Two Approaches using Deep Learning to solve Partial Differential Equation

PDE solver and operator learning

Center for AI and Natural Sciences
Korea Institute for advanced Study (KIAS)
JaeYong Lee

Workshop for Korea-UK AI/ML Research in Fundamental Sciences
2022.11.01

Index

- **PDE solver**
  - Physics Informed Neural Network (PINN)
  - My interests on PDE solver

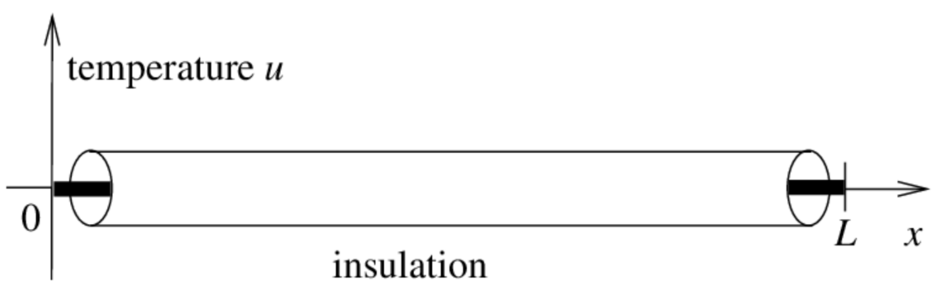
- **Operator learning**
  - CNN encoder-decoder
  - My interests on operator learning

Partial differential equation (PDE)

- **In mathematics, a differential equation is an equation that relates one or more functions and their derivatives.**
  - The functions generally represent **physical quantities.**
  - The derivatives represent **their rates of change.**
  - The differential equation defines a **relationship between the above two**.

(e.g.)

Let $u(t,x)$ denote the temperature at point x at time t.



Governing equation (heat equation) :

$$\frac{\partial u(t,x)}{\partial t} = k\frac{\partial^2 u(t,x)}{\partial x^2}.$$

Initial condition (IC) :

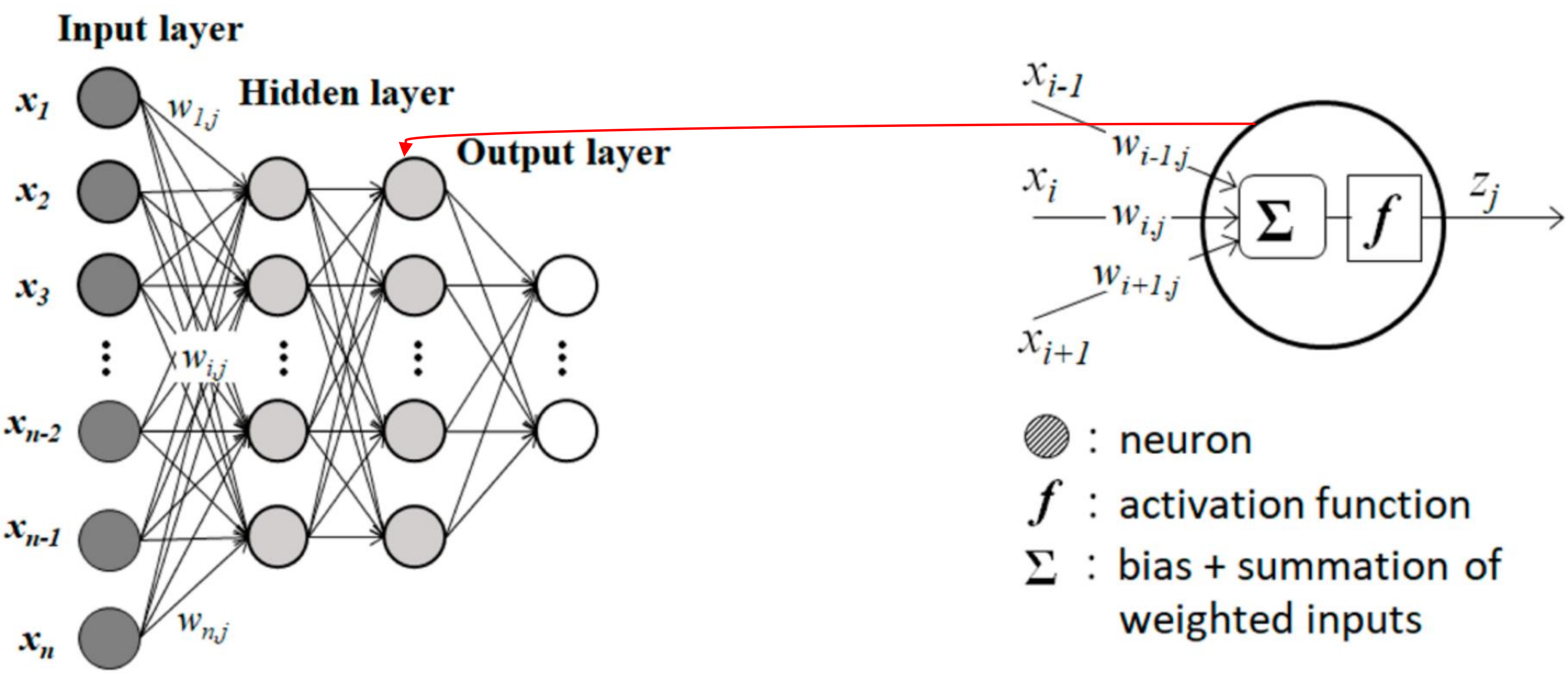
$$u(t=0,x) = f(x).$$

Boundary condition (BC) :

$$u(t,x=0) = h(x), \qquad u(t,x=L) = g(x).$$

https://math.libretexts.org/Bookshelves/Differential_Equations/Book%3A_Differential_Equations_for_Engineers_(Lebl)/4%3A_Fourier_series_and_PDEs/4.06%3A_PDEs_separation_of_variables_and_the_heat_equation

Deep Neural Network and Deep Learning



(**a**) Deep neural network (**b**) Inner structure of the neuron

https://www.researchgate.net/publication/338190342_Convolutional_Neural_Network_Coupled_with_a_Transfer-Learning_Approach_for_Time-Series_Flood_Predictions/figures?lo=1

Two mainstream on deep learning approach to PDEs

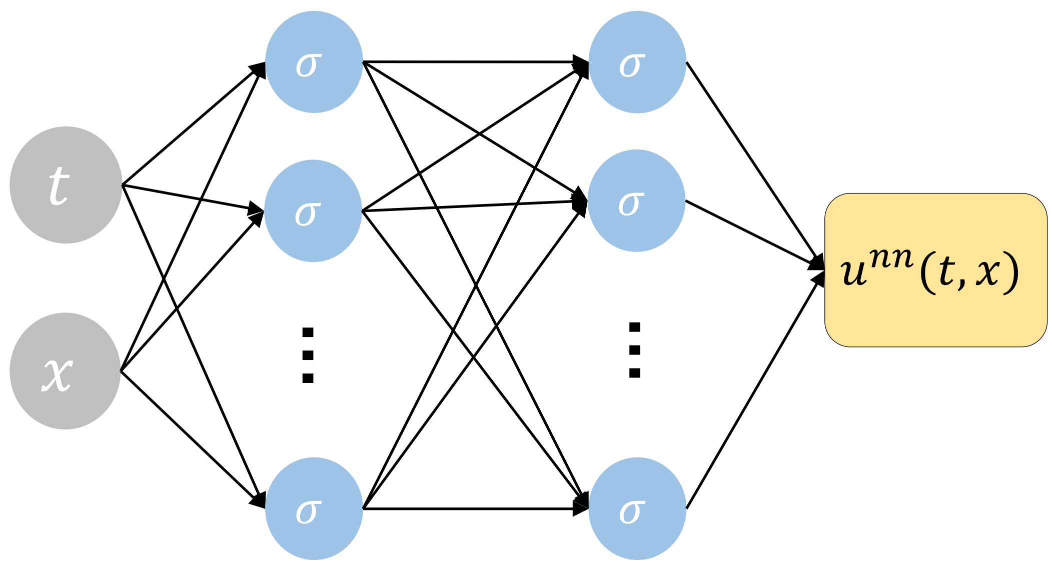
- **PDE solver**
  - Using neural networks directly to parametrize the solution to PDEs.
  - Solve one instance of PDE at a time.
  - Models
    - Deep Ritz Method (DRM)
    - Physics Informed Neural Network (PINN)

Find $\boldsymbol{u(t,x)}$ satisfying

$$\mathcal{L}_{PDE} = f(u, u_t, u_x, u_{xx}, \ldots) = 0$$

$$\mathcal{L}_{IC} = u(0,x) - g(x) = 0$$

$$\mathcal{L}_{BC} = u|_{\partial\Omega} - h(t,x)|_{\partial\Omega} = 0$$



Two mainstream on deep learning approach to PDEs

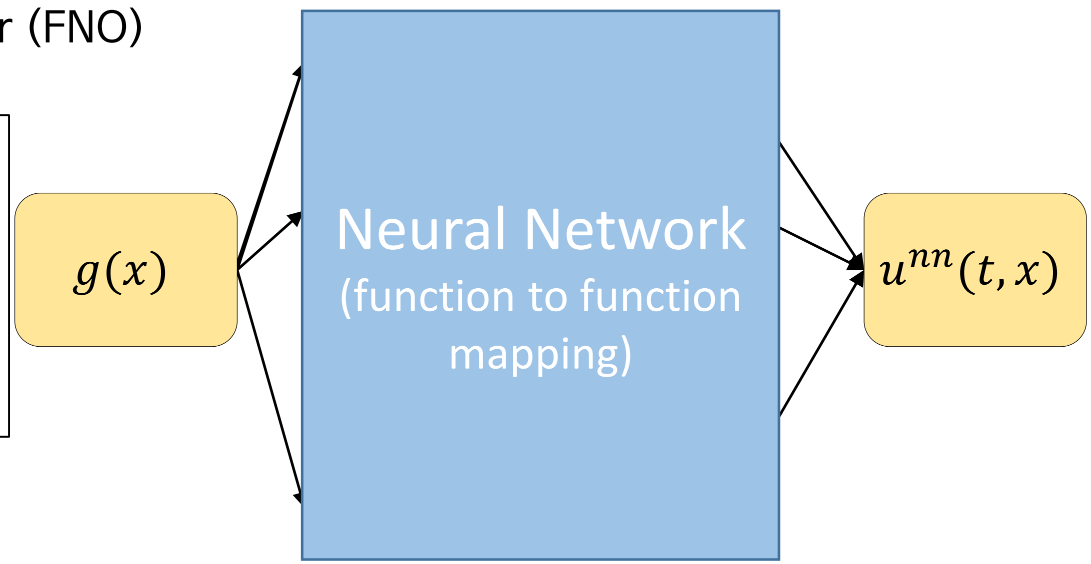
- **Operator learning**
  - Learning a mapping from the parameters (e.g. external force, initial, and boundary conditions) of the PDEs to the corresponding solution.
  - Learning a family of PDEs from data.
  - Models
    - Deep Operator Network (DeepONet)
    - Fourier Neural Operator (FNO)

Find a map $\boldsymbol{\mathcal{G}}: \boldsymbol{g(x)} \mapsto \boldsymbol{u(t,x)}$ satisfying

$$\mathcal{L}_{PDE} = f(u, u_t, u_x, u_{xx}, \dots) = 0$$

$$\mathcal{L}_{IC} = u(0,x) - g(x) = 0$$

$$\mathcal{L}_{BC} = u|_{\partial\Omega} - h(t,x)|_{\partial\Omega}$$



Part 1. PDE solver

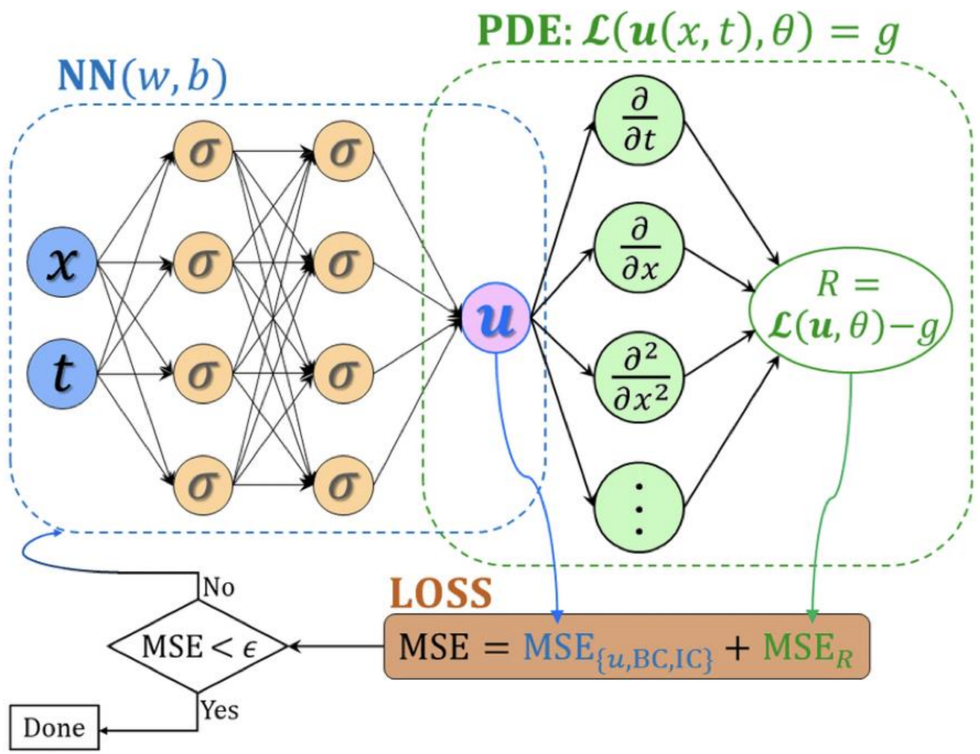
Physics-informed neural network (PINN)

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computation al physics*, *378*, 686-707.

Automatic differentiation and the back-propagation algorithm

“Here we revisit them using modern computational tools, and apply them to more challenging dynamic problems described by time-dependent nonlinear partial differential equations.”

[Schematic of a PINN]



Lu, L., Meng, X., Mao, Z., & Karniadakis, G. E. (2021). DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, *63*(1), 208-228

Physics-informed neural network (PINN) - Forward problem

(e.g.) Burgers' equation

Eq : $u_t + uu_x - (0.01/\pi)u_{xx} = 0$

IC : $u(0, x) = -\sin(\pi x)$ $\qquad x \in [-1,1], t \in [0,1]$

BC : $u(t, -1) = u(t, 1) = 0$

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, *378*, 686-707.

Physics-informed neural network (PINN) - Forward problem

(e.g.) Burgers' equation

Eq : $u_t + uu_x - (0.01/\pi)u_{xx} = 0$

IC : $u(0, x) = -\sin(\pi x)$ $\quad x \in [-1,1], t \in [0,1]$

BC : $u(t, -1) = u(t, 1) = 0$

Loss function

$$\boldsymbol{L}(\boldsymbol{\theta}) = \frac{1}{N_{Eq}} \sum_{i=1}^{N_{Eq}} \left| u_t(t^i, x^i) + u(t^i, x^i) u_x(t^i, x^i) - (0.01/\pi) u_{xx}(t^i, x^i) \right|^2 + \frac{1}{N_{IC}} \sum_{j=1}^{N_{IC}} \left| u(t^j, x^j) + \sin(\pi x^j) \right|^2 + \frac{1}{N_{BC}} \sum_{k=1}^{N_{BC}} \left| u(t^k, x^k) \right|^2$$

Data:

$(t_i, x_i) \in [0,1] \times [-1,1]$

$(t_j, x_j) \in \{0\} \times [-1,1]$

$(t_k, x_k) \in [0,1] \times \{-1,1\}$

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, *378*, 686-707.

Physics-informed neural network (PINN) - Forward problem

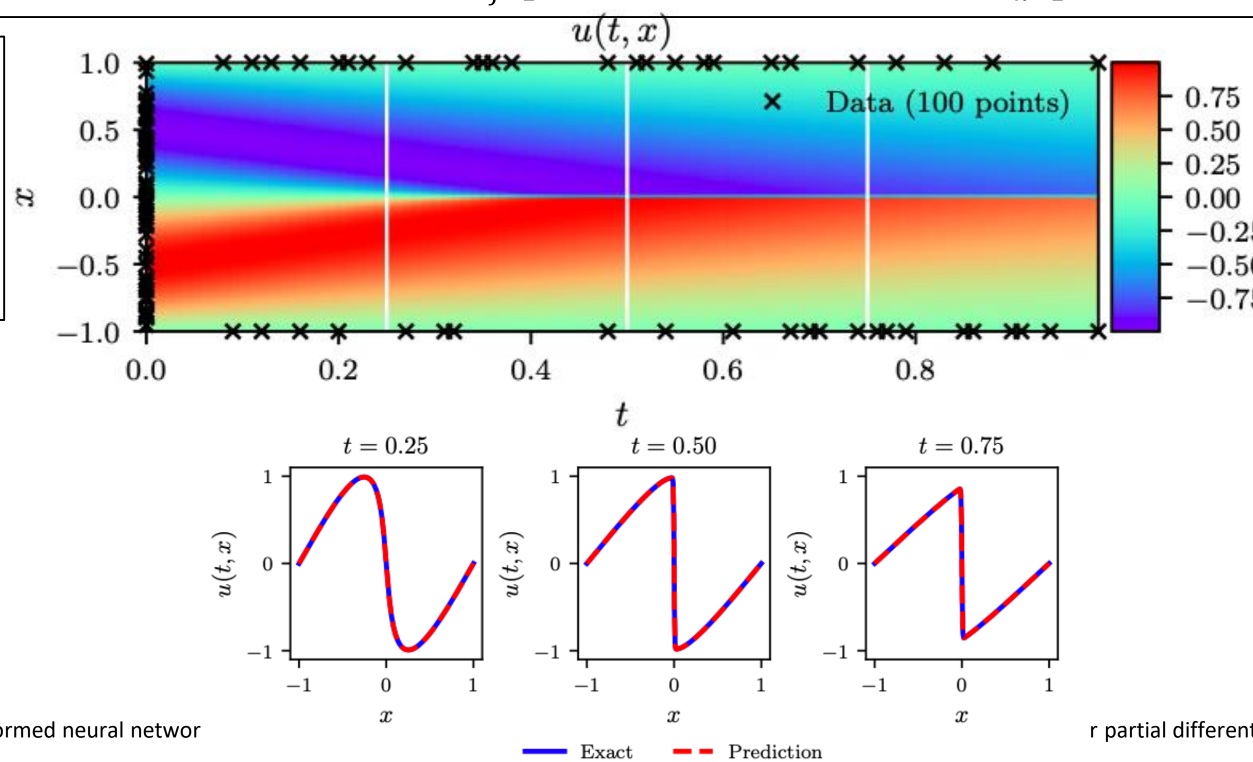
(e.g.) Burgers' equation

$$\text{Eq}: u_t + uu_x - (0.01/\pi)u_{xx} = 0$$
$$\text{IC}: u(0,x) = -\sin(\pi x) \qquad x \in [-1,1], t \in [0,1]$$
$$\text{BC}: u(t,-1) = u(t,1) = 0$$

Loss function

$$\boldsymbol{L(\theta)} = \frac{1}{N_{Eq}} \sum_{i=1}^{N_{Eq}} \left| u_t(t^i,x^i) + u(t^i,x^i)u_x(t^i,x^i) - (0.01/\pi)u_{xx}(t^i,x^i) \right|^2 + \frac{1}{N_{IC}} \sum_{j=1}^{N_{IC}} \left| u(t^j,x^j) + \sin(\pi x^j) \right|^2 + \frac{1}{N_{BC}} \sum_{k=1}^{N_{BC}} \left| u(t^k,x^k) \right|^2$$

Data:
$(t_i,x_i) \in [0,1] \times [-1,1]$
$(t_j,x_j) \in \{0\} \times [-1,1]$
$(t_k,x_k) \in [0,1] \times \{-1,1\}$



Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networ … r partial different ial equations. *Journal of Computational physics*, *378*, 686-707.

My interest [1]

Hwang, H. J., Jang, J. W., Jo, H., & **Lee, J. Y.** (2020). Trend to equilibrium for the kinetic Fokker-Planck equation via the neural network a pproach. *Journal of Computational Physics*, *419*, 109665.

- **Using PINN to the kinetic Fokker-Planck equation.**

The kinetic Fokker−Planck equation in a bounded interval $\Omega = [-1,1]$ is written as

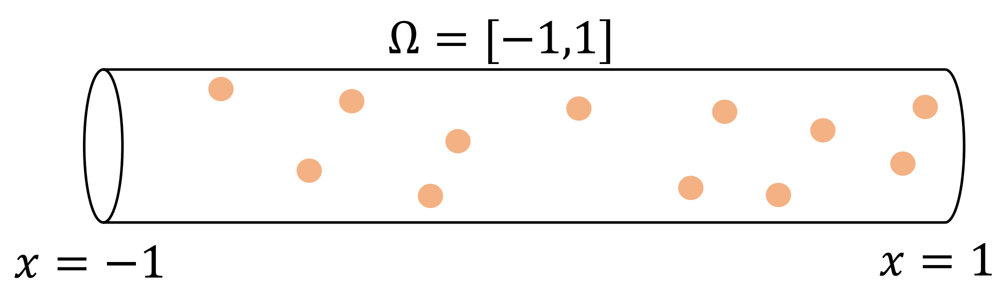
$$\partial_t f(t,x,v) + v \cdot \nabla_x f = \partial_v(\sigma \partial_v f + \beta v f),$$

subject to the initial condition

$$f(0,x,v) = f_0(x,v)$$

- $t \in [0,T]$ where T=5 or 10.
- $x \in [-1,1]$: unit ball in $\mathbb{R}^1$.
- $v \in [-10,10]$.

where σ is the diffusion coefficient, β is the friction coefficient, and the $f(t,x,v)$ is the pro babilistic density distribution of particles.



My interest [1]

Hwang, H. J., Jang, J. W., Jo, H., & **Lee, J. Y.** (2020). Trend to equilibrium for the kinetic Fokker-Planck equation via the neural network a pproach. *Journal of Computational Physics*, *419*, 109665.

- **Using PINN to the kinetic Fokker-Planck equation.**

The kinetic Fokker−Planck equation in a bounded interval $\Omega = [-1,1]$ is written as

$$\partial_t f(t,x,v) + v \cdot \nabla_x f = \partial_v(\sigma \partial_v f + \beta v f),$$

- $t \in [0,T]$ where T=5 or 10.
- $x \in [-1,1]$: unit ball in $\mathbb{R}^1$.
- $v \in [-10,10]$.

subject to the initial condition

$$f(0,x,v) = f_0(x,v)$$

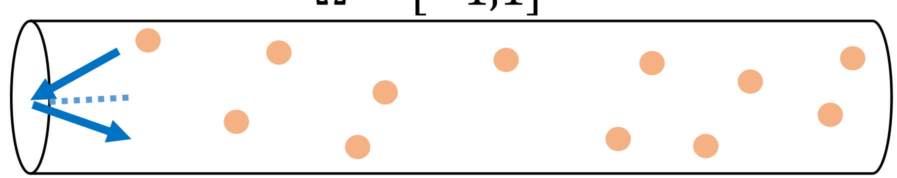
where σ is the diffusion coefficient, β is the friction coefficient, and the $f(t,x,v)$ is the pro babilistic density distribution of particles.

* Boundary conditions

– **Ex) Specular reflection** boundary condition

- $f(t,x,v) = f(t,x,-v)$, for $x=-1$ and $x=1$

$\Omega = [-1,1]$



$x=-1$ $x=1$

My interest [1]

Hwang, H. J., Jang, J. W., Jo, H., & **Lee, J. Y.** (2020). Trend to equilibrium for the kinetic Fokker-Planck equation via the neural network a pproach. *Journal of Computational Physics*, *419*, 109665.

- **Using PINN to the kinetic Fokker-Planck equation.**

The kinetic Fokker−Planck equation in a bounded interval $\Omega = [-1,1]$ is written as

$$\partial_t f(t,x,v) + v \cdot \nabla_x f = \partial_v(\sigma\partial_v f + \beta v f),$$

- $t \in [0,T]$ where T=5 or 10.
- $x \in [-1,1]$: unit ball in $\mathbb{R}^1$.
- $v \in [-10,10]$.

subject to the initial condition

$$f(0,x,v) = f_0(x,v)$$

where σ is the diffusion coefficient, β is the friction coefficient, and the $f(t,x,v)$ is the pro babilistic density distribution of particles.

* Boundary conditions

- **Absorbing** boundary condition: $f(t,x,v)|_{\gamma_-} = 0$, for x=−1 and 1.
- **Inflow** boundary condition: $f(t,x,v)|_{\gamma_-} = g(t,x,v)$, for x=−1 and 1 with a given function $g(t,x,v)$.
- **Specular reflection** boundary condition: $f(t,x,v) = f(t,x,-v)$, for x=−1 and 1.
- **Periodic** boundary condition: $f(t,x,v) = f(t,-x,v)$, for x=−1 and 1.
- **Diffusive** reflection boundary condition: $f(t,x,v) = C\mu(v)\int_{w\cdot n_x>0} f(t,x,w)|w\cdot n_x|dw,$ for x=−1 and 1, where $C = \left(\int_{v\cdot n<0}\mu(v)|v\cdot n|dv\right)^{-1}$ and $\mu(v) = e^{-\frac{v^2}{2}}$ for both $\mathrm{n} = \hat{x}$ and $-\hat{x}$

My interest [1]

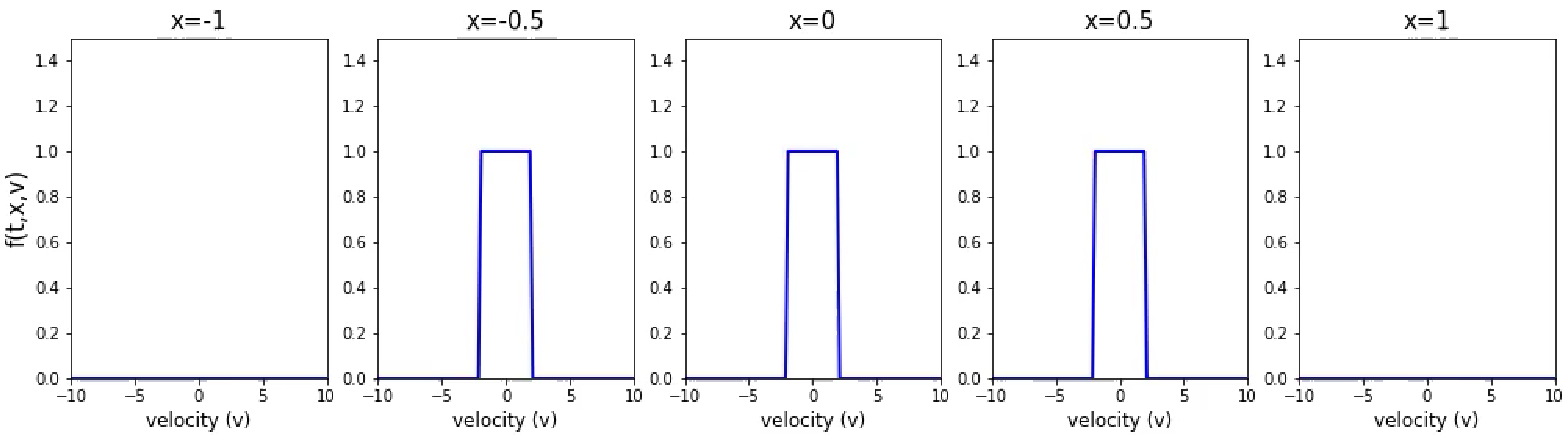
Hwang, H. J., Jang, J. W., Jo, H., & **Lee, J. Y.** (2020). Trend to equilibrium for the kinetic Fokker-Planck equation via the neural network a pproach. *Journal of Computational Physics*, *419*, 109665.

- **Using PINN to the kinetic Fokker-Planck equation.**
  - Easily changing the varied types of the physical boundary conditions.
  - Include a term regarding the conservation of the total mass of the system in the t otal loss function
  - Long time behaviors of neural network solution and its physical quantities
    - Apply various initial conditions and coefficients.
    - Convergence to the global Maxwellian.
  - Providing the theoretical supports for the pointwise convergence of the neural ne twork solutions to the a priori analytic solutions.

My interest [1]

Hwang, H. J., Jang, J. W., Jo, H., & **Lee, J. Y.** (2020). Trend to equilibrium for the kinetic Fokker-Planck equation via the neural network approach. *Journal of Computational Physics*, *419*, 109665.

- **Results : Specular boundary conditions which conserve the total mass**



Video: The pointwise values of $f^{nn}(t,x,v)$ as t varies at each x's for the **specular boundary condition**. x = ±1 stand for the boundary points, and x = ±0.5 and = 0 are the points away from the boundary.

Part 2. Operator learning

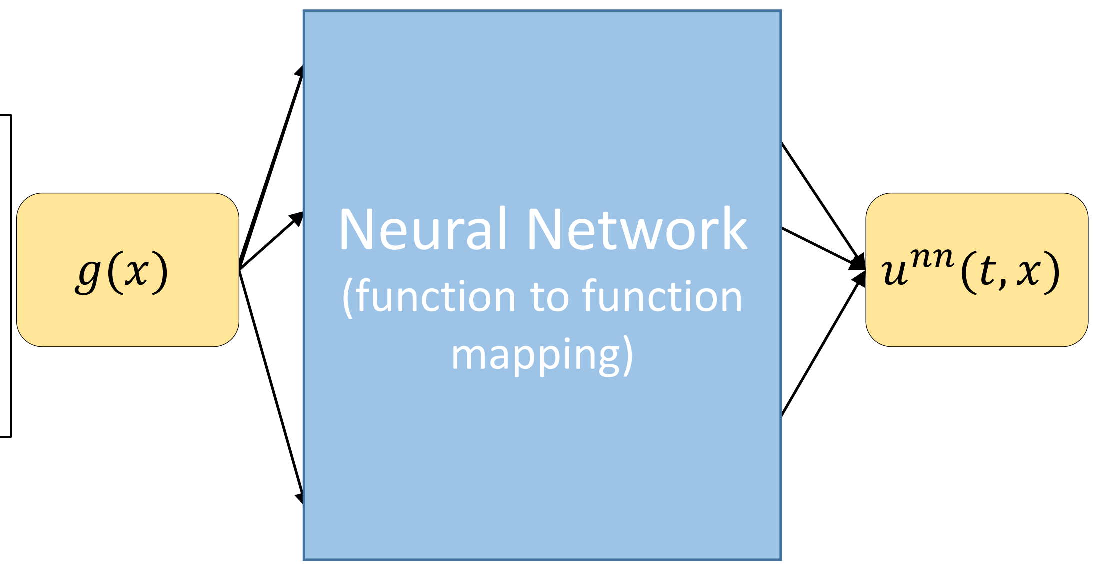
- **Operator learning**
  - Learning a mapping from the parameters (e.g. external force, initial, and boundary conditions) of the PDEs to the corresponding solution.
  - Learning a family of PDEs from data.
  - Models
    - Deep Operator Network (DeepONet)
    - Fourier Neural Operator (FNO)

Find a map $\boldsymbol{\mathcal{G}}: \boldsymbol{g(x)} \mapsto \boldsymbol{u(t,x)}$ satisfying

$$\mathcal{L}_{PDE} = f(u, u_t, u_x, u_{xx}, \dots) = 0$$

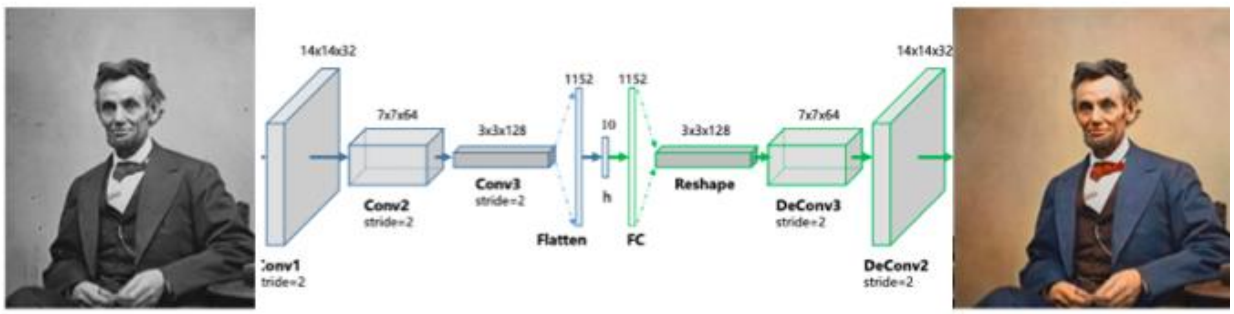
$$\mathcal{L}_{IC} = u(0,x) - g(x) = 0$$

$$\mathcal{L}_{BC} = u|_{\partial\Omega} - h(t,x)|_{\partial\Omega}$$



Data-driven operator learning

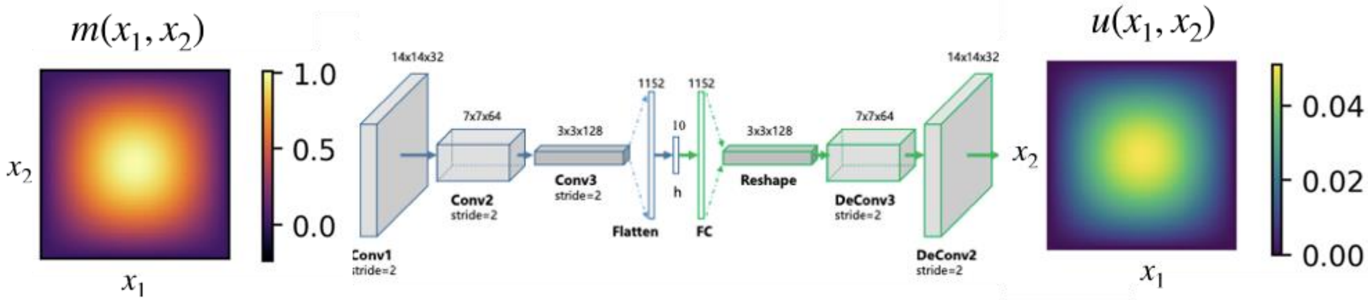
- **CNN encoder-decoder (Image-to-image regression)**
  - Image segmentation, Image Noise Reduction, Image coloring



https://towardsdatascience.com/convolutional-autoencoders-for-image-noise-reduction-32fce9fc1763

Data-driven operator learning

- **Operator learning (function-to-function regression)**



Ex) Poisson equation

$$-\Delta u - m = 0 \text{ in } \Omega \subset \mathbb{R}^2$$
$$u = 0 \text{ in } \partial\Omega$$

Goal) Find a solution $u(x_1, x_2)$ for different source term $m(x_1, x_2)$.
(Learning operator from the source term to the solution)

$$m(x_1, x_2) \mapsto u(x_1, x_2)$$

Ex) Burgers' equation

$$u_t + uu_x = \nu u_{xx}$$
$$u(x,0) = u_0(x)$$

Goal) For a fixed coefficient ν, find a solution $u(t, x)$ for different initial condition $u_0(x)$.
(Learning operator from the initial condition to the solution)

$u_0(x) \mapsto u(x, t = T)$ or $u_0(x) \mapsto u(x, t)$

New models for operator learning : DeepONet(2019), Fourier Neural Operator (FNO) [2020]

My interest [2]

Hwang, R., **Lee, J. Y.**, Shin, J. Y., & Hwang, H. J. (2022, June). Solving pde-constrained control problems using operator learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 36, No. 4, pp. 4504-4512).

- **Solving control problems using operator learning.**

Poisson equation with zero Dirichlet boundary conditions

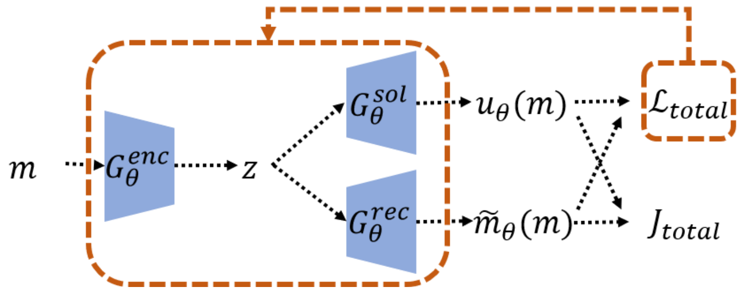
$$\begin{cases} -\Delta u(x) - m(x) = 0, & x \in \Omega, \\ u = 0, & x \in \partial\Omega, \\ m_a \leq m \leq m_b, & x \in \Omega, \end{cases}$$

- Ω : domain of interest
- $u : \Omega \to \mathbb{R}$ (unknown temperature)
- $u_d : \Omega \to \mathbb{R}$ (desired temperature)
- α : (unknown temperature)
- $m : \Omega \to \mathbb{R}$ (source term : control function)

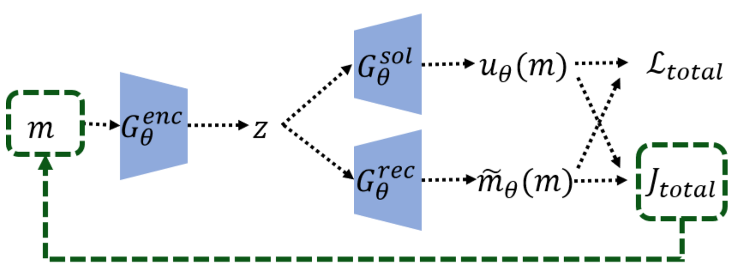
- We want to control $m(x)$ which minimize

$$J_{total}(m(x), u(x))$$

$$= \min \frac{1}{2}\int_\Omega (u - u_d)^2\, dx + \frac{\alpha}{2}\int_\Omega m^2 dx$$



Phase 1 : Operator learning
(Learning $m(x) \mapsto u(x)$)

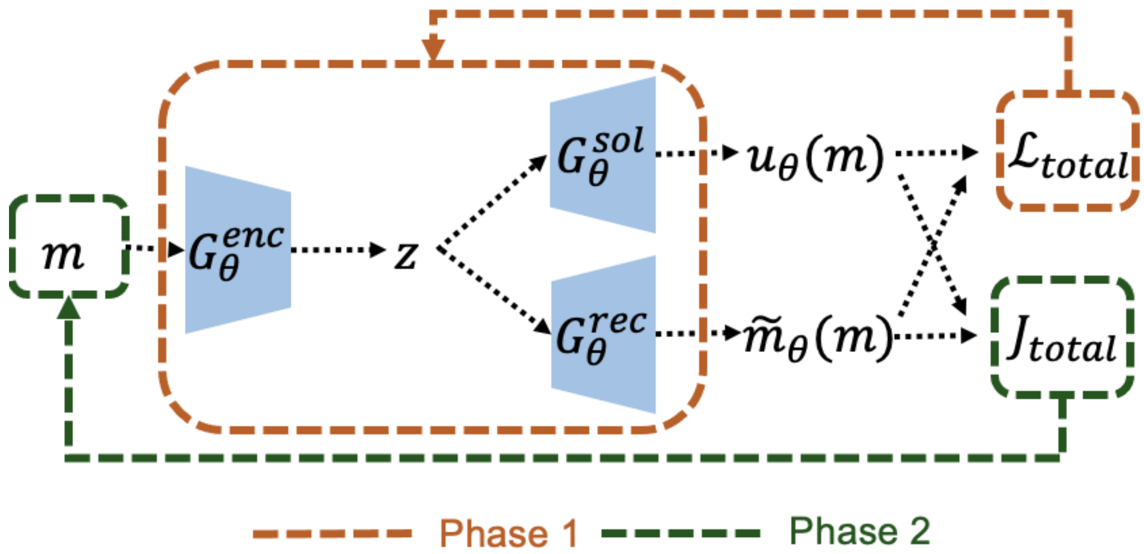


Phase 2 : Solving control problem
(Minimize J_{total} to find optimal $m(x)$)

My interest [2]

Hwang, R., **Lee, J. Y.**, Shin, J. Y., & Hwang, H. J. (2022, June). Solving pde-constrained control problems using operator learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 36, No. 4, pp. 4504-4512).

- **Solving control problems using operator learning.**
  - Our framework approximates a PDE solution with **sufficiently high accuracy** to search optimal controls while taking significantly **less time for inference.**

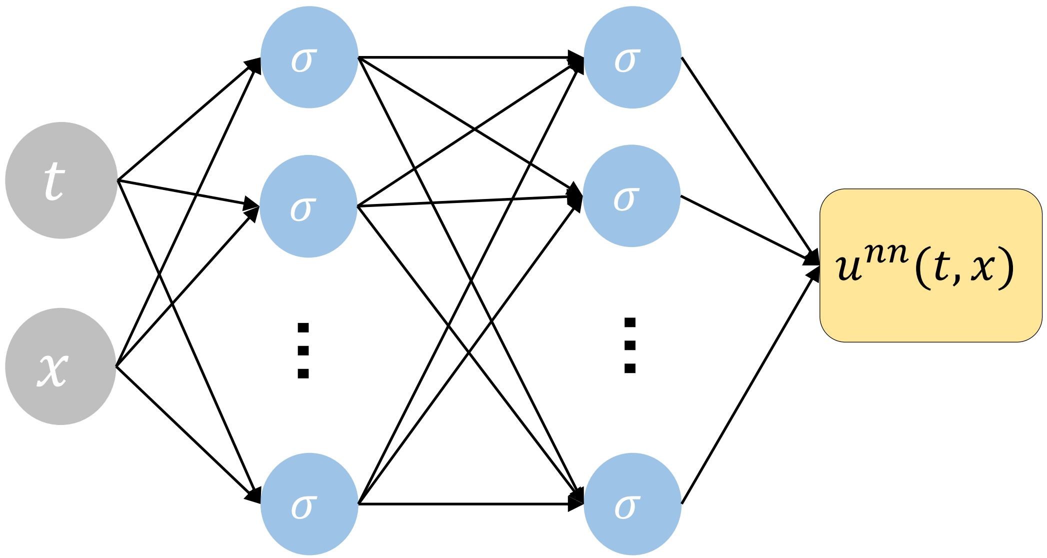


Conclusion

- **PDE solver**

Find $\boldsymbol{u(t,x)}$ satisfying

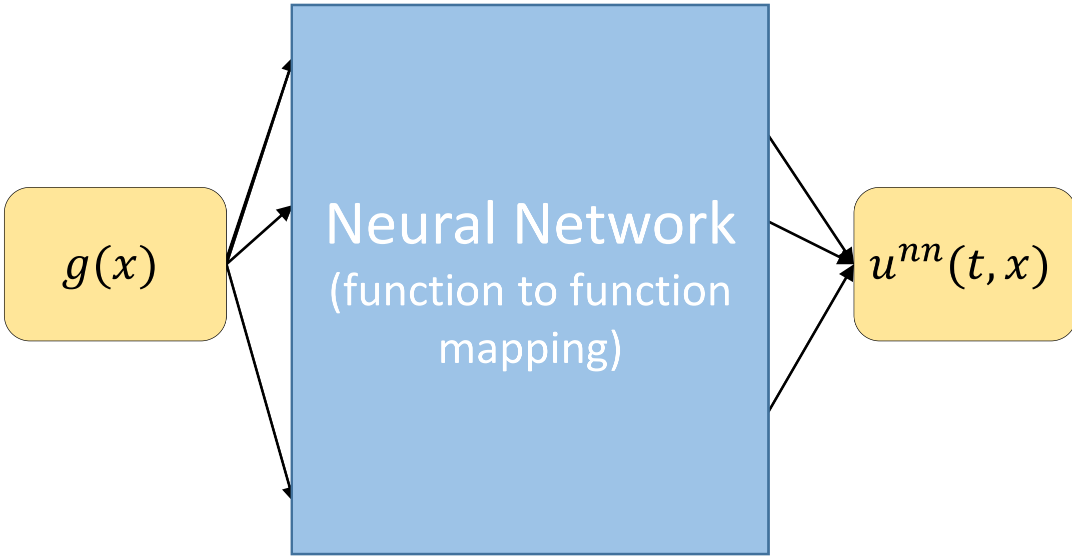
$$\mathcal{L}_{PDE} = f(u, u_t, u_x, u_{xx}, \dots) = 0$$
$$\mathcal{L}_{IC} = u(0,x) - g(x) = 0$$
$$\mathcal{L}_{BC} = u|_{\partial\Omega} - h(t,x)|_{\partial\Omega} = 0$$



- **Operator learning**

Find a map $\boldsymbol{\mathcal{G}}$: $\boldsymbol{g(x) \mapsto u(t,x)}$ satisfying

$$\mathcal{L}_{PDE} = f(u, u_t, u_x, u_{xx}, \dots) = 0$$
$$\mathcal{L}_{IC} = u(0,x) - g(x) = 0$$
$$\mathcal{L}_{BC} = u|_{\partial\Omega} - h(t,x)|_{\partial\Omega}$$



[1] Hwang, H. J., Jang, J. W., Jo, H., & **Lee, J. Y.** (2020). Trend to equilibrium for the kinetic Fokker-Planck equation via the neural network approach. *Journal of Computational Physics*, *419*, 109665.

[2] Hwang, R., **Lee, J. Y.**, Shin, J. Y., & Hwang, H. J. (2022, June). Solving pde-constrained control problems using o perator learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 36, No. 4, pp. 4504-4512)

Thank you