

Deep neural network methods for partial differential equations

Jiawei Sun: The Ohio State University

Vanessa López-Marrero: Brookhaven National Laboratory

Nathan Urban: Brookhaven National Laboratory

NYSDS'21, October 26-29, 2021

Introduction: keywords

- Neural network, examples of application: natural language processing, image processing, artificial intelligence etc.
- Partial differential equations (PDEs), examples of application: atmospheric and ocean dynamics, water wave theory, weather and climate science etc.
- Fourier neural operator (FNO): introduced by Li *et al.* in [1] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, Anima Anandkumar, Fourier Neural Operator for Parametric Partial Differential Equations, arXiv:2010.08895v3.

Neural network and numerical methods: comparison

- Classical numerical methods: Time consuming, high computational requirements on fine mesh
- Classical neural operator: Defined by matrix multiplication, Learn mappings between finite dimensional vector spaces.
- Fourier neural operator (FNO): Defined by integral convolution on functions.
- Advantage of FNO:
 - ▶ Time-efficient
 - ▶ Independent of spatial or temporal discretization
 - ▶ Independent of PDE parameters
 - ▶ Learn mappings between infinite dimensional Banach spaces, i.e. directly take functions as input and output data.

Purpose of this project

- Expand examples from [1] for Burgers' equation:
 - ▶ Including different types of initial conditions
 - ▶ modifying the neural network architecture
- Examine examples on Korteweg–De Vries (KdV) equation
- Test accuracy of FNO for both cases
- Neural network structure & initial implementation source:
https://github.com/zongyi-li/fourier_neural_operator

Structure of neural network

- Iteration updates:

$$v_{t+1}(x) = \sigma\left(Wv_t(x) + (\mathcal{K}(a; \phi)v_t)(x)\right), \quad (1)$$

- ▶ v : hidden layers
 - ▶ σ : activation function
 - ▶ W : linear weight transformation
 - ▶ \mathcal{K} : integral kernel
- Characterization of \mathcal{K} :

$$\left(\mathcal{K}(a; \phi)v_t\right)(x) = \int_D \kappa_\phi(x-y)v_t(y)dy, \quad (2)$$

- Apply Fourier and inverse Fourier transform onto (2):

$$\left(\mathcal{K}(\phi)v_t\right)(x) = \mathcal{F}^{-1}\left(\mathcal{F}(\kappa_\phi)\mathcal{F}(v_t(x))\right).$$

Structure of the neural network

- Definition of Fourier neural network:

$$\left(\mathcal{K}v_t\right)(x) = \mathcal{F}^{-1}\left(R_\phi\mathcal{F}(v_t)\right), \quad (3)$$

where R_ϕ is the Fourier transform of some periodic function κ .

- Other parameters of the neural network:
 - ▶ Activation function: ReLU
 - ▶ Optimizer: Adam
 - ▶ Learning rate: 0.001
 - ▶ Number of epochs: 500

Burgers' equation

Burgers' equation takes the form

$$\begin{cases} u_t + uu_x = \nu u_{xx}, & x \in (0, 1), t \in (0, 1) \\ u(x, 0) = u_0(x), & x \in (0, 1) \end{cases}$$

Learning settings

- input $a = (u_0(x), \nu)$
- output $u = u(x, 1)$
- Initial conditions u_0 are generated from a normal distribution
 $\mu = \mathcal{N}(0, \sigma^2(-\Delta + \tau^2 I)^{-\gamma})$
- ν is randomly selected in $[1/1000, 1/100]$

Burgers' equation: predictions on smooth initial curves

Training data set

- Neural network I (Tr1): 15 initial conditions \times 100 viscosities
- Neural network II (Tr2): 150 initial conditions \times 10 viscosities
- Neural network III (Tr3): 1500 initial conditions \times 1 viscosity

Testing data set

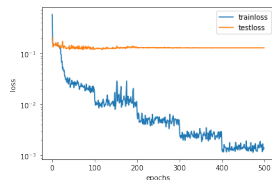
- Testing data set1 (T1): 40 initial conditions \times 10 viscosities
- Testing data set2 (T2): 4 initial conditions \times 100 viscosities

Burgers' equation: predictions on smooth initial curves

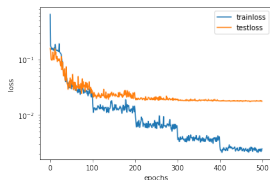
Training data set

- Neural network I (Tr1): 15 initial conditions \times 100 viscosities
- Neural network II (Tr2): 150 initial conditions \times 10 viscosities
- Neural network III (Tr3): 1500 initial conditions \times 1 viscosity

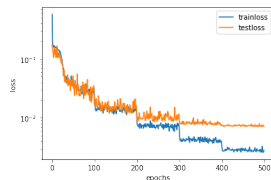
Plots of loss during epochs:



(a) Tr1



(b) Tr2



(c) Tr3

Figure 1: Loss during epochs for test data set T1

Burgers' equation: predictions on smooth initial curves

Training data set

- Neural network I (Tr1): 15 initial conditions \times 100 viscosities
- Neural network II (Tr2): 150 initial conditions \times 10 viscosities
- Neural network III (Tr3): 1500 initial conditions \times 1 viscosity

Plots of loss during epochs:

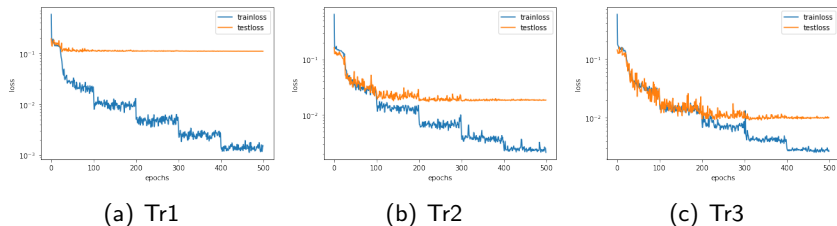
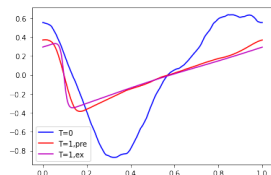


Figure 2: Loss during epochs for test data set T2

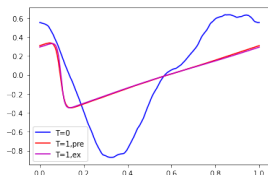
Burgers' equation: predictions on smooth initial curves

Training data set

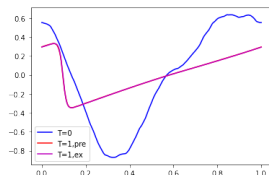
- Neural network I (Tr1): 15 initial conditions \times 100 viscosities
- Neural network II (Tr2): 150 initial conditions \times 10 viscosities
- Neural network III (Tr3): 1500 initial conditions \times 1 viscosity



(a) Tr1



(b) Tr2



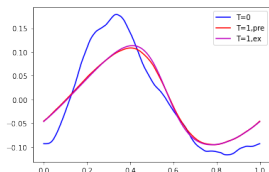
(c) Tr3

Figure 3: Examples of prediction for test data set T1

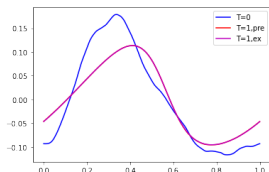
Burgers' equation: predictions on smooth initial curves

Training data set

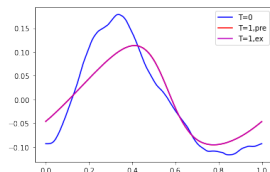
- Neural network I (Tr1): 15 initial conditions \times 100 viscosities
- Neural network II (Tr2): 150 initial conditions \times 10 viscosities
- Neural network III (Tr3): 1500 initial conditions \times 1 viscosity



(a) Tr1



(b) Tr2



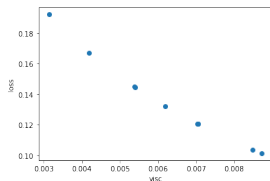
(c) Tr3

Figure 4: Examples of prediction for test data set T2

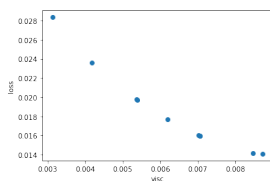
Burgers' equation: predictions on smooth initial curves

Training data set

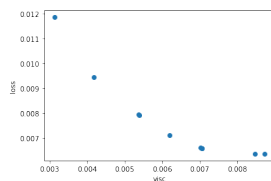
- Neural network I (Tr1): 15 initial conditions \times 100 viscosities
- Neural network II (Tr2): 150 initial conditions \times 10 viscosities
- Neural network III (Tr3): 1500 initial conditions \times 1 viscosity



(a) Tr1



(b) Tr2



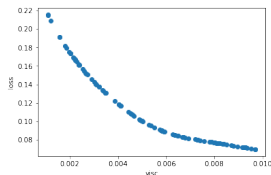
(c) Tr3

Figure 5: relationship between loss and viscosity for test data set T1

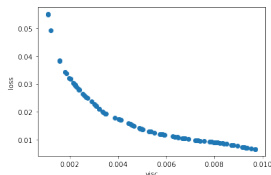
Burgers' equation: predictions on smooth initial curves

Training data set

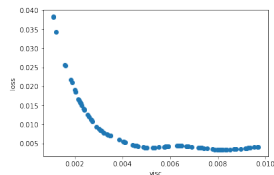
- Neural network I (Tr1): 15 initial conditions \times 100 viscosities
- Neural network II (Tr2): 150 initial conditions \times 10 viscosities
- Neural network III (Tr3): 1500 initial conditions \times 1 viscosity



(a) Tr1



(b) Tr2



(c) Tr3

Figure 6: relationship between loss and viscosity for test data set T2

Burgers' equation: predictions on sharp initial curves

Purpose

Try to predict solutions for sharp initial conditions (triangles and squares)

Using previously trained Tr3 to predict:

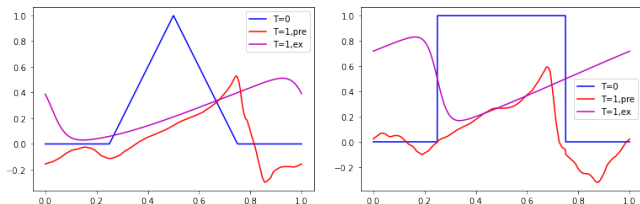


Figure 7: Predictions of sharp IC using Tr3

Burgers' equation: predictions on sharp initial curves

Training data set

- 1750 initial conditions \times 1 viscosity
- each viscosity is randomly selected from $1/100 - 2/100$
- each initial curve has probability $1/2$ to be a triangle or a square

Results of prediction:

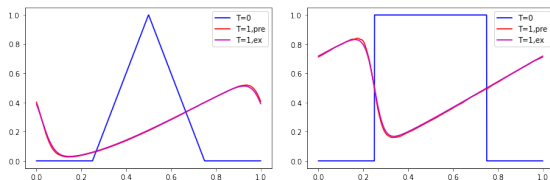


Figure 8: Predictions on sharp IC

Burgers' equation: predictions on sharp initial curves

Test back on normally distributed initial conditions:

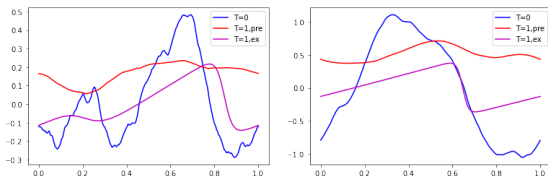


Figure 9: Predictions back on normally distributed IC

Merging training data

- Training data set: 1500 normally distributed initial curves and 1500 sharp functions (triangles and squares), each corresponds to a randomly chosen value of viscosity.
- Testing data set: 200 normally distributed initial curves and 200 sharp functions (triangles and squares).

Burgers' equation: Merging training data

Learning results:

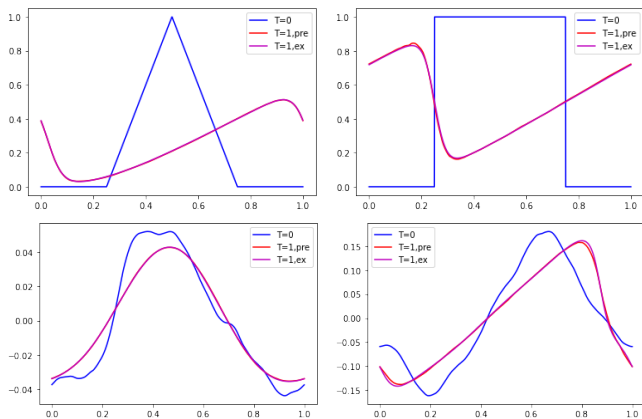


Figure 10: Predictions based on merging training data

Burgers' equation: conclusion

Conclusion

The neural network highly depends on the training data, and it only provides reliable predictions for data coming from the same family of the training data set.

KdV equation

The Korteweg–De Vries (KdV) equation is defined in the following form:

$$\begin{cases} u_t + uu_x + \delta^2 u_{xxx} = 0, & x \in (0, 1), t \in (0, 1) \\ u(x, 0) = u_0(x), & x \in (0, 1) \end{cases}$$

Learning settings

- input $a = (u_0(x), \delta^2)$
- output $u = u(x, 1)$
- Initial conditions u_0 are generated from a normal distribution $\mu = \mathcal{N}(0, \sigma^2(-\Delta + \tau^2 I)^{-\gamma})$

KdV equation: first attempt

Training data set

1500 initial conditions \times one δ^2

Learning result:

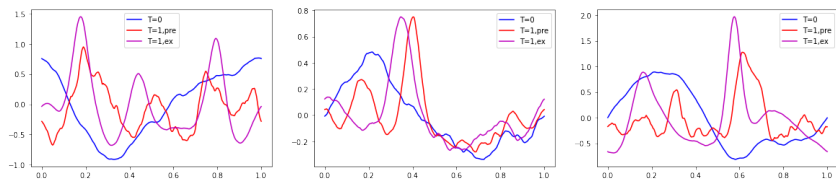


Figure 11: Predictions of KdV from model trained with 1500 data

KdV equation: first attempt

Using 7000 data and 10000 data:

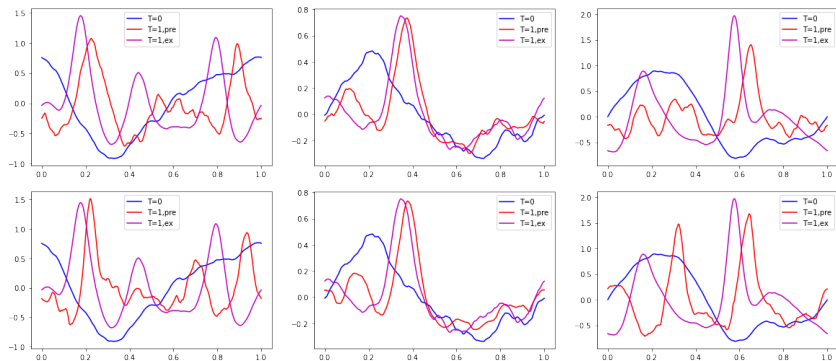


Figure 12: Predictions of KdV from model trained with 7000 data (upper) and 10000 data (lower)

KdV equation: fixing δ^2

Easier training: fix $\delta^2 = 2.4368E - 4$,

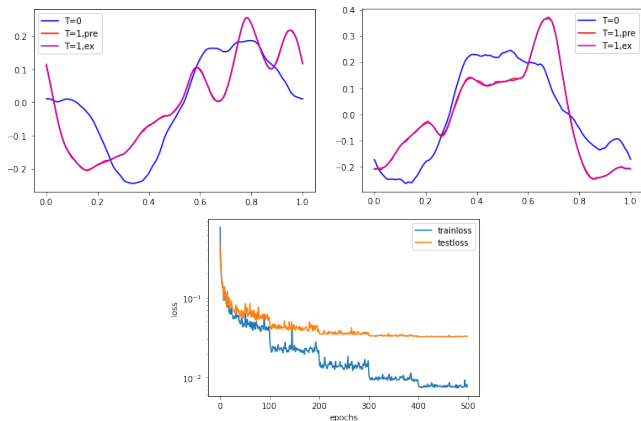


Figure 13: Prediction examples and plot for loss

KdV equation: fixing δ^2

More trainings with fixed δ^2

Using 30 different δ^2 to train 30 neural network models

Relationship between δ^2 and loss

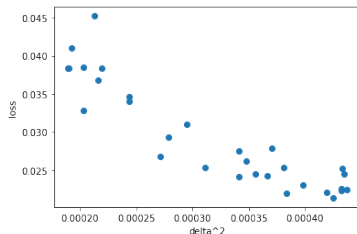


Figure 14: Plots of loss versus δ^2

Conclusion

Conclusion

The neural network can predict solutions very accurately

- for the KdV equation when fixing δ^2
- for Burgers' equation even if varying the parameter ν .


However, it is not able to learn solutions

- to the KdV equation if δ^2 is varied.

Possible reasons:

- discontinuous behaviour (or changes in the dynamics as δ^2 changes)
- solutions change too much when δ^2 changes
- neural network architecture may need to be modified
- further neural network hyperparameter tuning may be required

Reference

-  Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, Anima Anandkumar, *Fourier Neural Operator for Parametric Partial Differential Equations*, arXiv:2010.08895v3
-  Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, Anima Anandkumar, *Neural Operator: Graph Kernel Network for Partial Differential Equations*, arXiv:2003.03485v1

Acknowledgements

- National Science Foundation Mathematical Sciences Graduate Internship (NSF MSGI) Program
- Oak Ridge Institute for Science and Education (ORISE)
- Financial support from the NSF MSGI Program and the Empire State Development Corporation Funding, under Grant AA289