Proc. of the International Conference on Electrical, Computer and Energy Technologies (ICECET)
9-10 December 2021, Cape Town-South Africa

Continuous Convolutional Neural Networks: Coupled Neural PDE and ODE

Mansura Habiba
Cloud Solution Lead
IBM in Ireland Dublin,
Ireland
0000-0001-9051-1370

Barak A. Pearlmutter
Department of Computer Science & Hamilton Institute
Maynooth University
Maynooth, Ireland
0000-0003-0521-4553

2021 International Conference on Electrical, Computer and Energy Technologies (ICECET) | 978-1-6654-4231-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICECET52533.2021.9698766

***Abstract*—Recent work in deep learning focuses on solving physical systems in the Ordinary Differential Equation or Partial Differential Equation. This current work proposed a variant of Convolutional Neural Networks (CNNs) that can learn the hidden dynamics of a physical system using ordinary differential equation (ODEs) systems (ODEs) and Partial Differential Equation systems (PDEs). Instead of considering the physical system such as image, time -series as a system of multiple layers, this new technique can model a system in the form of Differential Equation (DEs). The proposed method has been assessed by solving several steady-state PDEs on irregular domains, including heat equations, Navier-Stokes equations.**

***Index Terms*—Convolutional Neural Network, Neural ODE, Neural PDE, Differential Equation**

I. Introduction

It is challenging to have an optimal balance between accuracy and efficiency while finding a solution for a Partial Differential Equation (PDE) system. Recent research in deep learning is focused on Physics informed neural networks. At the same time, data-driven neural networks that can be represented as a function of continuous-time series demonstrate the impressive result. Recently, Neural ODE (NODE) [2] is becoming promising for solving Ordinary Differential systems. NODE demonstrates promising result in case of continuous time series where the number of dimension is limited. For example, Fig 1 shows the learning vector of NODE model after 20 iterations for a butterfly curve, where the dimension is limited to eight only. Though the learning vector represents the hidden dynamics of the training data which constructs a butterfly curve, the training and test (at the left of figure) time series data are really struggling to reduce the training error. Neural ODE (NODE) models still need improvement to learn continuous time data with high dimensions.

We believe, NODE still needs significant improvement for time series with too many dimensions. Therefore, we are still looking for a robust model to solve PDE. At the same time, the steadiness of Convolutional Neural Network provide stability to the architecture of NODE models. In this paper, we presented a continuous CNN model that can continuously learn a PDE system. Traditional CNN models consider a continuous PDE system as a system with multiple layers, as shown in Fig.2. It is a challenge for learning continuous time series in real-time. Moreover, the sampling rate can be irregular wither higher frequency at the same time [5]. As a result, despite the excellent performance of CNN in several research areas, this model fails to exhibit similar performance with continuous data. One-dimensional CNN, aka a Time Delay Neural Network (TDNN) [3], are used in some use cases such as Natural Language Processing (NLP). However, a continuous system such as an integral PDE system is a very complex for TDNN to achieve high accuracy.

In this paper, we leverage the strength of a PDE solver and an ODE solver to define such a complex system. We proposed a new CNN model that acts like a one-dimensional TDNN to generate the integral of a PDE system and later convert that to an ODE system. Finally, an ODE solver is used to generate the output of the continuous system. Here, a continuous system is a system that takes inputs continuously with higher frequency. For example, an IoT device was continuously collecting sensor data over time duration T. We have adopted the architecture of a dimensional Convolutional Neural Network similar to TDNN and introduced continuous learning by coupling a PDE and an ODE system in the model. We call it Continuous Convolutional Neural Network (CCNN). The core component of the proposed work is to model PDE system using enhanced CNN models. The main contribution of this paper is as follows

- A novel Continuous Convolutional Neural Network (CCNN).
- A new technique to learn PDE system using deep learning model.
- Finally, a comprehensive performance evaluation of the proposed model using a simulated system for Chirps.

II. Model Design

The goal of this proposed model is to generalize Convolutional Neural Networks (CNNs) [4] to continuous space and time. First, let us consider a TDNN with a single unit and generalize it to continuous time. Fig. 2 shows the architecture of a TDNN model. TDNN model is used as the kernel function

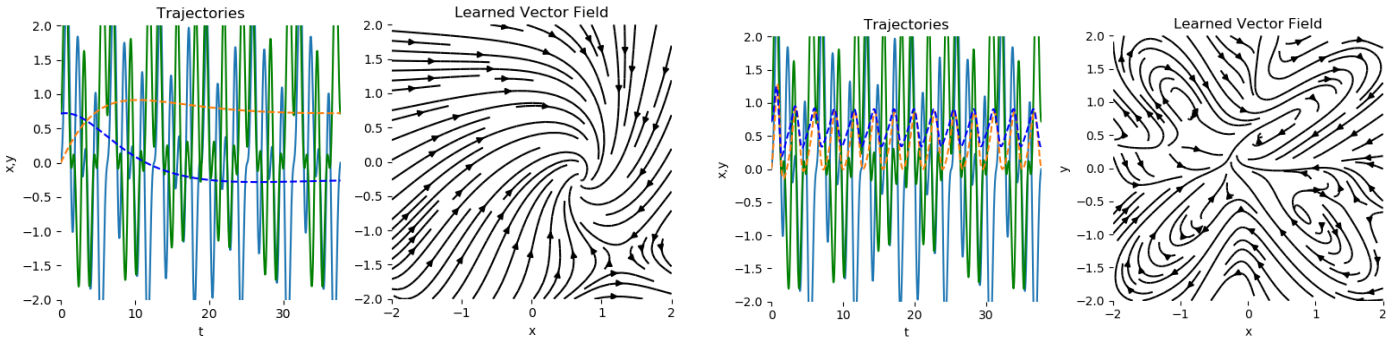

Fig. 1. Learning vector for butterfly curve at iteration 1 (Left) and 20 (Right)

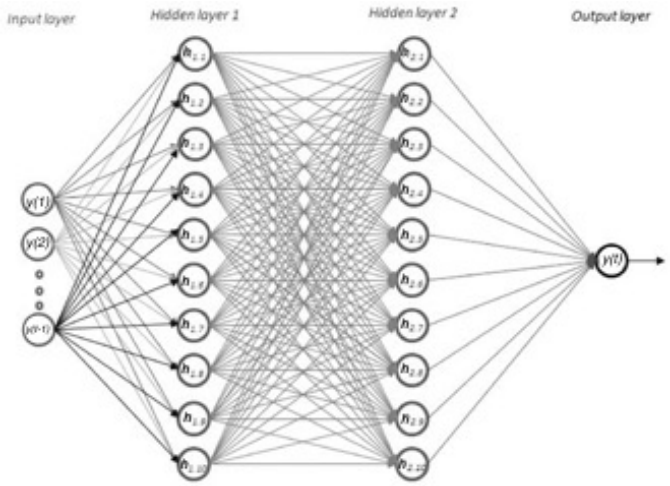

Fig. 2. Time-delay neural network (TDNN) with two hidden layers

shown in Eq. (3)(a). As TDNN model is also considered as one-dimensional CNN, this model helps to make the ODE solver more stable and solve the PDE in Eq. (3)(a).

In discrete time a TDNN model can be represented by Eq. (1)

$$y(t+1) = f(\sum_{\tau=0}^{T} K(\tau, W_K, t)y(t-\tau)) \tag{1}$$

where $y(t)$ is the activity of the unit at time t, the function f is a convenient nonlinearity, and $K(\tau, W_k)$ is a kernel function parameterized by W_k, typically using an explicit representation in the form of an array so $K(\tau, W_k) = W_k[\tau]$. This is defined for $T_0 \leq t \leq T_1$, with boundary conditions (i.e., initialization) handled specially.

To move to continuous time, a TDNN model can be described as Eq. (2).

$$\frac{\mathrm{d}}{\mathrm{d}t}y(t) = f(\int_{\tau=0}^{T} K(\tau, W_k, t)y(t-\tau)\mathrm{d}\tau) \tag{2}$$

Here, the generalized kernel function $K(\tau, W_k, t)$ is converted as parametric to accept parameters. This is causal but involves an integral. (This can be viewed as a delay-differential equation.) When coding, this would mean that the differential form

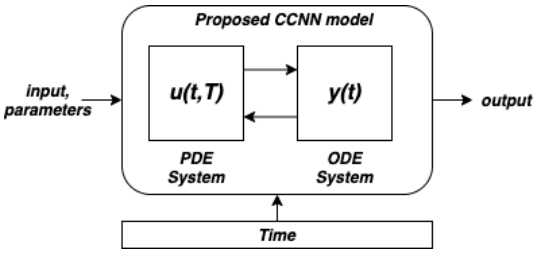

Fig. 3. The block diagram of proposed Continuous CNN model

passed to the ODE solver would contain an invocation of an integration operator, to which the solution of $y(t')$ for $t' \leq t$ would be passed. However, typically, this is not available: only the current value of $y(t)$ is available. So instead, we encode the integral in a PDE as shown in Eq. (3)(a).

$$\frac{\partial}{\partial\tau}u(t,\tau) = K(\tau, W_k, t)y(t-\tau) \tag{3a}$$

$$u(t,0) = 0 \tag{3b}$$

with $0 \leq \tau \leq T$. Note that

$$u(t,\tau') = \int_{\tau=0}^{\tau'} K(\tau, W_k, t)y(t-\tau)\mathrm{d}\tau \tag{4}$$

So in particular for a time duration $t \in [0, T]$, Eq. (3) provides the integral form of the PDE system.

$$u(t,T) = \int_{\tau=0}^{T} K(\tau, W_k, t)y(t-\tau)\mathrm{d}\tau \tag{5}$$

At this point, the integral from the driving term of y can be removed, instead expressing it in terms of u as shown in Eq. (5).

$$\frac{\mathrm{d}}{\mathrm{d}t}y(t) = f(u(t,T)) \tag{6}$$

Now we have a standard PDE for u as shown in Eq. (5) and an ODE for y as shown in Eq. (6), with no delays or integrals, and therefore Eq. (6) system is suitable for standard solvers. Fig.3 shows the block diagram of the CCNN model. In summary, CCNN is a coupled system of PDE and ODE systems.

Fig. 4 shows the architecture diagram of proposed model. This continuous model consists of a fully connected TDNN model which takes input $Y = \{y_1, y_2, \ldots y_{t-1}\}$ continuously at

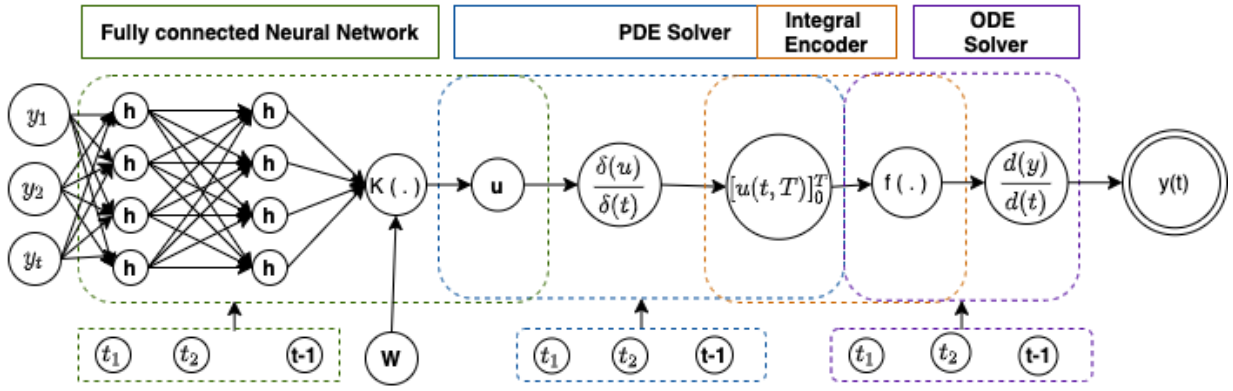

Fig. 4. The architecture diagram of proposed Continuous CNN model

each time step $T = \{t1, t2, ... t-1\}$. In addition, TDNN also takes the parameter W_k for the kernel function K as shown in Eq. (1). The second component in the model is a PDE solver which solves the input PDW system using Eq. (3)(a). On the next step, the Integral encoder converts the PDE system in to its integral form $u(t, T)$. $u(t, T)$ is a suitable continuous equation and can be passed to an ODE solver similar to NODE models. Finally, the ODE solver in Fig. 4 solves the continuous equation and generate the output y_t. The proposed CCNN model can also handle high dimensional inputs. For example, if $y(t) : \mathbb{R}^n$ for $n \geq 2$, i.e., a vector, then the system immediately generalizes by adding some extra indices.

Implementation

We have leveraged the neurodiffeq [1] library to implement the proposed model. For the implementation, we have a PDE system described as Eq. (5), and an ODE system described as Eq. (6). Using TDNN neural network, we can have solutions for both of the systems.

Challenges

To design the model, we need some additional sophisticated components for following challenges.

- The $\frac{\partial}{\partial \tau} u(t, \tau) = K(\tau, W_K, t) y(t - \tau)$ equation in Eq.(3)(a) has a shift, $t-\tau$, which might be hard to handle. Consider re-parameterizing u, with a change of variables, to get rid of the delay. Note that it would be okay to have a shift of the t argument to K, since K is fixed as far as the solver is concerned.
- A proper mechanism for driving term for an external input, instead of burying it in the initial conditions. Therefore, we have used the optimizer and model parameter as additional input to the model as shown in Fig. 4.
- Need to design error function and adjoint system. In future work, we will implement adjoint ODE solver, for Now we have leveraged the ODE solver from Neurodiffeq [1] library.

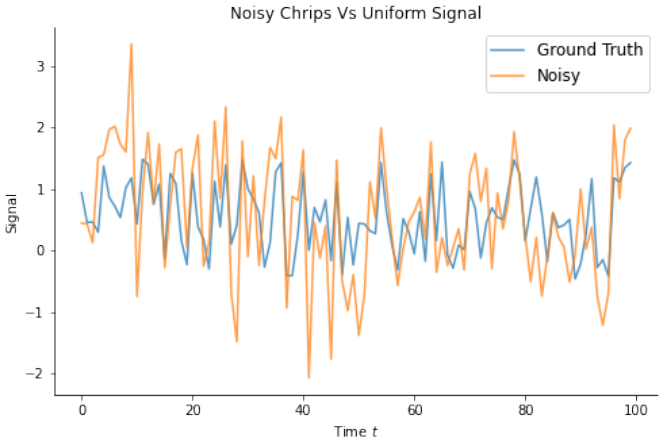

Fig. 5. One-dimensional input containing "chirps" for proposed model

Simulation with Chirps Signal

At this stage, we construct a one-dimensional input containing "chirps" to be detected against a background of white Gaussian noise, where the shape of a chirp varies systematically with time. This can be easily detected with a time-varying matched filter, and the performance evaluation shows that with learning, the kernel function K converges to the correct time-varying matched filter. Fig.5 shows the original signal in the blue curve and the noisy signal in the orange signal.

The ODE system solver function can be described as algorithm 1.

Algorithm 1 ODE System Solver Function

Input:: input signal (y), time series (t), internal (τ) and weight (W_k) as well as bias parameters (b)

1: **procedure** ODE_SOLVER($y, t, parameters$)
2: **return** $ODESOLVER(y, t)$

Similarly, the PDE system solver function can be described as algorithm 2. Here $kernel_function$ is the function shown in

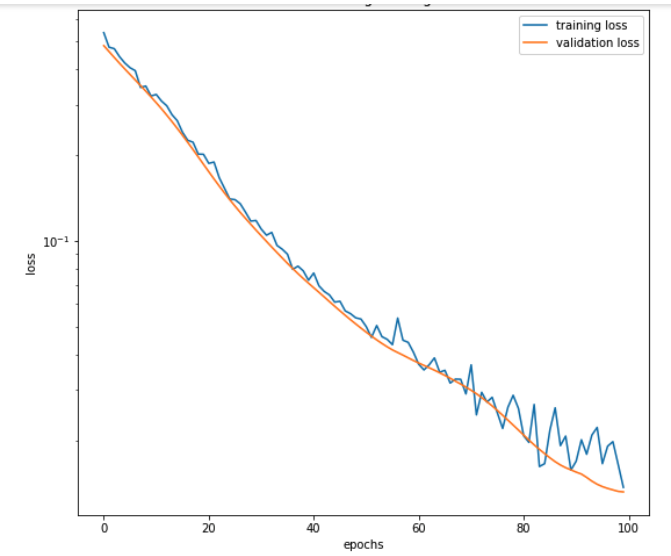

Fig. 6. Training loss for proposed CCNN model One-dimensional input containing "chirps"

Eq. (5). The ODE solver function described in algorithm 1 is used in algorithm 2 as the PDE and ODE system are coupled.

Algorithm 2 PDE System Solver Function

Input:: input signal (y), time series (t), internal (τ) and weight (W_k) as well as bias parameters (b)

1: **procedure** PDE_SOLVER_FUNCTION($y, t, parameters$)
2: $y \leftarrow kernel\_function(y, W_k, b)$
3: $u, v \leftarrow ode\_solver(y, t, parameters)$
4: **return** $[diff(u, t, order = 1) + diff(u, tau, order = 1)]$

The proposed model uses both ODE solver and PDE Solver modules. Fig. 6 shows the training loss for the noisy signal in Fig. 5.

III. Conclusion

This work proposed a new model for continuous Convolutional Neural Network (CCNN) and evaluated its performance for noisy signal (chips) processing. This model will help to process high dimensional time series with more than one dimension. This model also leverages the strength of Convolutional Neural Network (CNN) to process continuous data. In addition, any PDE system can be modelled and trained using this proposed model. We will work to integrate adjoint method in future work for PDE solver modules.

References

[1] F. Chen, D. Sondak, P. Protopapas, M. Mattheakis, S. Liu, D. Agarwal, and M. Di Giovanni. Neurodiffeq: A python package for solving differential equations with neural networks. *Journal of Open Source Software*, 5(46):1931, 2020.

[2] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31:6571–6583, 2018.

[3] J. Lang. On illuminations of c^2-surfaces in vector graphic description. *Comput. Graph.*, 12(1):33–38, 1988.

[4] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[5] D. Neil, M. Pfeiffer, and S.-C. Liu. Phased lstm: Accelerating recurrent network training for long or event-based sequences. In *Advances in Neural Information Processing Systems*, pages 3882–3890, 2016.