	1×1	1×1	5×5	1×1	$10\times14/10\times14$	192/192	Module A	fdeconv1	emodule2
fdeconv2	5×5	3×3	0×1	-	-	$10\times14/32\times42$	192/192	Deconv	fmodule2	-
fmodule3	3×3	1×1	1×1	5×5	1×1	$32\times42/32\times42$	192/64	Module A	fdeconv2	emodule1
fdeconv3	7×8	5×5	0×0	-	-	$32\times42/162\times213$	64/8	Deconv	fmodule3	-
fconv4	1×1	1×1	0×0	-	-	$162\times213/162\times213$	8/1	Conv	fdeconv3	-

Table 6: The parameter setting of the models based on PredRNN. We follow the structures in their paper for radar image prediction [53].The ConvRNNs used for their model are ST-LSTMs. As there are only two layers in the framework, *Pac*s are applied to both layers in the adjusted models, where the blocks are denoted as *Module B* in the table. *Ker, Str* are the size of kernel and stride for convolutions in its blocks. Pat is the patch size, where $Pat\times Pat$ grids of f dims are converted to 1 grid with $Pat\times Pat\times f$ dims. The other nominations are the same as Table.5.

Name	Ker	Str	Pad	In/Out M	In/Out F	Layer	Input
reshape1	–	–	4	$162\times213/41\times54$	3(1)/48(16)	Reshape	input
module2	3×3	1×1	-	$41\times54/41\times54$	48(16)/128	Module B	reshape1
module3	3×3	1×1	-	$41\times54/41\times54$	128/128	Module B	module2
conv4	1×1	1×1	-	$41\times54/41\times54$	128/48(16)	Conv	module3
reshape5	–	–	4	$41\times54/162\times213$	48(16)/3(1)	Reshape	conv4

Table 7: The parameter setting of the models based on MIM. The structures are the same as theirs for the radar echo dataset [54]. The modules which could be the *Pac*s-based or the original MIMs are named as *Module C* in the table. The meanings of items are the same as in Table.6.

Name	Ker	Str	Pad	In/Out M	In/Out F	Layer	Input
reshape1	–	–	4	$162\times213/41\times54$	3(1)/48(16)	Reshape	input
stlstm2	3×3	1×1	-	$41\times54/41\times54$	48(16)/64	ST-LSTM	reshape1
module3	3×3	1×1	-	$41\times54/41\times54$	64/64	Module C	stlstm2
mim4	3×3	1×1	-	$41\times54/41\times54$	64/64	MIM	module3
module5	3×3	1×1	-	$41\times54/41\times54$	64/64	Module C	mim4
conv6	1×1	1×1	-	$41\times54/41\times54$	64/48(16)	Conv	module5
reshape7	–	–	4	$41\times54/162\times213$	48(16)/3(1)	Reshape	conv6

Table 8: The evaluation results for precipitation forecasting. In each group, the best B-MSE and B-MAE are in bold font, where ↓ denotes the lower the better. Underlined results are the second-best scores.

Framework	Normalization	Activation Function	B-MSE ↓	B-MAE ↓
ConvGRU	Layer Norm	*Pac*2	3468 ± 123	6723 ± 142
		*Pac*3	3478 ± 319	7182 ± 94
		Sigmoid	3311 ± 201	7143 ± 260
		Relu	3337 ± 238	6758 ± 251
	Range Norm	*Pac*2	3300 ± 139	6897 ± 56
		*Pac*3	**3163 ± 67**	**6678 ± 77**
		Sigmoid	3609 ± 109	7827 ± 168
		Relu	3429 ± 240	7052 ± 137
PredRNN	Layer Norm	*Pac*2	3620 ± 209	6880 ± 79
		*Pac*3	3488 ± 91	6752 ± 44
		Sigmoid	3459 ± 251	7281 ± 330
		Relu	3359 ± 97	6710 ± 91
	Range Norm	*Pac*2	3258 ± 34	6732 ± 60
		*Pac*3	**3167 ± 102**	**6692 ± 23**
		Sigmoid	3654 ± 378	7889 ± 358
		Relu	3301 ± 98	6856 ± 35
MIM	Layer Norm	*Pac*2	3556 ± 170	6683 ± 110
		*Pac*3	**3259 ± 16**	6643 ± 28
		Sigmoid	3490 ± 159	6710 ± 146
		Relu	3421 ± 76	6645 ± 31
	Range Norm	*Pac*2	3396 ± 144	6645 ± 64
		*Pac*3	3375 ± 104	**6556 ± 136**
		Sigmoid	3395 ± 43	6733 ± 21
		Relu	3657 ± 76	6717 ± 127